Software Requirements Specification

for

WaterFall

Version 1.0 approved

Prepared by Mohamed Mohamed, Danial Syed,

Omavi Collison, Nabeel Azam,

UMN CSCI 5801

Feb, 01, 2023

Table of Contents

Revision History -

Name	Date	Reason For Changes	Version
Mohamed, Danial, Nabeel, Omavi	02/16/23	First Submission	v1.0
Mohamed, Danial, Nabeel, Omavi			

1. Introduction

1.1 Purpose

The purpose of this document is to showcase a detailed explanation of the system that will be developed for this project. This document will go over the purpose and the features of the system, and how it will be utilized. This document is intended for users of the software as well as potential testers.

1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement Specification Documents. The Requirements are displayed at the end.

1.3 Intended Audience and Reading Suggestions

The intended audience of this document would be anyone looking to utilize this system to calculate election winners based on IR or CPL election types. Any programmers looking to work on this system should also make use of this document.

1.4 Product Scope

The goal of this software is for officials to be able to calculate the winners of an election with the system we have created. Our goal is for it to be able to handle 2 different election types, and for it to generate a winner even in the result of ties. The two types of elections it will be able to handle are IR and CPL election types. The user will be able to simply feed the program a CSV file with election information and ballots and the system will be able to produce a winner based on the results.

1.5 References

Project GitHub page:

