CSCI 5081 Team 5

Instant Runoff and Closed Party List Voting System

Software Design Document

Name(s):

Codey Camerer	camer441
Zoe Foster	foste924
Adelaide Granse	grans036
Hailey Koster	koste116

Date: 03/03/2023

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to describe, in detail, a project to determine election winners via two voting algorithms, Instant Runoff Voting and Closed Party List Voting. This software design document will describe the architecture and system design of these two algorithms and the larger system they are to be contained within. This document will describe each of the functions of the system in turn, as well as the system within its environment, all through the use of detailed diagrams and descriptions. This document is intended for the developers of the system, potential future developers, as well as the users of this system.

1.2 Scope

This software will be a voting system, including two voting algorithms: Instant Runoff Voting and Closed Party Voting. The system is designed for the use of election officials when all votes have been casted and counted. The system will fairly calculate the election winners based off of the chosen voting algorithm, either Instant Runoff Voting or Closed Party List Voting, and user-provided ballot and cast votes. The system will ensure all votes are counted according to the algorithms in a fair and consistent manner, and the system will provide the information pertaining to the winners of each election.

1.3 Overview

This document describes the organizational and architectural understanding of the voting system handling both Instant Runoff Voting and Closed Party List Voting. Through the use of description, diagrams, and images, this document will outline the plans for development of this system.
Each section of this document provides specific and unique information as to the organization of this system:

- Sections 1 and 2 will give an outline of the system in more general terms, as well as prepare the reader for the more technical aspects of the system and its design in the later sections of this document.
- Section 3 describes the architecture of the system, and makes use of several diagrams to do so.
- Section 4 explains the data structures to be utilized in this system.
- Section 5 gives a closer look at each of the components of our system, this is where each function of the system will be closely demonstrated.
- Section 6 describes the user interfaces of this system.

- Section 7 details the requirements as defined by the Systems Requirements Specification for this system and how those requirements will be met within the architecture of the system.

1.4 Reference Material

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Design Descriptions. IEEE Computer Society, 1998.

Sommerville, Ian. Software Engineering. Available from: VitalSource Bookshelf, (10th Edition). Pearson Education (US), 2015.

Project 1 - Waterfall Methodology. CSCI 5081, Software Engineering 1. University of Minnesota, 2023.

1.5 Definitions and Acronyms

CSV File: A comma-separated value file, or CSV file, is a text file that follows a specified format where commas separate each value. Often these files also separate each data record onto a new row using a newline character.

CPL: Closed party listing voting, or CPL, is a proportional voting type. For this type of voting each party provides a list of candidates to fill the seats they are given after the election. Voters then vote for the party instead of the individual candidate. Once votes are cast the seats are proportionally split between each party based on the percentage of votes that they received. Candidates from each party are then appointed based on their position on the provided candidate list.

IEEE: Institute of Electrical and Electronics Engineers, or IEEE, is a professional group known for being a leader in development of industry standards for many engineering and technology related areas.

IR: Instant runoff voting, IR or IRV, is a plurality or majority type voting system. For this type of voting all candidates appear on the ballot. Voters then indicate their order of preference for each candidate, but are not required to rank all candidates. The winner is then calculated by determining if any candidate has a majority once votes are counted. If there is no majority the candidate with the least votes is eliminated, and any votes for that candidate move to the next ranked candidate indicated on the ballot. This process continues until there is a majority, and a winner is declared.

Majority: A majority in IR voting is considered to be over 50% of the total number of votes.

2. System Overview

This system is designed to aid in the election process. It is designed to handle a specific portion of the electoral process; the counting of votes and determination of a winner. The program will be broken into 3 major components, accepting an input file or ballot, running one (and only one) of two voting algorithms, and delivering the election results to the user. Upon the acceptance of the input file from the user, the system will conduct one of two voting algorithms as requested by the user, either Instant Runoff Voting or Closed Party List Voting. At the completion of the algorithm, an election winner(s) will be found and delivered to the user. The primary expected user is election officials, though we expect additional users such as system testers. All pre and post services related to the election will not be handled by this product. It is not a component of a larger system, though it has the potential to be added to a larger automated electoral process, and this document may be helpful in that possible future development.

3. System Architecture

3.1 Architectural Design

This system design has four main parts: the election, the votable items, the results, and the Ballots. Each of these main components are abstract classes, and extended by specific subclasses according to the election type. Additionally, we utilize two implementation classes to define actions for creating, editing, and managing our audit file, and for scanning and reading our input document.

Election Class (abstract): This class holds all election data for easier allocation to other classes. This includes any data read from the input file, and any data to be saved to the audit file, and results information. This class is extended by the Cpl and Ir classes.

- Cpl Class: This class extends the abstract election class to hold any cpl specific information. This includes party information, seat information, and functionality to allocate and determine seats.
- Ir Class: This class extends the abstract election class to hold any ir specific information. This includes candidate information, ballot ranking information, and functionality to remove candidates and determine majority.

Ballot Class (abstract): This class holds all data for each individual ballot. It is extended by the cplBallot and irBallot classes.

- cplBallot Class: This class holds cpl election specific ballot information, mainly the party voted for in the ballot.
- irBallot Class: This class holds ir election specific ballot information. This includes the rankings for the ballot, the current rank, as well as the current voted candidate.

VotableItem Class (abstract): This class holds data about the items that can be voted for like name, type, and a list of associated ballots. This class is extended by the Candidate and Party classes.

- Candidate Class: This class holds candidate specific information for ir elections, like the votes for the individual candidate.

- Party Class: This class holds party specific information for cpl elections. This includes their votes, candidates, seats won, and seats left over.

Result Class (abstract): This class holds information about the result of the election. This holds the functionality to display the results, and is extended by the CplResult and IrResult election specific classes.

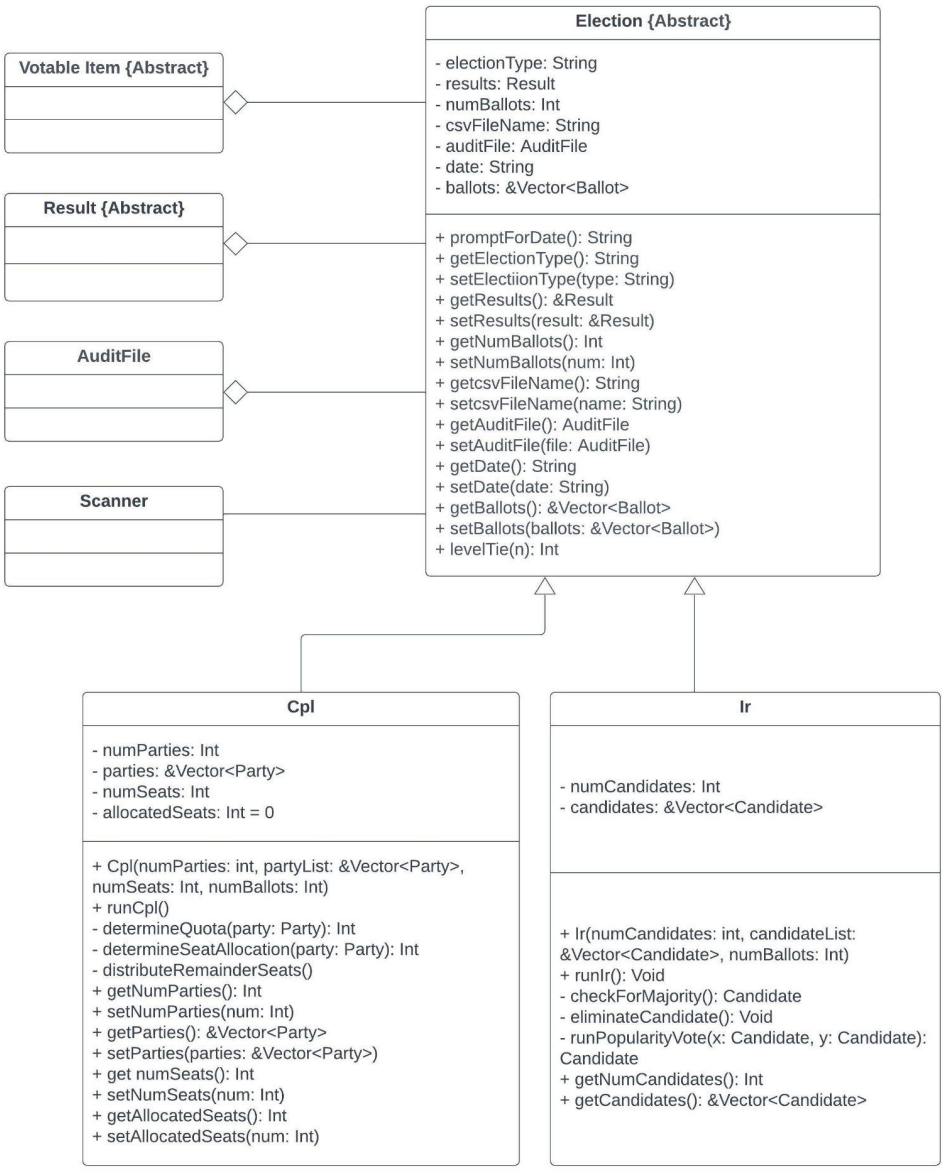
- CplResult: This class holds cpl election specific result information. This includes a list of the winning parties, and the Party class can be used for the seat allocation for each group.
- IrResult: This class holds ir election specific result information, namely a single winning candidate.

Scanner Class: This class holds the functionality to read the input file with the election information that is provided when the program is run.

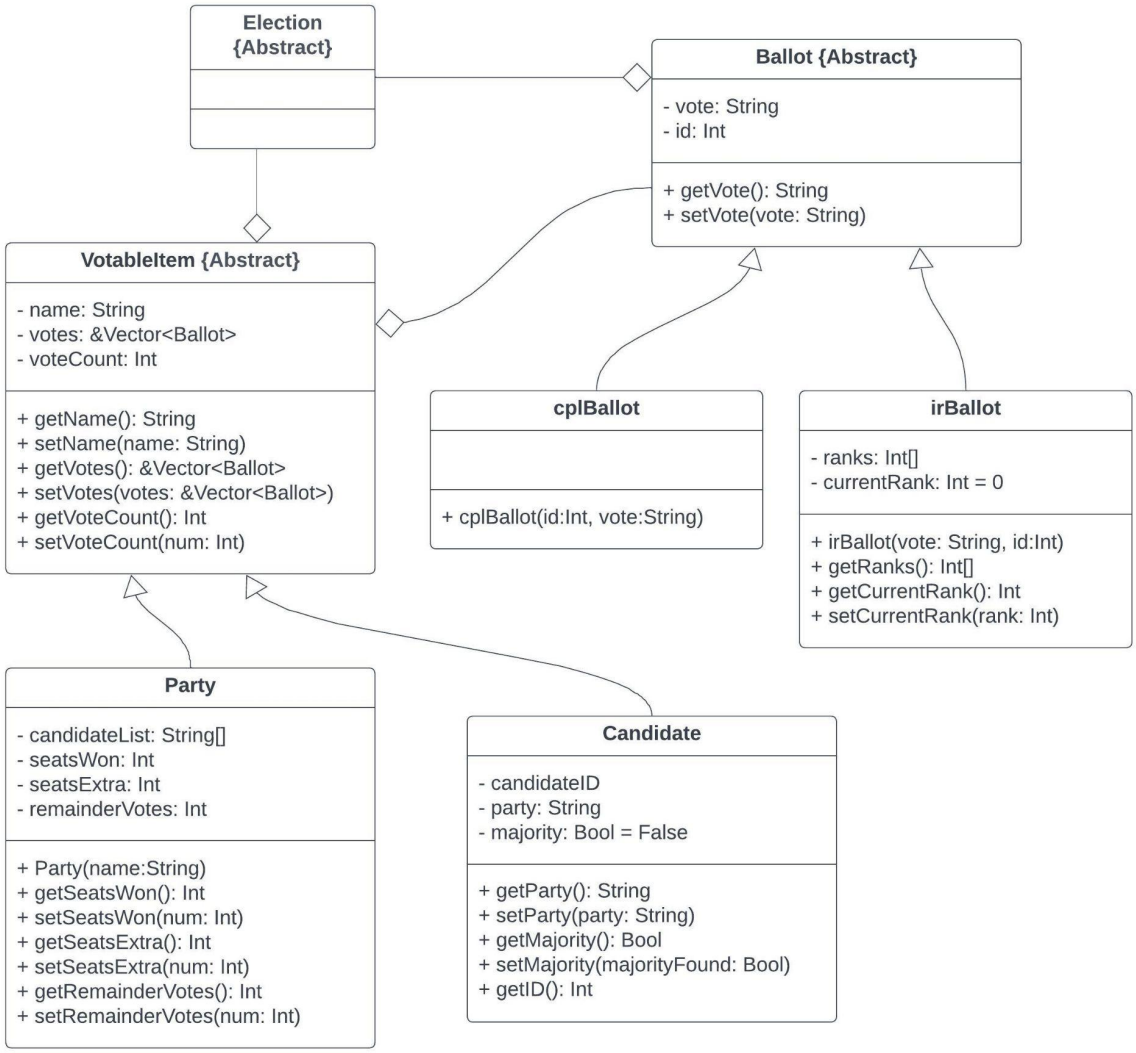
AuditFile: This class holds the information related to the audit file, created to keep track of how the election progressed and how the winner was declared.

