Software Design Document
Voting System
Team 16

Prepared by Adam Wall, Christopher Hanson,
Gabriel Jardio, Patrick Perez
Course CSCI 5801
03/01/2020

1. INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of the STV/Plurality Voting System (STV/P going forward).

The intended audience of this document is both users of the software and also potential developers looking to understand, or expand, its functionality.

1.2 Scope

STV/P is a program that runs elections with STV or plurality voting systems.It seeks to present a simple and intuitive interface to facilitate these elections. It also provides a robust auditing mechanism that clarifies election results, and demonstrates election fairness in a transparent and verifiable manner. The benefit of STV/P is the smooth running of a demonstrably fair election.

1.3 Overview

The remaining sections of this document are outlined below.

The rest of section 1 will contain any useful terms and reference material that will aid in the understanding of this document as a whole.

Section 2 will describe the functionality, context, and design of STV/P

Section 3 outlines the architectural design of STV/P, and the rationale behind the design plan.

Section 4 outlines this program's data structure.

Section 5 will examine each of the software's component parts in a systematic way.

Section 6 will outline the functionality of the system from a user's perspective.

Section 7 will provide a cross reference that traces components and data structures to the requirements in the SRS document.

1.4 Definitions and Acronyms

STV: Single Transferable Vote

STV/P: the STV/Plurality Voting System outlined in this document.

2. System Overview

STV/P is a software focusing strictly on the running and reporting of an STV or plurality election. It provides no mechanism for the organising of an election, or the collecting of an election's votes, focusing instead on taking votes already cast in a pre-arranged election, and determining that election's results. In other words, STV/P is a valuable tool for someone with a clear understanding of what election they are running, and already has the votes on hand which will determine the election results.

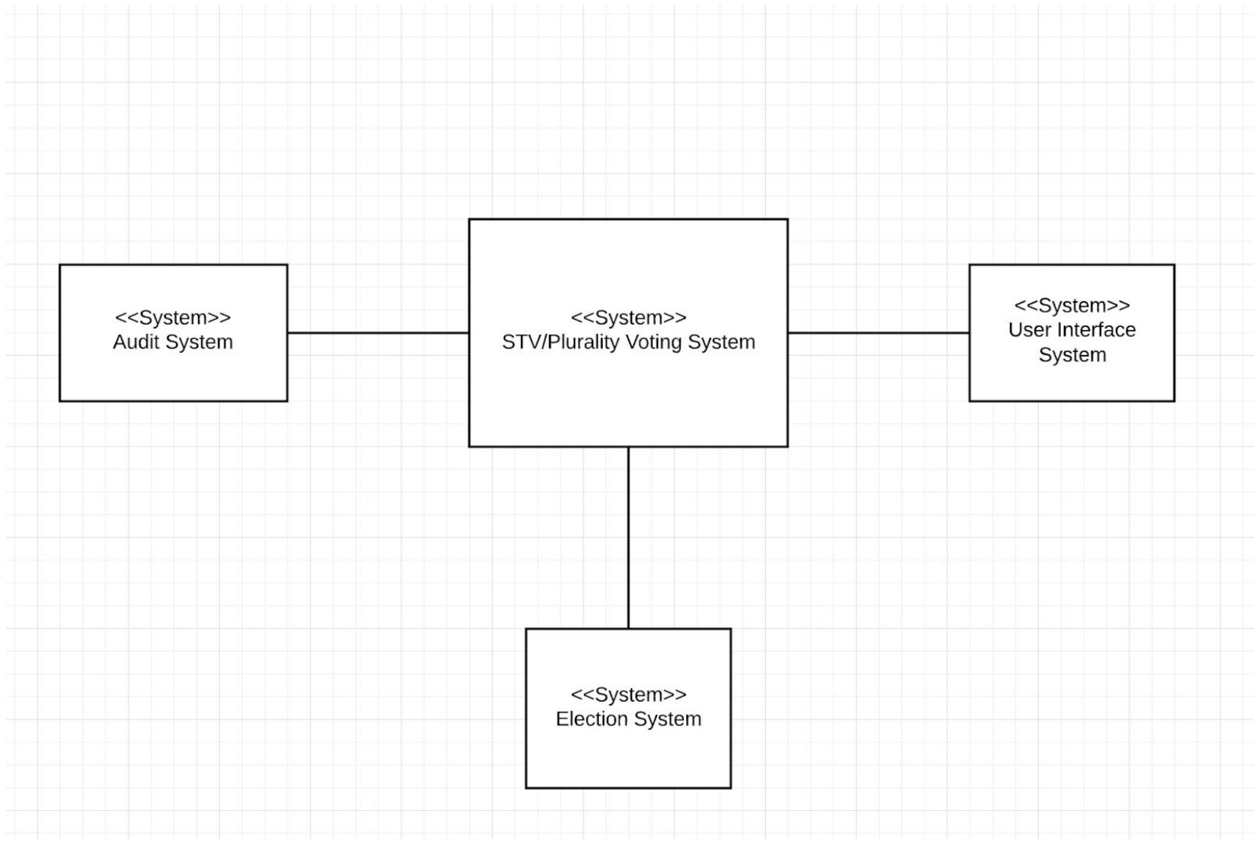
After running the election, STV/P further provides auditing information in the generation of a text file, and election results displayed on the user's screen.

