

Team 17

**V-Pal**

Software Design Document

Names:

Joel Alfveby, alfve012

Bat-Idar Ganbold, ganbo011

Kanoog Moua, mouax677

Ramanish Singh, singh891

Group: Team 17

Date: February 27, 2021

## TABLE OF CONTENTS

<b>1.</b>	<b>INTRODUCTION</b>	<b>2-4</b>
1.1	Purpose	2
1.2	Scope	2
1.3	Overview	3
1.4	Reference Material	3
1.5	Definitions and Acronyms	4
<b>2.</b>	<b>SYSTEM OVERVIEW</b>	<b>5</b>
<b>3.</b>	<b>SYSTEM ARCHITECTURE</b>	<b>6-10</b>
3.1	Architectural Design	6
3.2	Decomposition Description	7
3.3	Design Rationale	9
<b>4.</b>	<b>DATA DESIGN</b>	<b>10-16</b>
4.1	Data Description	10
4.2	Data Dictionary	10
<b>5.</b>	<b>COMPONENT DESIGN</b>	<b>17-31</b>
<b>6.</b>	<b>HUMAN INTERFACE DESIGN</b>	<b>32-35</b>
6.1	Overview of User Interface	32
6.2	Screen Images	33
6.3	Screen Objects and Actions	35
<b>7.</b>	<b>REQUIREMENTS MATRIX</b>	<b>36</b>

## Revision History

Name	Date	Reason For Changes	Version
V-Pal	February 27, 2021	System design document first creation	1.0

# 1. INTRODUCTION

## 1.1 Purpose

The purpose of this Software Design Document (SDD) is to describe the architecture and design of V-Pal's system. This document is meant to be read by people who will be using, testing, and modifying V-pal. Specifically, this document is intended for the election official who will be using V-Pal for determining the winner of the election, testers who will be testing for any bugs in V-Pal, and programmers who are responsible for developing and maintaining V-Pal.

## 1.2 Scope

V-Pal is designed to process the counting of ballots for the voting election(s) according to either IRV or OPVL voting systems (refer to section 1.5). When given an input of formatted ballots, this program will process the ballots and output a winner for the election. As well as an audit file and media summary file with the election information at the time.

The mission of V-Pal is to efficiently carry out IRV and OPLV ballot counting for the elections. It seeks to reduce the amount of time it takes to process ballot counts for these voting systems, with a high-throughput of 100k ballots in under 8 minutes. Thus allowing for the media to promptly receive a summary of the election results.

The simple and intuitive design of V-Pal allows for easy navigation of the software and increases the user's efficiency. Designated members of V-Pal will have full access to make changes, as they deem necessary. The changes include the edition of this document.

### 1.3 Overview

Section 2 shows the inner workings of the system and its organization. It depicts a clear explanation of where each design entity will reside as well as how each part works in unison to accomplish each component pertinent to the system.

Section 3 outlines the Architectural Design that specifies the subsystem that performs all of the functions included in the system. It contains a high-level diagram explaining each of the functions to its low-level description. It includes a UML activity diagram, UML class diagram and a sequence diagram.

Section 4 contains the Data Structure Design.

Section 5 includes the Component design of each function used in the V-Pal software.

Section 6 discusses User Interface Design.

Section 7 contains a cross-reference matrix of the requirements outlined in the corresponding SRS document.

### 1.4 Reference Material

1. Reaves, M. J. (2014, May 08). Software design description (sdd) sample. Retrieved February 23, 2021, from [https://www.slideshare.net/peny\\_mg/sdd-software-des-sample](https://www.slideshare.net/peny_mg/sdd-software-des-sample)
2. IEEE. (2003). Software Design Document (SDD) Template. IEEE SDD. <http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieeeSDD.pdf>
3. FairVoteMinnesota. (n.d.). How RCV Works | FairVote MN. Retrieved February 6, 2021, from <https://www.fairvotemn.org/rcv/>
4. Ian, S. (2015). Software Engineering Global Edition (10th ed.). PEARSON EDUCACION.

## 1.5 Definitions and Acronyms

**CSV:** Comma Separated Values, a type of file that uses commas to separate data.

Commonly used in Microsoft Excel.

**Election official:** A person in charge of running the election, as well as counting the votes

***IRV:** Instant runoff voting*

***OPLV:** Open Party Listing voting*

**Programmer:** Person who designs, develops, and tests software programs.

**SDD:** Software Design Document

**SRS:** Software Requirements System

**Tester:** Anybody wanting to test the integrity of the software

**UML:** Unified Modeling Language

**User:** Anybody using the software

## 2. SYSTEM OVERVIEW

V-pal was created in order to visualize and analyze the results of two different election methods .V-Pal will be able to process two types of ballots based on a voting system which are *IRV (Instant Runoff Voting)* and *OPLV (Open Party List Voting)*, respectively. When given an input of formatted ballots (CSV), this program will process the ballots and output a winner for the election as well as an audit file and media summary file. The Audit file will contain all the necessary information to replicate the election. The media summary file will contain all the necessary information extracted from the audit file. Insert IRV and OPLV explanations here, as well as the general functionality of the program (how to launch, what to input and when, what gets output and where)

## 3. SYSTEM ARCHITECTURE

### 3.1 Architectural Design

Please refer to UMLClassDiagram\_Team17.pdf

When the user runs the software in the command line, an instance of the Election class is initialized. Then the user provides the name of the csv file which is taken by the Election object and depending on the type of election mentioned in the csv file either the IRV class or the OPLV class is instantiated. If the csv file is not located in the current directory, the program displays the error on the screen that the file is not available in the current folder.

IRV:

If the election type mentioned in the ballot csv file is IRV then IRV child class of Election class is instantiated. After that the ballot data is read, and the IRVBallot objects of the IRVBallot class are instantiated and stored in the IRV object. Each ballot contains an attribute named “transformedVote” which is an ordered list of the priority of candidates mentioned in that ballot by the voter. For example, if that ballot reads “1,3,4,2” then the transformedVote attribute for that object will be stored as [1,4,2,3]. The order of the indices will represent the order of choice in the vote and values in the indices will represent the candidates. This transformation is carried out to easily determine to which candidate the ballots must be transferred in case the current owner of the ballots gets eliminated. The information about each candidate is stored in the IRVCandidate object. Each IRVCandidate object has an attribute myBallot which tracks the ballots which are currently owned by that candidate.