https://github.umn.edu/umn-csci-5801-01-S23/repo-Team20

2. Overall Description

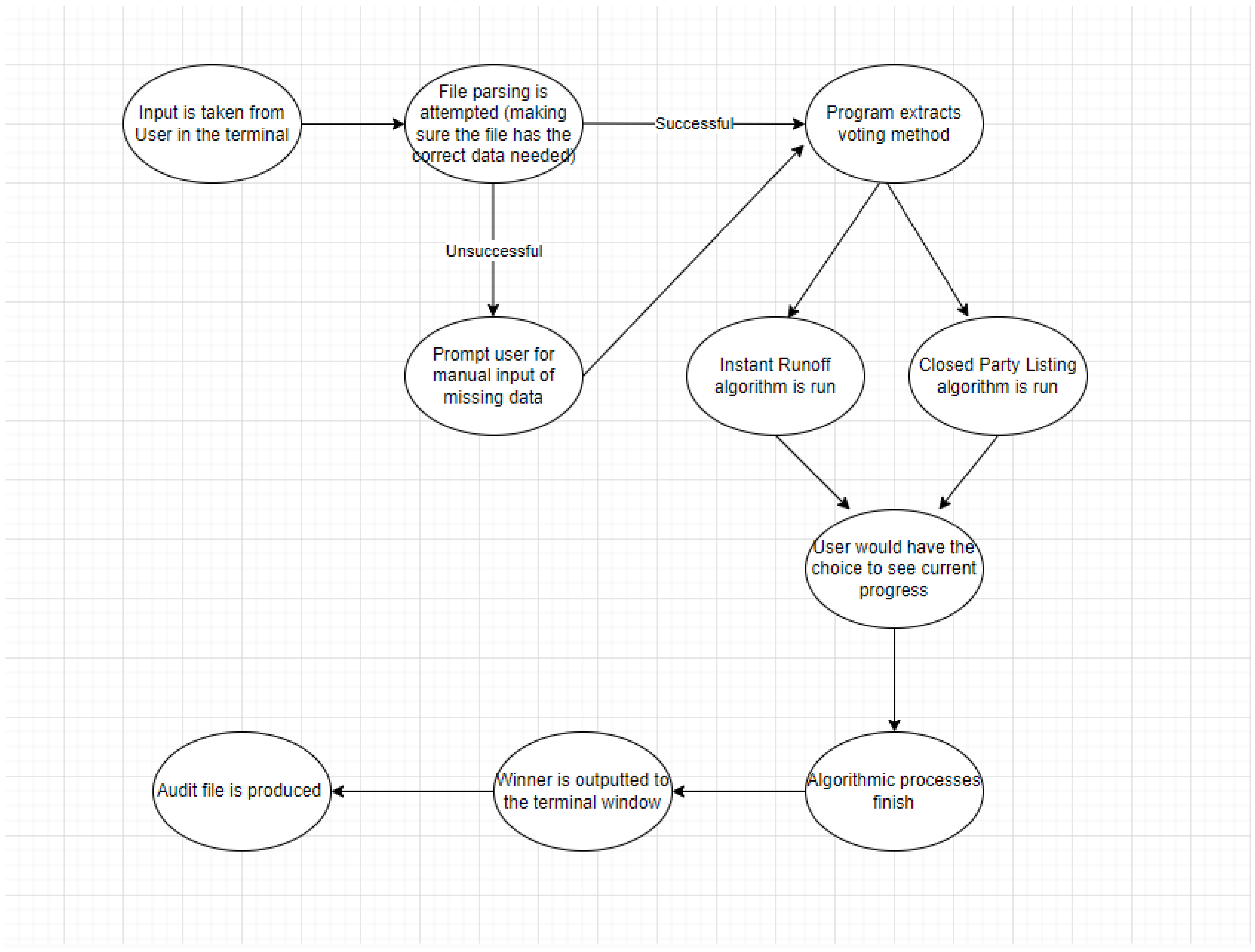
2.1 Product Perspective

The Voting System project was made to determine and produce a winning candidate based on given ballots. This voting algorithm supports Instant Runoff and Closed Party Listing elections given a single or multiple ballot files in CSV format.

This project was developed by 4 students under the direction of Professor Shana Watters. It was developed to run on Windows, Mac OS, Linux, and more specifically our University of Minnesota lab machines which also run on Linux.

2.2 Product Functions

High-level summary of the project (further details see section 3).


Input is taken from User in the terminal
File parsing is attempted (making sure the file has the correct data needed)
Successful
Program extracts voting method
Unsuccessful
Prompt user for manual input of missing data
Instant Runoff algorithm is run
Closed Party Listing algorithm is run
User would have the choice to see current progress
Algorithmic processes finish
Winner is outputted to the terminal window
Audit file is produced

2.3 User Classes and Characteristics

- Programmers: Developers who have access to the code and want to inspect algorithm choice and usage to determine the winning candidate
- Testers: Analyst who monitors the quality of the program with regards to what the election officials need
- Election Officials: Moderators who must view the results of elections

2.4 Operating Environment

Our program is able to run on the following environments:

- Windows 10-11
- MacOs 13
- Linux (More specifically University of Minnesota lab machines which always use the most up-to-date version)
    - Note: For local machines, SSH(secure shell connection) is also an option. SSH is used to remotely access University of Minnesota lab machines which allows operation and exploitation of the lab machine and its operating system

Our project was run and made through the Visual Studio Code IDE and using the JDK version 17 of Java. Visual Studio is available on all of the above environments. Our code was then pushed to a local GitHub repository using Git tools. Git allows local code that is only viewable on an individual's machine to be made publicly or privately available to others. This methodology was implemented in order to view and retrieve the progress of members of the project.

2.5 Design and Implementation Constraints

There are no corporate or regulatory policies that must be followed for this project. There are also no hardware requirements as this is not an intensive application to run. All that is needed to run the program is an environment that can run Java (this can be an online compiler, so no downloading is needed).

2.6 User Documentation

Guide on how Instant Runoff Voting works:

https://www.house.mn.gov/hrd/pubs/irvoting.pdf

Guide on how Closed Party Listing Voting works:

https://www.cheltenham.gov.uk/info/17/elections_and_voting/1213/voting_systems_in_the_uk#:~:text=Proportional%20representation%20

2.7 Assumptions and Dependencies

Assumptions:

- No more than one file is given per election
- First line of the ballot file will always contain the type of election
- No safety or security requirements are expected
- No errors in the ballots
    - For Instant Runoff ballots specifically, at least one candidate is ranked
    - Ranking Numbers will not have issues (for example, if there are 4 ranks, then expect candidates to be ranked 1,2,3,4 as oppose to a missing 3rd candidate)
    - Voting numbers are 100% accurate

Dependencies:

This project is built from scratch and therefore does not rely on any previous projects.

3. External Interface Requirements

3.1 User Interfaces

The software will be primarily use the command line interface as its user interface. This means that the majority of user interactions will be through text-based input and output, rather than a graphical user interface. The logical characteristics of this interface will be defined in a separate user interface specification.

