Software Requirements Specification

for

Vote Counting System

Version 0.6 approved

Prepared by Arpita Dev, Neha Bhatia, Praful Das, and Stuti Arora

CSCI 5801, University of Minnesota, Twin Cities

October 2023

Table of Contents

Revision History

Name	**Date**	**Reason For Changes**	**Version**
Vote Counting System	9/20	Given project details, understanding and documenting purpose	0.0
Vote Counting System	9/23	Additions to section 2, 3, and 5	0.1
Vote Counting System	9/27	Revised use cases after in-class elicitation	0.2
Vote Counting System	9/28	Specific System Requirements + cleaned up formatting	0.3
Vote Counting System	10/5	Cleaning up formatting and moving Use Cases to new document	0.4
Vote Counting System	10/6	Adding IRV and OPL algorithms in Appendix	0.5
Vote Counting System	10/6	Editing name of Use Case file	0.6

1. Introduction

1.1.Purpose

This document specifies the software requirements for a new standalone vote counting system. The system will be able to calculate election results based on two distinct voting algorithms: Instant Runoff Voting (IRV) and Party List Voting using Open Party List (OPL). The document serves as a comprehensive guide for the system's development, outlining essential features, functionalities, and constraints.

1.2.Document Conventions

The formatting and organization of this document follows the IEEE Template for System Requirement Specifications. Any code in this document is written in the Courier New font.

1.3. Intended Audience and Reading Suggestions

The intended audience for this document includes various stakeholders involved in the development and deployment of the voting system:

- **Software Developers:** This document serves as a guide for implementing the voting system and its associated features.
- **Testers:** The document can help testers understand the system's requirements and functionalities to ensure thorough testing and validation.
- **Election Officials:** Election officials, who will use the system, can gain an understanding of the system's capabilities and limitations.

To maximize the effectiveness of this document, readers are encouraged to start with the overview sections (Sections 1 and 2) to gain a high-level understanding of the project. Readers can then go to sections that are most pertinent to their roles and responsibilities to find relevant information.

1.4. Product Scope

The software will be a versatile voting system designed to handle two primary voting methods: Instant Runoff Voting (IRV) and Party List Voting using Open Party List (OPL). Its primary purpose will be to ensure efficient and accurate elections by processing electronic ballots, determining winners, and generating comprehensive audit reports. By providing a user-friendly and reliable platform for tallying votes, the software aims to uphold the integrity of elections and be a valuable tool for election officials.

1.5. References

IEEE Template for System Requirement Specifications taken from the CSCI 5801 Canvas page.
The voting algorithms used for Instant Runoff Voting and Open Party List voting have been referenced from FairVote.org.

2. Overall Description

2.1. Product Perspective

This software will be a standalone application designed to assist election officials by calculating and reporting election results for two distinct algorithms: IR Voting and OPL Voting. It will operate as a new self-contained product, not as a replacement of any existing product, and it will play no role in the vote collection process. It has a simple command line user interface for users to input an election file.

The project is a part of the CSCI 5801: Software Engineering 1 class at the University of Minnesota.

2.2.Product Functions

- **Input Election Data CSV File:** Election officials can input an election data CSV file, ensuring it is in the correct directory and the right format
- **Prompt for Correct Input Filename :** If the input filename is incorrect, the system will prompt the user to enter a valid filename.
- **Read Input File Header:** System will read the input file header to extract essential election information.
- **Run Instant Runoff (IR) Election:** Election officials can process an IR election ballot, where the most popular candidate wins if there is no clear majority.
- **Run Open Party List (OPL) Election:** Election officials can process an OPL election ballot, treating independents as grouped into one party.
- **Select Voting Algorithm:** Based on the header information, the system will automatically select the appropriate voting algorithm (IR or OPL).
- **Read Remaining Ballot Data:** The system will read the remaining file to extract and process the ballot data.
- **Shuffle Ballots:** When parsing OPL election data, the system must shuffle the ballots randomly to avoid bias in the election.