3. System Architecture

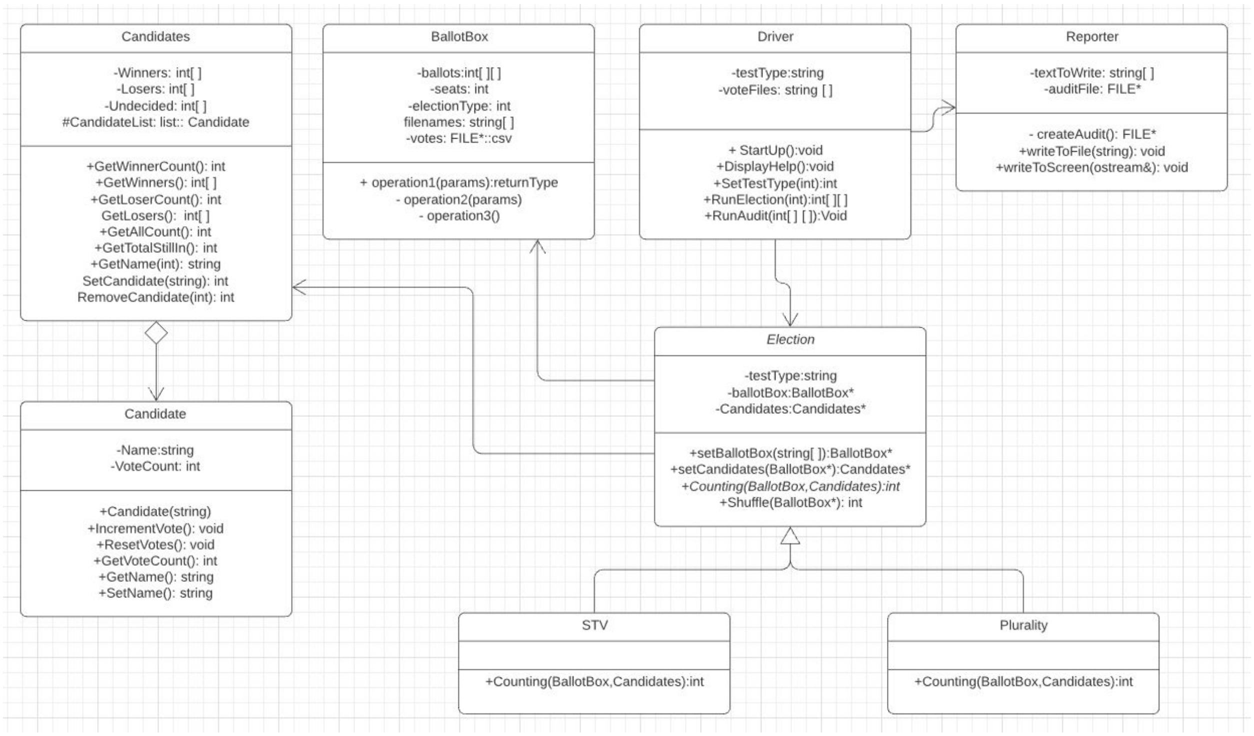
3.1 Architectural Design

STV/P does one thing: it runs an election, and reports its results as it goes. For this reason, STV/P is non-modular, or else mono-modular, in design. The program presents an interface to the user, who uses it to give the program all pertinent information for the running of the election (see section 6). It then runs an algorithm to translate the election input into an election result, loggint its actions to an audit file as it goes. Once the algorithm is complete, the program displays the election results on the user's screen and writes the results to the top of the audit file it has generated.

3.2 Decomposition Description



Subsystem Model



3.3 Design Rationale

STV/P is best understood as one potential module in a larger software structure. Since it's only intended for one purpose, it makes sense to keep it as a module in and of itself. This could cause problems if the intent were to develop a more comprehensive election software that would organize and receive votes in addition to running the election. However, it would make more sense to integrate STV/P into such a program, rather than have it serve as the basis for such a software itself.

4. Data Design

4.1
Data Description

There are three different data types we have in this voting system. The data that has been input by the user through the UI is spread throughout the system as a whole but mostly used within the driver and election class. The main database that will be holding the ballot for each election is stored as a 2D array. This 2D array comes from one test with each vote being a row and each column being a specific candidate to vote for. The counting result along with droop quota is stored in each Candidate object found within the Candidates class. Once an election has been run, the results are stored in a 2D array of integers. No higher level data structure like a hash table or list is needed as only counting is being completed. For a more detailed look at the data structure, look at the UML diagram or section 4.2.