**The Uml diagrams are displayed on the next few pages. If a class is empty of content on a diagram, it will be displayed in the following diagrams. **

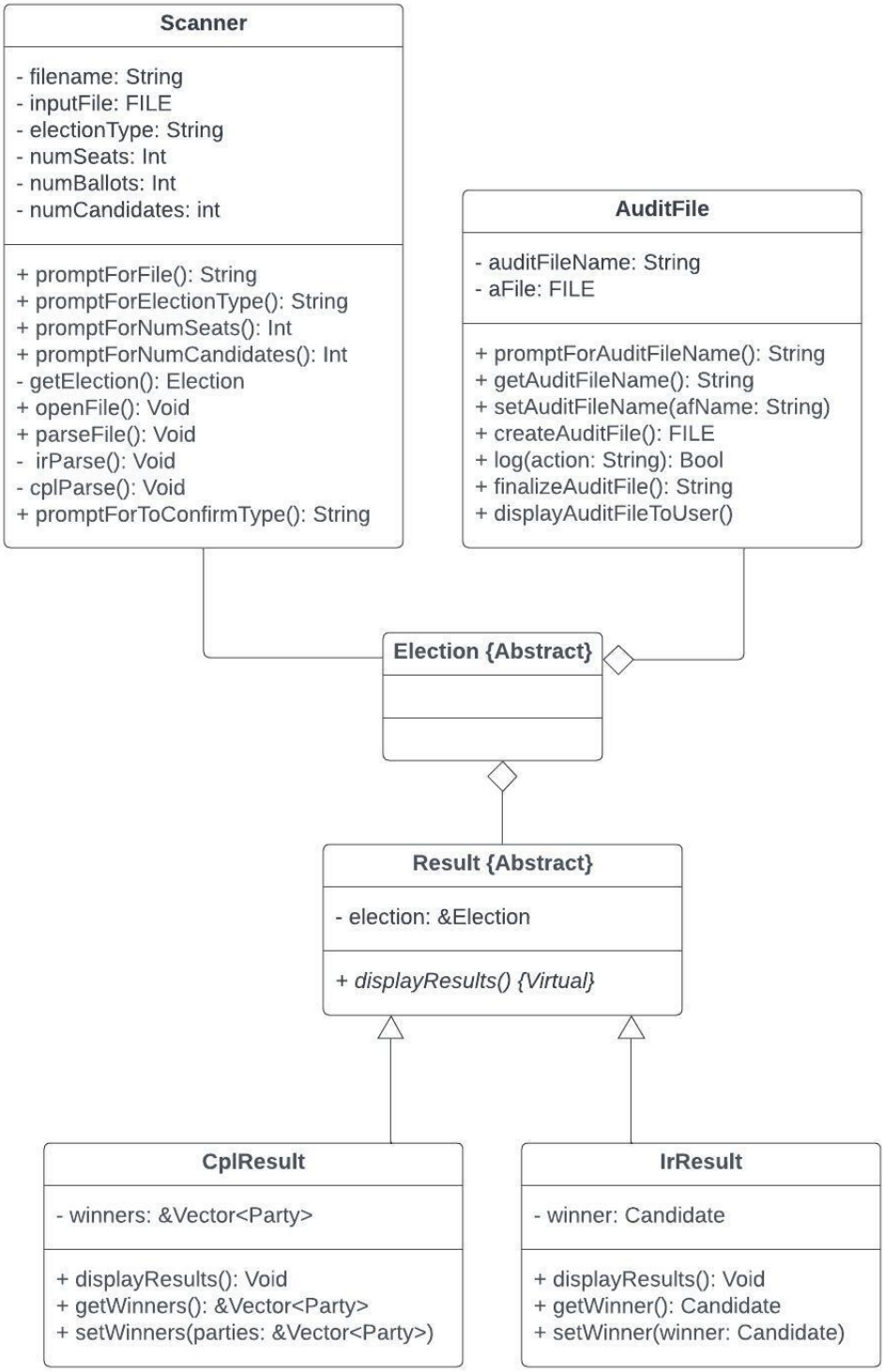
Uml Class Diagram 1:



Uml Class Diagram 2:

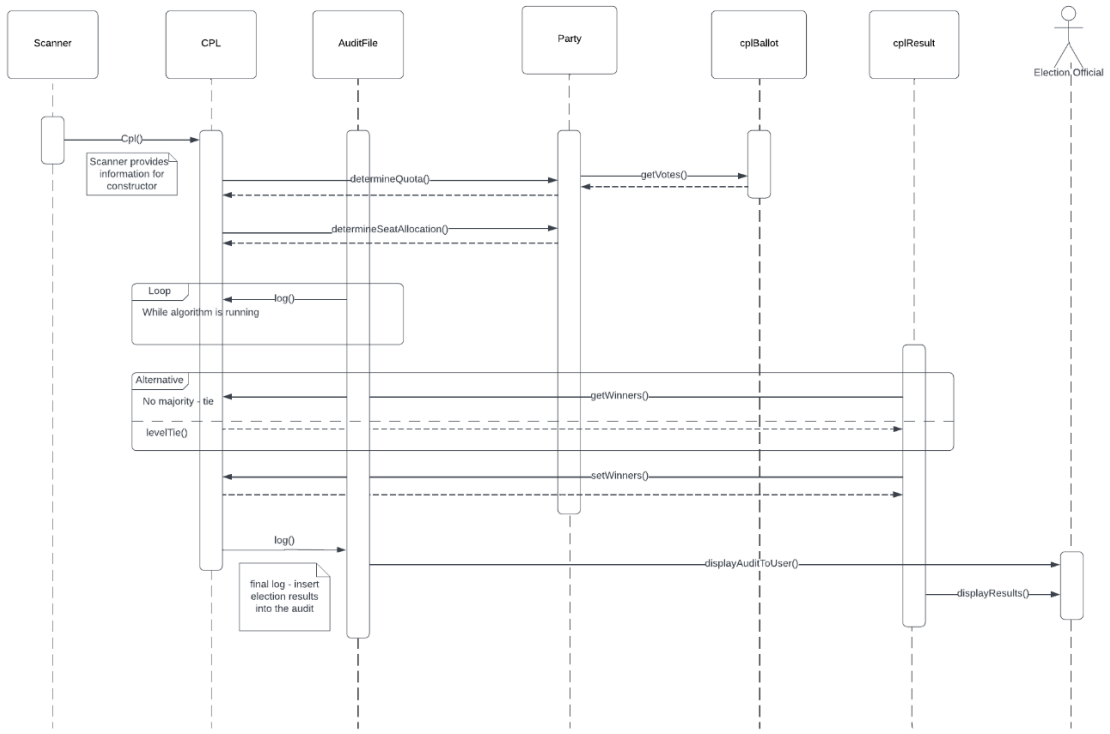


Uml Class Diagram 3:

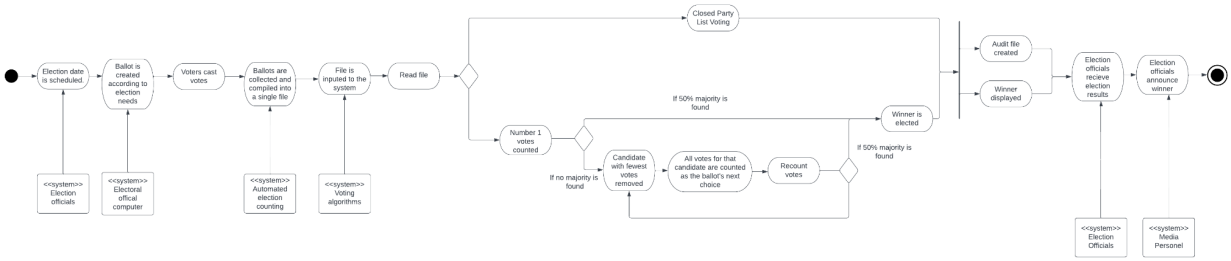


3.2 Decomposition Description

Sequence diagram:



Activity Diagram:



See appendix for zoomed images of this diagram.

3.3 Design Rationale

We decided to use our structure of the four main areas, with two additional helper classes because these were determined to be the four main sections of the program. We then chose to have an inherited class for each election type, inheriting from each of the four

main classes. This allows for more concise classes, without the need for unused attributes like what would occur with only the generalized main classes.

Additionally we chose to have a class to hold the audit file information, and the input file scanning functionality in their own classes. This again allowed us to have more concise classes, with clear boundaries around the sections of functionality needed in our program.

Although the decision to separate into this many classes will result in more sections of our code, the need to manage more pointers and memory, while also passing data between classes often, the decision will allow us an overall cleaner and more readable program. The cleaner program will result in slightly easier implementation of changes if needed later on in the development process.

4. Data Design

4.1 Data Description

The data is stored via the use of several object types. The bulk of the election information gained from the input file and user input will be stored in the Election object, either in the CPL or IR instantiation. Within those objects, there will be reference to the VoteableItem object which houses candidate or party information and votes. The votes themselves will be stored as a Ballot object. All of these data types and their use will be tracked at runtime through the audit file delivered to the user at the end of the system's completion. To be clear, there are 3 major objects used in data storage:

1. Election - either CPL or IR. Houses general election information and reference to all VoteableItems
2. VoteableItem - either Party or Candidate. Houses information pertaining to each Voteable Item and reference to all votes.
3. Ballot - cplBallot or irBallot. contains the votes themselves.

See section 4.2 to get a systematic and specific breakdown of each data type used in the system. See UML class diagram to see the composition of each of these objects and their housed data types.

4.2 Data Dictionary

Attribute Name	Attribute Type	Description
allocatedSeats	Int	The number of seats that have been

		allocated at any given point in the election.
aFile	FILE	Stores the file used to track data throughout the election process.
auditFile	AuditFile	Stores the AuditFile class object related to the election.
auditFileName	String	Stores the name of the audit file. Uses the format dateType.txt
ballots	&Vector<Ballot>	Stores a vector holding the related Ballot class objects.
candidateID	Int	This is equal to the order the candidate is listed in the input file.
candidateList	String[]	A string array holding the list of candidates.
candidates	Vector<Candidate>	Vector holding Candidate class type objects.
csvFile	String	The name of the csv input file, stored as a string.
date	String	The date of the election, provided by the user, stored as a string.
electionType	String	The type of election stored as a string.
filename	String	The csv input file name, stored as a string.
inputFile	FILE	The csv input file, stored as a FILE object.
name	String	The name of a votable item (party or candidate) stored as a string.
numBallots	Int	The total number of ballots in the election.
numCandidates	Int	The total number of candidates in an IR election.
numParties	Int	The total number of parties in a CPL election.
numSeats	Int	The total number of seats to be allocated in a CPL election.

parties	Vector<Party>	A Vector holding Party class type objects.
remainderVotes	Int	Holds the number of votes remaining for a party after calculations have been made and seats have been allocated. In other words, the votes that would result in a partial seat allocation.
results	Result	The results of the election as a Result class object.
seatsExtra	Int	The number of seats left after all candidates in a party have already been assigned to their allocated seats.
seatsWon	Int	The number of seats won by a party to be filled by their candidates.
vote	String	The current vote indicated on a ballot.
votes	&Vector<Ballot>	A vector of Ballot class type objects, indicating a current vote related to a common VotableItem.
voteCount	Int	A running count of votes given to a VotableItem.
winner	String	The name of the winner of an IR election, stored as a string.
winners	&Vector<Party>	The winners of a CPL election, stored as a vector of Party class type object.

5. Component Design

Election Class (Abstract)
Methods:

- getElectionType:
    - Description: This method is used as a getter for the electionType:String field.
    - Params: None

- Return: String that represents the type of election the Election Class represents.
- Flow:
  Return electionType

- getResults:
  - Description: This method is used as a getter for the results:&Results field.
  - Parameters: None
  - Return Type: A pointer to a Results object.
  - Flow:
    Return results

- levelTie:
  - Description: This method is used to level ties by picking a random number out from a chosen range of numbers.
  - Parameters: An Int *n* that represents the number of elements to be randomly picked from.
  - Return Type: Int
  - Flow:
    For i=0 to i=100:
        Let result = random number from the range 1 to n.
    Let result = random number from the range 1 to n
    Return result.

Cpl Class

Methods:

- runCpl:
  - Description: This is a driver function that holds most of the logic for running a cpl election. This function calls all other functions of Cpl in order to calculate the election. Creates a Result object with calculated election data.
  - Parameters: None
  - Return Type: Void
  - Flow:
    auditfile.log("running cpl election")
    For *i* in range(length(parties)):
        quota = determineQuota(parties[i])
        parties[i].seatsWon = parties[i].voteCount / quota
        auditfile.log("{parties[i].name} won {parties[i].seatsWon} seats")

```
            If parties[i].seatsWon > length(parties[i].getCandidateList):
                parties[i].seatsExtra =  parties[i].seatsWon -
            length(parties[i].candidateList)
            seatsAllocated += parties[i].getSeatsWon() - parties[i].get
        SeatsExtra()
        If (numSeats - seatsAllocated) != 0:
            determineSeatAllocation()
        For i in range( length(parties)):
            If parties[i].seatsExtra > 0:
                distributeRemainingSeats(parties[i])
        results = new CplResult(&parties)
        results.setWinner()
        auditfile.log("{winners} wins Cpl election")
```

- determineQuota:
    - Description: This method is used to determine the quota that will be used by the runCpl function.
    - Parameters: party:Party
    - Return Type: Int that represents the quota
    - Flow:
      Return party.getVoteCount() / numSeats.

- determineSeatAllocation:
    - Description: This method will be run after the initial seat allocation round is done in runCpl. This method will allocate any left over seats that have not been allocated to a party yet. It will use the Largest Remainder Formula as the logic behind this method.
    - Parameters: parties: None
    - Return Type: Void
    - Flow:
      auditfile.log("Allocating remaining seats with Largest Remainder Formula")
      Let max = -∞
      Let i = numSeats - allocatedSeats
      While i != 0:
        Let maxIndex = null
        For j in range(Length(parties)):
            If party.remainerVotes > max:

```
                    max = party.remainderVotes
                    Let maxCount = 0
            Let maxDuplicates[length(parties)] = []
            For k in range(parties):
                If parties[k].remainderVotes = max:
                    maxCount++
                    maxDuplicates.append(parties[i])
            If maxCount > 1:
                Let tb = tieLevel(maxCount)
                maxDuplicates.[tb].seatsWon++
                auditfile.log("{maxDuplicates.[tb].name} won +1 seat")
            Else:
                auditfile.log("{parties[maxIndex].name} +1 seat")
                parties[maxIndex].seatsWon++
           i--
```

- distributeRemainderSeats:
    - Description: This method is used when a party has won more seats than the number of candidates in the party. The remaining seats are distributed to parties with open seats by a lottery.
    - Parameters: party:Party is the party that is lottering off the extra seats
    - Return Type: Void
    - Flow:

```
auditfile.log("distributing {party.extraSeats} extra seats from
{party.name} to eligible parties.")
While party.seatsExtra > 0:
        eligibleParties[length(parties)] = []
        For i in range(parties):
            if parties[i] != party AND parties[i].seatsExtra = 0 AND
(length(parties[i].candidates) - parties[i].seatsWon > 0:
                eligibleParties.append(parties[i])
        Let n = tieLevel(length(eligibleParties))
        eligibleParties[n].seatsWon++
        auditfile.log("{eligibleParties[n].name} +1 seat ")
        party.seatsExtra--
```

Ir Class

Methods:

- runIr:

- Description: This is a driver function that holds most of the logic for running a Ir election. This function calls all other functions of Ir in order to calculate the election.
- Parameters: None
- Return Type: Void
- Flow:
  auditfile.log("running ir election")
  While checkForMajority() = Null:
      eliminateCandidate()
  results = new IrResult(&candidates)
  results.setWinner()
  auditfile.log("{winner} wins IR election")