- **Break Ties by Coin Flip:** In case of ties, the system will automatically break ties by flipping a fair coin.
- **Generate Audit File:** The system will create an audit file containing election information, candidate details, ballot statistics, and election progress.
- **Audit for IR Redistribution:** If IR Voting Algorithm is used, the audit file will display the order of candidate removal and redistributed ballots.
- **View Election Information:** Users can view winners, election type, number of seats, ballots cast, and candidate vote counts on the user interface.
- **Read-Only Audit and Results:** Audit file is set to read-only to prevent unintended modifications.
- **Test Voting System:** Users must be able to test the correctness of the voting algorithm and its individual components like parsing the header correctly, running the correct voting algorithm, having a fair coin toss, shuffling ballots fairly and more.

2.3. User Classes and Characteristics

- **Election Officials** will be the most frequent users of the product. They will use the program during election periods to process election data and determine winners for OPl and IR elections. They will have access to the audit file that can allow them to recreate the vote counting process for each election type from start to finish. They are considered experts in the election processes, but may not have technical expertise to use the program. Detailed instructions will be provided and help commands will be offered within the program.

- **Developers**: Developers of the program will use it for software creation. They possess technical expertise and knowledge of software development. They will reference rule requirements for the election process to ensure accuracy of voter counting.

- **Media Personnel and General Public**: They will access the election results generated by the program after an election cycle. They don't have expertise in the election and software development process. They will need easy access to clear, unchangeable, election result data.

2.4. Operating Environment

The software will run on the most up-to-date version of Linux on University of Minnesota's CSE Lab Machines. The lab machines must have the Java Runtime Environment and Java Development Kit installed.

2.5. Design and Implementation Constraints

The voting system software will be implemented in Java and run on up-to-date Linux on the CSE lab machines. The software should process election data files that are in a predefined comma delimited text format. The program cannot modify the input file structure. The software will not use an external database and will only record election results in an audit file. The software also has a timing constraint and must process 100,000 ballots in 8 minutes.

2.6. User Documentation

Upon starting the program, the user will be provided with guidelines on basic commands to navigate through the program. These can be referred to as many times as needed while the program is running. Guidance to initially run the program can be found in Section 3 and the accompanying README.md.

2.7. Assumptions and Dependencies

The voting system software assumes that the all election data files provided by the user will always be in the correct format and free from formatting errors. The system also assumes that there will be one election record per file. The user can only input one file at a time. Any inconsistencies in the input file could impact the software's ability to process the voting results accurately. We also assume that the input file will be in the same directory as the program.

For IR Voting, we expect the input file to look like the following:

```
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 if instant runoff
2nd Line: Number of Candidates
3rd Line: The candidates and their party separated by commas
4th Line: Number of ballots in the file

For OPL Voting, we expect the input file to look like the following:

