TEAM 18

Voting System

Software Design Document

Name(s):
Roshina Mohamed Rafee
Linh Duong
Thomas Haupert
Henry Huynh

Date: 02/28/2021

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The purpose of the Software Design Document is to provide the design details of the Voting System. The expected audience is considered to be the University of Minnesota department of Computer Science, along with Dr. Watters and the teaching assistants, as well as future developers of the application.

1.2 Scope

This document contains a complete description of the design of a Voting System.

The Voting System is capable of performing two types of voting: Instant Runoff Voting (plurality/majority type) and party list voting using Open Party List Voting (proportional voting). The goal of this system is to count the number of votes based on chosen voting algorithms, produce an audit file with election information that replicates the election progress itself, and display the name(s) of the winner(s) as well as other pertinent information both onscreen and in a media report file.

1.3 Overview

Section 2 provides an Overview of the Voting System. This allows a clear explanation of the general functionality, context and design of the Voting System.

Section 3 provides the Architecture Structure of the Voting System. It includes a high level architectural design which illustrates how the subsystems collaborate with each other to perform all the functionalities of the system. The system will have a decomposition description that includes a sequence diagram. This section also provides the design rationale, which discusses the critical issues and trade/offs of the design.

Section 4 provides Data Design of the Voting System which explains the data structures, objects, attributes, and methods that are used to build the Voting System.

Section 5 provides Component Design that summarizes each object member function for all the objects.

Section 6 provides Human Interface Design which discusses how the Voting system interacts with the user and optimizes the user experience and ease of use.

Section 7 provides the Requirements Matrix in a tabular format, which discusses how system components are used to satisfy each functional requirement of the Voting system.

Section 8 provides Appendices which are included and used in this Software Design Document.

1.4 Reference Material

Douglas, J. A. (n.d.). Instant Runoff Voting. FairVote. Retrieved from https://www.fairvote.org/instant_runoff_voting_no_substitute_for_pr

FairVote. (n.d.). How Proportional Representation Elections Work. Retrieved from https://www.fairvote.org/how_proportional_representation_elections_work

1.5 Definitions and Acronyms

Instant runoff voting (IRV): A type of voting system in which voters can rank the candidates to ensure their vote is transferred in case their primary choices cannot win the election.

Open party list ballot (OPL): A type of voting system in which voters vote for a particular party and particular candidates and are elected proportionally.

Comma-separated values (CSV): A type of file that stores a table of data where each column is separated by a comma and rows are separated by new lines.