- checkForMajority:
  - Description: This method iterates through candidates and checks whether a candidate has a majority yet.
  - Parameters: None
  - Return Type: Candidate type that represents the candidate with the majority.
  - Flow:
    For i in range(candidates):
        If length(candidates[i].getVotes) ./ numBallots > 0.5:
            candidates[i].setMajority()
            auditfile.log("{candidates[i].name} has obtained majority")
            return candidates[i]
    If length(candidates) = 2:
        Return runPopularityVote(candidates[0], candidates[1])
    Return Null

- eliminateCandidate:
  - Description: This method is used by runIr when no clear majority has been found. It iterates through the candidates and eliminates the candidate with the least number of votes.
  - Parameters: None
  - Return Type: Void
  - Flow:
    Let min = ∞
    minIndex = null

```
targetCandidate: &Candidate = null
For i in range(length(candidates))
        If length(candidates[i].getVotes()) < min:
                min = length(candidates[i].getVotes()
                minIndex = i
Let minCount = 0
Let minDuplicates[length(candidates)] = []
For j in range(length(candidates)):
        If length(candidates[j].getVotes()) = min:
                minDuplicates.append(candidates[j])
                minCount++
If minCount > 1:
        targetCandidate = minDuplicates[levelTie(minCount)]
Else:
        targetCandidate = candidates[minIndex]

targetVotes = targetCandidate.getVotes()
For k in targetVotes:
        k.rank ++
        If k.getCurrentRank() = candidates.getID:
                candidates.votes.append(k)
        targetCandidate.votes.remove(k)
candidates.remove.targetCandidate
auditfile.log("eliminated {targetCandidate.name}")
```

- runPopularityVote:
    - Description: This method takes two candidates and calculates who has the most votes.
    - Parameters: candidate1: Candidate, candidate2: Candidate
    - Return Type: winner: Candidate that represents the winner of the popular vote.
    - Flow:

```
auditfile.log("running popular vote between {candidate1} and
{candidate2}")
If length(candidate1.votes) == length(candidate2.votes):
        If levelTie(2) = 1:
                auditfile.log("{candidate1} wins popular vote")
                Return candidate1
        Else:
                auditfile.log("{candidate2} wins popular vote")
```

Return candidate2
Elif length(candidate1.votes) > length(candidate2.votes):
auditfile.log("{candidate1} wins popular vote")
Return candidate1
Else:
uditfile.log("{candidate2} wins popular vote")
return candidate2

AuditFile
Methods:

- promptForAuditFileName:
  - Description: Prompt the user for the election date. This information is needed to name the audit file
  - Parameters: None
  - Return Type: String
  - Flow:
    Prompt user - "Enter date of election: ";
    String date = user input;
    return date;

- getAuditFileName:
  - Description: Deliver Audit File Name.
  - Parameters: None
  - Return Type: String
  - Flow:
    return fileName;

- setAuditFileName:
  - Description: Set the file name for the AuditFile object.
  - Parameters: String afName
  - Return Type: void
  - Flow:
    auditFileName = afName;
    return;

- createAuditFile:
  - Description: Using the file name created from user input, create a FILE. Permissions should be write-only at this point.
  - Parameters: none

- Return Type: FILE
- Flow:

```
aFile = FILE auditFileName;
aFile set permissions = read-only;
aFile.open();
return aFile;
```

- log:
  - Description: To be called at the end of most voting algorithm processes. This function will take a string describing the action just completed, for example how many votes a specific candidate had, and write that string to the audit file in sequence.
  - Parameters: String action
  - Return Type: Bool
  - Flow:

```
aFile.write(action);
if error / fail to write
        return false;
return;
```

- finalizeAuditFile:
  - Description: Close the file. Change permissions of the audit file to read-only for the delivery of the file to the user.
  - Parameters: None
  - Return Type: String
  - Flow:

```
aFile.close();
aFile set permissions = read only;
return auditFileName;
```

- displayAuditFileToUser:
  - Description: Deliver the file to the user at the completion of the election.
  - Parameters: None
  - Return Type: void
  - Flow:

```
print (auditFileName file path);
return;
```

Scanner Class

Methods:

- promptForFile:
    - Description: Prompts the user for the filename of the input file that is to be opened.
    - Parameters: None
    - Return Type: String with the name of the file that the user input.
    - Flow:
      Let name = input(“Please enter filepath or filename.”)
      Filename = name
      Return name

- promptForElectionType:
    - Description: Prompts the user for the type of election the program is to process.
    - Parameters: None
    - Return Type: String
    - Flow:
      Let electionType = input(“Please enter election type (CPL/IR)”)
      Return electionType

- promptForNumSeats:
    - Description: Prompts the user for the number of seats in the election.
    - Parameters: None
    - Return Type: Int
    - Flow:
      Let numSeats = input(“Please enter number of seats (#).”)
      Return numSeats

- promptForNumCandidates:
    - Description: Prompts the user for the number of candidates
    - Parameters: None
    - Return Type: Int
    - Flow:
      Let numCandidates = input(“Please enter number of candidates (#).”)
      Return numCandidates

- getElection:
    - Description: This method instantiates an Election object with the needed information for either IR or CPL elections.
    - Parameters: None
    - Return Type: Election

- Flow:
  electionType = inputFile.readLine()
  If electionType = “Ir”:
  Return new Ir()
  Else if electionType = “Cpl”:
  Return new Cpl()

- openFile:
  - Description: This method is used to open the input file with the filename that was prompted for by getFileName.
  - Parameters: None
  - Return Type: Void
  - Flow:
    While inputFile = null
    Try:
    inputFile.open(fileName):
    Except:
    print(“Invalid file name. Try again”
    promptForFile()

- ParseFile():
  - Description: Determines what type of election needs to be parsed.
  - Parameters: None
  - Return Type: Void
  - Flow:
    If electionType = “Ir”:
    irParse()
    Else cplParse()

- irParse:
  - Description: Parses input file for information about an IR election.
  - Parameters: None
  - Return Type: Void
  - Flow:
    For i in numCandidates:
    Candidates.append(new Candidate)
    Line = input.readline()
    line.split(‘,’);
    For j in range(line):
    Candidates[j].name = lines[j]

```
            Candidates[j].ID = j
        Let numBallots = readline()
        For k in numBallots:
            ballot = New IrBallot()
            Ballot.votes = readline()
            ballot.setRanks()
            Let index = ranks.index(ballot.getCurrentRank)
            Candidates[index].append(ballot)
```

- cplParse:
    - Description: Parses input file for information about an CPL election.
    - Parameters: None
    - Return Type: Void
    - Flow:

```
        For i in numParties:
            Candidates.append(new Candidate)
        Line = input.readline()
        line.split(‘,’);
        For j in range(line):
            Parties[j].name = lines[ j]
            Parties[j].ID = j
        Let candidateNum = inputFile.readLine()
        For k in candidateNum:
            Parties[k].candiates.append(inputFile.readLine())
        Let numBallots = readline()
        For l in numBallots:
            ballot = New CplBallot()
            Ballot.vote = readline()
            Let index = parties.index(party.name = ballot.vote)
            parties[index].votes.append(ballot)
```

- PromptToConfirmType:
    - Description: This prompts to run a particular election after all information is obtained from input file and user.
    - Parameters: None
    - Return Type:
    - Flow:

```
        If electionType = “Ir”:
            Input = input(“Calculate IR election? Y/N”)
            If input = “Y”:
```

```
                    Return getElection()
                Else:
                    print("Canceling election calculation")
                    Exit
            Else:
                Input = input("Calculate CPL election? Y/N")
                If input = "Y":
                    Return getElection()
                Else:
                    print("Canceling election calculation")
                    Exit
```

Ballot Class {Abstract}

methods:

- getVote:
    - Description: This is a getter for the vote attribute.
    - Parameters: None
    - Return Type: String representing the vote on the ballot
    - Flow:
      Return vote

- setVote:
    - Description: This is a Setter for the vote attribute.
    - Parameters: vote: String
    - Return Type: Void
    - Flow:
      This.vote = vote

CplBallot Class

Methods:

- cplBallot:
    - Description: Constructor for the Cpl class.
    - Parameters: id:Int, vote: String
    - Return Type: CplBallot
    - Flow:
      This.vote = vote
      This.id = id

IrBallot Class

- irBallot:

- Description: Constructor class for the IrBallot class
- Parameters: vote:String, id:Int, ranks:Int[]
- Return Type: IrBallot
- Flow:
  This.vote = vote
  This.id = id
  Ranks = vote.split(“,”)

- getRanks:
  - Description: This is a getter method for the ranks attribute
  - Parameters: None
  - Return Type: ranks:int[]
  - Flow:
    Return ranks

- getCurrentRank:
  - Description: This is a getter for the currentRank attribute.
  - Parameters: None
  - Return Type: Int
  - Flow:
    Return currentRank

- setCurrentRank:
  - Description: This is a setter method for the currentRank attribute.
  - Parameters: rank:Int
  - Return Type: Void
  - Flow:
    currentRank = rank

VotableItem Class {Abstract}
Methods:

- getName():
  - Description: This is a getter method for the name attribute.
  - Parameters: None
  - Return Type: String
  - Flow:
    Return name

- setName():
  - Description: This is a setter method for the name attribute.

    - Parameters: name:String
    - Return Type: Void
    - Flow:
      This.name = name

- getVotes():
    - Description: This method returns a pointer to the votes this item has received.
    - Parameters: None
    - Return Type: Void
    - Flow:
      Return votes

- setVotes:
    - Description: This is a setter method for the votes attribute
    - Parameters: votes:&Vector<ballot>
    - Return Type: Void
    - Flow:
      This.votes = votes

- getVoteCount:
    - Description: This is a getter for the voteCount attribute.
    - Parameters: None
    - Return Type: Int
    - Flow:
      Return voteCount

- setVoteCount:
    - Description: This is a setter for the voteCount attribute.
    - Parameters: num:Int
    - Return Type: Void
    - Flow:
      voteCount = num

Party Class

Methods:

- getSeatsWon:
    - Description: A getter for seatsWon
    - Parameters: None
    - Return Type: Int

    - Flow:
      Return seatsWon

- setSeatsWon:
    - Description: This method is a setter for the seatWon attribute.
    - Parameters: num:Int
    - Return Type: Void
    - Flow:
      seatsWon = num

- getSeatsExtra:
    - Description: A getter for the seatsExtra attribute
    - Parameters: None
    - Return Type: Int
    - Flow:
      Return seatsExtra

- setSeatsExtra:
    - Description: This is a setter method for the seatsExtra attribute
    - Parameters: num:Int
    - Return Type: Void
    - Flow:
      seatsExtra = num

- getRemainderVotes:
    - Description: This is a getter method for the remainderVotes attribute.
    - Parameters: None
    - Return Type: Int
    - Flow:
      Return remainderVotes

- setRemainderVotes:
    - Description: This is a setter method for the remainderVotes attribute.
    - Parameters: num:Int
    - Return Type: Void
    - Flow:
      remainderVotes = num

Candidate Class
Methods:

- getParty:
    - Description: This is a getter method for the party attribute.
    - Parameters: None
    - Return Type: String
    - Flow:
      Return party

- setParty:
    - Description: This is a setter method for the party attribute.
    - Parameters: party:String
    - Return Type: Void
    - Flow:
      This.party = party

- getMajority:
    - Description: This is a getter method for the majority attribute.
    - Parameters: None
    - Return Type: Bool
    - Flow:
      Return majority

- setMajority:
    - Description: Sets the majority attribute to true when a clear majority is found.
    - Parameters: majorityFound:Bool
    - Return Type: Void
    - Flow:
      Majority = majorityFound

- getID:
    - Description: This is a getter for the candidateID attribute.
    - Parameters: None
    - Return Type: Int
    - Flow:
      Return candidateID

Result Class {Abstract}
Methods:

- *displayResults*:

- Description: This is a virtual function that will be overrode by Result's children classes. It will allow for the displaying of results to the terminal.
- Parameters: None
- Return Type: Void
- Flow:
  N/A

CplResults Class

Methods:

- displayResults:
  - Description: This function will realize Results virtual function, *displayResults*. It will display the results of the CPL election to the terminal.
  - Parameters: None
  - Return Type: Void
  - Flow:

```
print("Election results:
        Type of Election: Closed Party List
        Number of ballots cast: {numBallots}
        Number of seats: {numSeats}
        Parties and their candidates: ")
For i in range( parties):
        print(parties[i].name, ":")
        For k in range(parties[i].candidates):
                print(parties[i].candidates[k])
        print("Number of seats in party: {parties[i].votes}")
        print("Number of allocated seats: {parties[i].seatsWon}")
        print("Elected candidates of each party: ")
        For l in range(parties[i].seatsWon):
                print(parties[i].candidates[l])
```

- getWinners:
  - Description: This is a getter method for the winners attribute
  - Parameters: None
  - Return type: &Vector<Party>
  - Flow:
    Return winners

- setWinners:
  - Description: This is a setter for the winners attribute

- Parameters: parties:&Vector<Party>
- Return Type: Void
- Flow:
  Winners = parties

IrResults Class

Methods:

- displayResults:
  - Description: This function will realize Results virtual function, *displayResults*. It will display the results of the IR election to the terminal.
  - Parameters: None
  - Return Type: Void
  - Flow:
    print("Election results:
    Type of Election: Instant Runoff
    Number of ballots cast: {numBallots}
    Candidates: ")
    For i in range(candidates):
    print("{candidates[i].name}:
    Votes: {candidates[i].voteCount}
    Percentage: %{numBallots ./ candidates[i].voteCount}")
    print("Winner: {winner}")

- getWinner:
  - Description: This is a getter method for the winner attribute
  - Parameters: None
  - Return type: Candidate
  - Flow:
    Return winner

- setWinners:
  - Description: This is a setter for the winners attribute
  - Parameters: winner:Candidate
  - Return Type: Void
  - Flow:
    this.winner = winner

6. Human Interface Design

6.1 Overview of User Interface

Once the user has run the program, the user will be prompted to enter a ballot file. If there is any missing information from the ballot file the user will be prompted for additional information. Once the program has all of the necessary information the vote will be calculated and displayed to the terminal. The information the user will receive will be the winner(s) of the election, type of election, number of seats, number of ballots cast, number of votes for each candidate, and the percentages of votes received for each candidate.