```
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,,,,
```

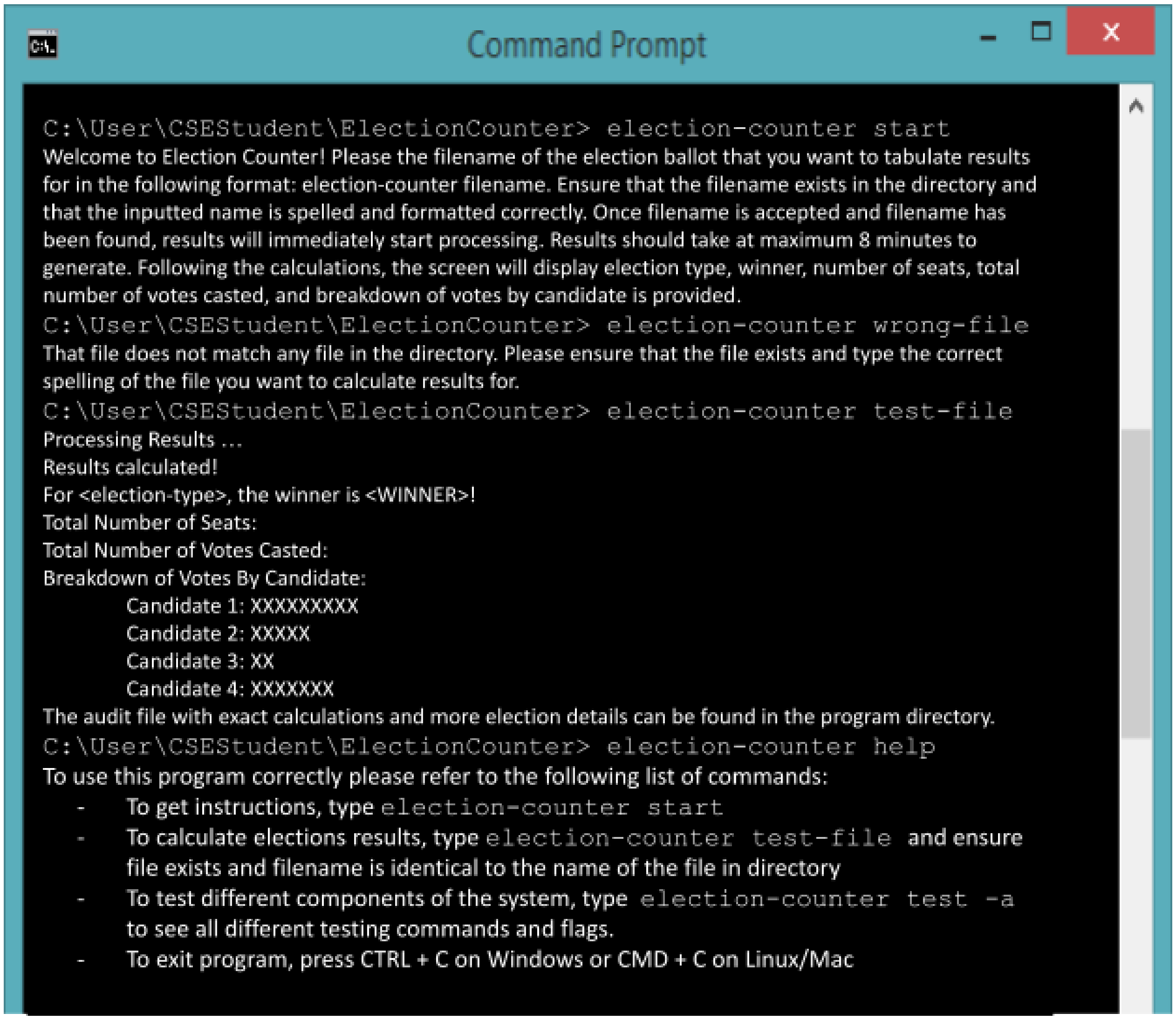
1st Line: OPL for open party listing
2nd Line: Number of Candidates
3rd Line: The candidates and their party separated by commas
4th Line: Number of Seats
5th Line: Number of Ballots

Another assumption is that the software will only run on a Linux based CSE Lab Machine with Java Runtime Environment installed. Running it on other operating systems can affect the software's performance and compatibility.

3. External Interface Requirements

3.1. User Interfaces

Overall the user interface will be a CLI (command line interface) with limited interactions from the users of the product. Namely, election official users will simply fill out the filename for the ballot of the election they want to determine and be subsequently prompted for incorrect or non-existent files in the directory. Following the calculation, the results screen will display election type, winner, number of seats, total number of votes casted, and voting breakdown by candidate.