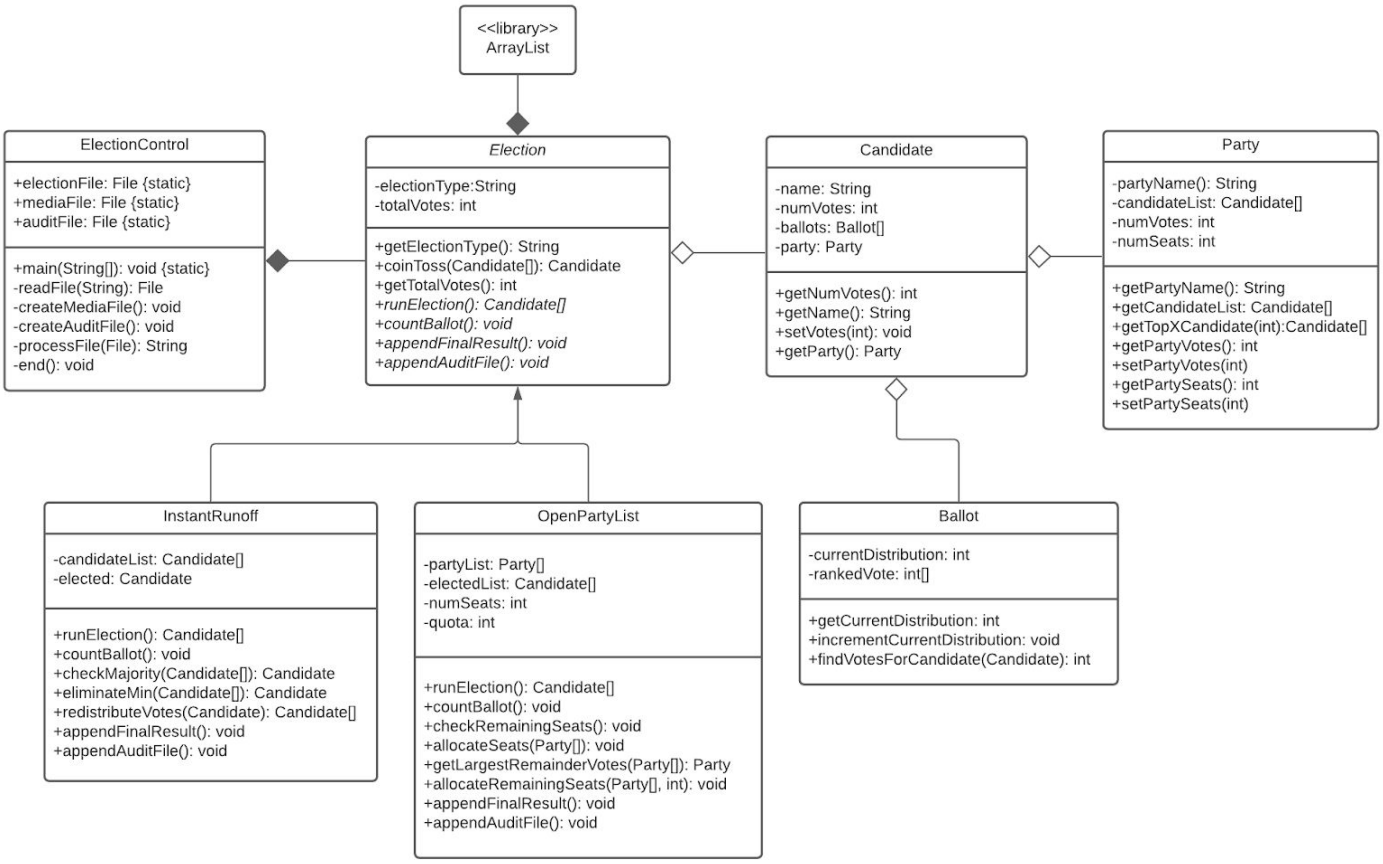
2. SYSTEM OVERVIEW

We will be creating a voting system that is capable of performing two voting methods: Instant Runoff voting and Open Party List voting. Our system will take a CSV file and process information provided and produce a result. Each of the methods will have different criterias that they follow to run the method to success. Once the respective method of voting choice is successfully done, there will be a media file and audit file produced declaring the winner with details such as their name, their party's name, and the number of votes they won. This program will run multiple times during normal election times and during special elections.

