

---

# **Software Requirements**

## **Specification**

**for**

## **IR/OPL Voting System**

Version 1.0 approved

Prepared by Andy Chen(chen6640), Sai Tallapragada(talla037),

Sree Pemma(pemma003), and Samuel Wong(wong0613)

2/19/2021

# Table of Contents

<b>Table of Contents.....</b>	<b>2</b>
<b>1. Introduction.....</b>	<b>3</b>
1.1. Purpose.....	3
1.2. Documentation Conventions.....	3
1.3. Intended Audience and Reading Suggestions.....	3
1.4. Product Scope.....	3
1.5. References.....	4
<b>2. Overall Description.....</b>	<b>2</b>
2.1. Product Perspective.....	4
2.2. Product Functions.....	4
2.3. User Classes and Characteristics.....	5
2.4. Operating Environment.....	5
2.5. Design and Implementation Constraints.....	5
2.6. User Documentation.....	5
2.7. Assumptions and Dependencies.....	6
<b>3. External Interface Requirements.....</b>	<b>6</b>
3.1. User Interfaces.....	6
3.2. Hardware Interfaces.....	6
3.3. Software Interfaces.....	6
3.4. Communication interfaces.....	7
<b>4. System Features.....</b>	<b>7</b>
4.1. Pass in filename.....	7
4.2. Read in CSV file.....	8
4.3. Run IR Election.....	9
4.4. Run OPL Election.....	11
4.5. Break Tie.....	12
4.6. Display Info to Screen.....	13
4.7. Create Audit File.....	14
4.8. Create Statistical Summary.....	15
4.9. ID Ballot.....	16
4.10. Count Ballots.....	17
<b>5. Other NonFunctional Requirements.....</b>	<b>19</b>
5.1. Performance Requirements.....	19
5.2. Safety Requirements.....	19
5.3. Security Requirements.....	19
5.4. Software Quality Attributes.....	19
<b>Appendix A:Glossary.....</b>	<b>19</b>

# **1. Introduction**

## **1.1. Purpose**

The purpose of this document is to present a detailed description of the IR/OPL vote counting software. It will explain the purpose and features of the software, what the software will do, and the constraints under which it must operate. This document is intended for the users of this software and also potential developers.

## **1.2. Document Conventions**

- This document was created based on the IEEE template for System Requirement Specification Documents.
- Each requirement will have its own priority
- The voting system will be referred to as the IR/OPL voting system throughout the document

## **1.3. Intended Audience and Reading Suggestions**

- Election officials who would use this software to generate an election data summary and results.
- Testers who will be running tests to discover bugs to be corrected.
- Software developers who would contribute to the software by developing additional features and/or fixing existing bugs.

## **1.4. Product Scope**

The IR/OPL voting system is a tool to assist election officials in determining the results of an election. Officials can use the software to easily process and generate an audit file containing a summary and the results of the election from an election information data file that's passed into the software, and display the results as well. The software is capable of running 100,000 ballots in under 8 minutes and is programmed to resolve any ties should one arise.

## 1.5. References

IEEE Template for System Requirement Specification Documents:  
<https://goo.gl/nsUFwy>

## 2. Overall Description

### 2.1. Product Perspective

The IR/OPL voting system was developed specifically to aid election officials in running Instant Runoff and Party List Voting style elections. It eliminates the need to hand count votes and provides a quick way to get the results of an election. Media may also find it useful as it also provides a statistical overview of these elections.

### 2.2. Product Functions

File:

- Read in file: Given the filename by the user, opens that file in the program
- Process file: Reads in information from the file such as the election type and its ballots

Voting:

- Instant Runoff: Run an Instant Runoff-style election
- Party List : run a Party List Voting-style election using Open Party Listing
- Determine winner: given either election type, determine the outcome of the election.
- Break Tie: In the event of a tie, execute a fair coin-toss to break it
- Display: At the conclusion of the election, display the winner and other relevant information to the screen

Output:

- Audit: At the end of the election produce an audit file tracking the assignment of each ballot.