3.2 Hardware Interfaces

The interface between the hardware and software components that make up the CSE lab machine is a key factor in determining the system's operation and performance. The processor is the main piece of hardware that carries out software instructions, and the operating system serves as a bridge between software applications and hardware. While the storage devices permanently store data and information, the memory stores data and processor-related instructions. Users can engage with the system using input/output devices, and communication protocols like USB make it easier for the various parts to talk to one another.

3.3 Software Interfaces

The software product interfaces with databases, OS, tools, libraries, and commercial components. Data input/output and purpose are specified. Requirements include CSV file access, support for macOS, Linux, Windows 11, JDK version 17, and version control via Github/Git.

3.4 Communications Interfaces

The product does not require any communication functions, as it operates locally and does not require remote communication. The user interface is mostly through the command line terminal or GUI. Communication between polling sites and election offices is outside the scope of this project. Operating locally offers benefits such as improved speed and efficiency, and eliminates the need for security and encryption. Not to mention communication between the polling station and the election officials are outside the scope of this project.

4. System Features

4.1 Exporting Audit File

4.1.1 Description and Priority:

This feature allows election officials to export the audit file once the election is complete and the file is produced. It is of high priority as it ensures transparency and accountability in the election process.

4.1.2 Stimulus/Response Sequences:

Stimulus:
The election official initiates the export of the audit file.
Response:
The system prompts the user to select the format and destination for the exported file.
Stimulus:
The user selects the desired format and destination.
Response:
The system exports the audit file to the selected destination in the selected format.

4.1.3 Functional Requirements:

REQ1-1: The system must allow the election official to initiate the export of the audit file.
REQ1-2: The system must prompt the user to select the format and destination for the exported file. The system must allow the user to select the desired format and destination.
REQ1-3: The system must export the audit file to the selected destination in the selected format.
REQ1-4: The system must confirm the successful export of the audit file.
If errors or issues arise with the audit file, the system must prompt the user to resolve the errors before exporting the file. If the user cancels the export process, the system must abort the export and return to the main menu.

4.2 Display the winner on the Screen

4.2.1 Description and Priority

This feature allows users to view the winner of the election on the screen. It is of a high priority since it is essential for providing integrity and trust in the election process.

4.2.2 Stimulus/Response Sequences

Stimulus:
User finishes casting a vote in the system
Response:
The system displays the current winner of the election on the screen

4.2.3 Functional Requirements

REQ2-1: The system should be able to record and store all votes cast in the election
REQ2-2: The system should be able to calculate the winner of the election based on the recorded votes
REQ2-3: The system should be able to display the winner of the election on the screen and
REQ2-4: In case of a tie, the system should be able to display The winner after the tie is broken on the screen
REQ2-5: The system should handle and display an appropriate message if there are no recorded votes in the system yet.

4.3 Running Instant Runoff (IR) Election

4.3.1 Description and Priority

This feature allows the system to use the Instant Runoff (IR) algorithm to determine the winner of the election and display the winner on screen. It is of a high priority since it is essential for providing transparency and accuracy in the election process.

4.3.2 Stimulus/Response Sequences

Stimulus:

The user selects IR as the election type and initiates the calculation of the winner

Response:

The system reads the CSV file with the votes, calculates the winner using the IR algorithm, and displays the winner on screen.

4.3.3 Functional Requirements

REQ3-1: The system should be able to read and store all votes cast in the election in a CSV file.

REQ3-2: The system should be able to use the IR algorithm to calculate the winner of the election based on the recorded votes.

REQ3-3: The system should be able to display the winner of the election on the screen.

REQ3-4: In case of a tie, the system should be able to display all winners on the screen.

REQ3-5: The system should handle and display an appropriate error message if no votes were cast in the election or there is an error with the CSV file.

REQ3-6: The user should be able to select another election type if they do not wish to use the IR algorithm, such as CPL (closed-party Election).

4.4 Produce an Audit file

4.4.1 Description and Priority

This feature allows the system to produce an audit file for an election, providing an official record of the election results. It is of a high priority since it is essential for ensuring accountability and transparency in the election process.

4.4.2 Stimulus/Response Sequences

Stimulus:

The user selects the option to produce an audit file for the election

Response:

The system generates an audit file containing the election results in a standardized format, such as CSV or tex-filet.

4.4.3 Functional Requirements

REQ4-1: The system should be able to generate an audit file for an election in a standardized format, such as CSV or txt(text).