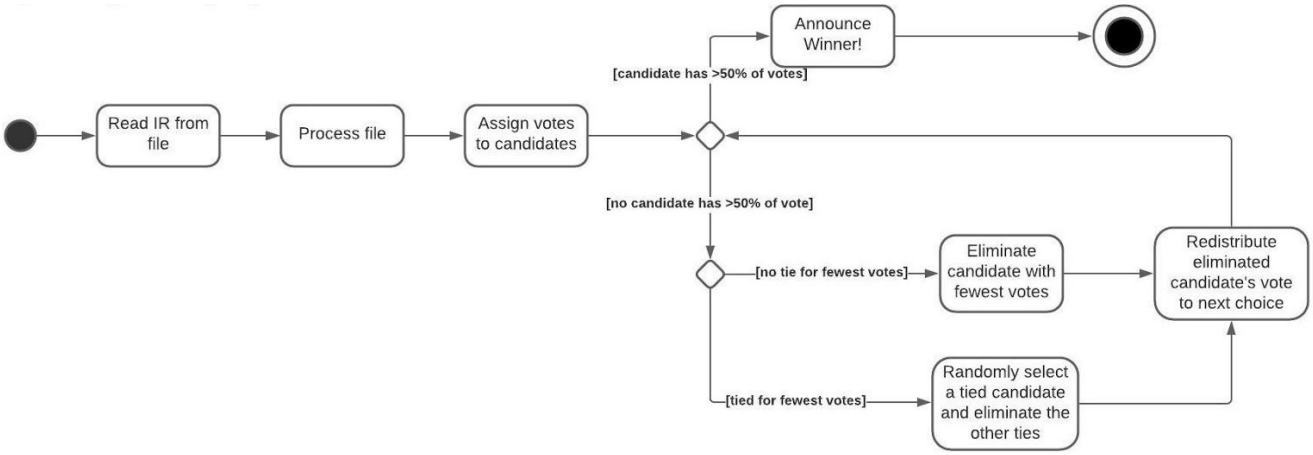
3. SYSTEM ARCHITECTURE

3.1 Architectural Design

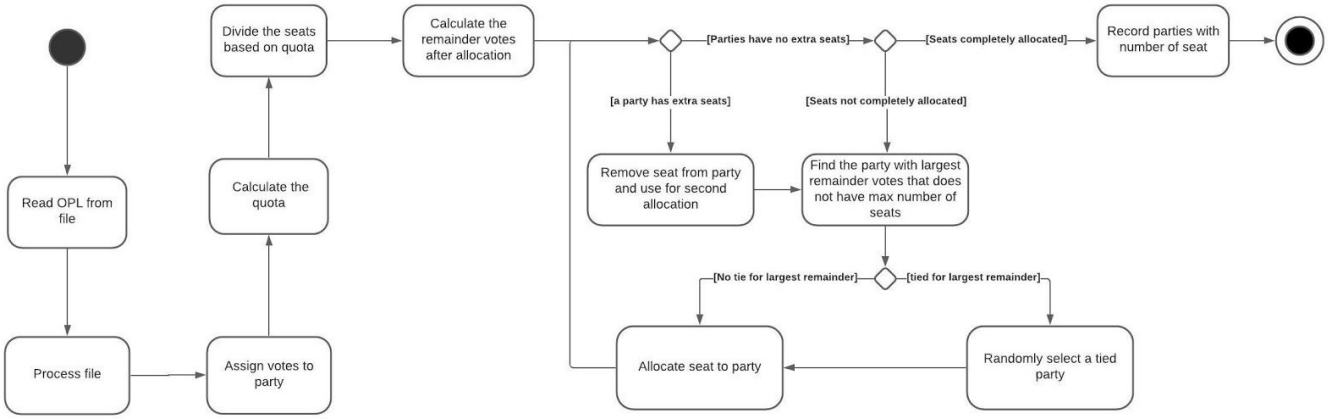
Class Diagram



Activity Diagram (IR)



Activity Diagram (OPL)

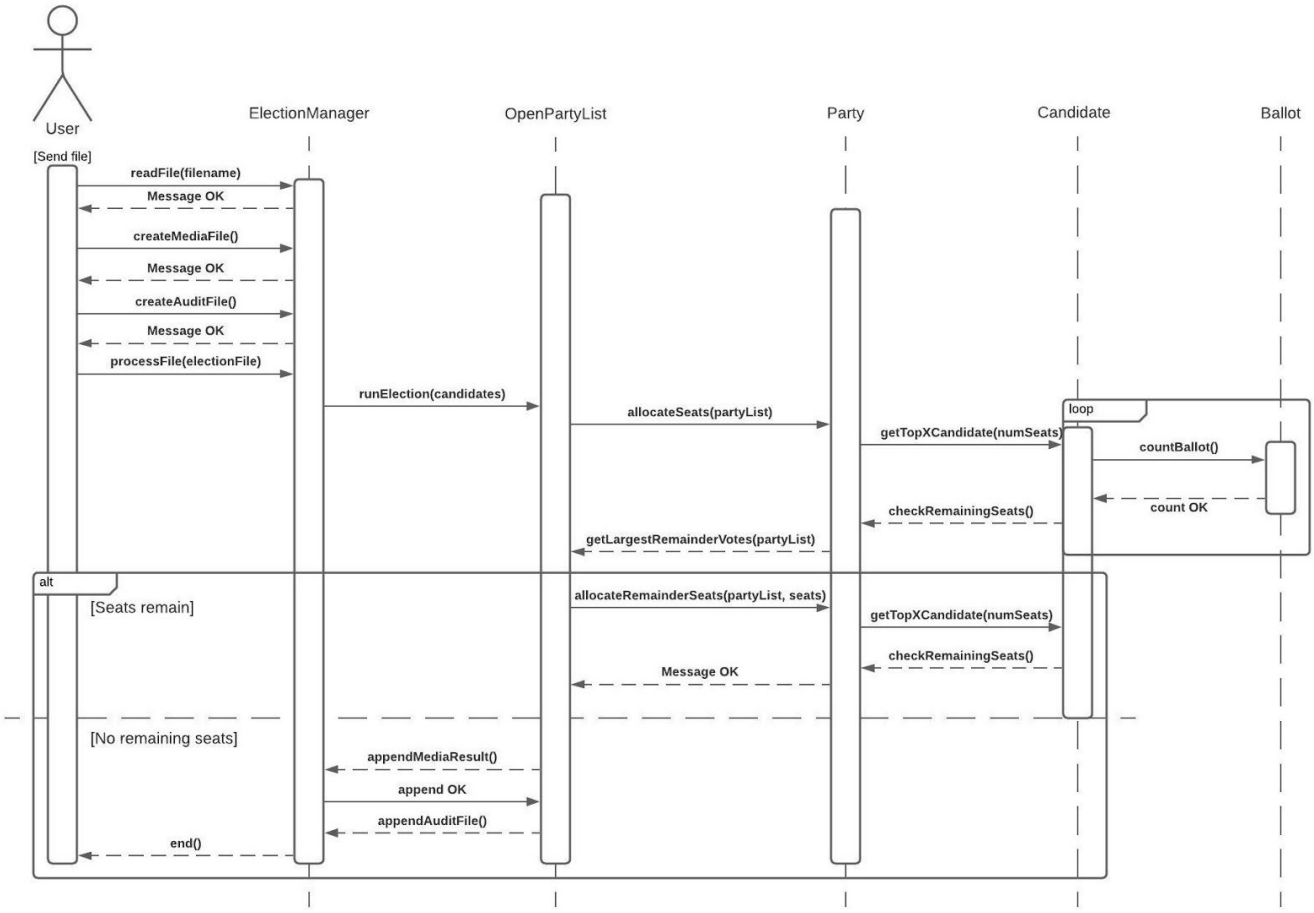


3.2 Decomposition Description

This design begins with ElectionControl, which is where the main method is and what will start off the election counting process. It will read and process the file to determine which election type it is. The Election class is abstract, with methods which are common to all election types and others that are unique to each election type. The two types of elections supported by the software are Instant Runoff and Open Party List, which inherit from the Election abstract class.