Once the IRVBallot objects and IRVCandidate objects are prepared, we call the FindWinner() function in the IRV object which conducts the election. More information about FindWinner() function in IRV is provided in the section 4.2 and 5. After the completion of FindWinner, we call the DisplayResults in the main program to print the results on screen and save the media summary file. All the intermediate calculations are stored in the Audit class which produces an audit file of the election.

OPLV:

If the election type mentioned in the ballot csv file is OPLV, then the OPLV child class of the Election class is instantiated. It stores attributes like numParties, quota, parties, etc. Mainly, it contains an array of OPLVParty objects in the parties attribute. The OPLVParty object is instantiated for each party in the election. OPLVParty objects have attributes like candidates (name of all the candidates contesting in the election who belong to that party), seatsWon (integer that keeps track of the seats won by that party), name, etc.

The candidates attribute in OPLVParty object is the array of OPLVCandidate class object which is a child class of Candidate class.

After the OPLV class is instantiated, the quota is calculated and updated using the CalculateQuota method. After that, seats won by each party in the first round of allocation is calculated using the FindWinner() method and the total number of seats allocated in the first round is updated in the seatsAllocated attribute. If seatsAllocated is less than numSeats, we execute the method UpdateRank which tells us the ranking of different parties based on their remaining votes. After that, we execute AllocateRemainingSeats to distribute the remaining seats between the parties based on their ranking. After this round of allocation, we call the DisplayResults in the main program to print the results on screen and save the media summary file. All the intermediate calculations are stored in the Audit class which produces the audit file of the election.

### 3.2 Decomposition Description

Refer to document UMLActivityDiagram\_Team17.pdf.

First the software will prompt the user to enter the name of the CSV file. The software will then validate the existence of the file by checking if the file exists in the same directory as the software. If it does not exist or can't be found, it will display an error message asking the user to enter a valid file name or the correct name for the CSV file. After the CSV file is successfully validated, the software will read the first line of the file to determine the election type. It will check if the election type is IRV, if it is election type IRV then it will split to the path of IRV, if not it will split to the path of OPLV.

IRV path:

First the rest of the CSV file will be read. number of ballots, number of candidates as well as their respective parties are going to be extracted from the CSV file. For each ballot an empty ballot object will be created to be filled in with information gathered from the CSV file. For each candidate IRVcandidate object will be created and each object will be filled in with information as previously mentioned. Then the ballot objects will be scanned and will be assigned to their respective candidates by their first choice votes. Then we will conduct an election or check if any of the candidates has a tie between them. If there are ties then the software chooses a winner at random. If there are no ties then the software will check if any of the candidates win by majority or more than 50% votes. If there is no majority then the candidate with the least votes will be eliminated and his ballot objects will be distributed among candidates by their second choice votes. This will repeat until there is a clear majority and a winner is decided. The winner will be displayed on the screen and an audit and media summary file will be created in the same directory as the software.



OPLV path:

First the rest of the CSV file will be read and the number of ballots, number of candidates and total number of seats will be extracted from the CSV file. Then the ballots will be read and the votes will be distributed to each of the candidates. Then the candidate's votes will be summed up for each of their respective parties and it will begin to calculate the seats won by each party. After the seats are calculated by using the quota, the remaining votes will be sorted in decreasing order with their respective parties. Then the remaining seats will be distributed from the top to bottom allocating one seat to each party. After all the seats are allocated to each of the parties, the software will calculate which candidates won seats from each party. The screen will display the amount of seats won by each party and an audit and media summary file will be created in the same directory as the software.

Please refer to OPLVSequenceDiagram\_Team17.pdf

If the election type mentioned in the ballot data csv file is OPLV, an OPLV class is instantiated which is the child class of Election class. Using the methods available in the OPLV class, we get the values of i) Total parties, ii) Total candidates, iii) Total ballots. Using that information, we create the candidate objects and the ballot objects. In each OPLVCandidate object, we store the information about the candidate, mainly, i) name, ii) party name, iii) number of votes won by that candidate. After that we create the party objects of the type OPLVParty. Each OPLVParty object contains the information about that party i.e. the name of the party, the names of the candidates in the party, etc. Using the SetRemainingVotes() function, we initialize the number of votes received by each party. Then the CalculateQuota() method in OPLV can be used to calculate the quota. We also need to rank the candidates in each party according to the votes they received, and that is achieved using the SetCurPartyRank() method. After all this information is entered, we call the FindWinner() method in OPLV which carries out the first round of allocation of seats within each party. That function updates the number of seats won by each party in the seatsWon attribute of the OPLVParty object. After that, we call the RemainingVotes() method to update the votes remaining and use the UpdateRank() method in the OPLV class to make an ordered list of parties sorted by their remainingVotes. We check if all the seats are allocated by comparing numSeats attribute to the seatsAllocated attribute, if seats are remaining, the second allocation is completed using the AllocateRemainingSeats() method in the OPLV object. Finally we use the DisplayResults() method in the Main function which stores the summary file of the election in the current folder and calls the Audit class to store the audit file of the election.

The UML Class Diagram which will be in the GitHub repository along with this document.

### 3.3 Design Rationale

For IRV, we had considered making ballot arrays for every candidate in the election, but we realized that transferring ballot objects between candidates could get complicated and messy. Instead, our method of having an array of all the ballots with separate integer arrays for each candidate makes it easy to check which ballots go to which candidates, along with adding up total ballots by summing up the array. Also, to easily track which ballot belongs to which candidate and to transfer ballots easily from one IRVCandidate to the other, we define a new way to store votes, the transformedVote. It is an ordered list of the priority of candidates mentioned in that ballot by the voter. For example, if that ballot reads “1,3,4,2” then the transformedVote attribute for that object will be stored as [1,4,2,3]. The order of the indices will represent the order of choice in the vote and values in the indices will represent the candidates. This transformation is carried out to easily determine to which candidate the ballots must be transferred in case the current owner of the ballots gets eliminated. We also maintain an attribute priorityIndex in each ballot class, which is simply the index of the choice we want to look at in that ballot (we update the priority index by one when the ballot owner gets eliminated, and the next choice should be the new ballot owner).