REQ4-2: The audit file should contain a record of all votes cast in the election and the winner of the election.

REQ4-3: The audit file should be easily accessible and exportable for election officials or auditors to review.

REQ4-4: The system should be able to handle and display an appropriate error message if there are no recorded votes in the system yet or if there is an error generating the audit file.
REQ4-5: The user should be able to select and produce an audit file for any past election in the system.

4.5 Break Tie

4.5.1 Description and Priority
This feature enables the system to break a tie in the election by randomly flipping a coin to select a winner. It is of high priority since it ensures an unbiased and efficient way of determining the winner.

4.5.2 Stimulus/Response Sequences
Stimulus:
The election results indicate a tie
Response:
The system randomly flips a coin to determine the winner
4.5.3 Functional Requirements
REQ5-1: The system should be able to detect a tie in the election results.
REQ5-2: The system should be able to randomly flip a coin to determine the winner in case of a tie.
REQ5-3: The system should notify the users of the result of the coin toss and the winner of the election.
REQ5-4: The system should log the coin toss and winner selection for auditing purposes.
REQ5-5: The system should handle and display an appropriate error message if there is an issue with the coin toss.
REQ5-6: The system should be able to handle multiple ties in a single election.

4.6 Read in the file

4.6.1 Description and Priority

This feature allows the system to read in an election file and record the details of the election, including candidate and party information as well as the votes cast. It is of high priority since it is essential for accurately recording and processing election data.

4.6.2 Stimulus/Response Sequences
Stimulus:
The user inputs an election file to be read by the system
Response:
The system reads the file and records the election data, including candidate and party information as well as the votes cast.
4.6.3 Functional Requirements

REQ6-1: The system should be able to read in one election file at a time.
REQ6-2: The system should be able to properly parse and interpret the data in the file, including candidate and party information as well as the votes cast.
REQ6-3: If the file contains errors or incorrect formatting, the system should display an appropriate error message and not record the data.
REQ6-3: The system should be able to record the election data accurately and store it in a database or file for further processing.
REQ6-5: The system should handle and display an appropriate error message if no file is inputted or if the file does not contain any election data.

4.7 Processing the Ballots

4.7.1 Description and Priority

This feature involves the system's ability to read a CSV file that contains ballots, ensuring that the file is correctly formatted and processed. It is of high priority since it is essential for the accuracy and fairness of the election results.

4.7.2 Stimulus/Response Sequences
Stimulus:
The user launches the program and initiates the processing of the ballots file.
Response:
The system reads the CSV file, checks the formatting and content, and determines the appropriate algorithm to use for selecting the winner.

4.7.3 Functional Requirements
REQ7-1: The system should be able to read and store all ballots cast in the election from the CSV file.
REQ7-2: The system should be able to check that the file is correctly formatted and contains the necessary election-type information.
REQ7-3: The system should be able to determine the appropriate algorithm to use for selecting the winner based on the election-type information.
REQ7-4: The system should handle and display an appropriate error message if the file is not correctly formatted or if it does not contain the necessary information.
REQ7-5: The user should be able to select another file to read if the current file is not suitable.
REQ7-6: The system should be able to process only one file at a time to avoid confusion and ensure accuracy.
REQ7-7: The system should be able to produce the winner based on the selected algorithm and display the results on the screen.
REQ7-8: The system should be able to provide an audit file of the election results.

4.8 Check File Format

4.8.1 Description and Priority:

The "Checking File Format" system feature is essential for ensuring that the correct file is inputted into the system. The feature is a high priority, as it affects the accuracy and efficiency of the election process.

4.8.2 Stimulus/Response Sequences:
Stimulus:
The user inputs a file name for the election into the system.
Response:
The system checks the format of the file name to ensure that it matches the expected format(csv or txt) for the election.
If the format of the file name is correct, the system proceeds to read the ballots from the file and the election process continues to the next stage.
If the format of the file name is incorrect, the system displays an error message to the user indicating that the file name is invalid and prompts the user to enter a valid file name.
Stimulus:
The file specified by the user does not exist, cannot be opened or is not in the expected format.
Response:
The system displays an error message indicating that the file cannot be found, or opened, or that the file format is invalid. The user is prompted to enter a valid file name.