The Candidate class has the Party class associated with it, as each candidate belongs to a party. Independent candidates are to be grouped into one party for this software. The Candidate class also has the Ballot class associated with it, so that each candidate will have the ballots of those that voted for them.

Sequence Diagram



3.3 Design Rationale

The ballots are stored with each candidate instead of in one large array. This decision was made to optimize for faster computation times. In Instant Runoff, ballots are redistributed when a candidate has the least number of votes. This design allows the software to simply redistribute the ballots associated with the candidate, instead of iterating through every single candidate's votes which would be much slower. One tradeoff of this approach is that it is slightly harder to implement.

The Election class was decided to be abstract, rather than a concrete class or an interface. This is because there are methods that are identical between all election types, which means that an interface would not be appropriate in this situation. These methods could have been copied between each class that implements the interface, but would be much more time consuming than to simply create the method once and have every election type inherit it. Another reason is that there are method signatures that are common to all election types, but the actual implementation is different for each type. This approach allows for easier future development of the software, as those wishing to add additional election types will have a blueprint to work from. Abstract classes allow for both the sharing of methods in a concrete class and the flexibility of an interface, all with no real tradeoffs.

4. DATA DESIGN

4.1 Data Description

There are three different files involved with the software. All files will be in the current working directory of the software.

The first file is an election file that must be present for the software to work properly. This file should be preformatted to contain basic information of the election followed by each ballot. No specific name of the file is required. The file type must be a CSV file.

The second file is a media file that is output at the conclusion of the program. This file contains the basic results of the election, such as the winner. This file will be named "report.txt" and will overwrite the file if it already exists.

The third file is an audit file that is output at the conclusion of the program. This file contains the full calculations that the program performed to determine the election. This file will be named "audit_MM-DD-YYYY_HH.MM.SS.txt" where MM is the current month, DD is the current day, YYYY is the current year, HH is the current hour, MM is the current minute, and SS is the current second. This ensures that the file name will be unique and will not overwrite any file.

4.2 Data Dictionary

Attribute Name	Access Modifier	Type
Class: Ballot		
currentDistribution	private	Int
findVotesForCandidate()	public	Candidate -> Int
getCurrentDistribution()	public	Void -> Int
incrementCurrentDistribution()	public	Void -> Void
rankedVote	private	Int Array
Class: Candidate		
ballots	private	Ballot Array
getName()	public	Void -> String
getNumVotes()	public	Void -> Int
getParty()	public	Void -> Party
name	private	String
numVotes	private	Int
party	private	Party
setVotes()	public	Int -> Void
Class: Election		
appendAuditFile()	public	Void -> Void
appendFinalResult()	public	Void -> Void
coinToss()	public	Candidate Array -> Candidate
electionType	private	String
getElectionType()	public	String -> Void
getTotalVotes()	public	Void -> Int
runElection()	public	Void -> Array of Candidate
totalVotes	private	Int

Class: ElectionControl		
auditFile	public	Static File
createAuditFile()	private	Void -> Void
createMediaFile()	private	Void -> Void
electionFile	public	Static File
end()	private	Void -> Void
main()	public static	String Array -> Void
mediaFile	public	Static File
processFile()	private	Void -> String
readFile()	private	String -> File
Class: InstantRunOff		
appendAuditFile()	public	Void -> Void
appendFinalResult()	public	Void -> Void
candidateList	private	Candidate Array
checkMajority()	public	Candidate Array -> Candidate
elected	private	Candidate
eliminateMin()	public	Candidate Array -> Candidate
redistributeVotes()	public	Candidate -> Candidate Array
runElection()	public	Void -> Candidate Array
Class: OpenPartyList		
allocateRemaindingSeats()	public	Party Array, Int -> Void
allocateSeats()	public	Party Array -> Void
appendAuditFile()	public	Void -> Void
appendFinalResult()	public	Void -> Void
checkRemainingSeats()	public	Void -> Void
electedList	private	Candidate Array
getLargestRemainderVotes()	public	Party Array -> Party

numSeats	private	Int
partyList	private	Party Array
quota	private	Int
runElection()	public	Void -> Candidate Array
Class: Party		
candidateList	private	Candidate Array
getCandidateList()	public	Void -> Candidate Array
getPartyName()	public	Void -> String
getPartySeats()	public	Void -> Int
getPartyVotes()	public	Void -> Int
getTopXCandidate()	public	Int -> Candidate Array
numSeats	private	Int
numVotes	private	Int
partyName	private	String
setPartySeats()	public	Int -> Void
setPartyVotes()	public	Int -> Void