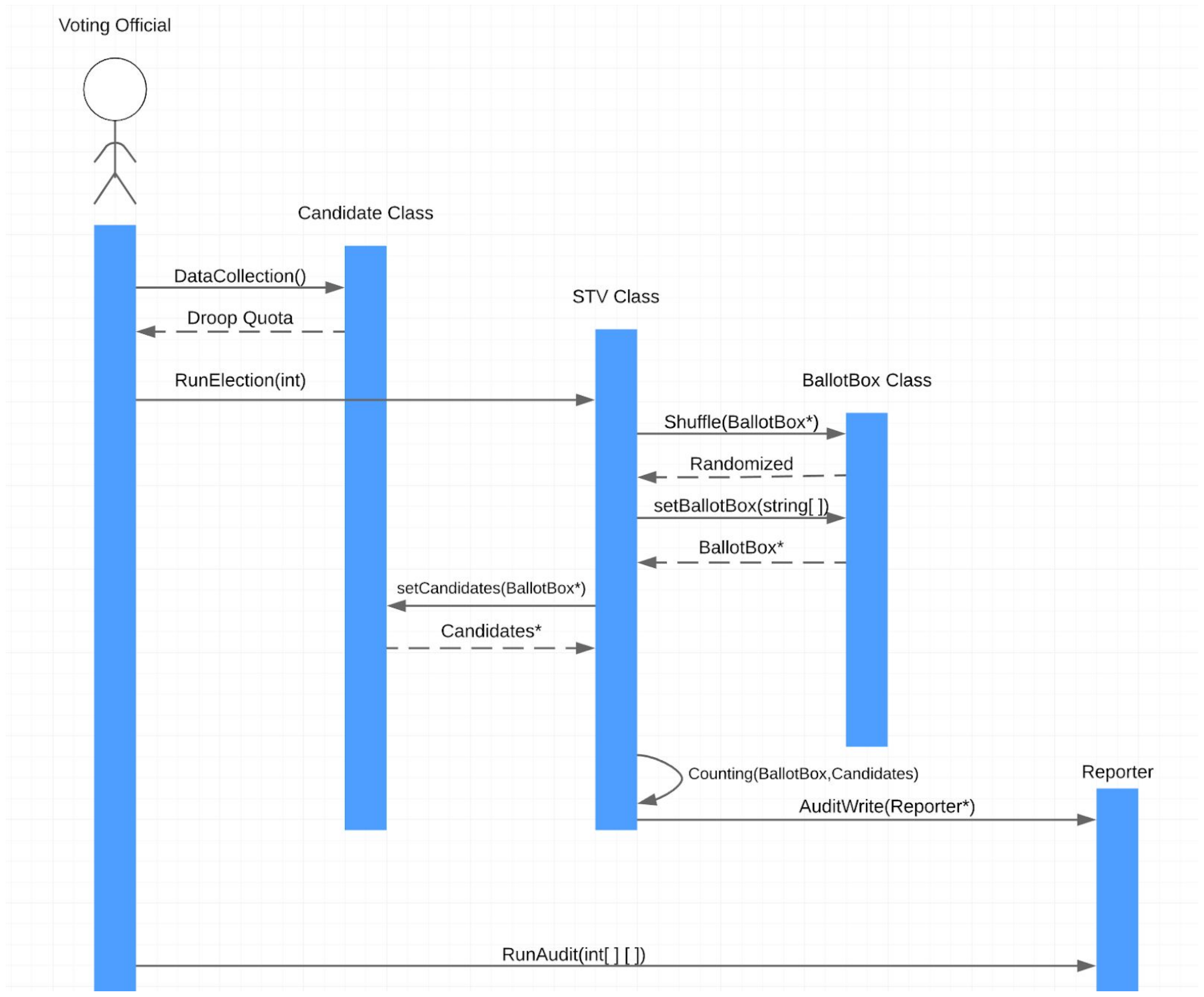
4.2
Data Dictionary

Objects

- Ballotbox Class
    - Contains the filenames of all .CSV files (String)
    - Contains pointers to the same .CSV files
    - Import Data function
        - Moves data from file to 2D array of integers
    - Will be contained individual ballots as a 2D array with each row being a voter, the initial column being the determined recipient of the vote, and each following column being a ranked vote for a candidate (2D array of integers)
- Candidate Class
    - Contains data point of whether a candidate is a winner or loser (possibly more options as code develops) (integer)
    - Contains data point of number of votes it has (integer)
    - Increment vote function
        - Adds vote to a specific candidate
    - Change winner or Loser function
        - Changes status between undetermined (0), winner(1), and loser (-1)
- Candidates Class
    - Contains a list of Candidate objects (see above)
    - Contains winner, loser, and undecided int arrays which have numbers corresponding to candidate order in the basic list
    - Contains data point for droop quota that is compared to the total votes if STV is being used (integer)
- Driver Class
    - Start-Up Function
        - Input output to determine if the user wants to get the help screen or start a election
    - Help Screen Function
        - Displays to the user the help screen
    - Data Collection Function
        - Collects filename for the Ballotbox class, testtype, and droop quota for candidate class
    - Contains testtype as string
        - Used to determine which election inheritance class should be run
    - Run Election function
        - Call election class
    - Call Audit function
        - Run Reporter class

            - Display results to user on screen in UI format
- Election Class
    - Contains 2 inheritance classes - Plurality and STV
        - Plurality
            - Counting Function
                - Runs the plurality voting algorithm that sifts through the 2D array and adds a count to the candidate class for each vote
            - Import Data function
                - Get 2D array from BallotBox class
            - Audit Write function
                - For each vote counted, a datapoint is added to the audit class
        - STV
            - Counting Function
                - Runs STV algorithm that sifts through the 2D array and continues to count until the droop quota (data point in driver class)
                - Once a candidate in STV is a winner, set data point in candidate class to winner
            - Import Data function
                - GEt 2D array from Ballots class
            - Audit write function
                - For each vote counted and droop achieved, a datapoint is added to the audit class to be written
        - These classes contain the random function that can shuffle the ballotbox class
- Reporter Class
    - Contains TextToWrite variable that is an array of strings written from candidate and election class (array of strings)
    - WriteToFile Function
        - Takes what data has been added into the TextToWrite variable and adds it to Audit File
    - Results Function
        - Displays results to use by collecting data from candidate class