For OPLV, we find the number of votes, then find the quota and total number of votes. With that, we can calculate the number of votes for the first allocation of seats. After that, with the remainingSeats attribute, we can find the second allocation of seats. Once all of the seats are calculated and allocated, each party can find a candidate for each seat based on their popularity.

## 4. DATA DESIGN

### 4.1 Data Description

V-Pal reads the csv file and finds all the necessary information to create the following objects:

- i) Election object (separate election object for IRV and OPLV): The election object will run and control the program.
- ii) Ballot objects (only for IRV): Keep track of the ballots from the csv.
- iii) Candidate objects (separate election object for IRV and OPLV): The candidate objects will be created from the csv. Will contain information about the candidates such as name, partyname, etc.
- iv) Party objects (only for OPLV): The party objects will be created from the parties described from the csv. Will contain information about the parties such as name, candidates, etc.
- v) Audit object: The audit object records the step-by-step process of calculating the ballots for the election.

### 4.2 Data Dictionary

Class	Data Item	Type	Definition
-------	-----------	------	------------

<b>Audit</b> Objects	auditFile	File	File to hold all audit information
Functions	Audit	() : Audit	Constructor for audit class, creates audit file
	Print	(string): void	Prints the string to the audit file

<b>Ballot</b> Objects	voterNum	int	Holds the number of the ballot (i.e. fourth ballot processed = 4)
Functions	Ballot	() = 0	Constructor for Ballot (is pure virtual, cannot instantiate Ballot)

<b><i>Candidate</i></b> Objects	name	string	Name of candidate
	party	char	Initial of party
Functions	Candidate	() = 0	Constructor for Candidate (is pure virtual, cannot instantiate Candidate)
	GetName	(): string	Returns name of candidate
	GetParty	(): char	Returns party

<b><i>Election</i></b> Objects	typeOfElection	string	Keeps track of the type of the election
	candidates	Candidate *	Array of all of the candidates (either OPLVCandidate or IRVCandidate)
	totalBallots	int	Holds total number of ballots in election
Functions	FindWinner	(FILE) = 0	Find winner of election (is pure virtual, cannot instantiate Election)
	GetTypeOfElection	(): string	Return type of election
	GetTotalCandidates	(): int	Return total number of candidates
	GetTotalBallots	(): int	Return total number of ballots
	GenerateCandidates	(): void	Creates candidates based on csv
	UpdateCandidates	(): void	Update candidate info

<b>IRV</b> Objects	numCandidates	int	The amount of candidates participating in the election
	ballots	IRVBallot *	Array of all ballots in election
Functions	IRV	() : IRV	Constructor for IRV
	FindWinner	(FILE): string	Find winner of IRV election
	TieBreaker	(string, string): string	If two candidates are tied, choose the winner
	ReadVote	() : void	Transforms the csv vote into our format, then stores in the ballots array
	GenerateBallots	() : void	Generates the ballots from the csv

<b>IRVBallot</b> Objects	priorityIndex	int	IRVBallots.transformedVote[priorityIndex] will give us the candidate number for which this ballot should count towards
	transformedVote	int *	An array that contains the ordered list of id of candidates, highest preference to lowest preference
	voteSize	int	Number of preferred candidates on ballot
Functions	IRVBallot	(int *, int): IRVBallot	Constructor for IRVBallot. takes the transformed vote and the vote size. Sets priorityIndex to 0

	Valid	(): boolean	If priorityIndex > voteSize, no candidate is preferred so ballot is invalid
	GetIndex	(): int	Return priorityIndex of ballot
	IncrementIndex	(): void	Increment priorityIndex; preferred candidate was eliminated
	GetVoteSize	(): int	Return voteSize
	GetCurrentPref	(): int	Return current preferred candidate: transformedVote[priority Index]

<b>IRVCandidate</b> Objects	stillInRace	boolean	Initialized as true and will be turned false if the candidate is eliminated
	candidateID	int	ID of candidate based on csv for ballot purposes
	myBallots	int *	List of all the ballots which have this candidate as the first choice. 0 - not for this candidate, 1 - for this candidate
Functions	IRVCandidate	(string, char, int): IRVCandidate	Constructor for IRVCandidate. Takes their name, party, and ID
	Eliminate	(): void	Eliminates candidate, transfers secondary votes to next candidate
	SetMyBallots	(IRVBallot *): void	Sets myBallots to be concurrent with the ballots from IRV
	CheckInRace	(): boolean	Check if candidate is still in race

	GetNumVotes	(): int	Returns number of votes for candidate. Does this by summing myBallots
--	-------------	---------	-----------------------------------------------------------------------

<b>Main Objects</b>	election	Election *	Holds election type for program, either IRV or OPLV
	audit	Audit	Audit object to output calculations to audit file
Functions	FindType	(FILE): void	Finds the election type from csv and checks for errors in file
	Run	(): void	Runs the election via the election object
	DisplayResults	(): void	Displays the results of the election and creates summary file

<b>OPLV Objects</b>	numParties	int	The number of parties in the election
	quota	int	The quota for determining the number of seats allotted to the parties
	parties	OPLVParty *	Array of all of the parties
	numSeats	int	The number of seats available in the election
	seatsAllocated	int	Keeps track of the number of seats that have been allocated
Functions	OPLV	(): OPLV	Constructor for OPLV
	FindWinner	(FILE): string	Find winner of OPLV election
	CalculateQuota	(int, int): int	Calculates the quota based on the total number of ballots and the total

			number of seats
	GenerateParties	(): void	Creates parties
	UpdateParties	(): void	Update party info
	UpdateRank	(): void	After all seats are awarded, update party rank of all candidates based on number of votes for candidate
	AllocateRemainingSeats	(): void	Function that allocates the remaining number of seats