5. COMPONENT DESIGN

ElectionControl

readFile(String filename):
return Open_file(filename) as read-only;
Returns a read-only file pointer for the file that is named.

createMediaFile():
create file "report.txt";
return;
Creates a file named "report.txt".

createAuditFile():
get date and time;
create file "audit_MM-DD-YYYY_HH.MM.SS.txt";
return;
Creates a file named "audit_MM-DD-YYYY_HH.MM.SS.txt" where MM is the current month, DD is the current day, YYYY is the current year, HH is the current hour, MM is the current minute, and SS is the current second.

processFile(File file):
if first line of file = "IR":
return "IR";
else if first line of file = "OPL":
return "OPL";
else return "None";
Returns the first line of the file passed in if it matches one of the election types.

Election

getElectionType():
return electionType;
Returns the type of election.

coinToss(Candidate[] clist):
rand = random from 1...clist.length;
return clist[rand];
Randomly selects a candidate from the list of candidates.

getTotalVotes():
return totalVotes;
Returns the total number of ballots in the election.

Ballot

getCurrentDistribution():
return currentDistribution;
Returns the current distribution as an integer.

incrementCurrentDistribution():
currentDistribution++;
return
Increments the current distribution by one.

findVotesForCandidate(Candidate cand):
return cand.getVotes();
Returns the number of votes for the candidate.

Candidate

getNumVotes():
return numVotes;
Returns the number of votes for the candidate as an integer.

getName():
return name;
Returns the name of the candidate as a string.

setVotes(int v):
votes = v;
return;
Sets the number of votes.

getParty():
return party;
Returns the party of the candidate.

Party

getPartyName():
return name;
Returns the name of the party.

getCandidateList():
return candidateList;
Returns a Candidate array containing the candidate objects affiliated with the party.

getTopXCandidate(int x):
topX = Candidate[x];

```
  iterate through the candidates in candidateList:
    if topX has less than x members:
      append to topX;
    else:
      iterate through the topX:
        if greater votes than topX member:
          remove last member of topX and insert in the current spot;
  return topX;
```
Returns a Candidate array of the corresponding number of candidates passed in as an int.

```
getPartyVotes():
  return numVotes;
```
Return the number of votes that the party has.

```
setPartyVotes(int votes):
  numVotes = votes;
  return
```
Change the number of votes of the party.

```
getPartySeats():
  return numSeats;
```
Return the number of seats that the party has as an integer.

```
setPartySeats(int seat):
  numSeats = seat;
  return
```
Change the number of seats that the party can get.

InstantRunoff

```
runElection(Candidate[] clist):
  majority = floor(total number of votes / 2) + 1;
  while checkMajority = null:
    loser = eliminateMin(clist);
    clist = redistributeVotes(loser);
  return [checkMajority];
```
Runs the Instant Runoff election.

```
countBallot():
  read next ballot from electionFile;
  add ballot to candidate with a 1;
  increment total number of votes;
  return;
```
Counts the next ballot in the election file.

checkMajority(Candidate[] clist):
iterate through clist:
if a candidate has more than 50% of total:
return candidate;
else:
return null;
Returns the candidate from the candidate list with the majority of votes or null if no majority.

eliminateMin(Candidate[] clist):
iterate through clist:
minCandidate = candidate with fewest vote;
return minCandidate;
Returns the candidate from the candidate list with the fewest votes.

redistributeVotes(Candidate loser):
clist = current candidate list;
remove loser from clist;
move loser's ballots to next choice;
return clist
Returns a candidate list with the loser's votes redistributed to other candidates.

appendFinalResult():
start with empty string ""
append to string with titles
iterate through candidate list:
append to string candidate name, candidate party, candidate votes
append to string to go to next line
return
Prepare a string of text for printing the final result

appendAuditFile():
start with empty string ""
n = maximum number of ballots.
append to string with titles
iterate through candidate list;
append to string candidate name, candidate party, num votes from 1st ballot...n-ballots
if candidate current ballot is null, append "--------" instead
append to string to go to next line
return
Prepare a string of text for writing to the audit file

OpenPartyList

```
runElection():
   calculate quota;
   allocateSeats(PartyList);
   checkRemainingSeats(PartyList);
   while (availableSeats > 0):
      find party with largest remainder votes that does not have max number of seats
      assign seat to the chosen party;
      decrement the available seats;
   return
```