5. COMPONENT DESIGN



STV Sequence Diagram

In the diagram above we can see how the different objects utilized in the system interact with each other when STV is used as the election type (refer to sec 3.2 for object diagrams). First the DataCollection function is used to obtain the droop quota for the Candidate class. Then the RunElection function is used in which the STV class shuffles the 2D array ballot data in the BallotBox class, then obtains pointers to both the BallotBox and Candidates class. The STV class then calls the Counting function which will count votes in the BallotBox class and set winners in the Candidate class. The STV class also calls the AuditWrite function after each ballot is read which will send data to the Reporter class. Finally the RunAudit function is called which will print data to a text file based on the data in the Reporter Class.

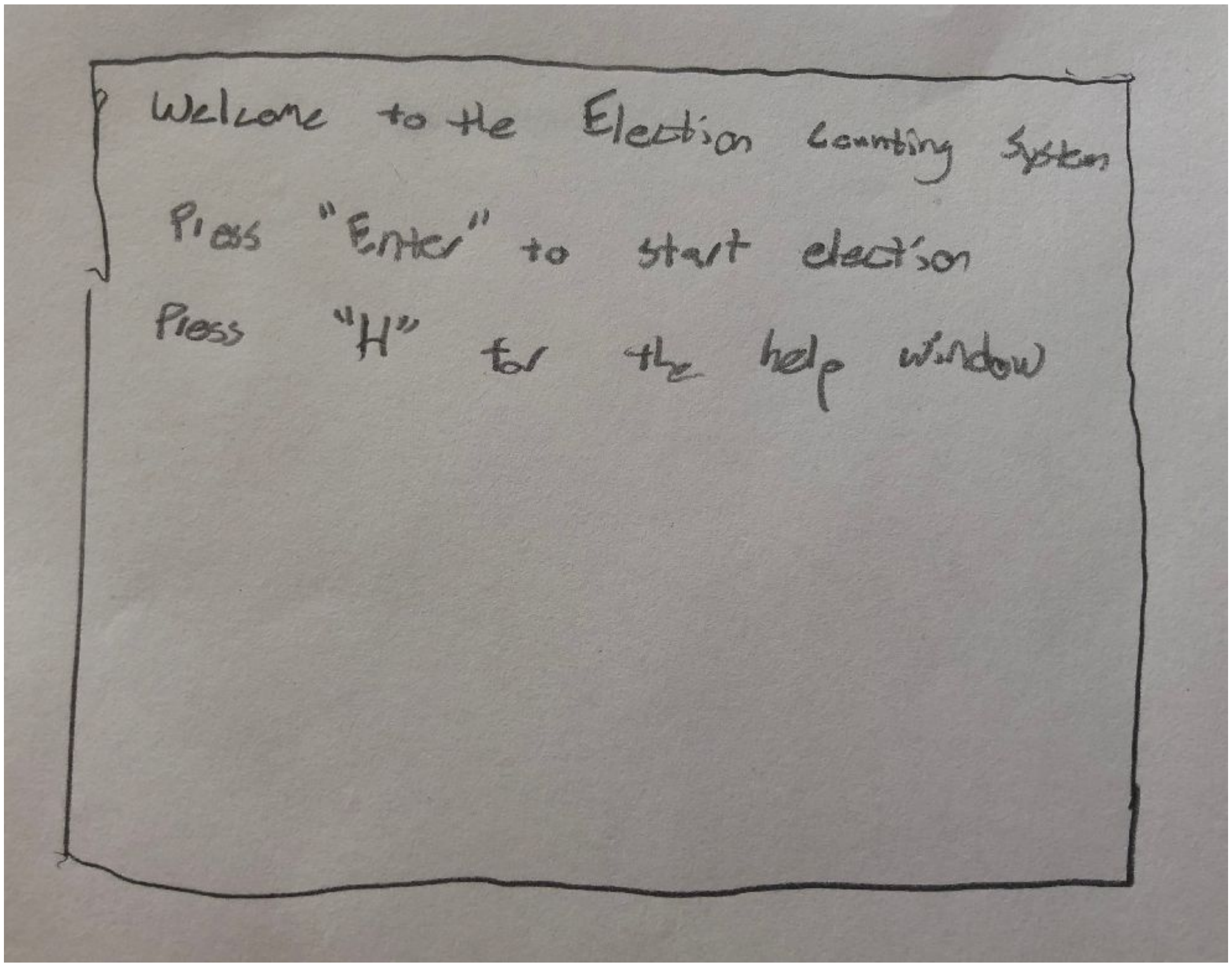
6. HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