<b>OPLVCandidate</b> Objects	curPartyRank	int	Their rank in the party according to the number of votes received
	numVotes	int	Holds number of votes for candidates
Functions	OPLVCandidate	(string, char): OPLVCandidate	Constructor for OPLVCandidate. Takes their name and party
	GetCurPartyRank	(): int	Returns current party rank
	SetCurPartyRank	(int): void	Sets current party rank

<b>OPLVParty</b> Objects	candidates	OPLVCandidate *	Array of candidates in party
	seatsWon	int	Total seats won for this party
	name	string	Name of party
	remainingVotes	int	Remaining votes after first seat allocation



Functions	OPLVParty	(OPLVCandidate *, string): OPLVParty	Constructor for OPLVParty. Takes its candidates and its name
	GetName	(): string	Return name of party
	GetSeats	(): int	Return number of seats won by party
	GetRemainingVotes	(): int	Return number of remaining votes for party
	SetRemainingVotes	(int): void	Set number of remaining votes for party
	SetSeatsWon	(int): void	Set seats won by party
	InitializeRemainingVotes	(int): void	Sets remaining votes to the total number of votes for the party. Will change after first allocation of seats

## 5. COMPONENT DESIGN

### 5.1 Main

#### 5.1.1.

Function	FindType()
Purpose	Find the election type from the csv file and checks the file for errors
Input	FILE
Pseudocode	Open csv file Get first line of csv If line = "IR" election = new IRV else if line = "OPL" election = new OPLV
Output	void

#### 5.1.2.

Function	Run()
Purpose	Runs the election via the election object
Input	none
Pseudocode	If election.typeOfElection = "IR" then GenerateBallots() GenerateCandidates() FindWinner(csv) If tie then TieBreaker(candidate 1, candidate 2) else CalculateQuota(total ballots, numSeats) GenerateCandidates() GenerateParties() FindWinner(csv)
Output	void

#### 5.1.3.

Function	DisplayResults
Purpose	Displays the results of the election to the screen
Input	none
Pseudocode	Call the getter functions for winner, seats, and number of votes Write the information retrieved to the summary.txt file Display the results to the screen

Output	none
--------	------

## 5.2 Audit

### 5.2.1.

Function	Audit
Purpose	Constructor for audit class, creates audit file
Input	none
Pseudocode	auditFile.open("audit_timestamp.txt")
Output	none

### 5.2.2.

Function	Print
Purpose	Prints the string to the audit file
Input	String
Pseudocode	fprintf(this->auditFile, string, %s)
Output	void

## 5.3 Ballot

### 5.3.1.

Function	Ballot
Purpose	Constructor for Ballot (is pure virtual, cannot instantiate Ballot)
Input	none
Pseudocode	Ballot::Ballot() = 0
Output	none

## 5.4 Candidate

### 5.4.1.

Function	Candidate
Purpose	Constructor for Candidate (is pure virtual, cannot instantiate Candidate)
Input	none
Pseudocode	FindWinner() = 0

Output	none
--------	------

**5.4.2.**

Function	GetName
Purpose	Returns name of candidate
Input	Candidate
Pseudocode	Return Candidate.name
Output	string

**5.4.3.**

Function	GetParty
Purpose	Returns Party
Input	Candidate
Pseudocode	Return Candidate.party
Output	char

**5.5 Election****5.5.1.**

Function	FindWinner
Purpose	Find winner of election (is pure virtual, cannot instantiate FindWinner)
Input	FILE
Pseudocode	FindWinner() = 0
Output	0

**5.5.2.**

Function	GetTypeOfElection
Purpose	Return type of election
Input	FILE
Pseudocode	ifstream readFile readFile.open("FILE")

	<pre>while (!readFile.eof()) string electionType = first line of csv readFile.close() return electionType</pre>
Output	string

**5.5.3.**

Function	GetTotalCandidates
Purpose	Return total number of candidates
Input	FILE
Pseudocode	<pre>ifstream readFile readFile.open("FILE") while (!readFile.eof()) int totalCandidates = find total candidates from csv end while readFile.close() return totalCandidates</pre>
Output	int

**5.5.4.**

Function	GetTotalBallots
Purpose	Return total number of ballots
Input	FILE
Pseudocode	<pre>ifstream readFile readFile.open("FILE") while (!readFile.eof()) int totalBallots = 4th line of csv end while readFile.close() return totalBallots</pre>
Output	int

**5.5.5.**

Function	GenerateCandidates
Purpose	Creates candidates based on csv file
Input	none

Pseudocode	<pre> while(on the candidates line in csv)   get string before comma   if IRV     candidates[index] = new IRVCandidate()   else candidates[index] = new OPLVCandidate() </pre>
Output	void

**5.5.6.**

Function	UpdateCandidates
Purpose	Update candidate info
Input	none
Pseudocode	<pre> while i&lt; numParties   Find the ranking of candidates in party i by candidate.numVotes   update CurPartyRank using their ranking </pre>
Output	void

**5.6 IRV****5.6.1.**

Function	IRV
Purpose	Constructor for IRV
Input	none
Pseudocode	typeOfElection = "IRV"
Output	IRV

**5.6.2.**

Function	FindWinner
Purpose	Find Winner of IRV election; calling all of the proper functions to find winner
Input	FILE
Pseudocode	<pre> votes=[]  while 1   i=0   while i&lt; numCandidatesinRace </pre>

	<pre> append sum(IRVCandidate[i].myBallot) to votes array  i=i+1  if max(votes[]) &gt; (totalBallots/2)     declare candidate with the max(votes[]) number of votes     break  if tie:     j is a random integer between 0 and numCandidates-1     declare candidate j the winner  else:     j is the index of smallest element in votes[]     get the myBallot attribute of candidate j     k=0     for k &lt; length(candidate[j].myBallot)         if candidate[j].myBallot[k]=0             k=k+1             continue         else             IRV.ballots[k].priorityIndex=IRV.ballots[k].priorityIndex+1             newOwner=IRV.ballots[k].transformedVote[priorityIndex]             candidate[newOwner].myBallot[k]=1              k=k+1     IRVCandidate[j].Eliminate  %%%% makes stillInRace=False </pre>
Output	string