6.2 Screen Images

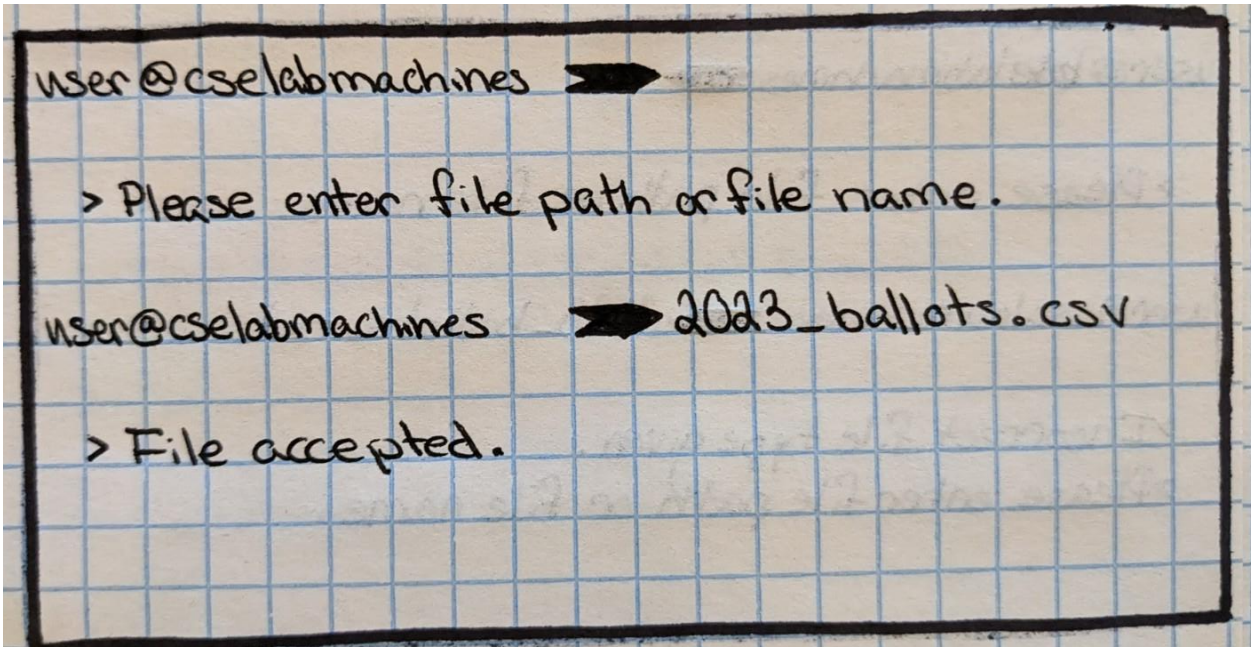


Figure 1: User prompted for file name, file is accepted.

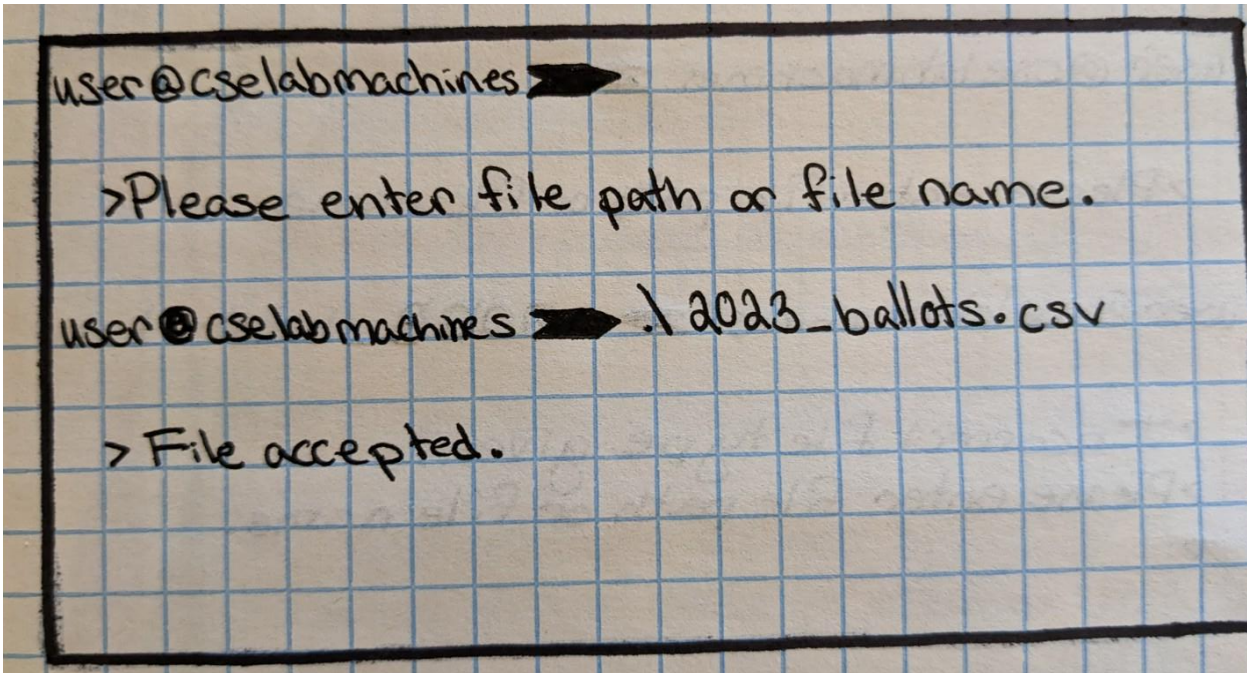


Figure 2: User prompted for file path, file is accepted.

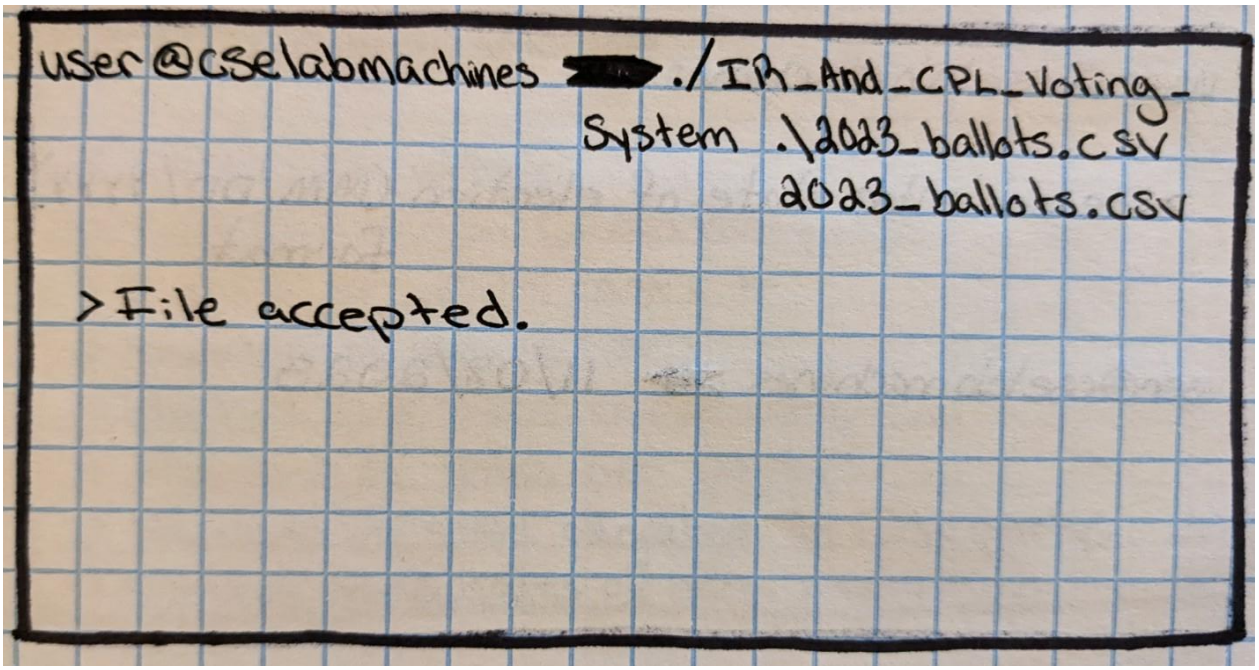


Figure 3: User runs program giving file path and name as command line arguments.

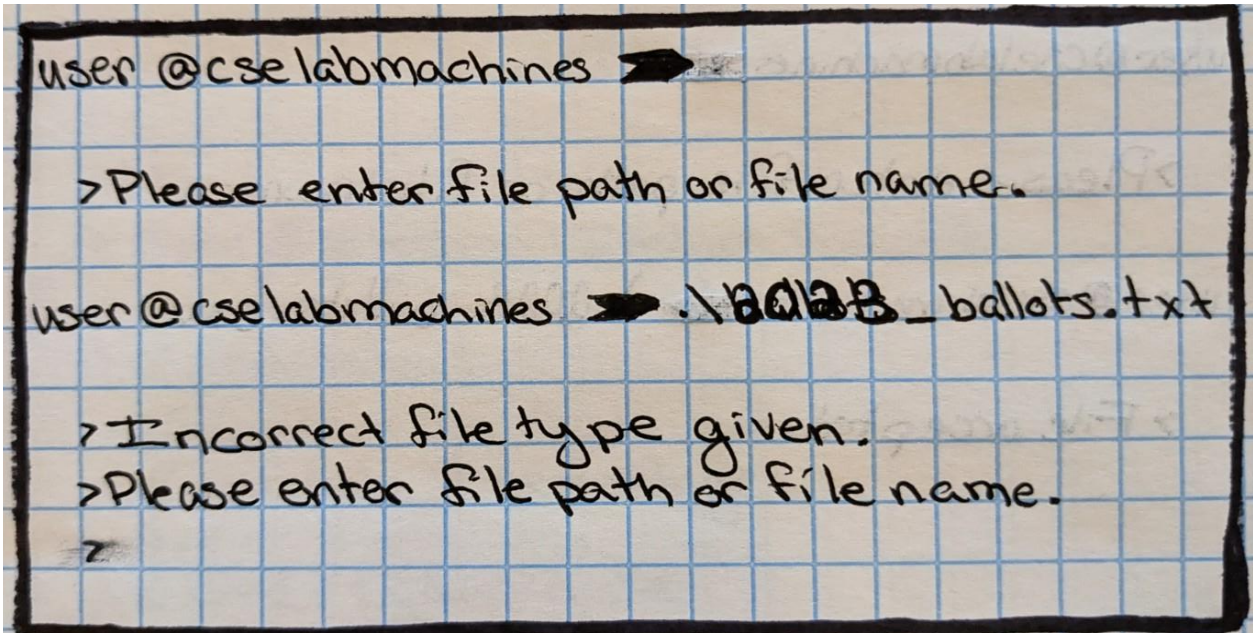


Figure 4: User is prompted for file path or file name. Incorrect file type is given.

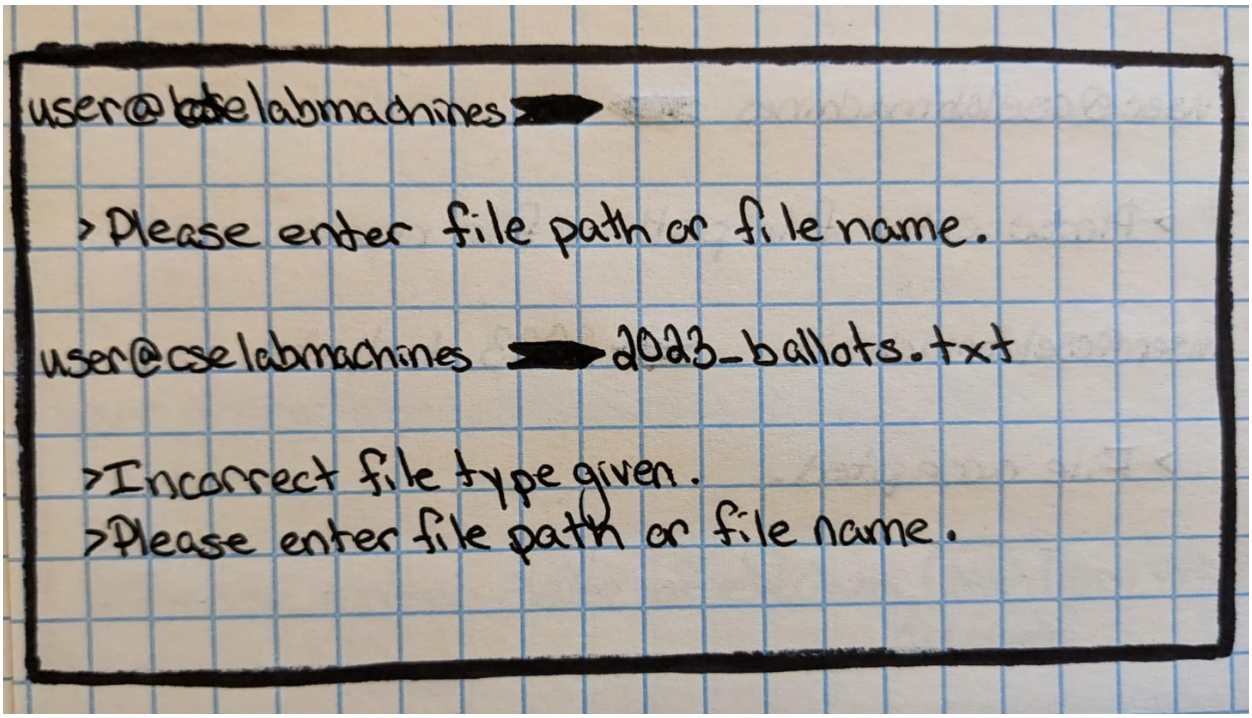


Figure 5: User is prompted for file path or file name. Incorrect file type is given.