4.8.3 Functional Requirements:
REQ8-1: The system shall check the file name format to ensure it matches the expected format.
REQ8-2: The system shall allow the election process to continue if the file name format is correct.
REQ8-3: The system shall display an error message if the file name format is incorrect, and prompt the user to enter a valid file name.
REQ8-4: The system shall display a warning message if the file name format is incorrect, but still proceed to read the ballots from the file.
REQ9-5: The system shall display an error message if the file specified by the user does not exist, cannot be opened, or is not in the expected format, and prompt the user to enter a valid file name.

4.9 Identify the File

4.9.1 Description and Priority:

Description: This feature allows different actors, such as programmers, testers, and election officials to identify and access the necessary file to perform their tasks, making it a high priority feature.

4.9.2 Stimulus/Response Sequences:

Stimulus:
User wants to access a file for processing
Response:
Program displays options for file selection
Stimulus:
User selects a file
Response:
Program confirms the selection and allows the user to perform their task with the selected file

4.9.3 Functional Requirements:
REQ9-1: The program must provide a user-friendly interface for file selection
REQ9-2: The program must display a list of files available for selection
REQ9-3: The program must allow the user to navigate to the desired directory to select a file
REQ9-4: The program must confirm the user's file selection before proceeding with processing the file
REQ9-5: The program must handle errors gracefully, such as notifying the user if the file is not found or is in the wrong format.

4.10 User Input Prompt

4.10.1 Description and Priority:

This system feature is of high priority and is essential for user interaction with the system. It allows the user to input their desired purpose for the system, whether it is to evaluate ballots, see results, or observe the current process. This is a Medium Priority Feature.

4.10.2 Stimulus/Response Sequences:
The system will display a message through the terminal asking the
Stimulus:
user to input their desired input.
Response:
The system will display a message through the terminal asking the
Upon receiving input, the system will store the input and use it to perform the corresponding action(reading in the file and processing it).
Alternate Response:
If the input is not recognized, the system will display an error message and prompt the user to enter a valid input.

4.10.3 Functional Requirements:
REQ10-1: The system must be able to display a message through the terminal asking for user input.
REQ10-2: The system must be able to store the user's input into a variable.

REQ10-3: The system must be able to recognize and perform the corresponding action based on the user's input.
REQ10-4: The system must be able to display an error message and prompt the user to enter a valid input if the input is not recognized.

4.11 Opening The Audit File

4.11.1 Description and Priority:
This feature allows authorized users to open and review the audit file to ensure the accuracy of the election results. It is a critical feature with high priority.

4.11.2 Stimulus/Response Sequences:
Stimulus:
The user opens the application used to view the audit file, and selects the file they wish to view.
Response:
The application will then load and display the audit file on the user's screen. The user reviews the audit file and takes any necessary actions.

4.11.3 Functional Requirements:
REQ11-1: The system must have an application or software that allows authorized users to view the audit file.
REQ11-2: The system must provide a list of available audit files for the user to select from.
REQ11-3: The system must be able to load and display the selected audit file.
REQ11-4: The system must allow the user to review the audit file and identify any discrepancies in the election results.
REQ11-5: The system must provide the user with the ability to take appropriate actions to resolve any issues or errors found in the audit file.
REQ11-6: The system must allow the user to close the program once the audit file has been reviewed and any necessary actions have been taken.
REQ11-7: The system should include a feature that allows the user to review the audit file in sections or narrow down the information displayed if the file is too large or complex.

4.12 Running a Closed Party Election

4.12.1 Description and Priority:
This feature allows election officials to run a 4.12 closed party election using the CPL voting method. It is a critical feature with high priority.

4.12.2 Stimulus/Response Sequences:
Stimulus:
The election official inputs the ballot information into the system.
Response:

The system calculates and allocates the seats for each party based on their percentage of votes.

4.12.3 Functional Requirements:

REQ12-1: The system must support the CPL voting method for a closed-party election.
REQ12-2: The system must allocate the number of seats for each party based on their percentage of votes in relation to the total votes cast.
REQ12-3: The system must distribute the seats to each party in ranked order.
REQ12-4: The system must display the seat allocation results to the election officials and candidates.
REQ12-5: The system must provide a tiebreaker mechanism if a tie occurs during seat allocation.
REQ12-6: The system must only allocate seats to authorized and registered parties.
REQ12-7: The system must have a way to verify and validate the inputted ballot information to ensure the accuracy of the results.
REQ12-9: The system should have a way to handle and report any system errors or technical issues that may occur during the seat allocation process.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

In order to run the program, it is imperative to have an OS/system that supports and can run Java programs. Depending on the size of the file inputted, the program would take longer to compute the election. We anticipate that it shouldn't take more than a few minutes to get your election results from an inputted file.