Runs the Open Party List election.

```
countBallot():
   read next ballot from electionFile;
   add ballot to candidate with lowest number ranking;
   increment total number of votes;
   return;
```

```
checkRemainingSeats():
   if (availableSeats = 0):
      return
   else:
      iterate through PartyList;
      if more than maximum seats is assigned to a party:
         increment availableSeats;
         decrement current partySeats;
   return;
```

Checks how many free seats are available and will free up a seat if a party has more seats than candidates.

```
allocateSeats(Party[] partyList):
   iterate through partyList;
      seats = partyList[i].getPartyVotes() / quota;
      partyList[i].setSeats(seats);
      remainVotes = partyList[i].getPartyVotes() % quota;
      partyList[i].setVotes(remainVotes);
   return;
```

Allocates seats to each party based on quota.

```
getLargestRemainderVotes(Party[] partyList):
   initialize largestParty and maxVotes;
   iterate through partyList:
      if a party has more votes than maxVotes AND has available seats:
```

maxVotes = current party votes
largestParty = current party
return largestParty;

Finds the party with the highest number of remainder votes.

allocateRemainingSeats(Party[] partyList, int availableSeats):
if availableSeats = 0:
return
else:
chosenParty = getLargestRemainderVotes(partyList);
increment a seat to chosen party;
allocateRemainingSeats(partyList, availableSeats--);
return

Gives the party with the largest number of remainder votes an additional seat.

appendFinalResult():
start with empty string “”
append to string with titles
iterate through partyList:
append to string partyName, partyVotes, partySeats
append to string to go to next line
return

Prepare a string of text for printing the final result.

appendAuditFile():
start with empty string “”
append to string with titles
iterate through partyList;
append to string partyName, partyVotes, firstBallot, remainingVotes
append to string secondBallot, finalSeat, % of Vote to % of Seats
append to string to go to next line
return

Prepare a string of text for writing to the audit file.

6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

The system will be displayed in a text-only form, with no graphical user interface. Upon starting the software, instructions will be displayed for the user to read. The only action the user will need to perform is to type the name of a file when prompted. The system will automatically parse the file and perform the features that pertain to the file (such as whether to use Instant Runoff or Open Party List), which means the user does not need any knowledge of the program to utilize all its features. If the user enters in an incorrect file name, the software will provide feedback that the file cannot be found and will present another opportunity to enter the file name.

6.2 Screen Images

```
Voting System
Please enter election file name: _
```

```
Winner(s) of election: John Smith
Type of election: Instant Runoff
Number of ballots cast: 100

John Smith (Orange Party): 52
Jane Doe (Blue Party): 25
Joe Clark (Purple Party): 23

An audit file has been created: audit_02-01-2021_15.20.50.txt
A media report file has been created: report.txt
```

6.3 Screen Objects and Actions

Only text will be displayed in the program. When the user is able to enter the file name, a text prompt will appear to alert the user that they can now type. When the user is done typing, they may press the Enter key to submit the text to the program. When the program is finished running, it will automatically exit without requiring an action from the user. While

the program is running, the user will be able to press the Control-C button combination at any point to send an interrupt and forcibly end the program.

7. REQUIREMENTS MATRIX

Use Case	Description	How to Test	Data Structure
PRC_001	Determine the file name for the election file.	Insert various file names that do and do not exist when the software prompts for a file name. Then check that the correct file is sent to the read file.	N/A
PRC_002	Read in the election file	Load a file that exists and check if it displays an error or not while reading.	N/A
PRC_003	Process the file	Once the file goes through PRC_001 and PRC_002, and check if all the details have been processed for election purposes.	N/A
ELEC_001	Process ballots received in the processed file according to the rules of IR.	Check if all the rules of IR have been executed correctly when the winner has been announced.	Array List
ELEC_002	Process ballots received in the processed file according to the rules of OPL.	Ensure if each member of the group get seats distributed based on the amount of votes received.	Array List
ELEC_003	For the purpose of an Open Party List election, the independents are grouped into one party.	Check to see if all the individual participants are grouped into one party and seats are allocated accordingly.	2D Array
ELEC_004	Toss the coin fairly to randomly select a winner if there is a tie in the votes.	There will tests to ensure that candidates are being picked at random.	Array

OUT_001	Print results to screen.	Display of any output of results.	N/A
OUT_002	Produce audit file.	Follows format required for the audit file.	N/A
OUT_003	Produce media report files.	See if all required information is included.	N/A

8. APPENDICES

The appendices lists the use cases that were in the SRS document. Section 7: Requirements Matrix references these use cases and are provided below for convenience.