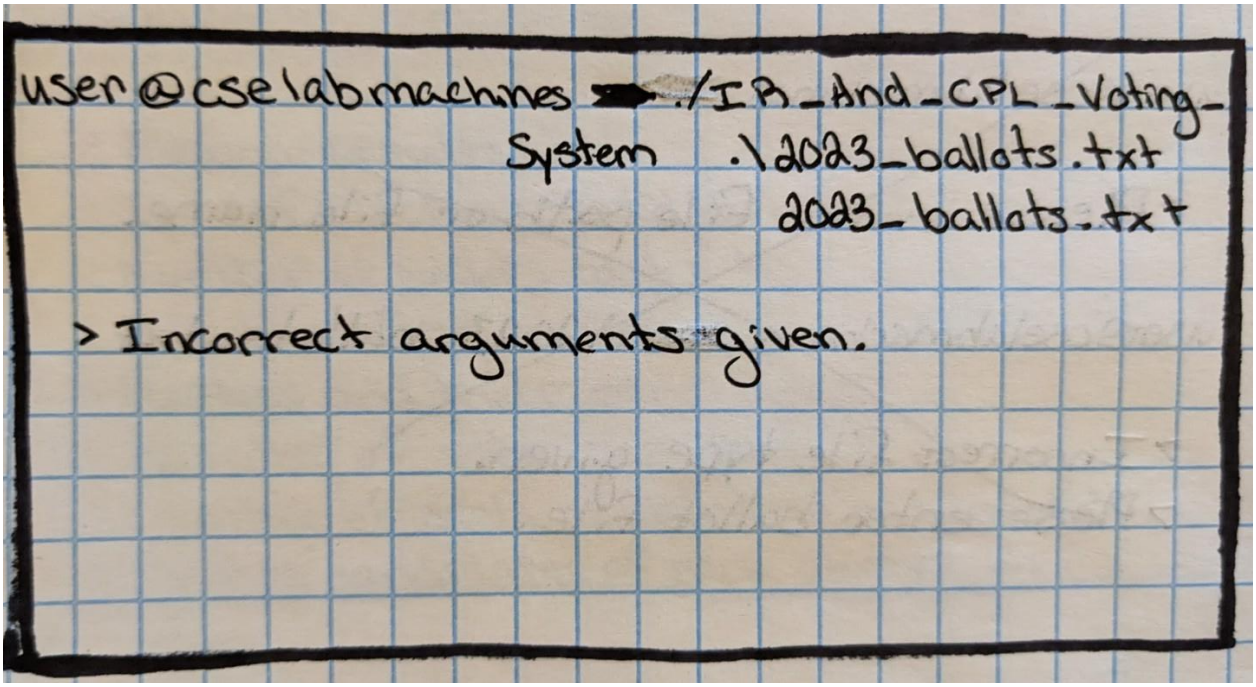


Figure 6: User runs program giving file path and name as command line arguments. Incorrect arguments are given.

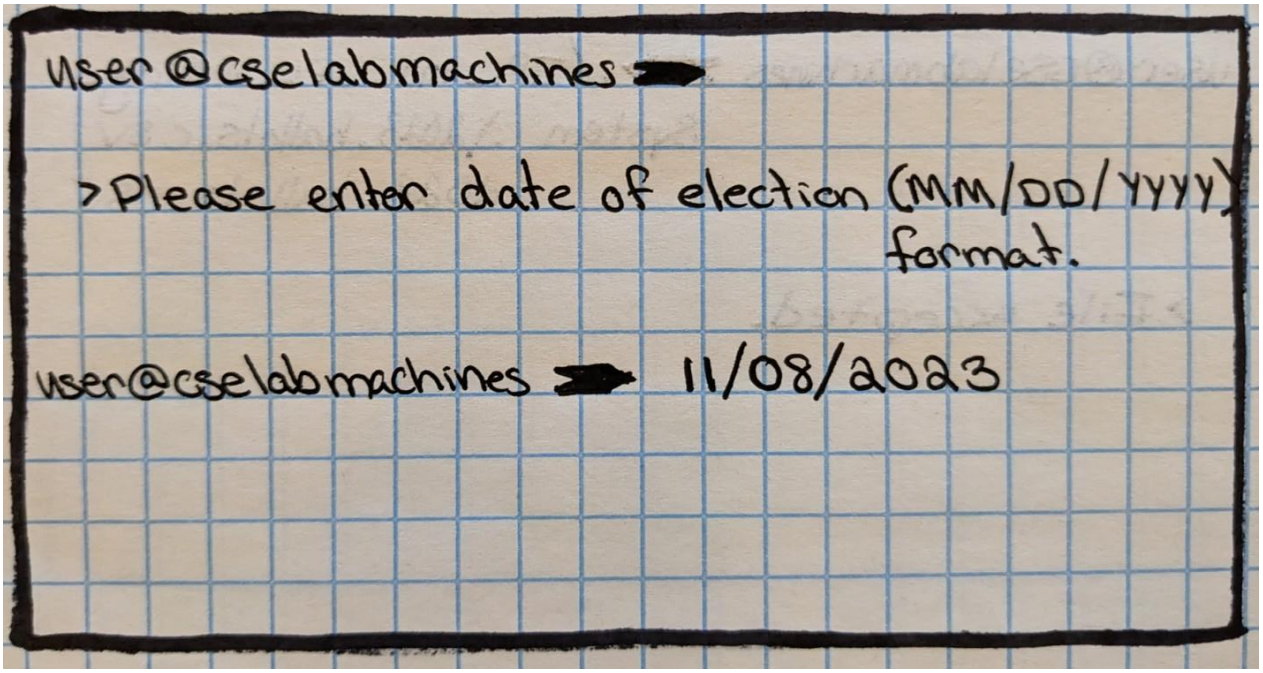


Figure 7: User is prompted for date of election with correct format.

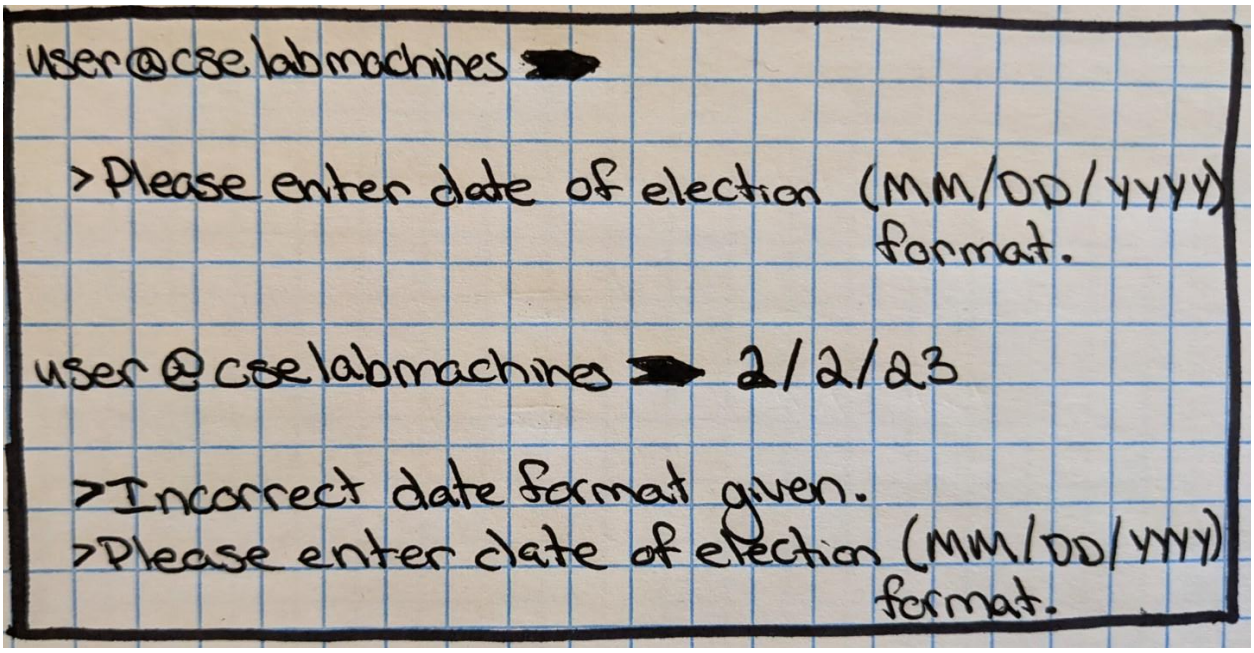


Figure 8: User is prompted for date of election with correct format. User inputs incorrect format and is notified.

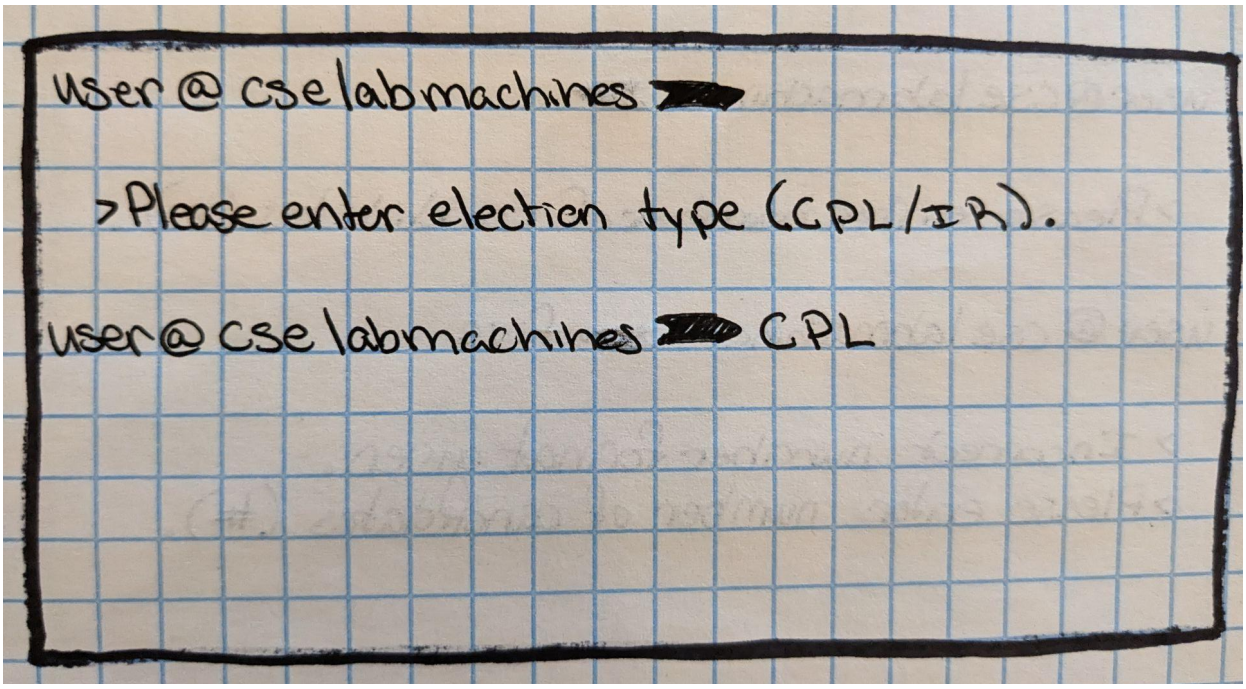


Figure 9: User is prompted for election type.

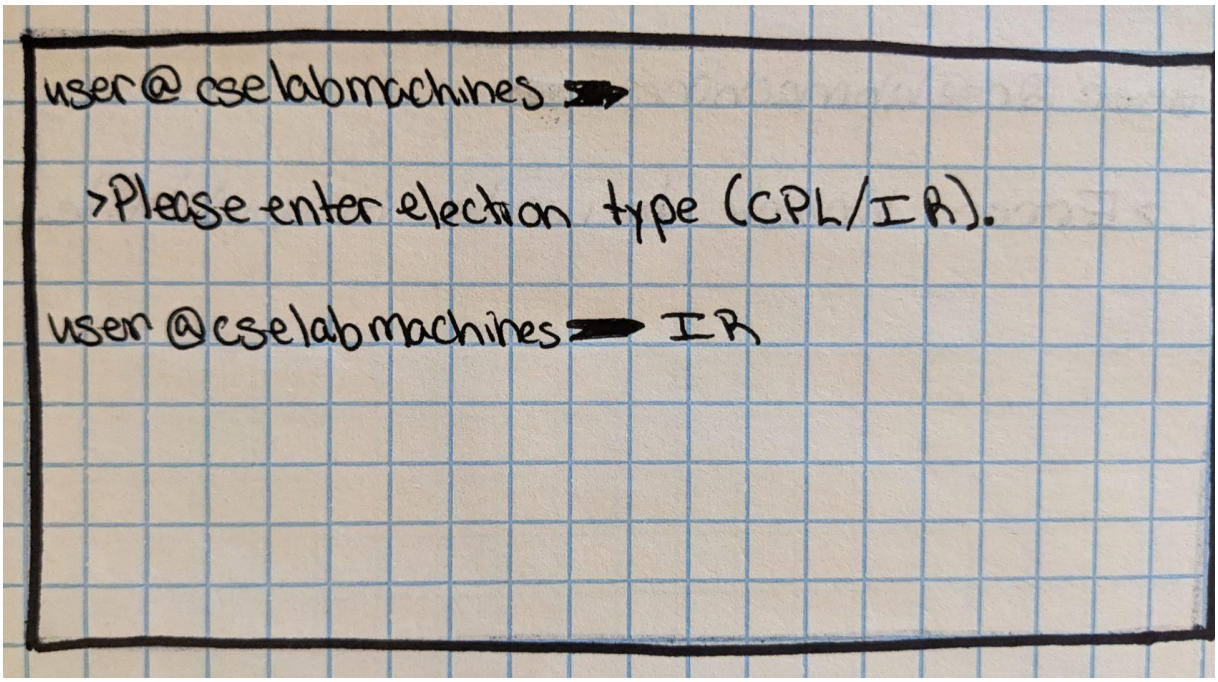


Figure 10: User is prompted for election type.

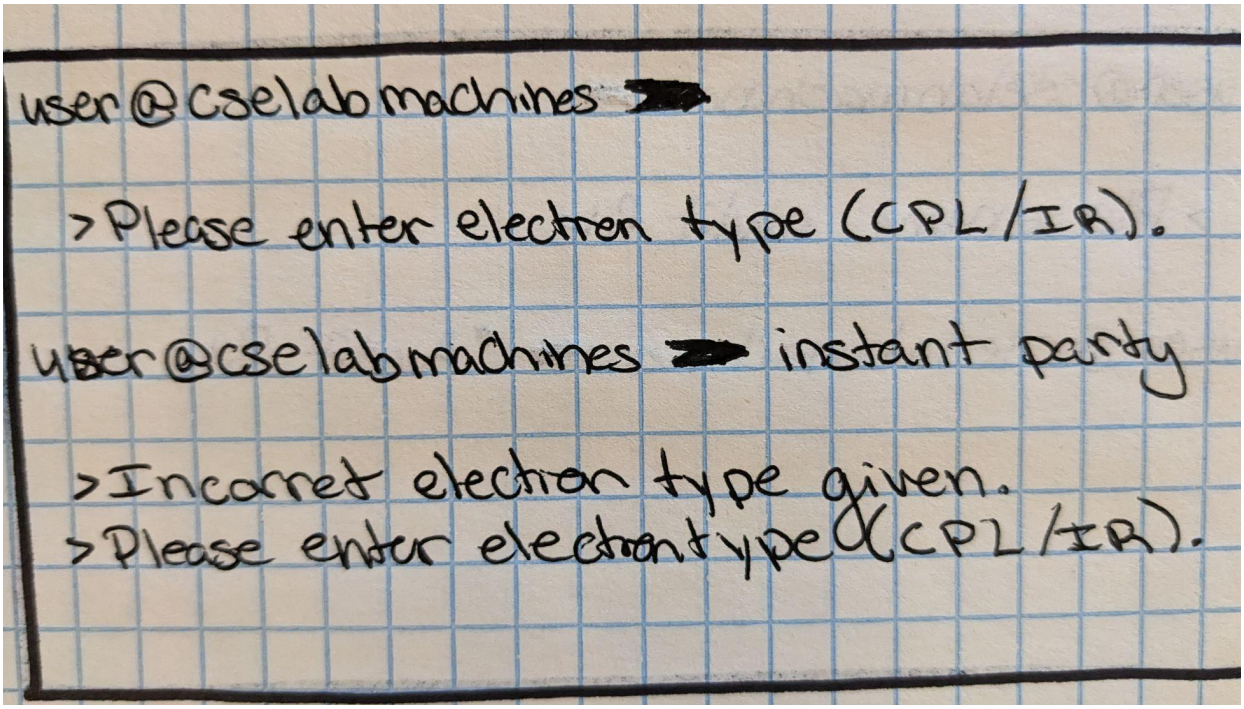


Figure 11: User is prompted for election type. User enters incorrect format.