### 5.6.3.

Function	TieBreaker
Purpose	Randomly choose winning candidate if there is a tie
Input	string, string
Pseudocode	<p>Calculate the total number of candidates in the race ( and have tied)</p> <p>Store all the candidates in an array (array of candidate objects)</p> <p>Pick a random natural number between 1 and total number of candidates (including 1 and total number of candidates)</p> <p>Return the name of the candidate which is present at the picked random number element in the array of candidates</p>
Output	string

**5.6.4.**

Function	ReadVote
Purpose	Transforms the csv ballots into our format for tracking votes and stores the ballots in an array
Input	none
Pseudocode	Read the csv file Determine the total number of ballots from line 4 totalBallots= $n$ Create empty $n$ ballot objects Jump to the line where the ballots start ( 5th line for IRV and 6th line for OPLV)
Output	void

**5.6.5.**

Function	GenerateBallots
Purpose	Generate the ballots from the csv
Input	FILE
Pseudocode	ifstream readFile // open file stream readFile.open("FILE")//open file while (!readFile.eof())// while the file has not reached end of file int totalBallots = ~read second line and store ~ end while for i = 0 to totalBallots ~Generate ballot objects~ end for readFile.close()
/cl	void

**5.7 IRVBallot****5.7.1.**

Function	IRVBallot
Purpose	Constructor for IRVBallot. Takes the transformed vote and the vote size. Sets priorityIndex to 0
Input	(int *, int)
Pseudocode	transformedVote = int * voteSize = # candidates priorityIndex = 0



Output	IRVBallot
--------	-----------

**5.7.2.**

Function	Valid
Purpose	If priorityIndex > voteSize, no candidate is preferred so ballot is invalid
Input	none
Pseudocode	if(priorityIndex > voteSize) ballot is not valid for count else ballot is still valid for redistribution to other candidates
Output	boolean

**5.7.3.**

Function	GetIndex
Purpose	Return priorityIndex of ballot
Input	none
Pseudocode	return priorityIndex of ballot
Output	int

**5.7.4.**

Function	IncrementIndex
Purpose	Increment priorityIndex; preferred candidate was eliminated
Input	none
Pseudocode	increment priorityIndex
Output	void

**5.7.5.**

Function	GetVoteSize
Purpose	Return VoteSize
Input	none
Pseudocode	return VoteSize

Output	int
--------	-----

**5.7.6.**

Function	GetCurrentPref
Purpose	Return the current preferred candidate in the ballot: transformedVote[priorityIndex]
Input	none
Pseudocode	return priorityIndex value
Output	int

**5.8 IRVCandidate****5.8.1.**

Function	IRVCandidate
Purpose	Constructor for IRVCandidate. Takes their name, party, and ID
Input	(string, char, int)
Pseudocode	name = string party = char stillInRace = true candidateID = 0 myBallots = for i=csv.length; i < 1: i<length; i++ ballots[i] = new IRVBallot(vote, # candidates)
Output	IRVCandidate

**5.8.2.**

Function	Eliminate
Purpose	Eliminates candidate, transfers secondary votes to next candidate
Input	none
Pseudocode	if myBallots.length < other ballots.length stillInRace = false
Output	void

**5.8.3.**

Function	SetMyBallots
----------	--------------

Purpose	Sets myBallots to be concurrent with the ballots from IRV
Input	(IRVBallot *)
Pseudocode	<pre>int something.this = int * for(int i = 0; i &lt; ballots.length; i++)     ballots[i] = IRVBallot()</pre>
Output	void

**5.8.4.**

Function	CheckInRace
Purpose	Check if the candidate is still in the race
Input	none
Pseudocode	<pre>if(stillInRace is true)     return TRUE else     return FALSE</pre>
Output	boolean

**5.8.5.**

Function	GetNumVotes
Purpose	Returns number of votes for candidate. Does this by summing myBallots
Input	none
Pseudocode	<pre>sumVotes = sum(myBallots) return sumVotes</pre>
Output	int

**5.9 OPLV****5.9.1.**

Function	OPLV
Purpose	Constructor for OPLV
Input	none
Pseudocode	OPLV = new OPLV
Output	OPLV

**5.9.2.**

Function	FindWinner
Purpose	Find winner of OPLV election; calls the relevant functions to determine winner
Input	FILE
Pseudocode	<pre> find numParties quota = CalculateQuota(totalBallots, numSeats) forNumParties     parties = new Party TotalnumSeats = GetSeats()  for NumParties AND numSeats!=0     NumVotes= getRemainingVotes(Party)     RemainingVotes = NumVotes % quota     Seatswon= NumVotes / quota     if (TotalnumSeats&gt;= Seatswon)         TotalnumSeats= numSeats - Seatswon     end for if numSeats !=0 then     AllocateRemainingSeats()  else     Break </pre>
Output	string

**5.9.3.**

Function	CalculateQuota
Purpose	Calculates the quota based on the total number of ballots and the total number of seats
Input	(int, int)
Pseudocode	<pre> quota = totalBallots % numSeats return quota </pre>
Output	int

**5.9.4.**

Function	GenerateParties
Purpose	Creates the party objects according to csv file

Input	none
Pseudocode	for (parties found) OPLVParty(Candidate, name)
Output	void

**5.9.5.**

Function	UpdateParties
Purpose	Update the info for the party
Input	none
Pseudocode	UpdateParties() // updates with seatsWon, remainingVotes
Output	void

**5.9.6.**

Function	UpdateRank
Purpose	After all seats are awarded, update party rank of all candidates based on number of votes for candidate
Input	none
Pseudocode	update curPartyRank
Output	void

**5.9.7.**

Function	AllocateRemainingSeats
Purpose	Function that allocates the remaining number of seats
Input	none
Pseudocode	for Candidates in party SetSeatsWon(seatsWon)
Output	void