- Summary: At the end of the election produce a statistical summary of the election with important information relevant to media personnel

### **2.3. User Classes and Characteristics**

- Election officials, those who are running elections are responsible for counting ballots and determining winners who want to use the software to make this process easier and much faster.
- Media Personnel, media covering elections who want to use the software to produce important statistics from the election.
- Programmers/testers who want to validate/improve the software.

### **2.4. Operating Environment**

- Windows XP
- Windows 7 32/64 bit
- Windows 8 32/64 bit
- Windows 10 32/64 bit
- General Unix/Linux systems
- CSE Lab Unix/Linux computers(Ubuntu)
- Mac OS
- Ubuntu
- 16 GB RAM or more available for each OS

### **2.5. Design and Implementation Constraints**

- It is a runtime constraint that the program must be able to run 100,000 ballots in under 8 minutes.
- There is no way to change the file structure outside of the program, all files come into the program in a predetermined format.
- There are no security constraints on this program.

### **2.6. User Documentation**

Upon starting the program the user will be prompted to enter the file name containing the election information. From there the program will determine which type of election is to be run and run that type of election. The program will automatically produce a statistical summary to be shared with media personnel and an audit file in the same directory as the program and input file upon completion of the election.

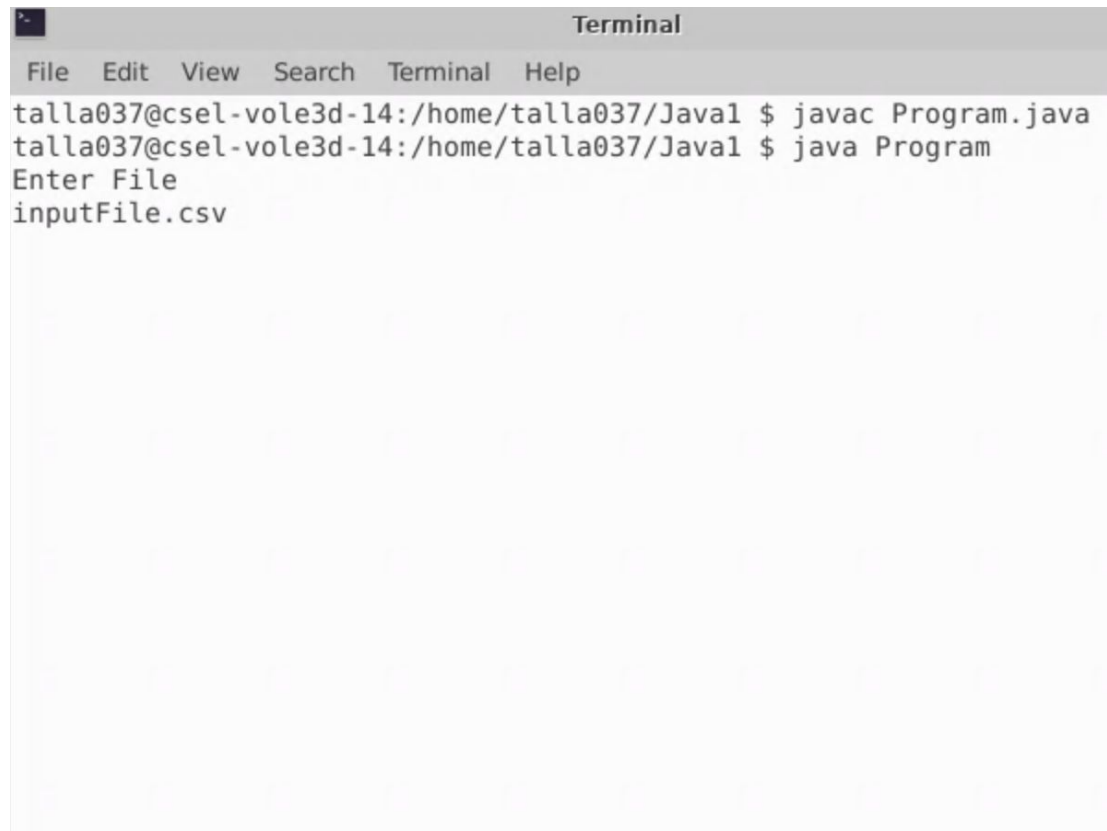
## **2.7. Assumptions and Dependencies**

- It is assumed that the election file will be located in the same directory as the program itself
- All ballots are assumed to be correct and completely filled out (i.e. for IR voting, all ballots have choices 1-4)
- The program is written in java and hence requires java to be installed on the users machine.