```
C:\User\CSEStudent\ElectionCounter> election-counter start
Welcome to Election Counter! Please the filename of the election ballot that you want to tabulate results
for in the following format: election-counter filename. Ensure that the filename exists in the directory and
that the inputted name is spelled and formatted correctly. Once filename is accepted and filename has
been found, results will immediately start processing. Results should take at maximum 8 minutes to
generate. Following the calculations, the screen will display election type, winner, number of seats, total
number of votes casted, and breakdown of votes by candidate is provided.
C:\User\CSEStudent\ElectionCounter> election-counter wrong-file
That file does not match any file in the directory. Please ensure that the file exists and type the correct
spelling of the file you want to calculate results for.
C:\User\CSEStudent\ElectionCounter> election-counter test-file
Processing Results ...
Results calculated!
For <election-type>, the winner is <WINNER>!
Total Number of Seats:
Total Number of Votes Casted:
Breakdown of Votes By Candidate:
        Candidate 1: XXXXXXXXX
        Candidate 2: XXXXX
        Candidate 3: XX
        Candidate 4: XXXXXXX
The audit file with exact calculations and more election details can be found in the program directory.
C:\User\CSEStudent\ElectionCounter> election-counter help
To use this program correctly please refer to the following list of commands:
   -    To get instructions, type election-counter start
   -    To calculate elections results, type election-counter test-file and ensure
        file exists and filename is identical to the name of the file in directory
   -    To test different components of the system, type election-counter test -a
        to see all different testing commands and flags.
   -    To exit program, press CTRL + C on Windows or CMD + C on Linux/Mac
```

Figure 1. Sample command line interface with possible user commands and program's anticipated responses.

3.2. Hardware Interfaces

Not applicable as the software operates independently without direct hardware interactions.

3.3. Software Interfaces

Not applicable as the software does not integrate with other specific software components, databases, or external tools.

3.4. Communications Interfaces

Not applicable as the software does not require external communication functions, such as e-mail, web browsers, or network servers, and does not involve message formatting or data transfer rates.

4. System Features

System Use Cases can be found in the UseCases_Team9.pdf located in the same directory as this file.

4.1. User Instructions on Program Usage

4.1.1. Description and Priority

This feature is of high priority. It involves providing clear instructions to users on how to use the program and interpret the results once the program has been opened.

4.1.2. Stimulus/Response Sequences

Stimulus: User opens the program.

Response: The system displays a set of user instructions and guidance on program usage and result interpretation.

4.1.3. Functional Requirements

REQ-1: Upon opening the program, the system should present users with a user-friendly and informative set of instructions on how to use the program effectively.

REQ-2: The instructions should include clear guidance on the steps to input election data, select a voting algorithm, run the election, view results, and access the audit file.

REQ-3: Additionally, the instructions should explain how to interpret the election results, understand the audit file, and perform any necessary actions within the program.

4.2. Interface Shows File Upload Status

4.2.1. Description and Priority

This feature is of medium priority. It involves displaying a message on the interface to inform users when a file has been successfully uploaded, with a statement like "File found, processing results.”

4.2.2. Stimulus/Response Sequences

Stimulus: User uploads a CSV file containing election data.

Response: The system displays a message on the interface indicating that the file has been found and that the processing of results is underway.

4.2.3. Functional Requirements

REQ-4: When a user uploads an election data CSV file, the system shall check the file's presence and validity.

REQ-5: If the uploaded file is found and valid, the system shall display a message on the interface with the text "File found, processing results" to inform the user.

REQ-6: If the uploaded file is not found or is invalid, the system shall display an appropriate error message to guide the user on next steps.

4.3. Display FAQ and Debugging Instructions on "Program Help"

4.3.1. Description and Priority

This feature is of medium priority. It allows users to access a Frequently Asked Questions (FAQ) section and debugging instructions when they type "program help."

4.3.2. Stimulus/Response Sequences

Stimulus: User enters the command "election-counter help" in the CLI.

Response: The system displays an FAQ section and debugging instructions along with the program's usage instructions.

4.3.3. Functional Requirements

REQ-7: When the user enters the command "election-counter help," the system shall respond by displaying a comprehensive FAQ section addressing common user queries and issues. Users shall have the option to access this FAQ and debugging information at any time.

REQ-8: The FAQ section shall cover topics such as troubleshooting, error handling, and common scenarios that users might encounter.

REQ-9: In addition to the FAQ, the system shall provide debugging instructions, including steps to diagnose and resolve issues within the program.

4.4. Display Tabulated Election Results and Audit File

4.4.1. Description and Priority

This is a high priority feature. Once the election processing is complete, the program will generate a tabulated version of vote counts per candidate, type of election and display the winner to the user. It will also let the user know the name of the audit file and which directory to find it. The audit file contains granular information about exact steps taken to process the ballots, count the votes and assign the winner.