**5.10 OPLVCandidate****5.10.1.**

Function	OPLVCandidate
Purpose	Constructor for OPLVCandidate. Takes their name and party

Input	(string, char)
Pseudocode	new OPLVCandidate(name, party)
Output	OPLVCandidate

**5.10.2.**

Function	GetCurPartyRank
Purpose	Returns current party rank
Input	none
Pseudocode	return curPartyRank
Output	int

**5.10.3.**

Function	SetCurPartyRank
Purpose	set the current rank of the candidate within the party
Input	int
Pseudocode	curPartyRank = int
Output	void

**5.11 OPLVParty****5.11.1.**

Function	OPLVParty
Purpose	Constructor for OPLVParty. Takes its candidates and its name
Input	(OPLVCandidate *, string)
Pseudocode	new OPLVParty(Candidate, name)
Output	OPLVParty

**5.11.2.**

Function	GetName
Purpose	Return name of the party
Input	none

Pseudocode	return OPLVParty.name
Output	string

**5.11.3.**

Function	GetSeats
Purpose	Return number of seats won by the party
Input	none
Pseudocode	return seatsWon
Output	int

**5.11.4.**

Function	GetRemainingVotes
Purpose	Return number of remaining votes for party
Input	none
Pseudocode	return remainingVotes
Output	int

**5.11.5.**

Function	SetRemainingVotes
Purpose	Set number of remaining votes for party
Input	int
Pseudocode	remainingVotes = int
Output	void

**5.11.6.**

Function	SetSeatsWon
Purpose	Set seats won by party
Input	int
Pseudocode	seatsWon = int
Output	void

**5.11.7.**

Function	InitializeRemainingVotes
Purpose	Sets remaining votes to the total number of votes for the party. Will change after first allocation of seats
Input	int
Pseudocode	<pre>votes=[]  j=0 while j &lt; numCandidates in the party     append votes won by candidate j in the party to votes  Remaining votes = sum of all the elements of votes[] OPLVParty.RemainingVotes=Remaining votes</pre>
Output	void



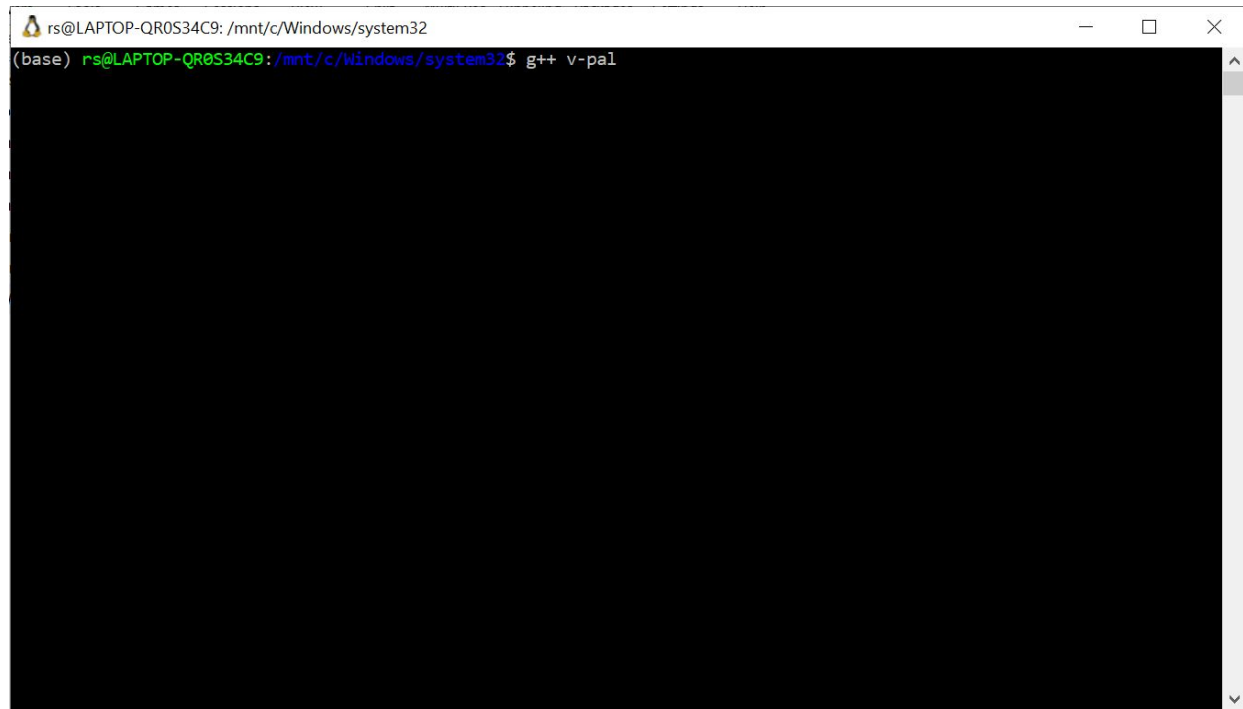
## **6. HUMAN INTERFACE DESIGN**

### **6.1 Overview of User Interface**

V-Pal is a software application aimed at determining the results of an election based on the ballots files. The users (election officials) can use this C++ based software easily to process the ballot files they have acquired from the election, and determine the winning candidate or number of seats won by each party (depending on the type of election). V-Pal also creates an audit file of the election. The audit file contains the information about each ballot and votes/seats were allotted to a candidate. This makes the whole election process transparent and reproducible. Moreover, V-Pal also creates a summary file of the election which can be sent to the media so that they can publish the results. Summary file contains the information about the winners and losers and how many votes each candidate received/ how many seats each party won.