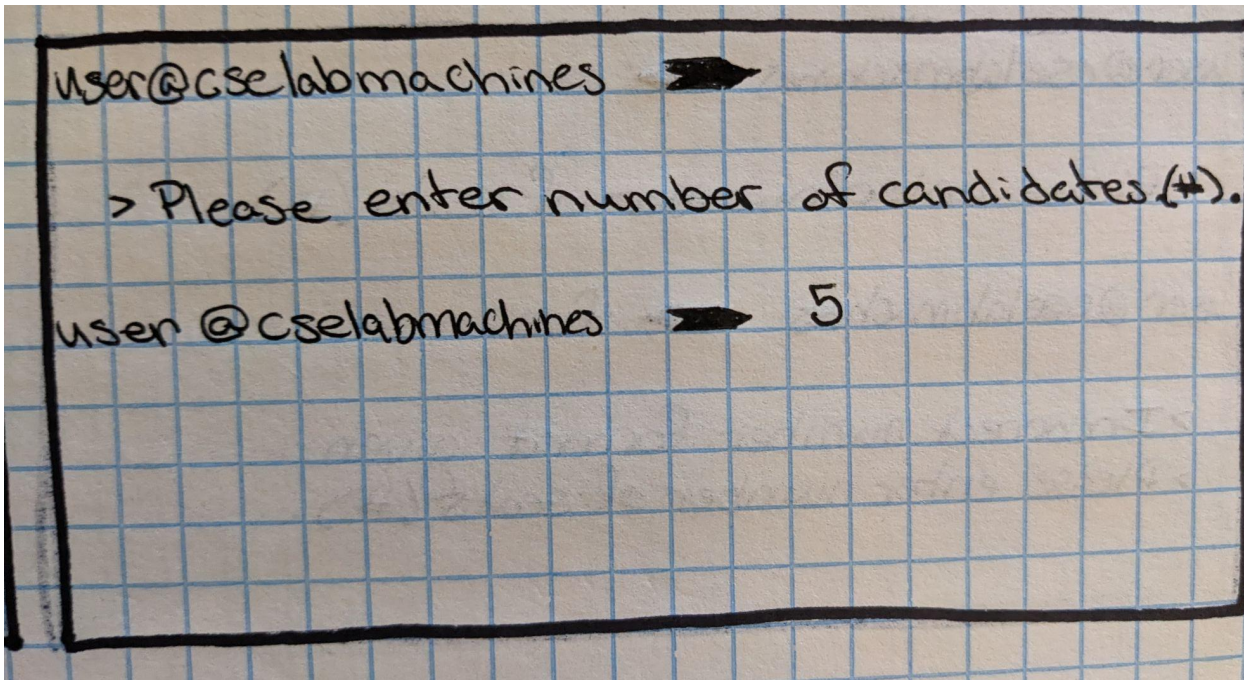


Figure 12: User is prompted for number of candidates.

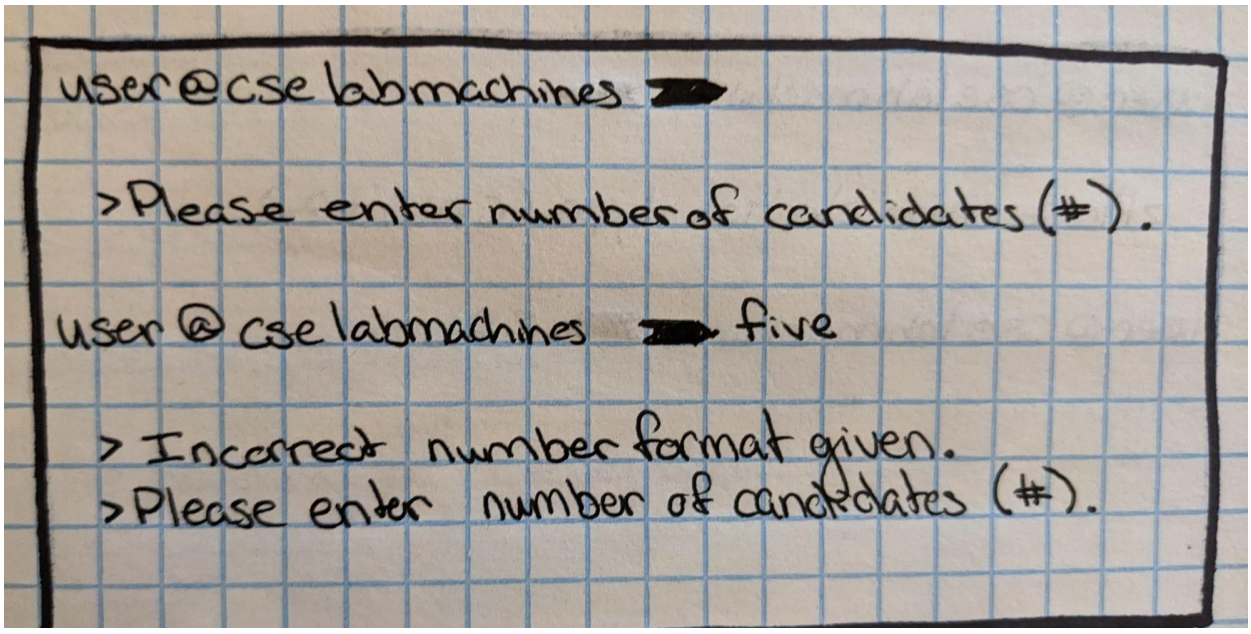


Figure 13: User is prompted for number of candidates in election. User enters incorrect information.

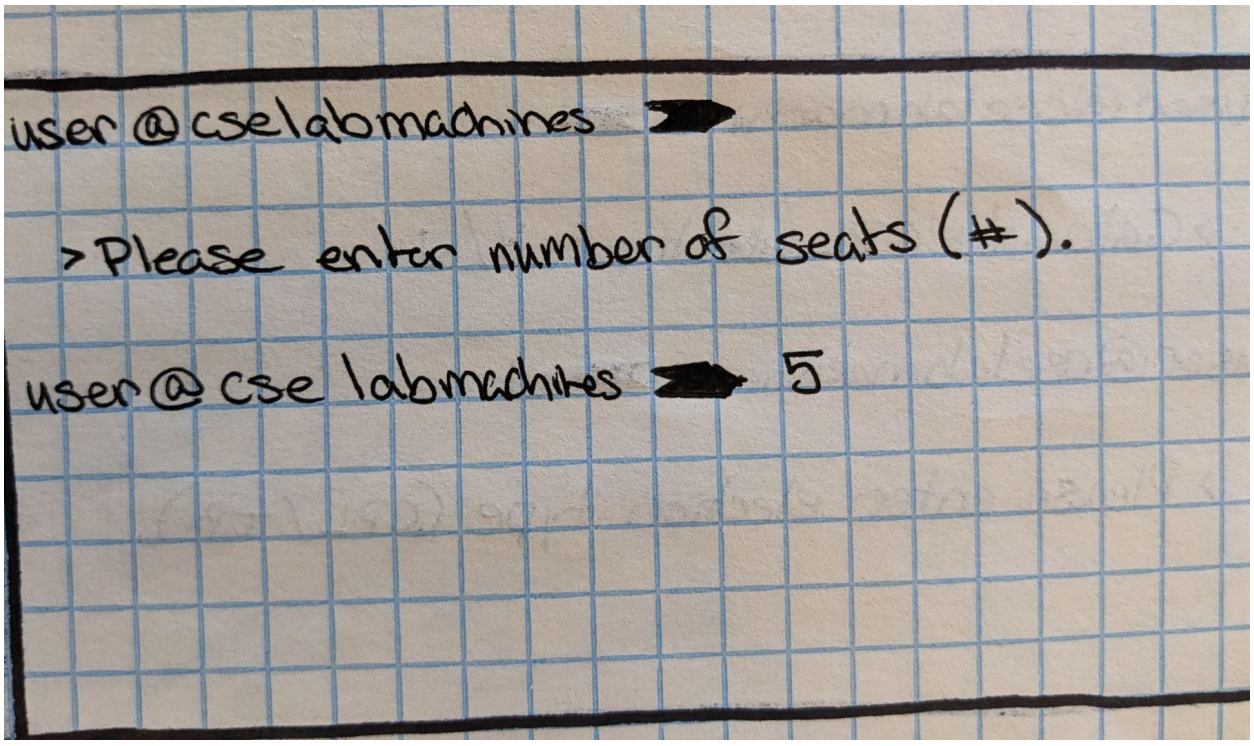


Figure 14: User is prompted for number of seats available.

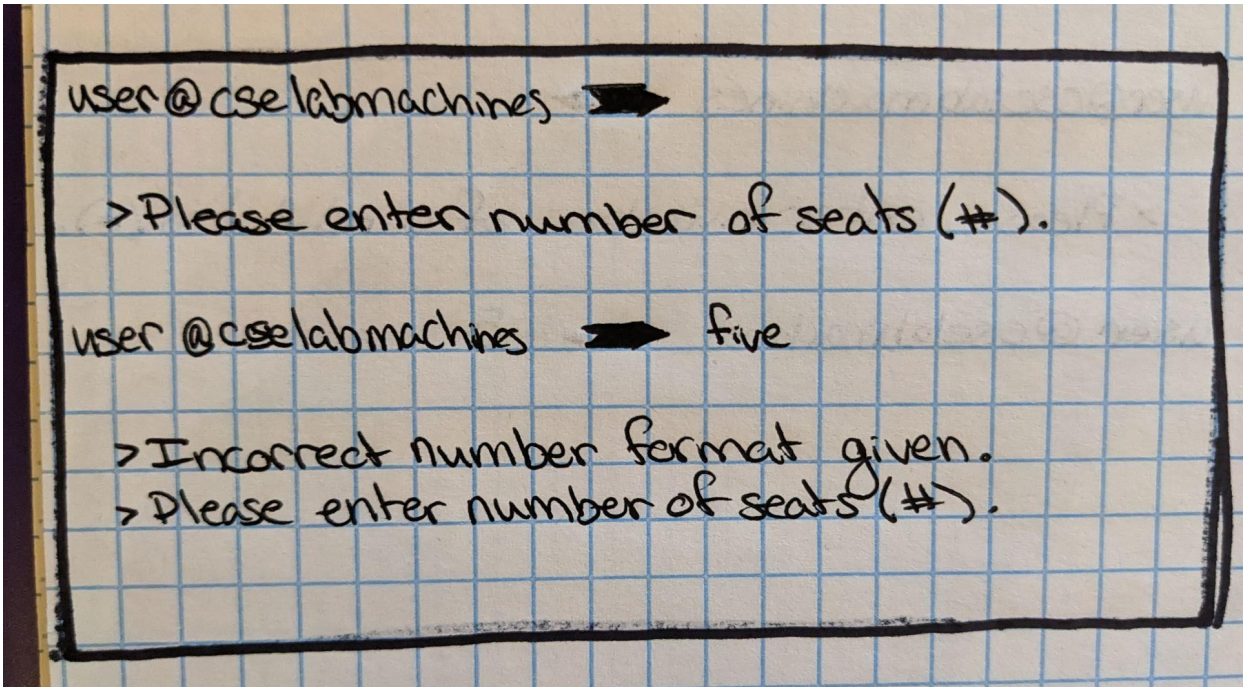


Figure 15: User is prompted for number of seats available. User enters incorrect information.

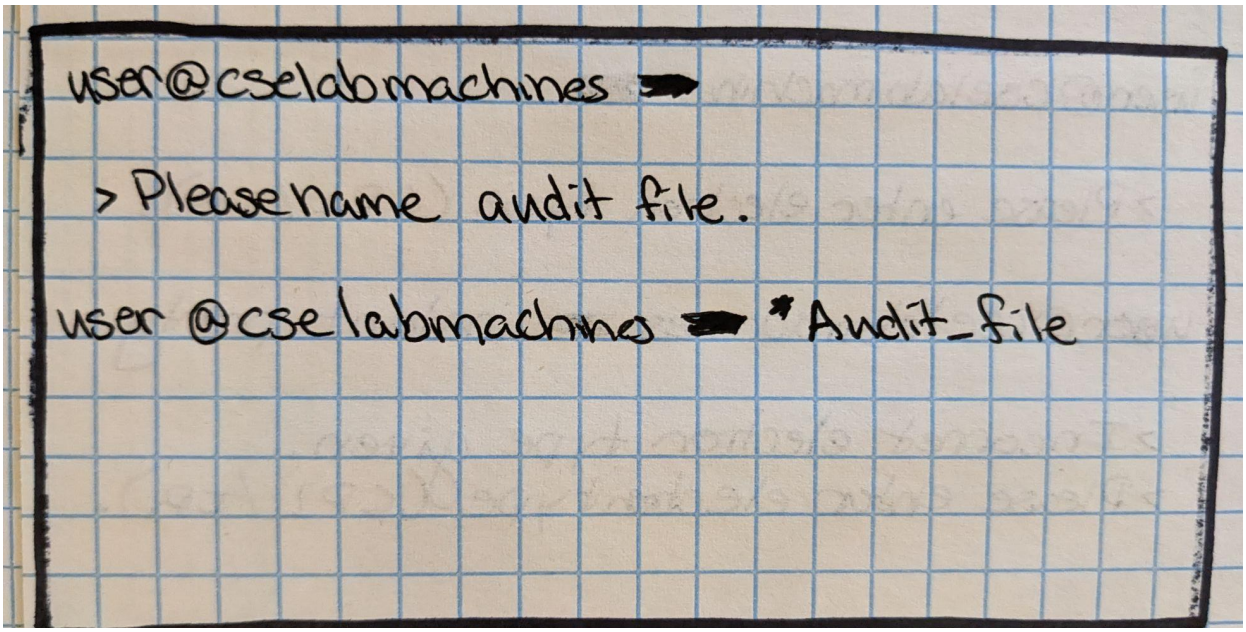


Figure 16: User is prompted to name audit file.