### 3. External Interface Requirements

#### 3.1. User Interfaces

We will be using the terminal window to compile and run our java program. Please refer to the following image below which shows how to compile a java program with javac and how to run the program after compilation.

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal shows the following commands and output:

```
talla037@csel-vole3d-14:/home/talla037/Java1 $ javac Program.java
talla037@csel-vole3d-14:/home/talla037/Java1 $ java Program
Enter File
inputFile.csv
```

#### 3.2. Hardware Interfaces

A keyboard, mouse or trackpad, and monitor or other display device will be required to interact with the software.

The minimum hardware requirement for our software is atleast a intel core i3 4th generation or higher with at least 16 GB of ram to run everything smoothly along with other system processes. We also need a keyboard or a similar input device to type.

### 3.3. Software Interfaces

Our software system will require Java to be installed on the system, specifically preferring the version Java 10.

### 3.4. Communication Interfaces

An internet connection would be required in the case that it's necessary to download additional Java plugins that may not have been originally installed. After that, no internet connection is required to run the system.

## 4. System Features

### 4.1. Pass in filename

<b>Name</b>	Pass in filename
<b>ID</b>	UC_001
<b>Description</b>	Election official types in the name of the file that the program is to run on.
<b>Actors</b>	Election Official
<b>Organizational Benefits</b>	Allows files storing election/ballot data to be passed into the IR/OPL voting system.
<b>Frequency of Use</b>	Each time an election needs to be processed, the file name needs to be passed into the system.
<b>Triggers</b>	Upon starting the system the system will prompt the user to input the filename.
<b>Preconditions</b>	An election needs to have been run and the election official has started the IR/OPL system.
<b>Postconditions</b>	IR/OPL system has found the file with



	the name that has been passed in. The election official can see confirmation that the file has been found.
<b>Main Course</b>	<ol style="list-style-type: none"> <li>1. User runs IR/OPL voting system</li> <li>2. User is prompted to type in the filename</li> <li>3. User types in file name</li> <li>4. Confirmation that the file is found/not found is printed to the screen</li> </ol>
<b>Alternate Courses</b>	AC1 User quits out of program <ol style="list-style-type: none"> <li>1. Provide user with prompt providing an option to exit the program instead of inputting the filename</li> </ol>
<b>Exceptions</b>	EX1 File not found <ol style="list-style-type: none"> <li>1. System prompts user to input filename a second time.</li> <li>2. If filename is found user continues at Main Course #4</li> <li>3. If file is still not found print out error message</li> </ol>

## 4.2 Read in CSV file

<b>Name</b>	Read in CSV file
<b>ID</b>	UC_002
<b>Description</b>	For every election, we will receive a file containing all the ballots and the election information in a CSV format.
<b>Actors</b>	System
<b>Organizational Benefits</b>	By reading in the CSV file, the system is given the election data from the file and is capable of calculating election winners and summaries.
<b>Frequency of Use</b>	Each time an election needs to be processed the file name needs to be passed into the program.

<b>Triggers</b>	Upon starting the system, it prompts the user to input a file. The user inputs a CSV file
<b>Preconditions</b>	An election needs to be run and the election official has started the IR/OPL software. The user must also have already input the CSV file containing the election information to be used for processing the results of the election.
<b>Postconditions</b>	After reading the CSV, we will store the election information which includes, candidates, ballots, type of election, etc.
<b>Main Course</b>	The program reads the CSV file.
<b>Alternate Courses</b>	No alternate courses.
<b>Exceptions</b>	EX1 The system determines that the input file is not in the correct CSV format <ol style="list-style-type: none"> <li>1. Notify the user if there is something wrong with the file</li> </ol>

### 4.3 Run IR Election

<b>Name</b>	Run IR Election
<b>ID</b>	UC_003
<b>Description</b>	Read the inputted CSV file to generate election results and a statistical summary for an IR type election.
<b>Actors</b>	System
<b>Organizational Benefits</b>	Election results and a statistical summary can be generated automatically by the system without requiring human effort. It can be done more quickly and accurately compared to what can be done by human calculations.
<b>Frequency of Use</b>	Every time an IR election occurs, the system will be given the ballots to calculate the results.