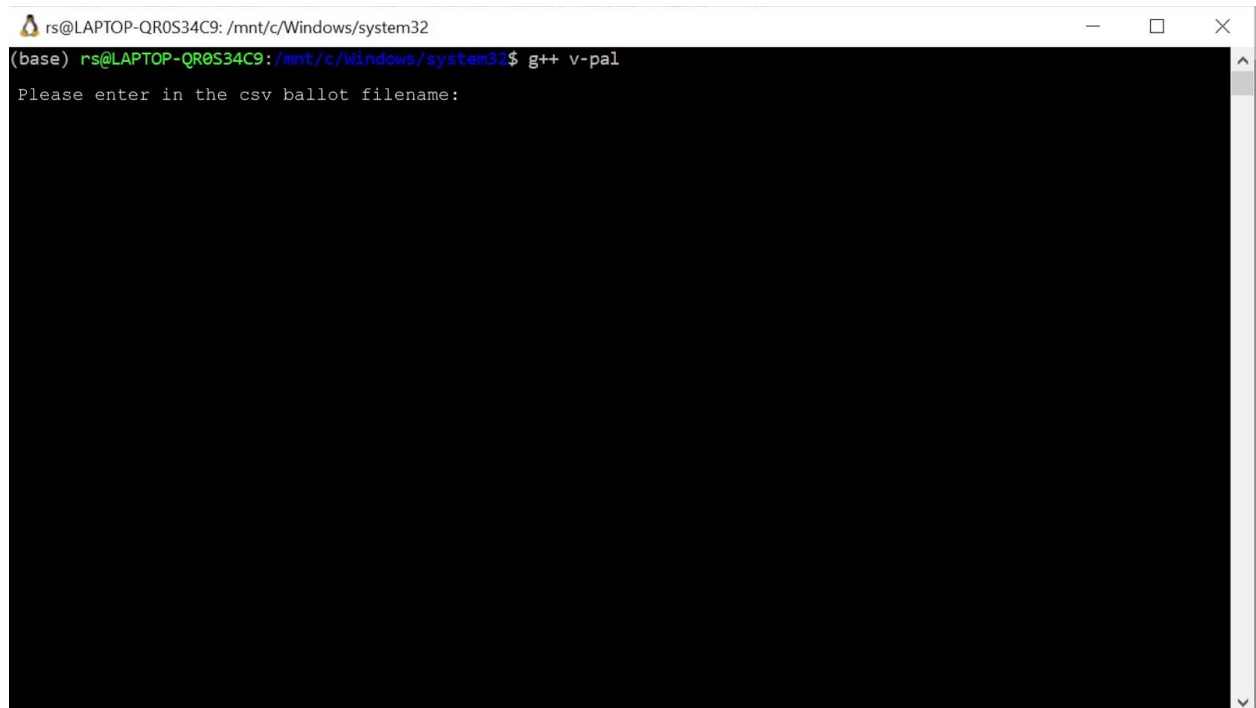
Any errors with the file will be reported back to the user on the screen. V-Pal will also report the process of calculating the election and what type it is. At the end of the process, the results of the election will be displayed on the screen. This information includes the name of the winner(s), their party(ies), and how many votes they received. Detailed information about the process will be contained in the audit and media summary file.

## 6.2 Screen Images

A screenshot of a terminal window. The title bar at the top reads 'rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32'. The terminal content shows a prompt '(base) rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32\$' followed by the command 'g++ v-pal'. The rest of the terminal area is black, indicating the command is waiting for input or the output is not visible.

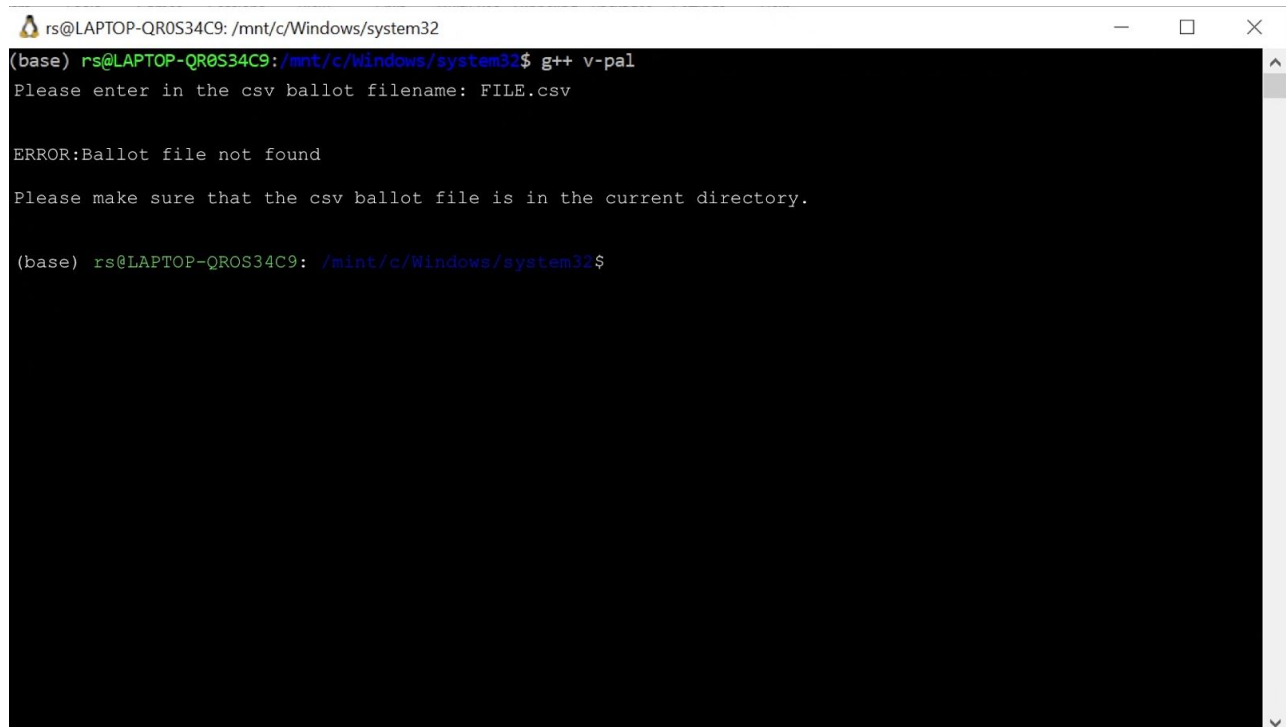
```
rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32
(base) rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32$ g++ v-pal
```

After the program is executed in the terminal, the system will ask for the name of the ballot file.



```
rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32
(base) rs@LAPTOP-QR0S34C9:/mnt/c/Windows/system32$ g++ v-pal
Please enter in the csv ballot filename:
```

The user will need to place the ballot file in the current working directory (cwd) and provide the name in the terminal followed by the return key (Enter).