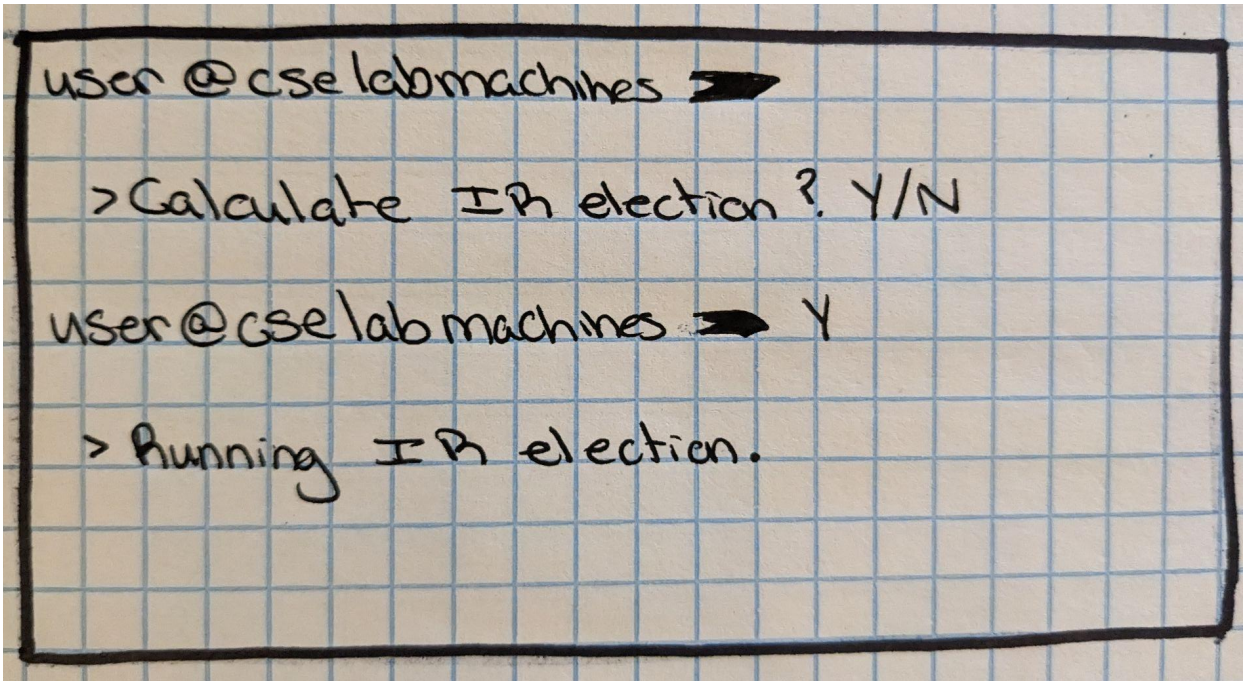


Figure 17: User is prompted to verify the election type.

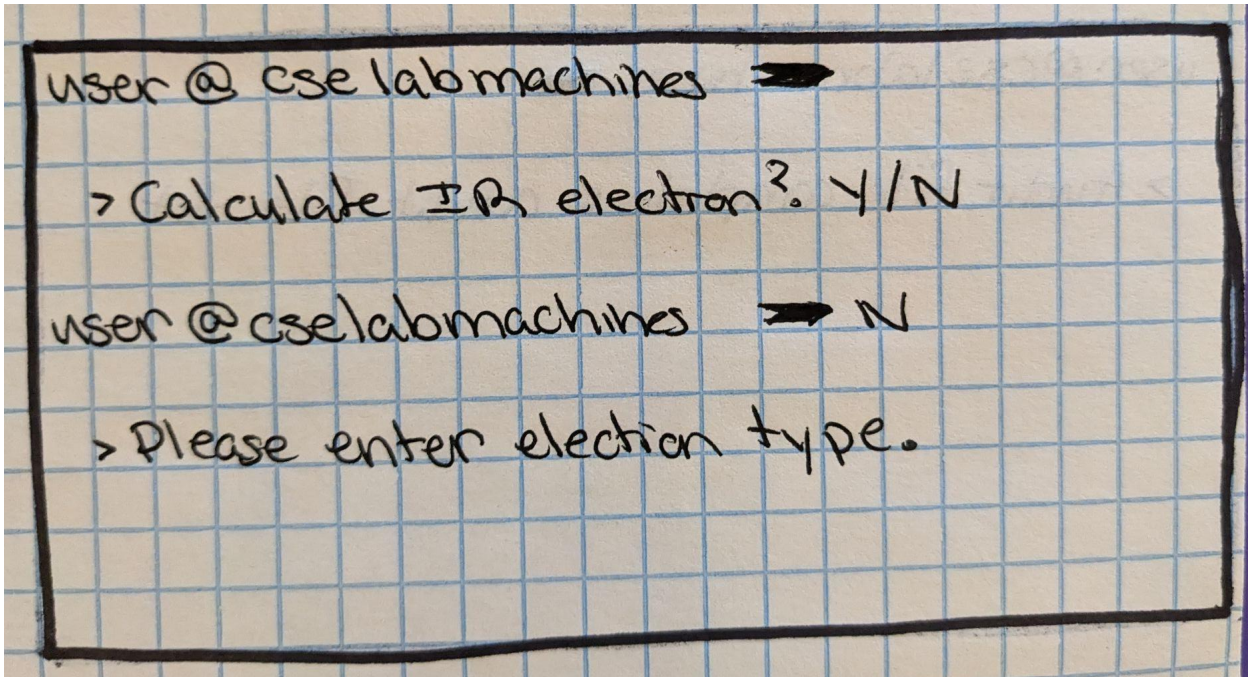


Figure 18: User is prompted to verify the election type. User says no to calculating type.

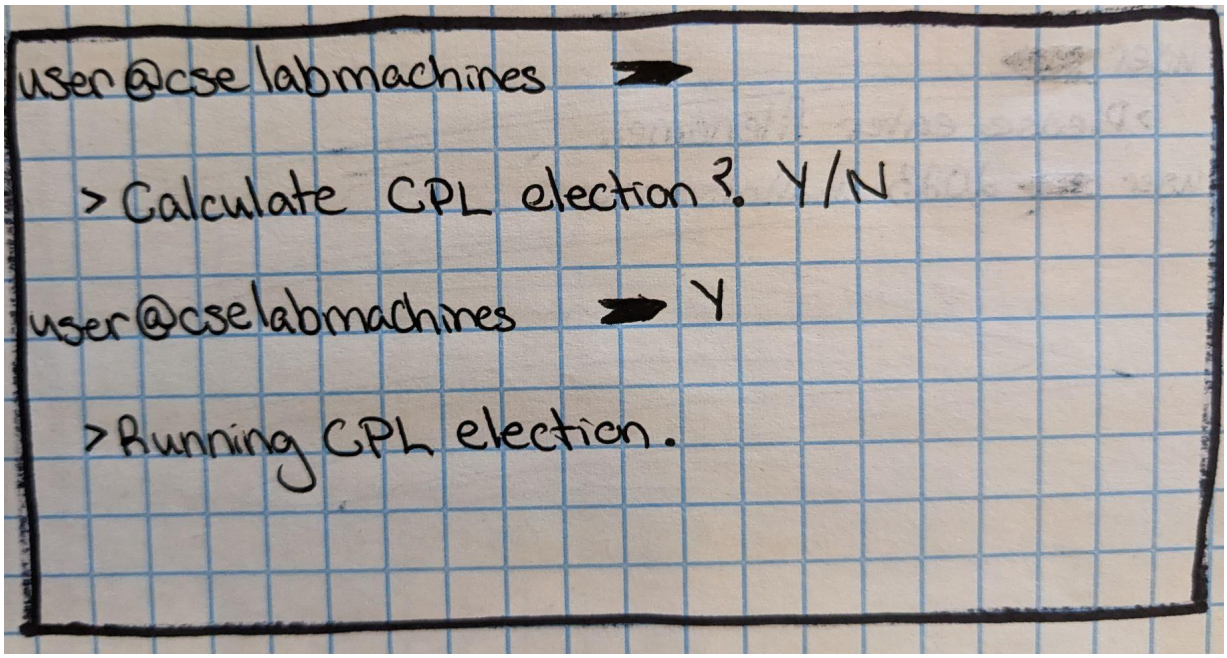


Figure 19: User is prompted to verify the election type.

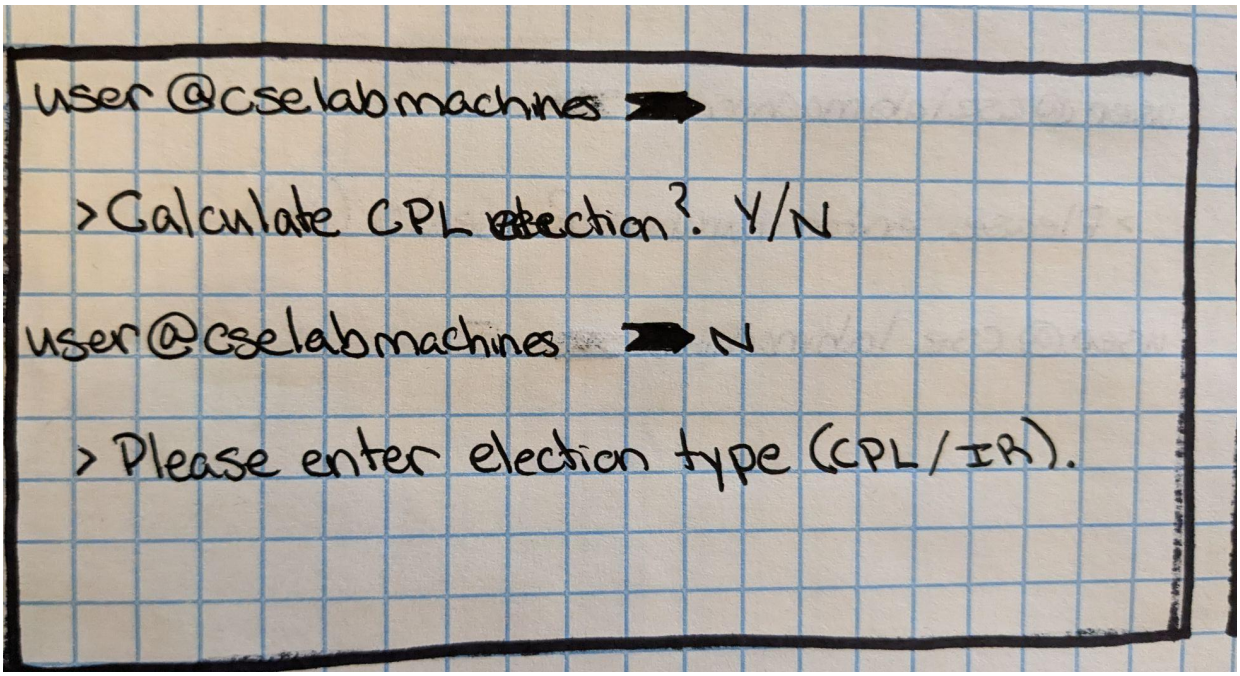


Figure 20 : User is prompted to verify the election type. User says no to calculating type.

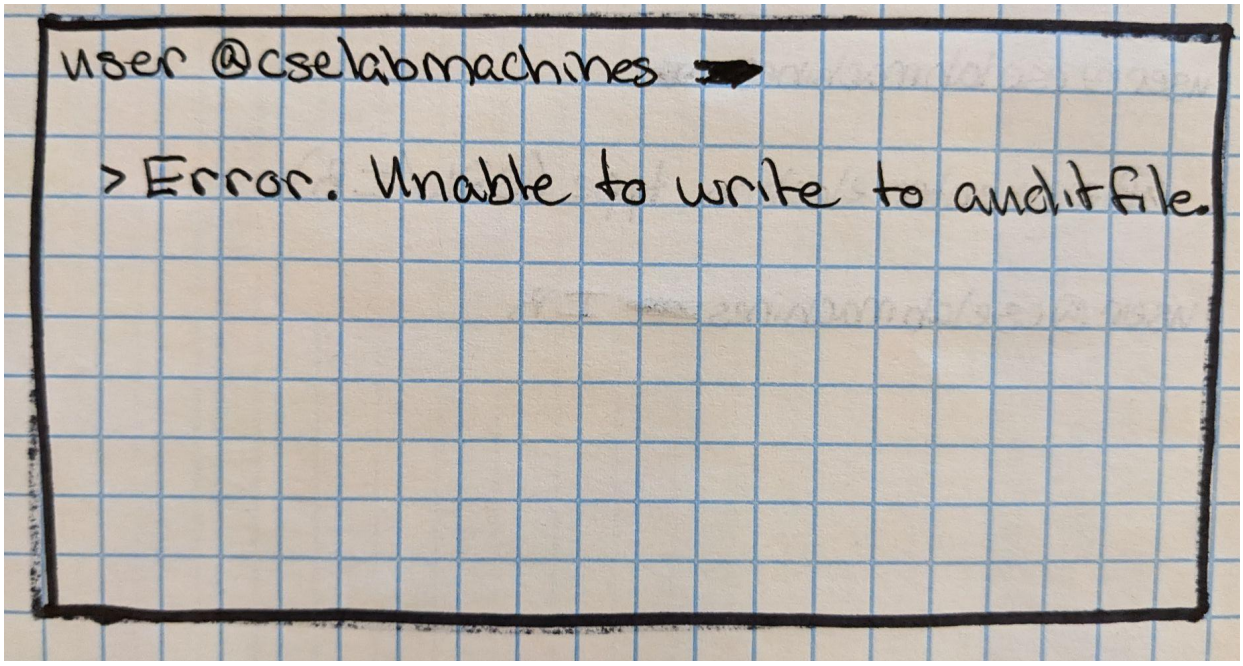


Figure 21: Program is unable to write to audit file and user is notified.