<b>Triggers</b>	When the input CSV file has been read in and the election type is identified as IR, the IR election program will be triggered.
<b>Preconditions</b>	<p>An election has been run and it's details and ballots have been stored in a CSV file.</p> <p>A file has been passed into the IR/OPL system.</p> <p>The input CSV file read in previous steps needs to say IR election in the first line.</p>
<b>Postconditions</b>	Election results and the statistical summary are generated by the system for the OPL election.
<b>Main Course</b>	<ol style="list-style-type: none"> <li>1. Reads number of candidates from CSV file</li> <li>2. Reads in each candidate listed in the file</li> <li>3. Reads in total number of ballots</li> <li>4. Reads and IDs each ballot one by one until all ballots have been read (see UC_009 and UC_010)</li> <li>5. Calculate a summary and election results from the ballot count and other details.</li> <li>6. Go to use case UC_006 to display a summary.</li> </ol>
<b>Alternate Courses</b>	<p>AC1 on event of a tie, the election winner cannot be determined without randomization</p> <ol style="list-style-type: none"> <li>1. Tie will be broken in UC_005</li> </ol>
<b>Exceptions</b>	As validity of file format is verified in the reading of the CSV files and we can assume that there are no errors in the ballots, we can say there are no exceptions unless the system stops working/breaks down. Under these circumstances, users will be notified that summary and results cannot be calculated.

#### 4.4 Run OPL Election

<b>Name</b>	Run OPL Election
<b>ID</b>	UC_004
<b>Description</b>	Read the inputted CSV file to generate election results and a statistical summary for an OPL type election.
<b>Actors</b>	System
<b>Organizational Benefits</b>	Election results and a statistical summary can be generated automatically by the system without requiring human effort. It can be done more quickly and accurately compared to human calculations.
<b>Frequency of Use</b>	Every time an OPL election occurs, the system will be given the ballots to calculate the results.
<b>Triggers</b>	When the input CSV file has been read in and the type is identified as OPL, the OPL election program will be triggered.
<b>Preconditions</b>	An election has been run and it's details and ballots have been stored in a CSV file. All independents grouped into one party. The input CSV file read in previous steps needs to say OPL election in the first line.
<b>Postconditions</b>	Election results and the statistical summary are generated by the system for the OPL election.
<b>Main Course (Unfinished)</b>	<ol style="list-style-type: none"> <li>1. Reads number of candidates from CSV file</li> <li>2. Reads in each candidate and party from file</li> <li>3. Reads in number of seats</li> <li>4. Reads in number of ballots</li> <li>5. Reads in each ballot from the file</li> <li>6. Calculates the results from</li> </ol>

	reading in each ballot from file 7. Go to use case UC_006 to display a summary.
<b>Alternate Courses</b>	AC1 on event of a tie, the election winner cannot be determined without randomization 1. Tie will be broken in UC_005
<b>Exceptions</b>	As validity of file format is verified in the reading of the CSV files and we can assume that there are no errors in the ballots, we can say there are no exceptions unless the system stops working/breaks down. Under these circumstances, users will be notified that summary and results cannot be calculated.

#### 4.5 Break Tie

<b>Name</b>	Break Tie
<b>ID</b>	UC_005
<b>Description</b>	A virtual “coin” simulated by Java’s random number generation capabilities is “flipped” many times to ensure fairness to determine the winner of an election when there is a tie.
<b>Actors</b>	Election Official
<b>Organizational Benefits</b>	Further voting will be unnecessary by flipping a coin to determine the winner of a tied election. Time and other resources such as money will be saved by doing so.
<b>Frequency of Use</b>	Whenever a tie occurs in an election, the process of breaking the tie will be necessary.
<b>Triggers</b>	A tie has occurred either in the IR or OPL election processing.
<b>Preconditions</b>	Assuming the preconditions for the previous use cases have been met,

	the only precondition is that the alternate course has been taken for either UC_003 or UC_004 to lead to this use case.
<b>Postconditions</b>	The tie is broken by flipping a virtual coin multiple times for fairness and selecting the 1001th flip to determine the winner.
<b>Main Course</b>	1. A virtual coin will be flipped 1000 times for fairness and the 1001 coin flip result will be used to determine winner
<b>Alternate Courses</b>	No alternate courses
<b>Exceptions</b>	No exceptions