4.4.2. Stimulus/Response Sequences

Stimulus: A successful file upload from the user and status that the file is being processed.

Response: The system displays the election type, winning candidate and tabulated results for vote counts per candidate. The system tells the user the name of the audit file and where to find it.

4.4.3. Functional Requirements

REQ-11: After the system displays a successful file upload and lets the user know that the file is processed, within a maximum of 8 minutes the system should generate a successful election output to the user on the CLI.

REQ-12: This output must clearly show the election type, candidates and votes received, the winner and the name and location of the audit file.

REQ-13: The audit file must contain the election type, number of ballots, candidate names and all the steps taken by the system to allot votes to each candidate to determine the winner. Once generated, this file must be read only.

4.5. Display and Run Testing Scripts

4.5.1. Description and Priority

This is a high-medium priority feature. Using the command line interface, the user can access the testing system for the program. The system provides instructions on what kind of tests can be performed and how a user can run them.

4.5.2. Stimulus/Response Sequences

Stimulus: User types in "election-counter test -a" in the CLI.

Response: The system displays all the possible tests that can be performed and instructions on how to run them.

4.5.3. Functional Requirements

REQ-14: The system must provide clear instructions to run specific tests. If instruction is incorrect, the system must re-prompt the user to type in a command.

REQ-16: The system allows testing for individual voting algorithm correctness, if the file header was recognized correctly, if ballots for OPL election were shuffled appropriately, fairness of the coin flip in tiebreakers and other components of the voting system.

5. Other Nonfunctional Requirements

5.1. Performance Requirements

Application should be able to handle 100,000 ballots within 8 minutes and run on the latest version of the Linux based CSE lab machines.

5.2. Safety Requirements

Calculation processes for both IR and OPL must be according to algorithm standards to prevent election fraud and possible repercussions from the spread of incorrect election result information. In the case of a tie, tie break methods need to be fairly executed, without bias. Once results are generated, the created audit file must be read-only so the results and algorithm steps can be shared but not tampered with.

5.3. Security Requirements

The ballot file must be read-only to prevent any external or internal modifications on sensitive election information and votes. This helps maintain the integrity of the election process and the vote counting system.

5.4. Software Quality Attributes

The application must be:

- **Reliable:** The election results should be accurate and fair.
- **Reusable:** The product will initially be designed to handle IR and OPL voting but future versions might need to be able to handle and integrate different kinds of voting mechanisms.
- **User-Friendly:** Users should be able to easily input a file, interpret results, and find the respective audit file.

For future work, the product must be able to eventually integrate into a web application for online voting as an all inclusive system.

5.5. Business Rules

System is to be used by election officials, developers, and testers. All parties can input files but files must be in the same directory as the program and in correct format. Audit files can be viewed by anyone and shared publicly but will not have edit access to anyone.

6. Other Requirements

All necessary requirements have been covered in previous sections.

Appendix A: Glossary

- **Audit File/Report:** A comprehensive read-only document generated by a voting system or election software that provides a detailed account of an electoral process, including vote counting procedures, candidate results, ballot statistics, and any relevant information pertaining to the election.
- **Bias-Free Coin Flip**: A randomization process employed in elections or decision-making scenarios to break ties or make impartial choices. A bias-free coin flip ensures that each possible outcome (e.g., "heads" or "tails") has an equal probability of occurring, eliminating any favoritism or unfairness in the selection process.
- **CLI:** Abbreviation for Command Line Interface; a text-based user interface that allows users to interact with a computer program or operating system by typing commands into a terminal or command prompt. In a CLI, users provide instructions to the software by entering text-based commands, which are then interpreted and executed by the system.
- **CSV File**: CSV stands for "Comma-Separated Values." It is a widely used plain text file format used to store and exchange structured data. In a CSV file, data is organized into rows and columns, with each line representing a record and each field separated by a comma or other specified delimiter, such as a semicolon or tab. CSV files are commonly employed for data

import/export, database management, and spreadsheet applications due to their simplicity and ease of readability by both humans and software programs.

- **IEEE**: Abbreviation for the Institute of Electrical and Electronics Engineers; a globally recognized professional organization dedicated to advancing technology and innovation in the fields of electrical engineering, electronics, computer science, and related disciplines.