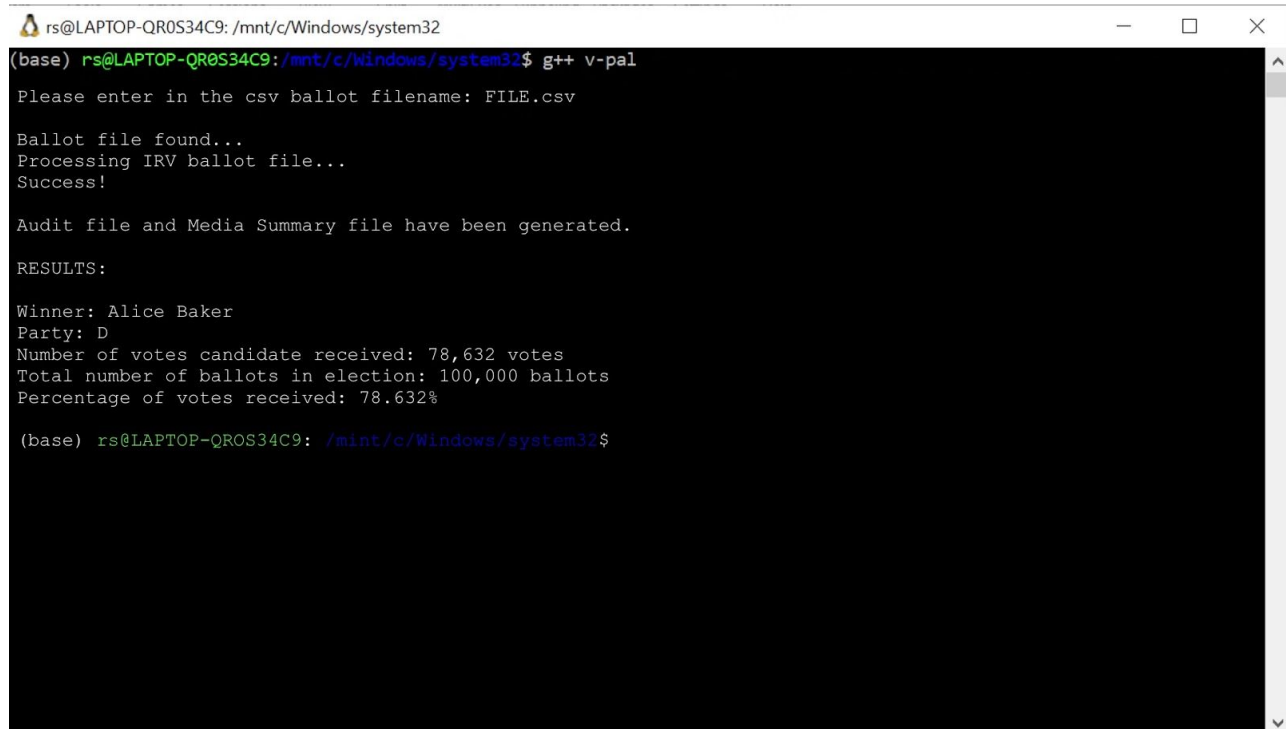
```
rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32
(base) rs@LAPTOP-QR0S34C9:/mnt/c/Windows/system32$ g++ v-pal
Please enter in the csv ballot filename: FILE.csv

ERROR:Ballot file not found
Please make sure that the csv ballot file is in the current directory.

(base) rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32$
```

If the ballot file can not be found in the current directory, then an error message (as shown above), will be displayed. The user should then proceed to check that the csv ballot

file that they are trying to process through V-Pal exists in the current directory and run V-Pal again.



```
rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32
(base) rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32$ g++ v-pal
Please enter in the csv ballot filename: FILE.csv
Ballot file found...
Processing IRV ballot file...
Success!
Audit file and Media Summary file have been generated.
RESULTS:
Winner: Alice Baker
Party: D
Number of votes candidate received: 78,632 votes
Total number of ballots in election: 100,000 ballots
Percentage of votes received: 78.632%
(base) rs@LAPTOP-QR0S34C9: /mnt/c/Windows/system32$
```

If the ballot file was found, then their screen will look like the example above. The above picture shows an example of the results that will be generated for the successful processing of an IRV election ballot.

### 6.3 Screen Objects and Actions

The user needs to run V-Pal in the same directory where the ballot file is being stored, otherwise V-Pal will output an error message on the screen if not found. If the file is found, then a prompt will be displayed: “Ballot file found...”. Another message will be displayed indicating that V-Pal is processing the file: “Processing \_\_\_\_ ballot file...”. If the operation is successful, a “Success!” a message will appear on the screen and that the unique audit and media summary files have been generated. These two documents can be found in the same directory under the unique name (i.e. audit\_file\_{election\_type}\_{date}\_{time}.txt.) It will then process to display the results of the election on the screen.

## 7. REQUIREMENTS MATRIX

SDD Class and Function	Functional Requirement	Location in SRS
Class: Main() function: FindType()	<b>REQ-1:</b> The input file must be provided in csv (comma separated values) format. <b>REQ-2:</b> Input file should have the format as defined in section 3.2 in the SRS. <b>REQ-3:</b> Input data must be stored on the system on which the software is used.	<b>SRS Section 4.1.3 and 4.2.3</b>
Class: Main() function: DisplayResults() Class: Audit function: Print()	<b>REQ-1:</b> The program ran successfully and found a winner <b>REQ-2:</b> The program displays the winner, party, and votes on screen <b>REQ-3:</b> The program generates a summary and audit file	<b>SRS Section 4.3.3</b>