#### 4.6 Display Info to Screen

<b>Name</b>	Display info to screen
<b>ID</b>	UC_006
<b>Description</b>	Display election information that includes things such as the type of election, the winner, and number of casted ballots to the screen.
<b>Actors</b>	System
<b>Organizational Benefits</b>	The information that is displayed to the screen is the main summary statistics/information that is necessary after an election. It is concise and provides an organized way of presenting the results.
<b>Frequency of Use</b>	Anytime an election occurs, the processed election information will need to be displayed to the screen.
<b>Triggers</b>	After the election has ended and a winner has been declared either post tie or without a tie break, the results and summary stats will be generated and this will trigger the display.

<b>Preconditions</b>	Assuming all the preconditions have been met in the use cases that lead to this, an election ending and a winner being determined is the only precondition.
<b>Postconditions</b>	The election results along with other summary stats will be displayed to the screen.
<b>Main Course</b>	<ol style="list-style-type: none"> <li>1. The winner of the election will be displayed to screen</li> <li>2. Type of election and number of seats will also be on screen</li> <li>3. Number of casted ballots and the number of votes each candidate received will be displayed as well</li> </ol>
<b>Alternate Courses</b>	No alternate courses
<b>Exceptions</b>	<p>EX1 Displaying to screen goes wrong</p> <ol style="list-style-type: none"> <li>1. The system notifies to the screen that displaying the information is not possible.</li> <li>2. The cancel of try again buttons will show up for the viewer to choose from</li> <li>3. If try again is chosen, the main course steps are repeated</li> </ol>

#### 4.7 Create Audit File

<b>Name</b>	Create audit file
<b>ID</b>	UC_007
<b>Description</b>	At the end of the program an audit file is automatically produced containing information necessary to validate the election.
<b>Actors</b>	System
<b>Organizational Benefits</b>	Allows an election's results to be validated by tracking each ballot in the audit file.
<b>Frequency of Use</b>	Every time the election is run there will

	be an audit file produced.
<b>Triggers</b>	Election is finished running, winners have been determined.
<b>Preconditions</b>	Each ballot is assigned an ID and belongs to one candidate. The election has finished running and a winner has been determined
<b>Postconditions</b>	An audit file has been created in the same directory the program and the initial election file are located. Each ballot in the audit file is able to be tracked to a single candidate.
<b>Main Course</b>	<ol style="list-style-type: none"> <li>1. IR/OPL voting system is started</li> <li>2. User is prompted for filename</li> <li>3. User inputs filename</li> <li>4. IR/OPL ID's each ballot as it's counted</li> <li>5. System uses ballot ID's to track where each ballot is going</li> <li>6. Ballot information is stored</li> <li>7. Ballot information is written to a file created in the same directory as the program itself</li> </ol>
<b>Alternate Courses</b>	No alternate courses
<b>Exceptions</b>	EX1: <ol style="list-style-type: none"> <li>1. Program is killed before an audit file can be created</li> <li>2. In order to produce an audit file the program must be run again with the same input file</li> </ol>

#### 4.8 Create Statistical Summary

<b>Name</b>	Create statistical summary
<b>ID</b>	UC_008
<b>Description</b>	Once the program finishes running the election, the program will create a file containing relevant statistical information such as, winners, vote