- **IR Voting:** Abbreviation for Instant Runoff Voting; a voting method used in elections to select a single winner from a list of candidates. In IR Voting, voters rank candidates in order of preference on their ballots. The counting process involves multiple rounds, where the candidate with the fewest first-preference votes is eliminated in each round. The votes of the eliminated candidate's supporters are then redistributed to their next preferred candidate. This iterative process continues until one candidate accumulates a majority of the votes, making them the winner. IR Voting is designed to ensure that the ultimate winner has the broadest support among the voters, even if they were not the first choice for all voters.
- **OPL Voting**: Abbreviation for Open Party List Voting; a voting system used in elections where voters cast their ballots for political parties rather than individual candidates. In OPL Voting, political parties present lists of candidates, and voters choose a party rather than selecting individual candidates. Seats in the legislative body are then allocated to parties based on the proportion of the total vote each party receives. This voting method is designed to represent the preferences of voters for political parties rather than individual candidates and is often used in proportional representation systems to ensure fair representation of various political groups.

Appendix B: Analysis Models

There are no additional analysis models.

Appendix C: To Be Determined List

There are no additional references to be determined.

Appendix D: Instant Runoff Voting Algorithm

The following segment is taken from FairVote.org to describe how the algorithm must work:
“Ranked choice voting (RCV) — also known as instant runoff voting (IRV) — makes our elections better by allowing voters to rank candidates in order of preference.

RCV is straightforward: Voters have the option to rank candidates in order of preference: first, second, third and so forth. Ballots that do not help voters’ top choices win count for their next choice.

It works in all types of elections and supports more representative outcomes. RCV means better choices, better campaigns, and better representation. That’s why it’s the fastest-growing nonpartisan voting reform in the nation.” (FairVote.org)

The following algorithm is built using the information provided by the Requirements Document for Waterfall Voting in CSCI 5801 and FairVote.org:

1. Ballots are appropriately shuffled and all the number one preferences of the voters are counted.
2. If a candidate receives over 50% of the first choice votes, they are declared elected. If no candidate receives a majority, then the candidate with the fewest votes is eliminated.
3. The ballots of supporters of this eliminated candidate are transferred to whichever of the remaining candidates they marked as their number two choice.
4. Votes are recounted to see if any candidate now receives a majority of the vote.
5. The process of eliminating the lowest candidate and transferring their votes continues until one candidate receives a majority of the continuing votes and wins the election.
6. If there is not a clear majority in instant runoff (IR), then popularity wins after all votes have been handed out.
7. If there is a tie between candidates, it must be broken with a fair coin flip.

Appendix E: Open Party List Voting Algorithm

The following excerpt is taken from the Waterfall Voting document provided on Canvas:

“Legislators are elected in large, multi-member districts. Each party puts up a list or slate of candidates equal to the number of seats in the district. Independent candidates may also run, and they are listed separately on the ballot as if they were their own party (see below). On the ballot, voters indicate their preference for a particular party and the parties then receive seats in proportion to their share of the vote. So in a five-member district, if the Democrats win 40% of the vote, they would win two of the five seats. The two winning Democratic candidates would be chosen according to their position on the list.

There are two broad types of list systems: closed list and open list. In a closed list system--the original form of party list voting--the party fixes the order in which the candidates are listed and elected, and the voter simply casts a vote for the party as a whole. This is shown in the first ballot below, which illustrates an election for the House of Representatives in a five-seat district. Voters are not able to indicate their preference for any candidates on the list, but must accept the list in the order presented by the party. Winning candidates are selected in the exact order they appear on the original list. So in the example here, if the Democrats won two seats, the first two candidates on the pre-ordered list--Foster and Rosen-Amy--would be elected.” (Project 1 Waterfall Voting SRS).

The following algorithm is built using the information provided by the Requirements Document for Waterfall Voting in CSCI 5801 and FairVote.org. Seat allocation is done through the Largest Remainder Approach.

1. The ballots are appropriately shuffled and total votes are tallied for each candidate as well as their representative party. For example, if candidate X belongs to the Orange party, a vote towards candidate X will count towards both them and the party. All independent candidates are grouped into one party.
2. The process of seat allocation through the Largest Remainder approach starts with calculating the quota, which is the total number of valid votes in the district and divided by the number of seats. For example, if there are 100,000 votes and 10 seats, the quota will be 10,000 seats.
3. For every whole number quota met by a party, they get allotted 1 seat. For example, if the Orange party has 38,000 votes and the quota is 10,000, they will get allotted 3 seats since 38,000/10,000 rounded down to the nearest whole number is 3.
4. Once the initial number of seats is allotted, the remaining seats are given out to parties with the largest remainder of votes until all seats are filled.
5. The candidates from each party are assigned seats in order of the number of votes they received within the party. For example, if Candidate X received 5,000 votes and Candidate Y received 2,000 votes and their party has 1 allotted seat, the seat will go to Candidate X.