Team 3

Voting System

Software Design Document

Name (s): Sidney Huong Nguyen, Chiemeka Nwakama,
Ted Casey, and Phuong Lieu

x500's: nguy4257, nwaka013, casey526, lieu0009

CSCI 5801 Software Engineer I, University of Minnesota

February, 2023

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of the Voting System that is capable of performing IR and CPL voting.

The intended audiences of this document are the Election officials, who will need the result of the winner after election ballots are processed. The testers and developers are audiences with the authorization of how the Voting System will run.

1.2 Scope

This document contains a comprehensive description of the design that will be used to implement the Voting System. The Voting System will support two major voting systems: Closed Party List Voting (CPL) and Instant Run-off Voting (IR). Implementation of both algorithms in our system will allow for the simple way to count ballots in an election since the process will be almost completely automated making it so that the election judges never need to even open up the election file.

All that an election judge will need to do is type in the name of the election file name into the prompt, and the rest will be handled by the Voting System.

Developers and Testers will have full access to the intricacies of the system; however, election judges will only be able to interact with the system via the prompts that it provides. Election Judges will be able to start the election process by inputting the file name. Once the election is done, the results will be displayed to the judges confirming that the election has finished and the results of said election.

1.3 Overview

This document describes an overview of the system, the design considerations leading to the system architecture, and description of the system architecture and design,

The remaining chapters and their contents are listed below.

Section 2 is the System Overview which will describe the Voting System in general and showing the features that will be included into the system. The system is designed to run every time there is an election.

Section 3 is the System Architecture that specifies the design entities that will be used to perform the system. Each section will contain different descriptions of the system that will give a better understanding of the Voting System.

Section 4 is about Data Design or how the data will be used and stored in the system.

Section 5 contains the Component Design, which will include the descriptions of UML diagram, CPL Sequence diagram, and IR Activity diagram.

Section 6 is the Human Interface Design, which will show how the user interacts with the system by giving it commands or automated processes.

Section 7 covers the Requirements Matrix.

Section 8 is Appendices, but it is not necessary for this document or software

1.4 Reference Material

N/A.

1.5 Definitions and Acronyms

Audit file	In this system, an audit file is a text based file containing the generated results of the election
Ballot	A device used to cast votes in an election
Ballot File, Election File	The CSV file that contains the details of the election and the votes. Note “ballot file” and “election file” refer to the same file.
Closed Party Listing Voting (CPLV or CPL)	A type of voting method where voters vote for

	one political party as a whole, rather than individual candidates. The voters do not have influence on which candidates are elected from the winning party.
CSV	Comma-Separated Values file
Class	A block of code that performs a specific function. There may be multiple classes used in combination for larger features.
Election	A formal and organized choice by vote of a person for a political office or other position
Git	A distributed version control system that tracks changes in any set of computer files.
Github	An internet hosting service for software development and version control for Git
IEEE	An acronym that stands for Institute of Electrical and Electronics Engineer. A professional association for electronic engineering and electrical engineering with the mission of advancing technology for the benefit of humanity.
Instant Runoff Voting (IRV or IR)	A type of ranked preferential voting method that uses the majority voting rule, which is the principle that the group with the most supporters is chosen. Voters rank the election candidates in order of preference.
Java	A high level, class based, object oriented programming language that is designed to have as few implementation dependencies as possible. Its latest release is Java SE 19.02
Software Requirements Specification (SRS)	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. This document is a software requirements specification document.

User(s)	Election official, Developer, Tester
Version control	The practice of tracking and managing changes to software code. It is systems of software tools that help software teams manage changes to software code over time.
Waterfall Model	A sequential model that divides software development into pre-defined phases where each phase must be completed before the next phase can begin with no overlap.

2. System Overview

The voting system is equipped to handle both Instant-Runoff and Closed Party List voting making what could be a long, grueling, and error prone process if done by hand, to a seamless, quick, and automated process with this program.

This system is also easily extendable meaning that if there were a need to add an additional voting algorithm, doing so would be relatively quick since all of the underlying infrastructure for the voting system is already there.

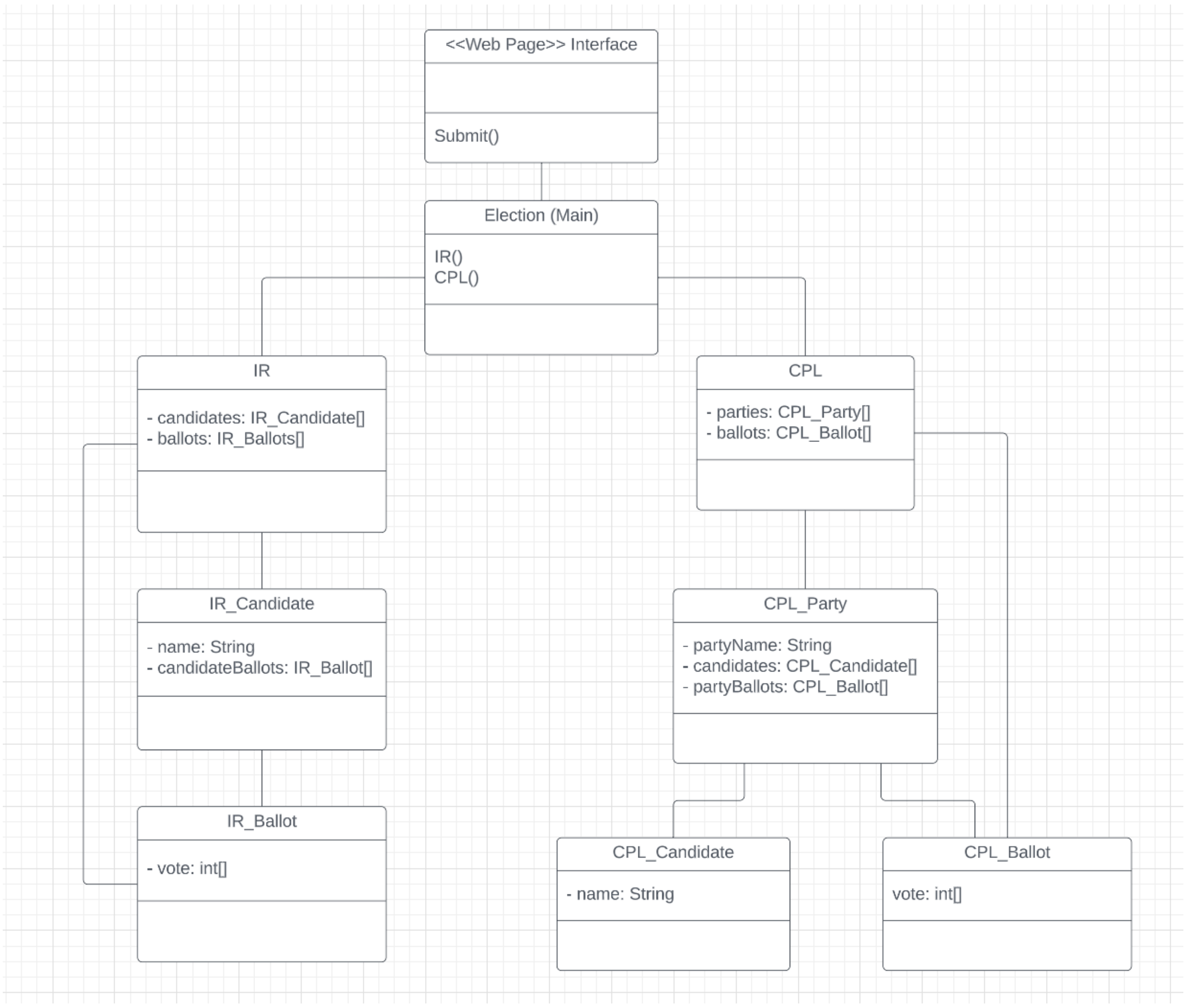
The system is designed to be run one time per election. The system can be run multiple times as many times as needed.

When starting an election, all that is needed by the election official is to input the name of the election file and the voting system will handle the rest, producing an audit file with detailed steps taken during the election process as well as displaying the winner to the screen at the very end of the election.

3. System Architecture

3.1 Architectural Design

3.1.1 Figure 1: UML Diagram of the Voting System



3.2 Decomposition Description

We have multiple objects to keep track of important information throughout the election process. The election file can be input either through the interface or through the command line.

It will then parse the information from the document and populate a number of candidate, ballot, and party objects.

Ballot Objects:

We keep track of these in multiple places, both in the voting algorithm itself and in the candidate or party object. This facilitates counting the votes and adds a

Outside Systems:

There will be outside systems: Election System, Ballot System, CSV Generation System.

These systems maintain the election during its duration, collect the ballots and process them. The Voting System relies on these systems, however, they are outside of this system.

The Voting System does not have access to theses systems

Note that the audit file will be large in size and lengthy. It will be required to have enough space on the user's device to store this file.

3.3 Design Rationale

The main trade off we made was some speed for clarity/accuracy. We created multiple objects, such as ballot or candidate objects, to make sure we were able to easily audit and debug our results.

Another architecture we considered was not having candidate or ballot objects. While this may have sped our program up a bit, we decided that for accuracy and consistency we should have those objects.

We also decided for the same reasons that our audit file was going to be large and comprehensive. While again, we sacrifice some speed, we are able to thoroughly show our process and guarantee accuracy.

Due to the nature of this program, we decided that accuracy was our top priority. Speed is important, and we still prioritize that, but we made design tradeoffs to ensure our program is accurate, even at the cost of some speed.

4. Data Design

4.1 Data Description

In main, we will use a scanner to scan in the file to determine the requirement voting system. Depending on whether it is IR or CPL, then we will call the IR or CPL object for that class.

In the IR class, we will create a IR_Ballot[] to store all of the IR Ballots. We will also have an IR_Candidate that will hold IR Ballots that the candidate has earned. This will be used in determining which ballots need to be redistributed if a candidate loses. An IR_Audit_File will be created to keep track of the process.

In the CPL class, then we will first create a CPL_Ballot[] to store all the ballots in the election. Then a Party[] to store all of the parties in the ballot. Then, we will have a Party class to check for the election process. This will be used to determine the seats in each party. The parties will assign seats based on their list of candidates who will get their seat. An CPL_Audit_File will be created to keep track of the process.

4.2 Data Dictionary

Name	Type
Election (main)	
+ args	string []
+ scanner_obj	scanner
IR	
- audit	IR_Audit_File
+ ballots	IR_Ballot []
- candidates	IR_Candidate []
- candidateBallots	IR_Ballot []

+ numBallots	int
- numCandidates	int
IR_Ballot	
- ballot	int []
- currentCandidate	int
- currentVote	int
IR_Audit_File	
# auditFile	File
CPL	
- audit	CPL_Audit_File
- ballots	CPL_Ballot []
- numBallots	int
- numParties	int
- parties	Party []
- totalSeats	int
CPL_Ballot	
- ballot	int []
- partyVote	int
CPL_Audit_File	
# auditFile	File
Party	
- candidates	CPL_candidates []
- name	string
- remainderVotes	int

- seats	int
- totalVotes	int
- votes	int
Candidate	
# name	string
IR_Candidate	
- ballots	IR_Ballot[]
- status	bool
- voters	IR_Ballot []
CPL_Candidate	
- seats	int

Variables marked with an ‘-’ are private variables and required variables. Variables marked with ‘+’ are public variables, which can be used by other classes/interfaces. Variables marked with ‘#’ are protected variables and will only be used within its own and child classes.

5. Component Design

5.1 Figure 2 UML Diagram Description

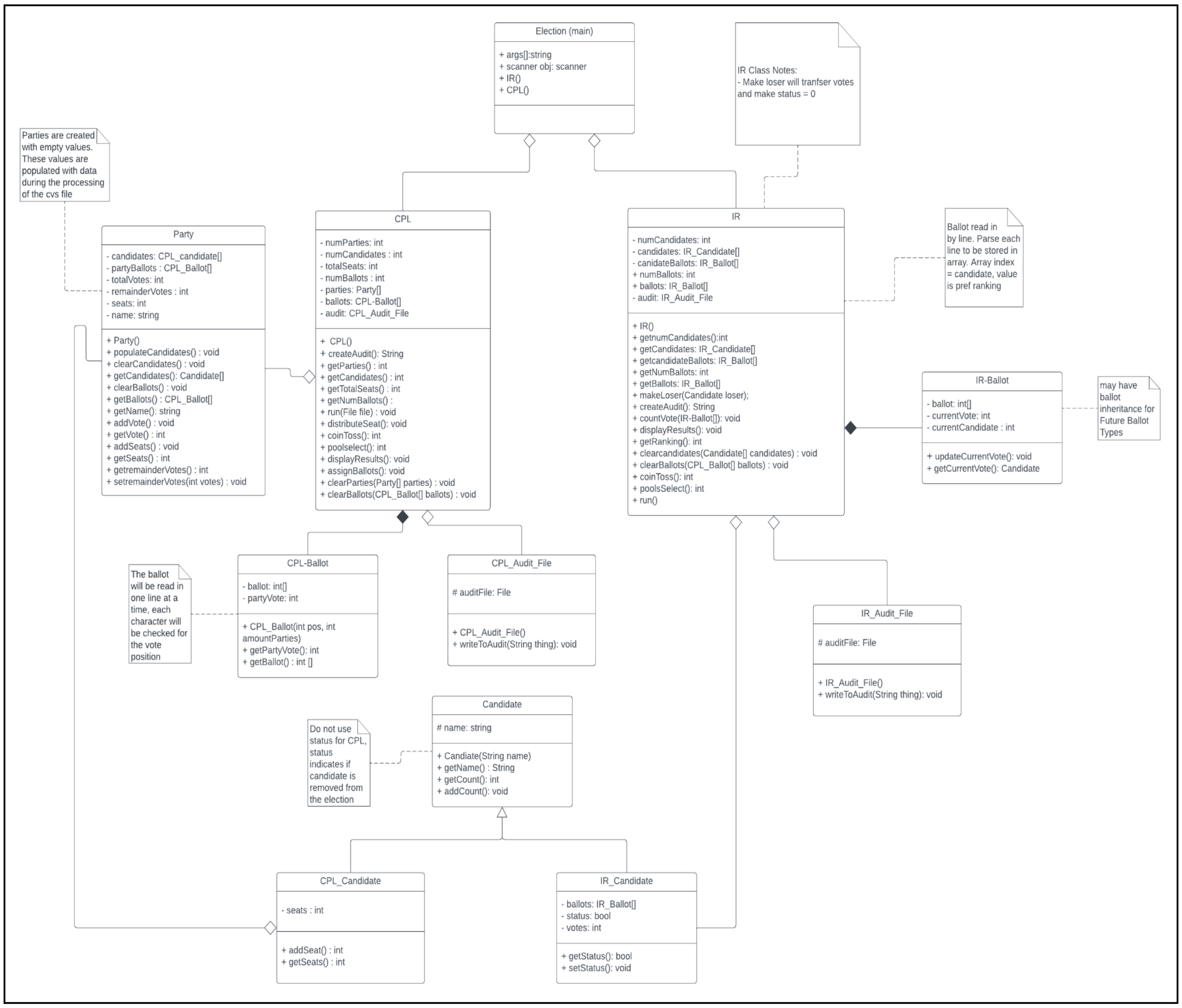
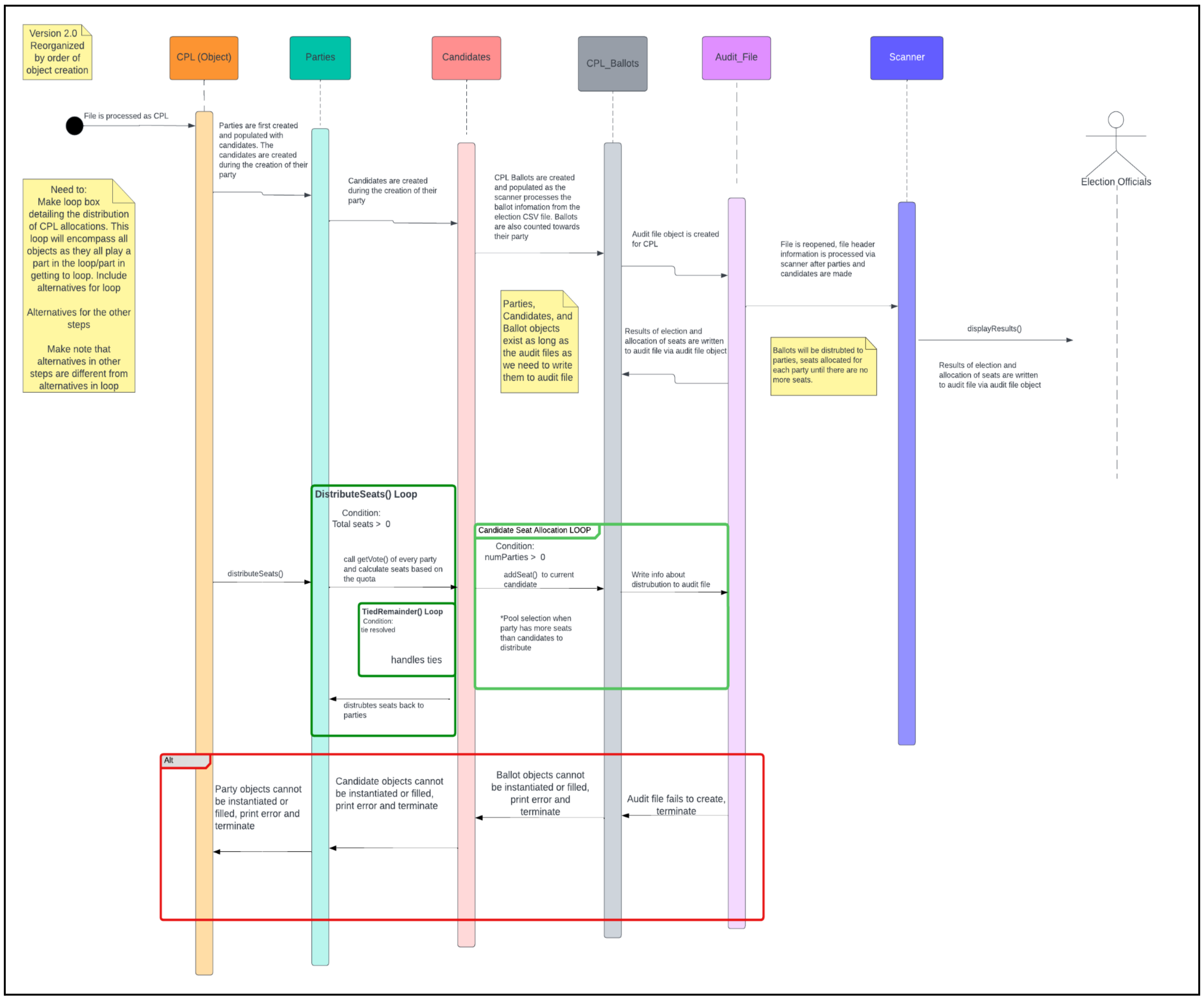


Figure 2. Describes the relationships between the subcomponents of the system. The Voting System contains 10 classes, one of which is an abstraction designed for future extensibility.

5.2 Figure 3 Sequence Diagram Description


Version 2.0
Reorganized by order of object creation
CPL (Object)
Parties
Candidates
CPL_Ballots
Audit_File
Scanner
Election Officials
File is processed as CPL
Parties are first created and populated with candidates. The candidates are created during the creation of their party
Candidates are created during the creation of their party
CPL Ballots are created and populated as the scanner processes the ballot infomation from the election CSV file. Ballots are also counted towards their party
Audit file object is created for CPL
File is reopened, file header information is processed via scanner after parties and candidates are made
Need to: Make loop box detailing the distribution of CPL allocations. This loop will encompass all objects as they all play a part in the loop/part in getting to loop. Include alternatives for loop
Alternatives for the other steps
Make note that alternatives in other steps are different from alternatives in loop
Parties, Candidates, and Ballot objects exist as long as the audit files as we need to write them to audit file
Results of election and allocation of seats are written to audit file via audit file object
Ballots will be distrubted to parties, seats allocated for each party until there are no more seats.
displayResults()
Results of election and allocation of seats are written to audit file via audit file object
DistributeSeats() Loop
Condition: Total seats > 0
distributeSeats()
call getVote() of every party and calculate seats based on the quota
TiedRemainder() Loop
Condition: tie resolved
handles ties
distrubtes seats back to parties
Candidate Seat Allocation LOOP
Condition: numParties > 0
addSeat() to current candidate
Write info about distrubution to audit file
*Pool selection when party has more seats than candidates to distribute
Alt
Party objects cannot be instantiated or filled, print error and terminate
Candidate objects cannot be instantiated or filled, print error and terminate
Ballot objects cannot be instantiated or filled, print error and terminate
Audit file fails to create, terminate

Finger 3. Sequence diagram for the running of the closed party listing (CPL) election: is from the moment you start processing the file itself for information and ballots to the end of declaring the winner(s)

5.3 Figure 4 Activity Diagram Description

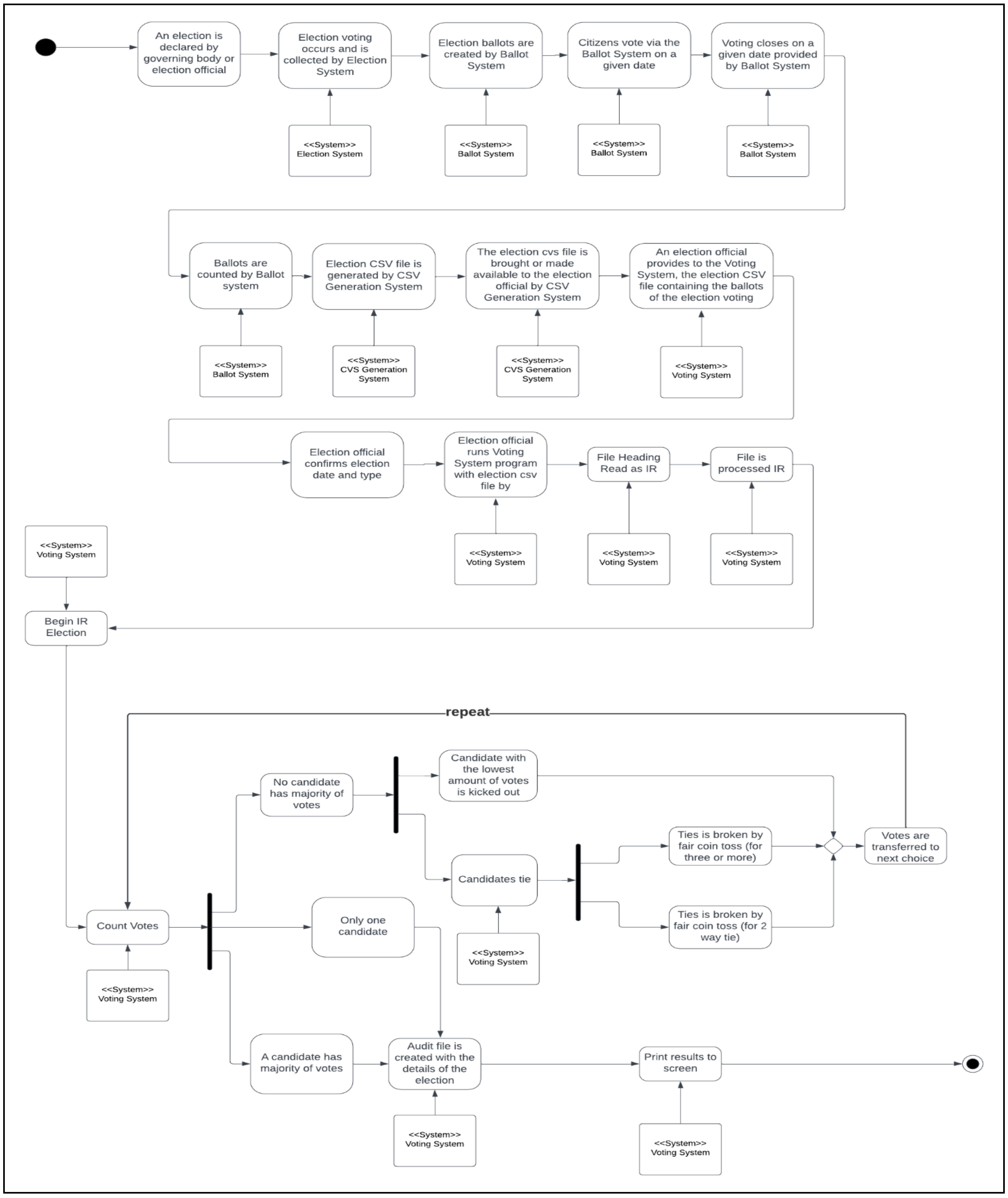

An election is declared by governing body or election official
Election voting occurs and is collected by Election System
<<System>> Election System
Election ballots are created by Ballot System
<<System>> Ballot System
Citizens vote via the Ballot System on a given date
<<System>> Ballot System
Voting closes on a given date provided by Ballot System
<<System>> Ballot System
Ballots are counted by Ballot system
<<System>> Ballot System
Election CSV file is generated by CSV Generation System
<<System>> CVS Generation System
The election cvs file is brought or made available to the election official by CSV Generation System
<<System>> CVS Generation System
An election official provides to the Voting System, the election CSV file containing the ballots of the election voting
<<System>> Voting System
Election official confirms election date and type
Election official runs Voting System program with election csv file by
<<System>> Voting System
File Heading Read as IR
<<System>> Voting System
File is processed IR
<<System>> Voting System
<<System>> Voting System
Begin IR Election
repeat
Count Votes
<<System>> Voting System
No candidate has majority of votes
Candidate with the lowest amount of votes is kicked out
Candidates tie
<<System>> Voting System
Ties is broken by fair coin toss (for three or more)
Ties is broken by fair coin toss (for 2 way tie)
Votes are transferred to next choice
Only one candidate
A candidate has majority of votes
Audit file is created with the details of the election
<<System>> Voting System
Print results to screen
<<System>> Voting System

Figure 4. Activity diagram for the Instant Runoff (IR) election: from start to finish of declaring a winner. Starts at the "election has been set/called for a given date" and ends at "we have a winner."

5.4 Subcomponent Algorithm Design

5.4.1 Main

```
if user did not enter file by command (argument count)
    prompt user to enter file name
    create scanner object and scan in file name

open file and read first line
if first line = IR
    Call IR algorithm
else if first line = CPL
    Call CPL algorithm
else
    error
```

5.4.2 CPL

Constructor: Initialize values to empty

Create Scanner

Create Audit_File (auditing takes place after each step take via writeToAudit())

Open file

Scanner scans in starting at the 2nd line list of parties, how amount of seats, candidates in each party, and each ballot one by one and stores them into respective object instances

Gets votes from each party and starts distributing seats based on party votes/quota with the remainder that each party has (anything left over after a modulus) based on order of highest remainder seats in each party

```
if two or more parties have = remainder and there is a seat dispute
        coinflip or pool selection to determine what party gets seat
after parties all have seats now distribute seats to candidates
while distributing seats to candidates, do in order of candidates one by one
        if there are more seats to give out after giving seats to all 5 candidates (could be less):
                do pool selection to give out each additional seat
Once all seats are given to candidates audit file is finished being written to
```

Display results;

5.4.3 Party

Constructor: Initialize values to empty

During the processing of the file for information

Set name, candidates, party ballots

5.4.4 CPL_Candidate

Constructor: Initialize values to empty

Read in line, parsed by commas

For each token, assign Candidate object and set name

5.4.5 CPL_Ballot

Constructor: Initialize values to 0

Set partyVote to party position in file

Read in each line and search each character for a 1

Use the location of the 1 to designate where in the ballot variable, the vote should be

5.4.6 CPL_Audit

Constructor: Initialize values to empty

Write in the date, parties, name, etc.

Write into the file a new line for each result at the end of each round in the CPL election.

5.4.7 Candidate

This is an abstraction, so IR_Candidate and CPL_Candidate must include this in their instantiations

Constructor: Initialize values to empty

Set parameter to name

5.4.8 IR

Constructor: Initialize values to empty

Create Scanner

Create Audit file

```
 Open file
 while file descriptor is not at the end of the file
        set numCandidates, numBallots
        read in ballot line, assign to candidateBallots variable
        Locate the first choice and assign ballot to that candidate's ballot variable
        Add vote to that candidate

Once ballots have been assigned and votes counted, write initial votes to audit file
run algorithm

run algorithm:
If there is only one candidate
  Audit file details winner
  Print results to screen
  End program
Else

While 1
If no candidate has majority
        If there is no tie
                Candidate with lowest votes is removed from election
                Loop through this candidate's ballot array, for each ballot,
                        if their next choice has not yet been removed, add their ballot to that candidate
                        and add their vote
                        else
                                repeat with next choice
        else  If there is a tie
                implement fair coin toss, losing candidate is removed and their votes are transferred
else
        Candidate has majority vote
        Audit file details winner
        Print results to screen
        End program
```

5.4.9 IR_Candidate

Constructor: Initialize values to empty

will be set later: names, IR_ballots[], status, votes (number)

5.4.10 IR_Ballot

Constructor: Initialize values to empty

will be set later: all IR_Ballots read in and stored, current vote (rank), current candidate

5.4.11 IR_Audit

Constructor: Initialize values to empty

Write in the date, number of candidates, their name, initial of their parties, number of seats, etc.

Write into the file a new line for each result at the end of each round in the CPL election.

6. Human Interface Design

6.1 Overview of User Interface

When the Voting System is running, there will be a display for the election judge to prompt in the election file name. Then, the system will automatically calculate the result based on the voting system type. There will be a display pop up to show the results of the elections. The result will be displayed with the winner(s) and information about the election.

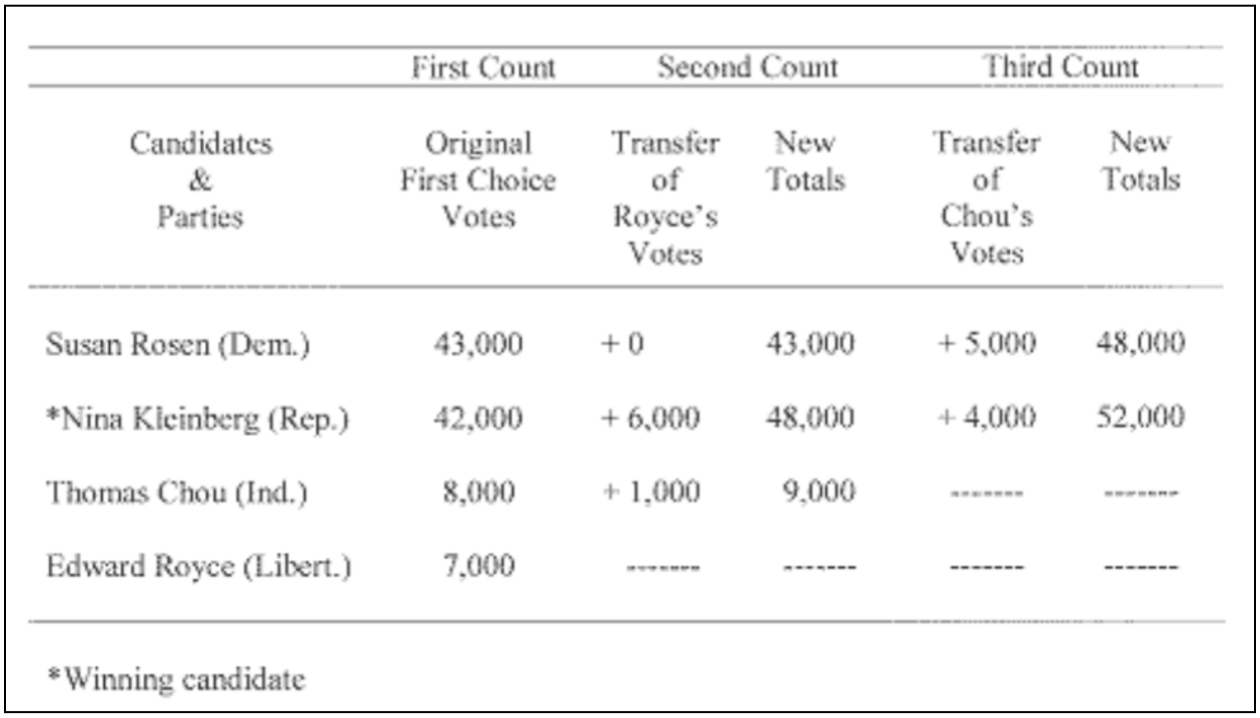
6.2 Screen Images

Prompt Display:

Enter the election file:

Result Display:

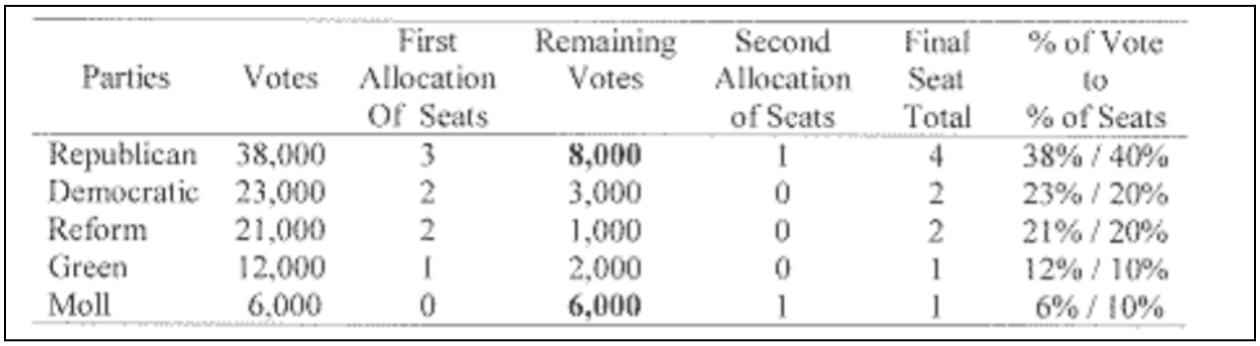
IR



	First Count	Second Count		Third Count	
Candidates & Parties	Original First Choice Votes	Transfer of Royce's Votes	New Totals	Transfer of Chou's Votes	New Totals
Susan Rosen (Dem.)	43,000	+ 0	43,000	+ 5,000	48,000
*Nina Kleinberg (Rep.)	42,000	+ 6,000	48,000	+ 4,000	52,000
Thomas Chou (Ind.)	8,000	+ 1,000	9,000	-------	-------
Edward Royce (Libert.)	7,000	-------	-------	-------	-------

*Winning candidate

CPL



Parties	Votes	First Allocation Of Seats	Remaining Votes	Second Allocation of Seats	Final Seat Total	% of Vote to % of Seats
Republican	38,000	3	**8,000**	1	4	38% / 40%
Democratic	23,000	2	3,000	0	2	23% / 20%
Reform	21,000	2	1,000	0	2	21% / 20%
Green	12,000	1	2,000	0	1	12% / 10%
Moll	6,000	0	**6,000**	1	1	6% / 10%

6.3 Screen Objects and Actions

The user just needs to choose whether they want to manually enter in a file name or has the system automatically do it.

7. Requirements Matrix

Use Cases	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
UML Class Diagram	X					X		X			X	X	X	X	X
UML Activity Diagram	X	X	X	X		X	X		X	X	X		X		X
UML Sequence Diagram					X			X	X	X		X	X	X	X

Use Cases:

UC_001: Start program

UC_002: Prompt user for information

UC_003: File identification - file exist

UC_004: Processing IRV Vote File

UC_005: CPL Processing of the CSV file

UC_006: Instant Runoff Voting Algorithm

UC_007: Instant Runoff No Clear Winner

UC_008: Closed Party List Algorithm

UC_009: Coin Toss

UC_010: Pool Selection

UC_011: Create Audit File - IR

UC_012: Create Audit File - CPL

UC_013: Update Audit File

UC_014: Naming/storage scheme of Audit file

UC_015: Display Election Results

8. APPENDICES

N/A