Determine File Name

Name	Determine File Name
ID	PRC_001
Description	Determine the file name of the election file.
Actor(s)	Election Official, System
Organizational Benefits	To gather all the data required to announce the winner of the election.
Frequency of Use	Multiple times per year at normal election times and special elections.
Trigger	When the election officials receive an election file after an election has been conducted and the software is run.
Precondition	The election file should be present in the working directory of the software.
Postcondition	The election filename will be stored for later use.
Main Course	1. The system will prompt the user for the file name. 2. User will enter in the file name (see EX1). 3. The system will save the file name that was entered.
Alternate Courses	No alternate courses
Exceptions	EX1 File name entered is not found 1. System prompts user for file name another time. 2. If the file is found, proceed to Main Course 3. 3. If the file is not found, return to EX1.

Read In File

Name	Read In File
ID	PRC_002
Description	Read in the election file.
Actor(s)	Election Official, System
Organizational Benefits	To gather all the data required to announce the winner of the election.
Frequency of Use	The file will be read every time the user enters the name of the file.
Trigger	The software will automatically run this after the “Determine File Name” use case.
Precondition	The file name must have been determined and successfully found.
Postcondition	The election file will be loaded by the program for reading.
Main Course	1. The software will load the file that was named earlier in read-only mode.
Alternate Courses	No alternate courses
Exceptions	No exceptions

Process File

Name	Process File
ID	PRC_003
Description	Process all the information listed in the file and store the information in the software.
Actor(s)	Election Official, System
Organizational Benefits	To process data within the file to help us determine the information that is needed to finalize the results
Frequency of Use	The file will be processed every time the user enters the name of the file and the file is read.
Trigger	The software will automatically run this after the “Read In File” use case.
Precondition	An election file must be loaded by the file for reading. The first line of the election file must read “IR” or “OPL”.
Postcondition	The election file will be loaded by the program for processing information.
Main Course	1. The software will read the first line of the election file to determine the type of election (see AC1, AC2).
Alternate Courses	AC1 If the first line of the election file is “IR”: 1. The software will proceed to run the “Run Instant Runoff” use case. AC2 If the first line of the election file is “OPL”: 1. The software will proceed to run the “Run Open Party List” use case.
Exceptions	No exceptions

Run Instant Runoff

Name	Run Instant Runoff
ID	ELEC_001
Description	For elections of the Instant Runoff type, this will process ballots received in the processed file according to the rules of an Instant Runoff type election.
Actor(s)	Election Official, System
Organizational Benefits	Automation of the Instant Runoff election process.
Frequency of Use	Every time there is an election of the type Instant Runoff.
Trigger	The election results file received is of the Instant Runoff type.
Precondition	The elections results file has been identified and processed, with the election type specified as Instant Runoff.
Postcondition	Two separate files are generated. One audit file which details the process of the automated Instant Runoff election, and one media report file which details the winner and final results of the election.
Main Course	1. The system checks to see if a candidate has received greater than half of the votes. 2. If it does, declares that candidate as the winner and generates the audit and media report files. If a candidate does not have the majority, the program continues to step 3. 3. The current state of the election is stored for the audit file. 4. The candidate with the fewest votes is eliminated and all of the votes of the eliminated candidate are given to the candidate whose name shows up next on their ranked ballot. 5. The system repeats steps 1-4 until a winner is declared. 6. Audit and media report files are produced. 7. The results are printed to screen.
Alternate Courses	AC1 In the case that there is a complete tie at some point in the election where there is no candidate with the fewest votes: 1. The system proceeds to the coin flip process.
Exceptions	EX1 In step 3, if a ballot does not have a next choice: 1. The ballot is tossed and the vote is not given to any candidate.

Run Open Party List

Name	Run Open Party List
ID	ELEC_002
Description	For elections of the Open Party List type, this will process ballots received in the processed file according to the rules of an Open Party List election.
Actor(s)	Election Official, System
Organizational Benefits	Automation of Open Party List election process.
Frequency of Use	Every time there is an election of an Open Party List type.
Trigger	The election results file received is of the Open Party List type.
Precondition	The elections results file has been identified and processed, with the election type specified as Open Party List.
Postcondition	Two separate files are generated. One audit file which details the process of the automated Open Party List election, and one media report file which details the winner and final results of the election.
Main Course	1. The system determines the quota by dividing the total number of votes by the number of seats available. 2. The system then tallies the total votes received by each party. 3. The total votes of each party are divided by the quota and a seat is given to that party for each whole number produced. If not all seats are given, see AC1. 4. For each seat given to a party, the candidates with the highest votes are chosen until all seats are filled. 5. Audit and media report files are produced. 6. The results are printed to screen.
Alternate Courses	AC1 In the case that not all seats are given after the initial counts: 1. The seats are awarded to the parties closest to the next whole number (highest remaining votes). If there is a tie for the majority, proceed to the coin flip process. 2. The system proceeds to Main Course 4.
Exceptions	No exceptions

Independent Candidates

Name	Independent Candidates
ID	ELEC_003
Description	For the purpose of an Open Party List election, the independents are grouped into one party.
Actor(s)	Election Official
Organizational Benefits	Due to the nature of how seats are given based on the number of total votes given to a party, the independents must be grouped together in order for them to obtain seats.
Frequency of Use	This will occur whenever there is an Open Party List election with independent candidates.
Trigger	There is an Open Party List election with independents running.
Precondition	All of the votes have been counted and party listings created.
Postcondition	The independents are grouped together to form the “Independents” party for the purpose of the Open Party List election process.
Main Course	1. All of the independent candidates are grouped into one party called the “Independent party”.
Alternate Courses	No alternate courses
Exceptions	EX1 In the case that there is only one independent candidate: 1. The independent party is still created, with that candidate as the sole member.