	percentages, ect. And store this file in the same directory as the program and audit file.
<b>Actors</b>	System
<b>Organizational Benefits</b>	Allows the organization to easily share relevant statistical information about the election to media personnel.
<b>Frequency of Use</b>	Each time the program is run it will produce a statistical summary file.
<b>Triggers</b>	The program finishes running an election
<b>Preconditions</b>	Election has finished running and a winner has been determined. Each ballot is tied to one candidate and this information can be found.
<b>Postconditions</b>	A file containing relevant statistics to the election is present in the directory that the program is located in.
<b>Main Course</b>	<ol style="list-style-type: none"> <li>1. User runs IR/OPL voting system</li> <li>2. User is prompted to input filename</li> <li>3. User inputs filename</li> <li>4. Program runs election and determines winners</li> <li>5. While running election each ballot is tracked and given an ID</li> <li>6. Using ballot information determines statistics like winner, percentages of votes, total number of votes, ect.</li> <li>7. Stores statistical information in a file that is created in the same directory as the program</li> </ol>
<b>Alternate Courses</b>	No alternate courses
<b>Exceptions</b>	EX1: <ol style="list-style-type: none"> <li>1. Program is killed before it finishes creating statistical summary file</li> <li>2. In order to create this file, the</li> </ol>

	program will have to be run again with the same input file
--	--

#### 4.9 ID Ballot

<b>Name</b>	ID Ballot
<b>ID</b>	UC_009
<b>Description</b>	Each ballot that was cast in the election will be assigned when it is counted.
<b>Actors</b>	The IR/OPL system will have a method which assigns IDs for every ballot.
<b>Organizational Benefits</b>	Automatically assigns IDs to every ballot without manual input by the election official. This helps us track each and every individual ballot as it is being counted throughout the election.
<b>Frequency of Use</b>	We will assign an ID to every ballot the first time the program encounters them.
<b>Triggers</b>	When reading the file containing the election information, the program will come across information about ballots so we will assign IDS to each of the ballots casted in the election at that time.
<b>Preconditions</b>	The Program is reading the CSV file.
<b>Postconditions</b>	Every Ballot has an ID attached to it.
<b>Main Course</b>	<ol style="list-style-type: none"> <li>1. The Program will get started up by an election official.</li> <li>2. The election official will input a csv file containing the election information. Our program will read the election information and process it.</li> <li>3. As we come across information regarding ballots casted in the election, we will</li> </ol>

	give each ballot a unique ID.
<b>Alternate Courses</b>	No alternate courses
<b>Exceptions</b>	If the format of how the ballots are represented is not consistent in the CSV file, the problem may call an error or run into one.

#### 4.10 Count Ballots

<b>Name</b>	Count Ballots
<b>ID</b>	UC_010
<b>Description</b>	After assigning an ID to each ballot, we will have to start counting the number of ballots casted for each candidate
<b>Actors</b>	The Election Official will be observing the results of this process.
<b>Organizational Benefits</b>	There is no need for election officials to manually count the ballots as our program will automatically count all the ballots casted for each candidate.
<b>Frequency of Use</b>	Everytime a CSV file has been read by the system, the number of ballots will be counted for each candidate.
<b>Triggers</b>	Once a CSV file has been fully read, and the IR or OPL election runner calls count.
<b>Preconditions</b>	We would need to read the CSV file and store all the election information regarding ballots in the system.
<b>Postconditions</b>	Each candidate will have a count of how many ballots were casted for them.
<b>Main Course</b>	<ol style="list-style-type: none"> <li>1. File has been fully read</li> <li>2. Depending on the election type, the IR or OPL election</li> </ol>

	runners will call for count 3. Count the number of ballots for each candidate.
<b>Alternate Courses</b>	AC1 in an IR election recount ballots once a candidate has been eliminated 1. Recount (run UC_010 again)
<b>Exceptions</b>	No exceptions

## 5. Other NonFunctional Requirements

### 5.1. Performance Requirements

As listed in the design and implementation constraints, the program must be able to run 100,000 ballots in under 8 minutes so our program requires a system with at least 16 GB RAM.

### 5.2. Safety Requirements

There are no specific safety requirements as such for our voting system.

### 5.3. Security Requirements

Our program does not have any security requirements and thus any type of user can use it without any additional privileges.

### 5.4. Software Quality Attributes

Users will be provided with simple steps to use the system and complete their intended tasks. The system can be used by both experts and typical users with ease.

### 5.5 Business Rules

We are not looking to launch this product into the business world so we don't have any current requirements regarding that.

## **6. Other Requirements**

### **Appendix A: Glossary**

- IR - Instant Runoff election type
- OPL - Open Party List election type