5.2 Safety Requirements

The biggest concern regarding safety would be the safety of the inputted file in order to make sure that it can not be tampered with, and is in a format suitable to be read for the election. It is required to be a read-only CSV file.

5.3 Security Requirements -

The program will assure the integrity and security of the election as the file input will be identified and read seamlessly. Votes will then be accurately counted and allocated to the designated candidate or party, given the type of voting being simulated. In order to prevent tampering, there is a clause that the file is to only be read and not edited. Once received the file with the votes is read and cannot be edited or changed in any way. Furthermore, the file being imputed must be provided in the correct format and name in order for it to be processed by the system.

5.4 Software Quality Attributes

Regarding Software Quality Attributes, some we considered are:

- Availability: this program will be available to be used every election cycle throughout the year making it available 100% of the time when it is necessary
- Correctness: The ballots will be seamlessly read by a machine meaning there is practically no margin for error in contrast to if it was being counted by human hands where mistakes are prominent. Furthermore, the ballots are all cast electronically and stored in a CSV file format to be read by the program. So every ballot would be recorded accurately rather than having to interpret a handwritten ballot.
- Interoperability: The program will be easy to use, simply prompting the user for file input and the file name. This is straightforward and the only prompt the user needs to take action on; afterward, the file is identified and read and the votes are counted and allotted with the results and audit file produced.
- Portability: The program won't be too large in size and can run on any machine that supports Java. The only other thing it would need is the CSV file to be read.
- Reliability: The program has no downtime being a standalone software with no server to depend on. It is 100% reliable.
- Reusability: Once the file is read and the winner is displayed and the audit file is produced; the program regresses to its factory settings. It can instantly forget the past input and prompts the user for a new input file for another election.
- Robustness: For security reasons, the program is quite inflexible. It will only run if the file imputed is in its proper format, otherwise, it will give an error.

5.5 Business Rules

The system is unbiased towards any outside party and fully automated in order to maintain the integrity of the election. The file is imputed and the votes are counted and results are displayed on the screen at the end reflecting what was read. In this, two types of election voting are supported: Instant runoff voting (IRV) and Party-list voting using Closed Party Listing (CPL). For users, a screen with the polling results is displayed at the end of the terminal. Furthermore, if there is a tie in the voting, a flipping coin algorithm executes in order to select the winner.

For IRV, voters rank all candidates based on their preferences. The system computes this information and selects the candidate with the majority first preference votes as the winner. If no majority is present the candidate with the fewest votes is eliminated with their ballots going to the voters' second choice. This algorithm is executed until a candidate with the majority first-place votes is found.

Regarding CPL, there are parties on the ballot each with an ordered list of candidates. The voters vote for a party and the percentage of votes for a party would determine how many seats they've won for their candidates.

For both types of voting, there is a mutual way of deciding a tie-breaker via a fair coin toss. In the event of a tie in the IRV or CPL, the system would execute an algorithm simulating a fair random coin toss to decide the winner.

Glossary:

- Algorithm: a process or set of rules to be followed in calculations or other problem-solving operations.
- Terminal: A display that allows for execution of a program
- SSH (secure shell): A secure connection typically used to access remote connections (such as a computer/server)
- CSV (Comma Separated Files): A type of text file where the data is separated line by line via a comma
- Git: A set of tools that allows code to be transferred locally to an external repository
- GitHub: The landing platform for our code and acts as the above mentioned external repository
- Environment: An area that withholds all tools needed to run code such compilers and other associated developmental tools
- Compiler: A translator that changes programming language code (such as Java) into the lowest form (machine code) which allows the computer to understand and output what has been programmed
- Visual Studio: A tool used to code and program projects, can code in several different programming languages such as Java and allows for extensions to be installed in order to run the given language
- IR: a ranked preferential voting method. It uses a majority voting rule in single-winner elections where there are more than two candidates.
- CPL (closed p): A voting system in which voters can effectively only vote for political parties as a whole.