Coin Flip on Tie

Name	Coin Flip on Tie
ID	ELEC_004
Description	Toss the coin fairly to randomly select a winner if there is a tie in the votes.
Actor(s)	Election Official, System
Organizational Benefits	To keep the counting process fair without repeating the election again.
Frequency of Use	The coin flip is used every time there is a tie in regards to the election process.
Trigger	There is a tie vote cast between the candidates or between parties.
Precondition	The input file has been processed; the candidates have been assigned with the number of votes cast.
Postcondition	The name of the chosen winner will be stored for further processing.
Main Course	1. The system checks the candidates whose votes are tied then group these candidates together. 2. The system checks the number of candidates (winners) needed to be chosen from this group. If there is more than 1 candidate needed to be chosen, proceed to AC1. 3. The system then generates a random seed to choose one candidate out of this group. 4. The candidate chosen by the random seed is stored and returned as a winner within the group.
Alternate Courses	AC1 In the case that the system is required to choose more than 1 candidate in the group of candidates with tie votes: 1. The system generates a random seed and chooses one candidate out of this group. The chosen candidate is stored in another group for winners. 2. The system then removes the chosen candidate from the tie-vote group. 3. The system repeats steps 1 and 2 until the system satisfies the number of candidates needed to be chosen. 4. The system returns the winner group with the stored name of candidate(s).
Exceptions	No exceptions

Print Results to Screen

Name	Print Results to Screen
ID	OUT_001
Description	Prints the results of an election to standard output.
Actor(s)	System
Organizational Benefits	This process is useful when the results of an election need to be visible. This is also useful in the debugging process and to ensure there are no errors.
Frequency of Use	Everytime an Instant Runoff or Open Party List election process is run, the final results will be printed to screen.
Trigger	The final step in the Instant Runoff and Open Party List after producing media and audit files are to print the results to screen.
Precondition	The Instant Runoff or Open Party List process has finished.
Postcondition	The results of the election are printed out to standard output.
Main Course	1. The system determines the type of election. 2. If it was an Instant Runoff election, displays a list of candidates along with the votes at each count. 3. Otherwise, in the case of Open Party List elections, displays a list of the parties along with total votes, seats allocated initially, final seat allocation, and selected candidates from each party.
Alternate Courses	No alternate courses
Exceptions	EX1 If the election process was run but an error occurred: 1. System prints out “An error occurred in the election process.”

Produce Audit File

Name	Produce Audit File
ID	OUT_002
Description	After an election process concludes, an audit file detailing the steps taken by the system in generating the results is produced.
Actor(s)	System
Organizational Benefits	The audit file is a highly important file which allows the election process to be visible and thus validated. Should the results of an election be challenged, the audit file is a key piece of evidence.
Frequency of Use	At the end of every election process run by the system, an audit file will be produced.
Trigger	The produce audit file process is triggered at the end of every election process.
Precondition	An election process of either type concludes.
Postcondition	An audit file is produced which details the steps taken by the system in determining the winner of the election.
Main Course	1. The system determines which election type the audit file will be for. 2. During the election process, the initial, intermediary, and final states of the process were stored. The system compiles these states into a readable file that displays how the election process occurred according to the format of the election type.
Alternate Courses	No alternate courses
Exceptions	EX1 If the election process was run but an error occurred: 1. System prints out “An error occurred in the election process.”

Produce Media Report File

Name	Produce Media Report File
ID	OUT_003
Description	After an election process concludes, a media report file will be produced which details the final winner and results of the election.
Actor(s)	System
Organizational Benefits	The media report is a document ment to report the final results of an election process in a readable and concise manner.
Frequency of Use	At the end of every election process run by the system, a media report file will be produced.
Trigger	The produce media report process is triggered at the end of every election process.
Precondition	An election process of either type concludes.
Postcondition	A media report is produced which details the final results of the election.
Main Course	1. The system determines what type of election the media report is being produced for, to determine the format. 2. The election results are shown according to the format.
Alternate Courses	No alternate courses
Exceptions	EX1 If the election process was run but an error occurred: 1. System prints out “An error occurred in the election process.”