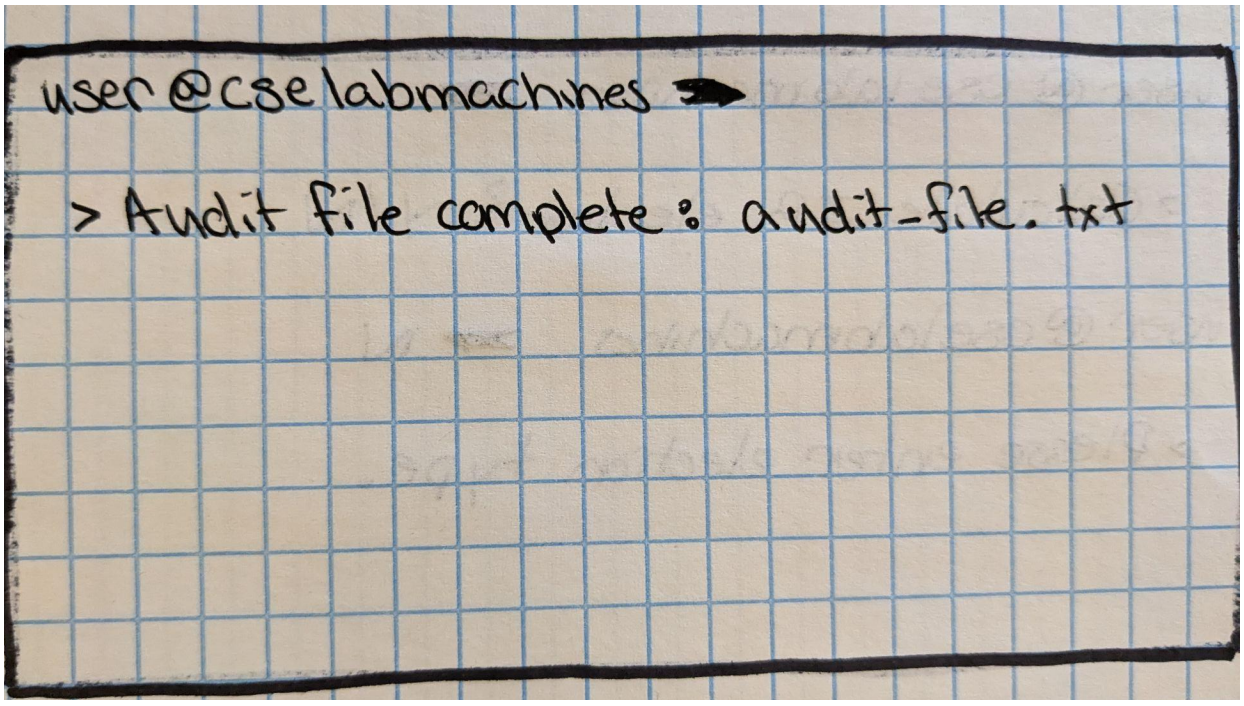


Figure 22: Audit file is complete. User is notified and given location of file.

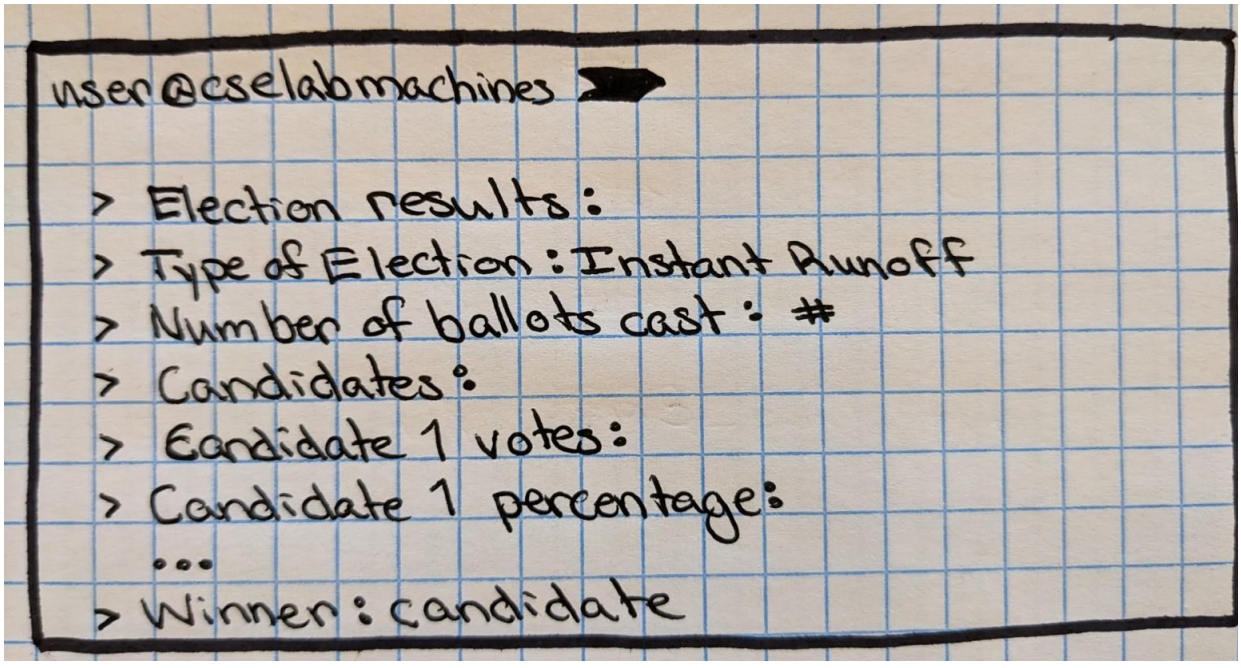


Figure 23: Election has been determined for Instant Runoff and details are printed to the user.

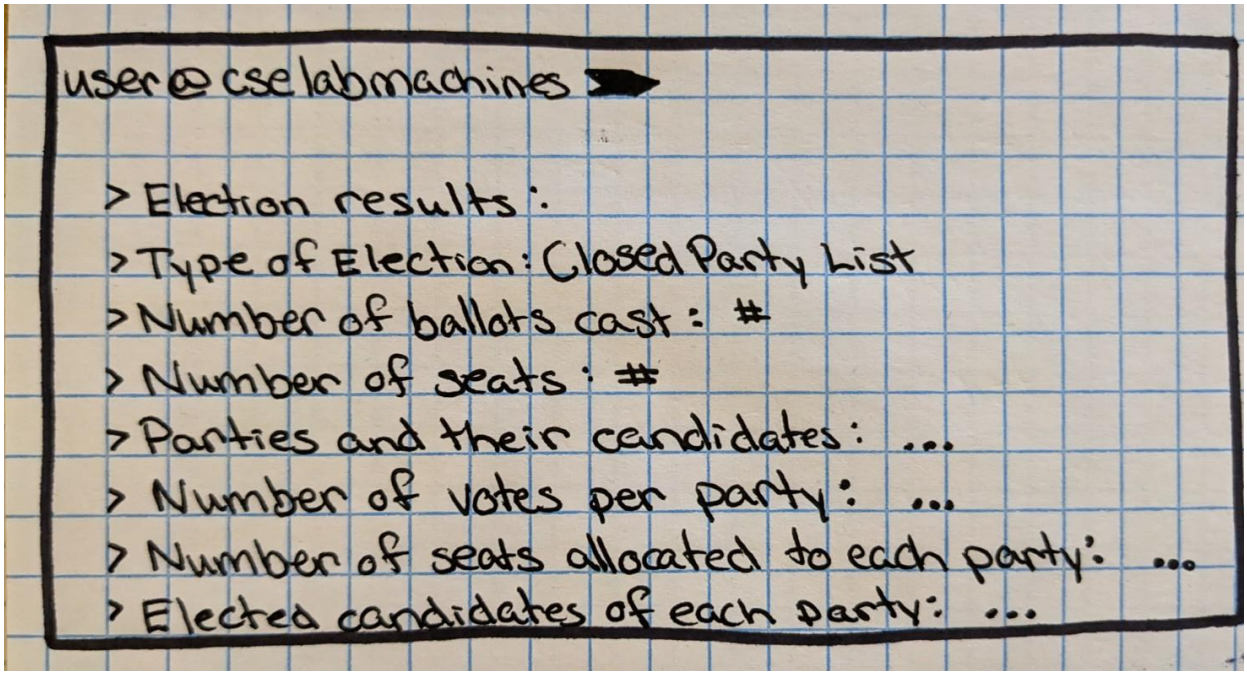


Figure 24: Election has been determined for Closed Party List and details are printed to the user.

6.3 Screen Objects and Actions

The program uses a text based interface and therefore there are no screen objects used.

7. Requirements Matrix

Components accounted for in Section 3 class diagram:

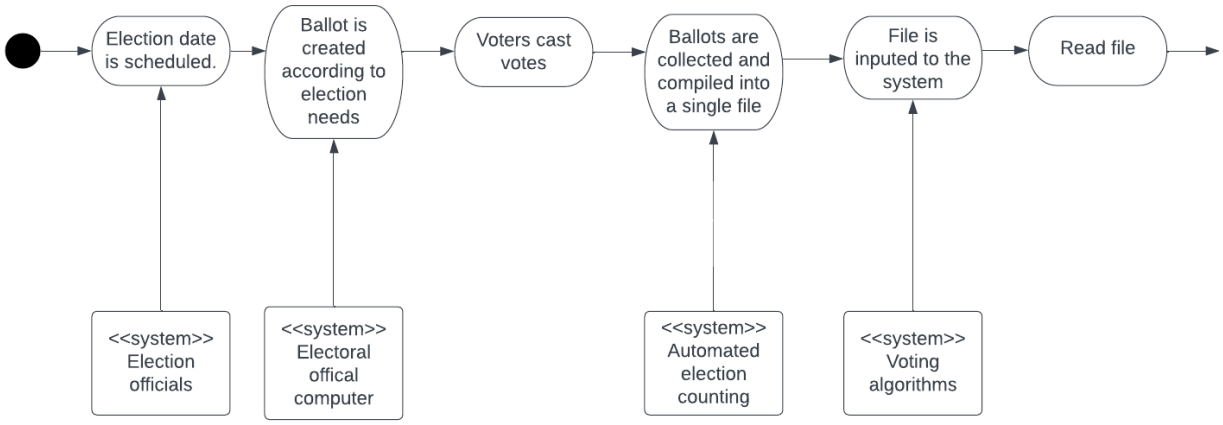
Accepting an Election Data Input File UC_001	After an election has occurred a file with information about the election and the ballot data needs to be uploaded to the program to determine the type of election and the winner of the election. Class diagram: Scanner
Parse Input File UC_002	The system will take the opened input file and parse it for election information it needs to make an election calculation. Class diagram: Scanner

Prompt User for Additional Election Information UC_003	Users will be prompted for any needed information that cannot be extracted from the file. (e.g. number of candidates, number of seats, etc.) Class diagram: Scanner, Election
Verify Voting Type UC_004	A user should be able to verify the type of election that the system is going to compute prior to any actual computation. Class diagram: Scanner
Creating Audit File UC_005	An audit file must be created and named. This audit file will log the calculations of the election computation. Class diagram: AuditFile
Writing to Audit File UC_006	The system will write to the audit file to log election type, election process and election winner. Class diagram: AuditFile
Running Instant Runoff Election Calculation UC_007	Instant runoff voting process will be used to establish a clear majority of the Instant Runoff Election and will log the calculation and results. Class diagram: Ir
Running a Closed Party List Election Calculation UC_008	The Closed Party List (CPL) process will be used to calculate how many seats are allocated to each party. Class diagram: Cpl
Determining seat allocation for Closed Party List Voting UC_009	Closed Party List (CPL) voting will be used to allocate seats appropriately. Class diagram: Cpl
Running Popularity Election Calculation UC_010	A popular vote is used to determine who has the most votes between two people. Class diagram: Ir
Leveling a Tie by Fair Coin Toss UC_011	If a tie between candidates occurs, randomly select the winner in a fair coin toss via the computer. Class diagram: Election {Abstract}
Leveling a Tie With Lottery UC_012	If a tie between 3 or more candidates occurs, randomly select the winner in a lottery process via the computer. Class diagram: Election {Abstract}
Finalize Audit File UC_013	After election data has been processed and a winner has been determined, an audit file outlining the

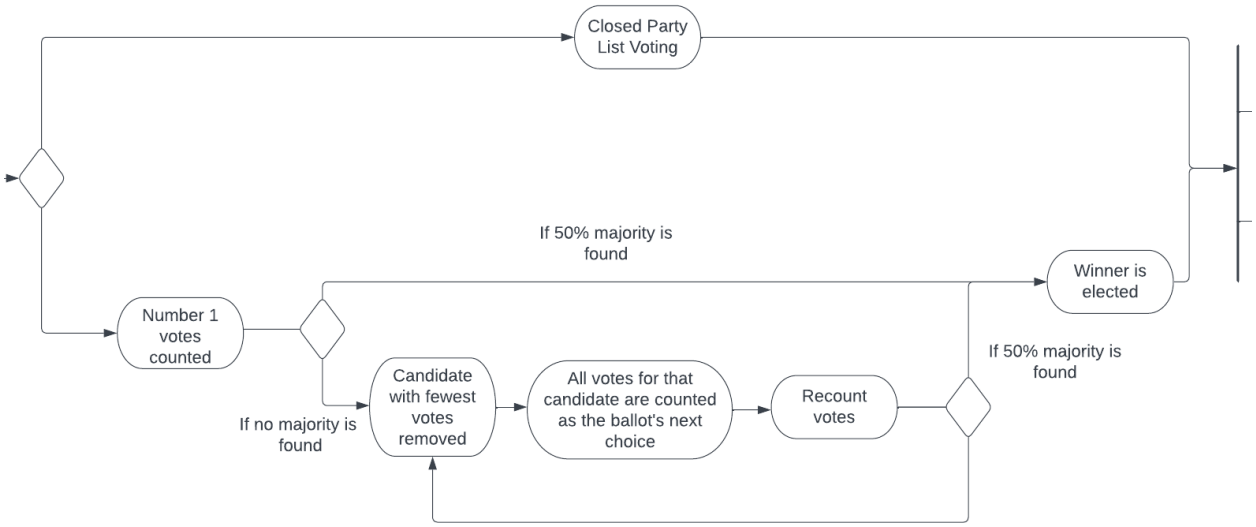
	process of how the winner was determined as well as the final result will be finalized and converted to a read only file. Class diagram: AuditFile
Display Winner and Percentage UC_014	Display to the screen the winner(s) and information about the election (type of election, number of seats, number of ballots cast, winners, number of votes received, percentage of votes received). Class diagram: Result {Abstract}

8. Appendices

Activity Diagram 1:



Activity Diagram 2:



Activity Diagram 3: