Voting System
Software Design Document

Name (s): Nikhil Srikanth, Peter Linden, Alex Bohm, Andrew Petrescu

Date: 02/28/2021

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The purpose of this software design document is to illustrate the overall architecture of the Voting System, as well as the process flows of counting votes for the Instant Runoff and Open Party List voting methods. This document is intended for those who are interested in maintaining this software in the future or testers who wish to ensure features are working as intended.

1.2 Scope

The Voting System will be able to determine the results of an Instant Runoff or Open Party List election within 8 minutes. It will use the information from a final ballot file (which includes important information and votes from the election) in order to determine which process flow it must follow in order to achieve the desired outcome. For media representatives and auditors, separate files will be generated in order to meet their needs. This system benefits election officials and those who are interested in the outcome of an election as it will accurately determine the winner as well as provide some useful metrics.

1.3 Overview

This document provides the necessary information regarding the Voting System's overall architecture as well as the design and flows of the important software components. An overview of the system is provided first followed by the system architecture and the design of the important components. The document finishes with the design of the interface used for the system and a requirements matrix which maps the design choices to the requirements. Section titles and subtitles are bolded and larger in font with the content that follows in a regular format.

1.4 Reference Material

https://github.umn.edu/umn-csci-5801-S21-001/repo-Team20/blob/master/SRS/SRS_Team20.pdf

1.5 Definitions and Acronyms

- IR - Instant Runoff
- OPL - Open Party List

- CSV - Comma separated values
- SDD - Software Design Document
- BID - Individual Ballot ID
- CHOICE - Candidate selected for individual ballot
- CTALLY - Count of ballots associated with individual candidate
- PTALLY - Count of ballots associated with individual party

2. SYSTEM OVERVIEW

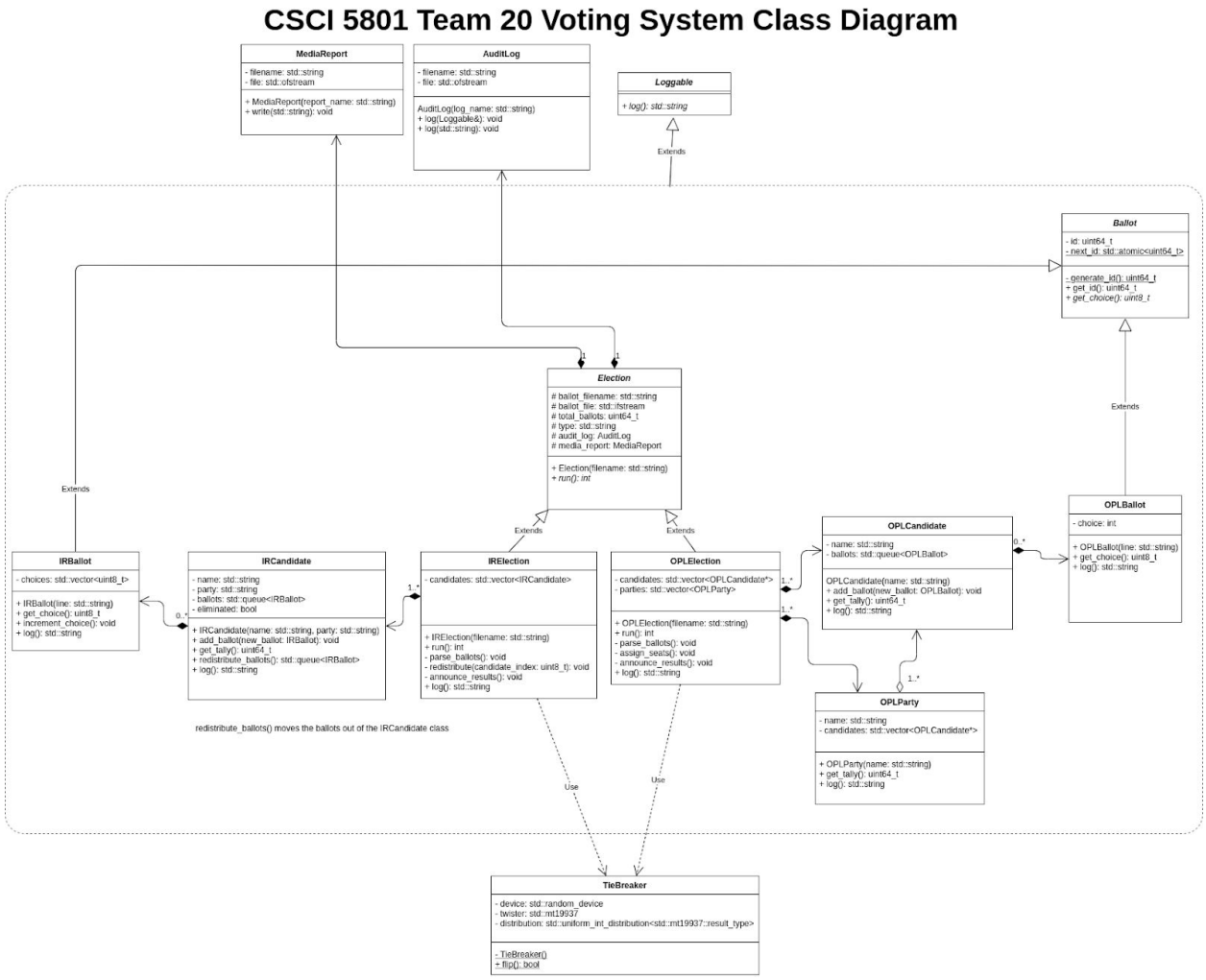
The Voting System is used by election officials to receive results from open party list (OPL) and instant runoff (IR) elections. The Voting System we create receives comma separated value (CSV) ballots which contain information regarding the election. When the type of election is selected, a specific algorithm will be run by the system that will analyse the ballots and retrieve information from them to produce results for the election. When the algorithm is calculating the winner of the election, the system will periodically output results to see the step by step process of how the winner was selected. The system will finally output an audit file which will contain information of the winner as well as more in depth information containing step by step processes of how the system selected the winner.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

Class Diagram

Refer to `voting_system_class_diagram_team20.pdf`. A low resolution PNG has been included for quick reference.



Loggable Class

The Loggable class is an abstract class that specifies that any inheriting class shall contain a log() method that produces necessary information about that class as a string.

AuditLog Class

The AuditLog class is a class that contains all the methods necessary for producing an audit log. It manages opening, writing, and closing the audit log file. The Loggable class is used in order to provide the AuditLog class with a simple interface where any Loggable object can be passed into the AutditLog's log(Loggable&) method. This enforces classes having a dedicated way of logging activity.

Election Class

The Election class is a base class that holds common information about an election. In our software two election types inherit from this base class and have different behavior based on their individual algorithms.

Ballot Class

The ballot class is an abstract base class that outlines some basic methods of how a ballot works. It handles ID generation automatically on construction. Each subclass takes in a line in the ballot file and parses it into the appropriate indices of candidates. The candidate choice can then be retrieved with get_choice(). Each inheriting ballot class slightly modifies what get_choice() returns.

IRElection Class

This class runs the logic for an independent runoff election. It holds a dynamic array of IRCandidate. As ballots are parsed from the ballots file, the IRElection class distributes ballots into the array of IRCandidate.

OPLElection Class

This class runs the logic for an open party listing election. As the OPLElection class parses ballots from the ballot file, it distributes ballots into specific candidates.

OPLParty Class

This class holds a list of references to the candidates for each party. This class supplies some basic helpers to determine which classes have a candidate.

OPLCandidate Class

The OPLCandidate class is a container for ballots.

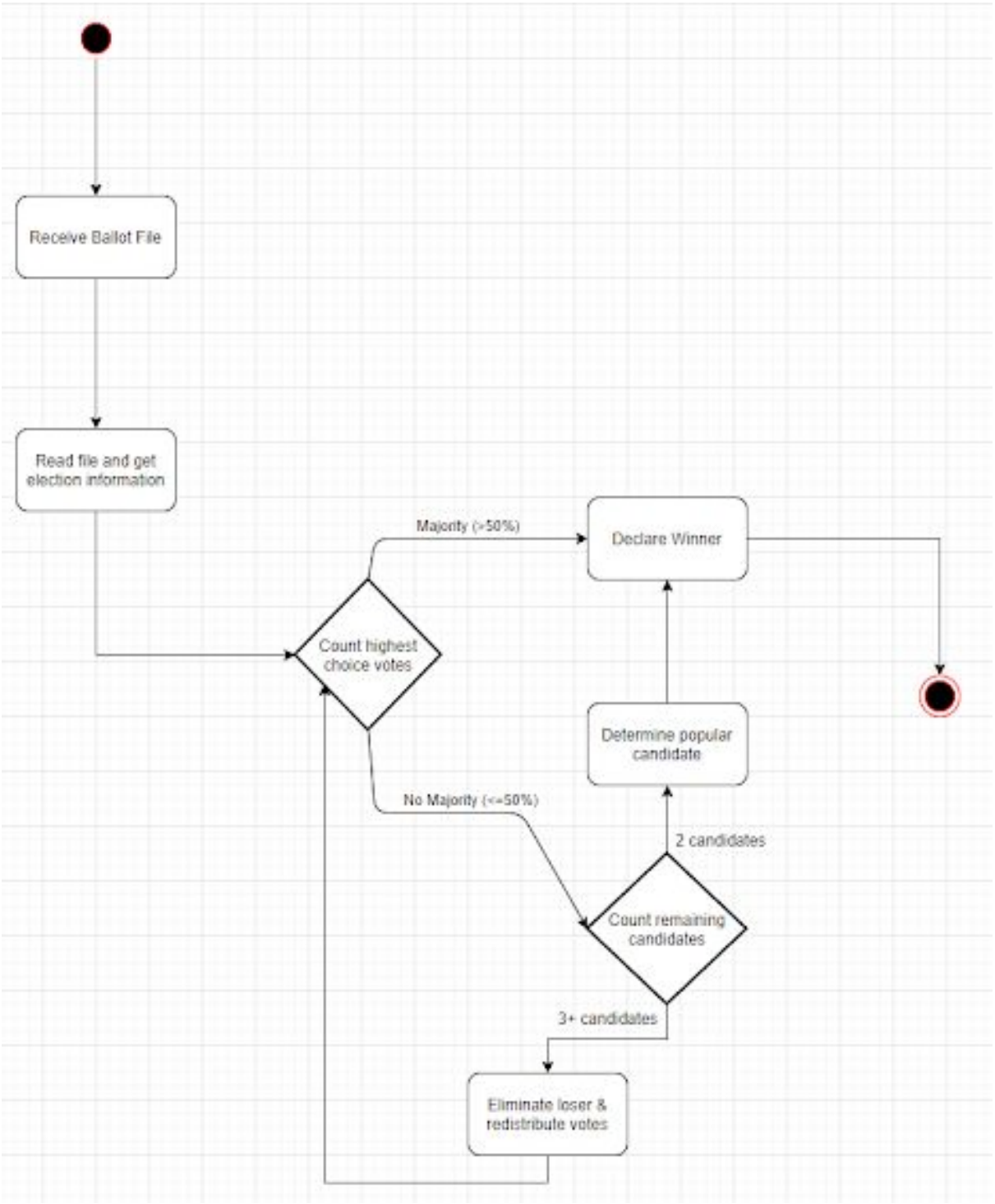
TieBreaker Class

This class provides a static method for breaking ties. It either returns true or false randomly.

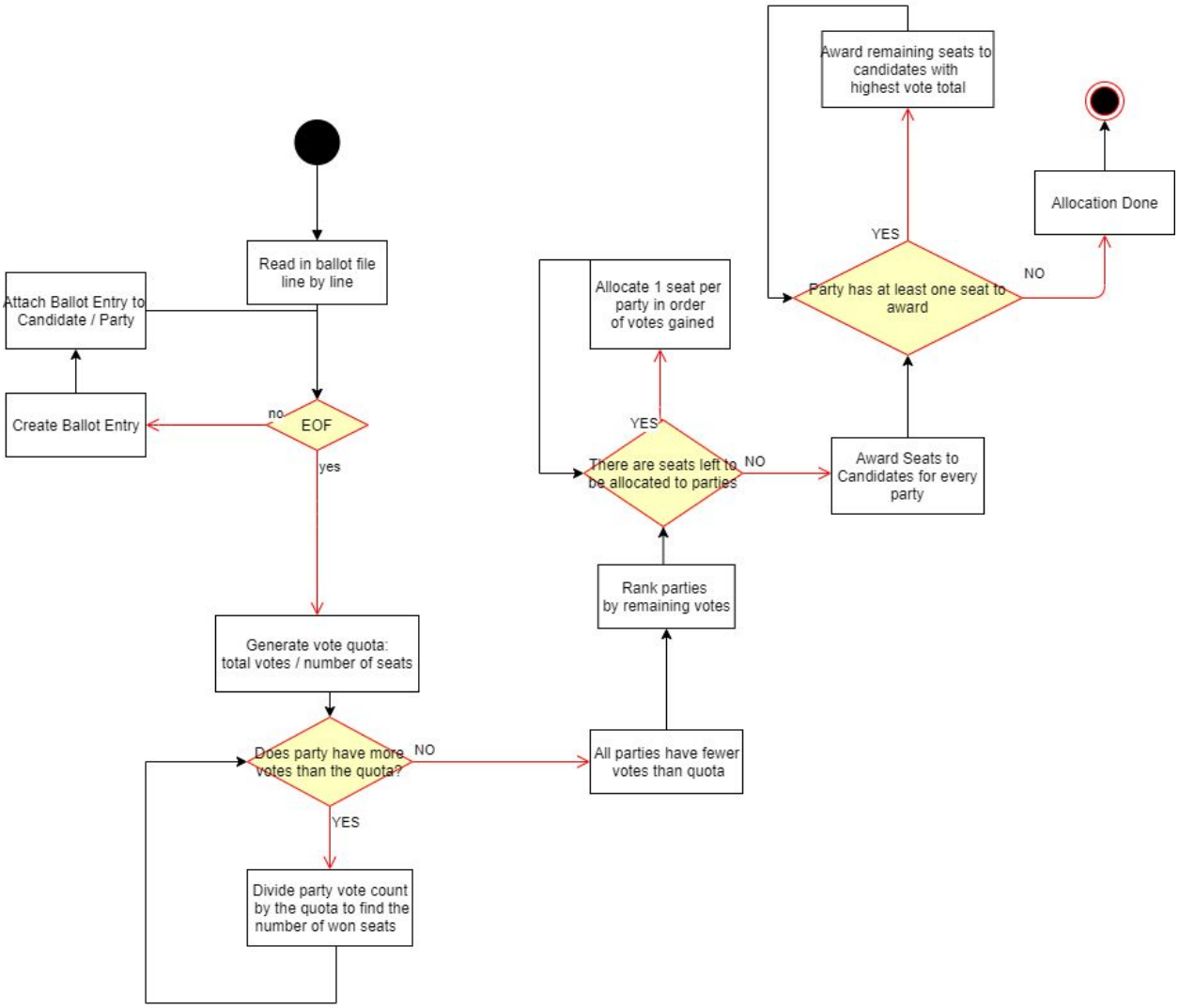
MediaReport Class

The MediaReport class handles writing a media report file. During construction and destruction of the class the file is properly opened and closed.

3.1.1 IR Activity Diagram

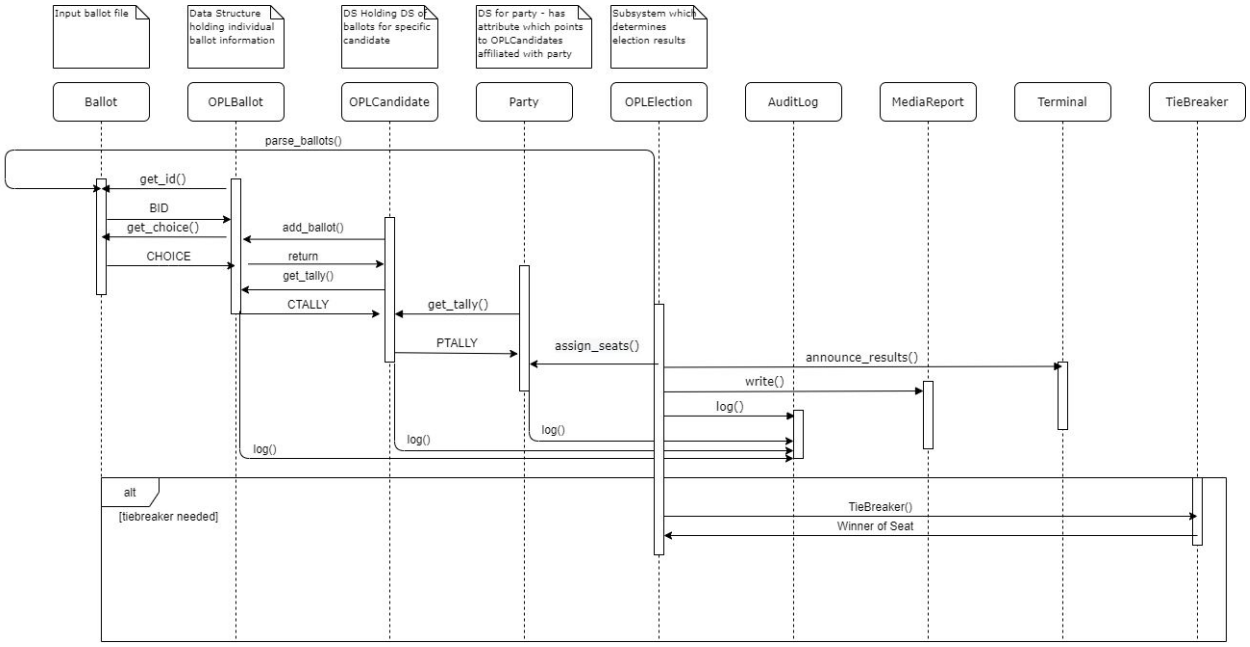


3.1.2 OPL Activity Diagram



3.2 Decomposition Description

3.2.1 OPL Sequence Diagram



3.3 Design Rationale

The main subsections were decided to be the ballot file, and data structures for ballot entries, candidate vote tallies, and party vote tallies, and a class used to run each election type. This categorization would allow for an intuitive flow of information from the imported CSV file data to the individual candidate vote totals and party totals, as the information in the previous level of organization would fold into the next level of organization (the candidate data structure would hold the list of ballot entries, and the party data structure would do the same with individual candidate tallies). The architecture was designed so that both election types would follow similar data hierarchies (ballot:candidate:election data structures), which streamlines design and forces us to come up with solutions that can be utilized by both election processes. Each path culminates in an election subsection which has access to all the data from the ballots, and follows the correct election methodology - this organization decision was made with the intent of compartmentalizing the actual vote counting and allocation - the initial data structures will consolidate the data, and the election subsystem will utilize it to generate a result.

The structure of the Ballot classes had to be considered carefully. The current method stores indices of candidates. This makes the size of each ballot class relatively small in memory, however it adds complexity to the candidate list. The candidates must stay in the exact order that they are in the file. We can see the effects of this decision in the IRCandidate class with a "eliminated" boolean field. Despite using dynamic arrays, we cannot remove a candidate since we rely on their index not changing.

4. DATA DESIGN

4.1 Data Description

The information from our system is parsed by the CSV and transformed in the following categories: Election System, Ballot, and Candidate. The approach we chose to use is to "parse as we go" rather than parsing the entire CSV at the beginning. An audit log is used to show all steps of the election as well as the winner. The major data structure is the election system, where many of the important elements of the election and ballot information are created here. This information is used by the Ballot and candidate classes in order to determine the election type to be running as well as keeping ballot information separate from one another.

4.2 Data Dictionary

Getter methods for the following objects and classes are assumed to be present when necessary.

Election System

Item	Type	Description
ballot_filename	std::string	Name of ballot input file
ballot_file	std::ifstream	Input file stream handler for interacting with the OS
Election()	filename: std::string	Base constructor for elections (used/modified by OPL and IR)
run()	int	Base run function for elections (used/modified by OPL and IR)
total_ballots	uint64_t	Number of ballots total in the election
type	std::string	Type of election taking place
audit_log	AuditLog	Creates an audit log
media_report	MediaReport	Creates a media report

OPL:

Item	Type	Description
announce_results()	void	Displays results of OPL election
assign_seats()	void	Assigns a seat to a candidate based on votes
candidates	std::vector<OPLCandidate>	All candidates in the election
log()	std::string	Logs election information to AuditLog object
OPLElection(filename: std::string)	constructor	Constructor for OPLElection objects
parse_ballots()	void	Parses through ballot file and loads ballots into candidates
parties	std::vector<OPLParty>	All parties in the election
run()	int	Runs an OPL Election instance

IR:

Item	Type	Description
announce_results()	void	Displays results of IR election
candidates	std::vector<IRCandidate>	All candidates in the

		election
IRElection(filename: std::string)	constructor	Constructor for IRElection objects
log()	std::string	Logs election information to AuditLog object
parse_ballots()	void	Parses through ballot file and loads ballots into candidates
redistribute(uint8_t)	void	Initiates the redistribution process
run()	int	Runs an IR Election instance

Tie Breaker

Item	Type	Description
device	std::random_device	Random device of the machine. Note: may be pseudorandom if no physical random device is present.
distribution	std::uniform_int_distribution <std::mt19937::result_type>	A uniform distribution between 0 and 1 representing two sides of a coin (lose/win)
flip()	bool	Initiates coin flip
TieBreaker()	constructor	Private constructor for TieBreaker object
twister	std::mt19937	A pseudorandom number generator.

Ballot

Item	Type	Description
generate_id()	uint64_t	Generates unique id for each new ballot.
id	uint64_t	A system generated unique id to distinguish every ballot
next_id	std::atomic<uint64_t>	The id of the next ballot

OPL Ballot:

Item	Type	Description
choice	int	Increments the number of ballots the candidate receives
log()	std::string	Base log function used/modified by other classes
OPLBallot(line: std::string)	constructor	Constructor for OPLBallot object

IR Ballot:

Item	Type	Description
choices	std::vector<uint8_t>	List of votes that the candidate received.

increment_choice()	void	Another choice is added
log()	std::string	Logs ballot information to AuditLog object
IRBallot(line: std::string)	constructor	Constructor for IRBallot object

OPL Party:

Item	Type	Description
candidates	std::vector<OPLCandidate>	Names of the candidates in a given party
log()	std:string	Logs party information to AuditLog object
name	std::string	Name of the party
OPLParty(name: std::string)	constructor	Constructor for OPLParty object

Candidate

OPL Candidate:

Item	Type	Description
ballots	std::queue<OPLBallot>	Number of ballots total
add_ballot(OPLBallot)	void	Another ballot gets added
log()	std::string	Logs candidate information

		to AuditLog object
name	std::string	Name of the candidate
OPLCandidate(name: std::string)	constructor	Constructor for OPLCandidate object

IR Candidate:

Item	Type	Description
add_ballot(IRBallot)	void	Another ballot gets added
ballots	std::queue<IRBallot>	Number of ballots total
eliminated	bool	Ballots that are eliminated due to candidates not having a majority
IRCandidate(name: std::string, party: std::string)	constructor	Constructor for IRCandidate object
log():	std::string	Logs IR candidate information to AuditLog object
name	std::string	Name of the candidate
party	std::string	Name of the party
redistribute_ballots()	std::queue<IRBallot>	When no clear majority or there is a looser the ballots get redistributed

AuditLog

Item	Type	Description
AuditLog(log_name: std::string)	constructor	Constructor for AuditLog object
file	std::ofstream	String that contains the results from the election
filename	std::string	Output file stream handler for interacting with the OS
log(Loggable&)	void	Base log function which determines if an object/process is loggable (used/modified by other classes)
log(std::string)	void	Base log function used/modified by other classes

MediaReport

Item	Type	Description
file	std::ofstream	Output file stream handler for interacting with the OS
filename	std::string	Name of the Media report
MediaReport(report_name: std::string)	constructor	Constructor for a Media Report object
write(std::string)	void	Writes a string to the media report file.

Loggable

Item	Type	Description
log()	std::string	Base function for determining if an object is loggable or not (utilized by the log class)

5. COMPONENT DESIGN

Both Election Types

add_ballot(): Adds ballot to a specific candidate once read from the ballot file.

announce_results(): Returns the vote totals and seat allocations for each party and candidate to the terminal for viewing after the election has been tallied.

generate_id(): Is called on construction of a new Ballot. This function shall always produce a new, unique id on every call.

get_id(): Returns the id of a Ballot instance (OPLBallot, IRBallot).

get_choice(): Returns the candidate selected by an individual ballot.

get_tally(): Returns the number of ballots attributed to either a specific candidate or specific party.

log():

- Logs movement of a specific ballot between candidates / party
- Logs ballots attributed to a candidate
- Logs ballots attributed to a party

parse_ballots(): Reads in ballots and attributes to a specific candidate and party.

run(): Executed by either the IRElection or OPLElection class to properly count votes and allocate seats according to the specific election style as dictated in the ballot file.

TieBreaker(): Runs a sequence of simulated coin flips and returns a 1 or a 0 depending on the result of the next coin flip. Used in tied vote scenarios.

write(): Returns a set of metrics and results about the election to the MediaReport class to be used as a summary of the election.

IR

redistribute_ballots(): Used to allocate ballots to the n-th choice after their n-1-th choice is eliminated from the election

OPL

assign_seats(): Distribute available seats based first on simple division of each party's vote total

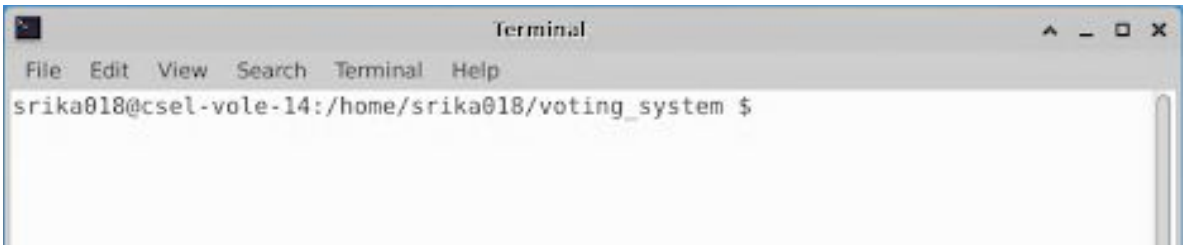
versus the decided quota value, and then based on which party has the highest remaining votes

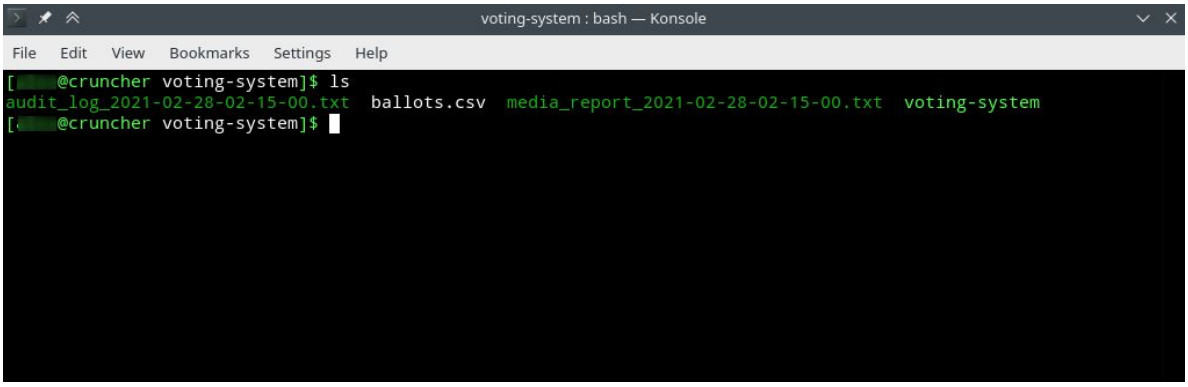
6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

The user will use the system solely through the command line interface. This can be accessed directly from a physical CSE IT lab machine, or through an SSH connection or Vole. To use the system, the user will enter a command to run the program with a ballot file and the program will handle the rest. Upon completion, the results along with important information will be displayed on the screen. The audit and media files will have been generated and stored in the same directory as the program.

6.2 Screen Images





6.3 Screen Objects and Actions

- Command Line Interface
  - Running program
  - Locate/Select ballot file

- Locate media file
- Locate audit file

7. REQUIREMENTS MATRIX

ID	Name	Where in code
SF_001	Run Program	Main/Driver function
SF_002	Select Ballot File	N/A
SF_003	Read File	Election class
SF_004	Indicating Election Type	Election class
SF_005	Determine winner in case of clear majority in IR election	IR candidate, IR ballot, IR class
SF_006	Determine winner in case of no clear majority in IR election	IR candidate, IR ballot, IR class
SF_007	Group OPL Candidates into respective parties	OPL candidate, OPL ballot, OPL party, OPL class
SF_008	Display Election Winner	Election class
SF_009	Start Audit Log	Election class, AuditLog class
SF_010	Log Ballot Ingest	Election class, AuditLog class
SF_011	Log Redistribution of Vote	Election class, AuditLog class
SF_012	Log Election Results	Election class, AuditLog class

SF_013	Resolving a Tie	TieBreaker class
SF_014	Produce Report for Media Personnel	Election class, MediaReport class
SF_015	Retrieve Audit File	Election class, AuditLog class

8. APPENDICES

Class Diagram: `voting_system_class_diagram_team20.pdf`