This system will take a linear approach overall. First, the user will be prompted with a set of questions in a textual UI format, these questions include file location and what test must be done. These questions will be plainly listed to the user and will let the user know if an incorrect input was entered. The system will then allow the user to reenter the data asked for. Once this data has been entered, an algorithm will run to count and determine a winner. The user will then be shown the results of winners and losers of an election. This section will include a list of the winners and losers and in what order or a total count of their votes. A Help screen can also be reached at the initial screen that will direct the user to show how to work the system.

6.2 Screen Images

Start Window

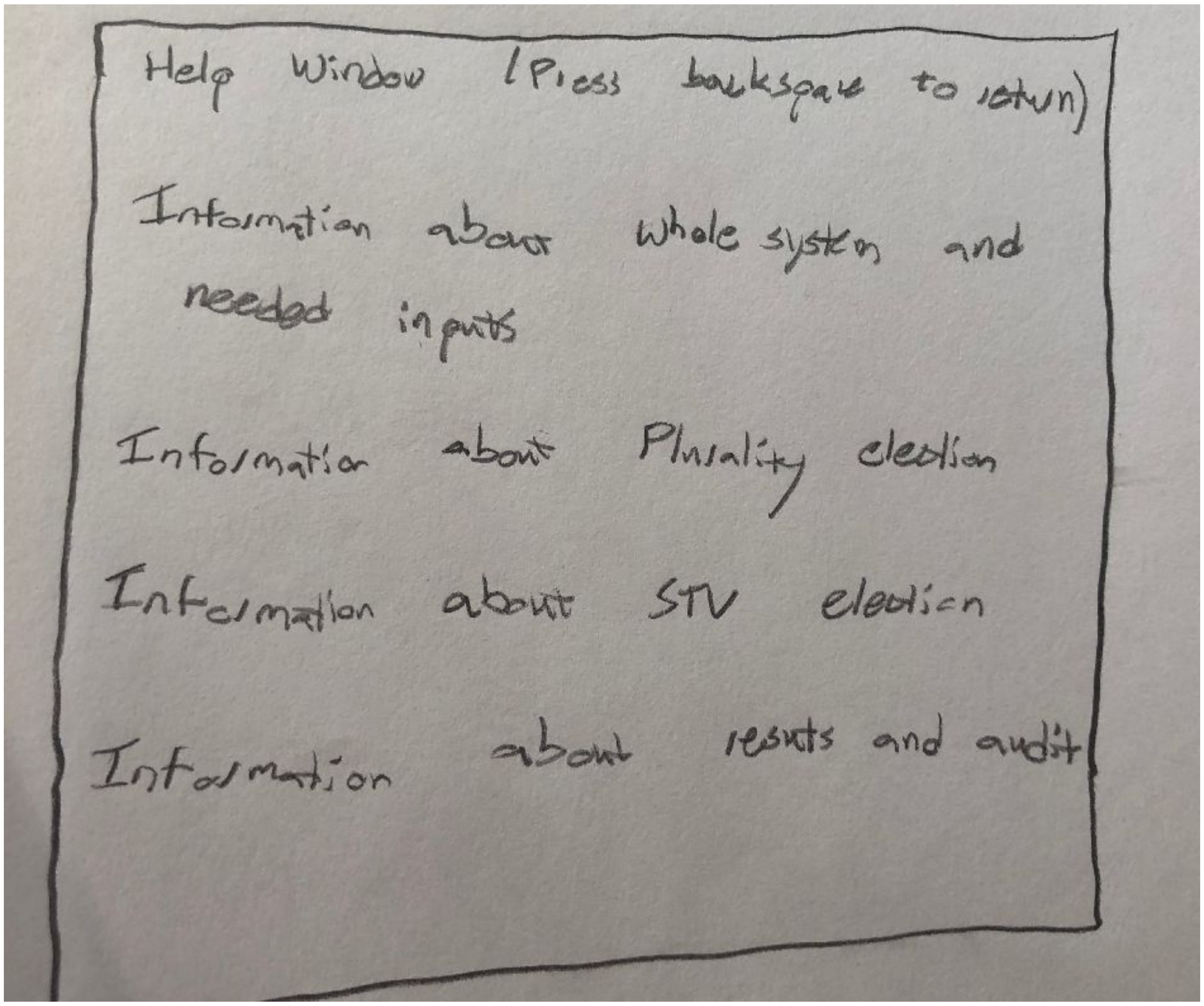


Welcome to the Election Counting System

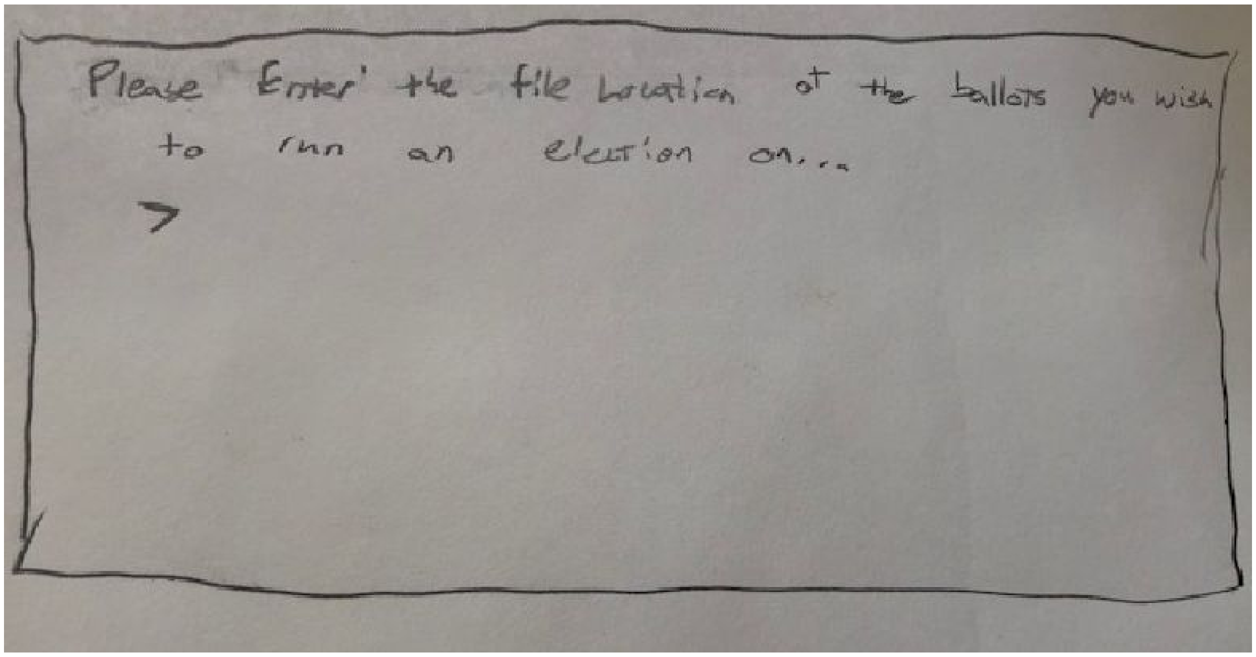
Press "Enter" to start election

Press "H" for the help window

Help Window


Help Window (Press backspace to return)
Information about whole system and needed inputs
Information about Plurality election
Information about STV election
Information about results and audit

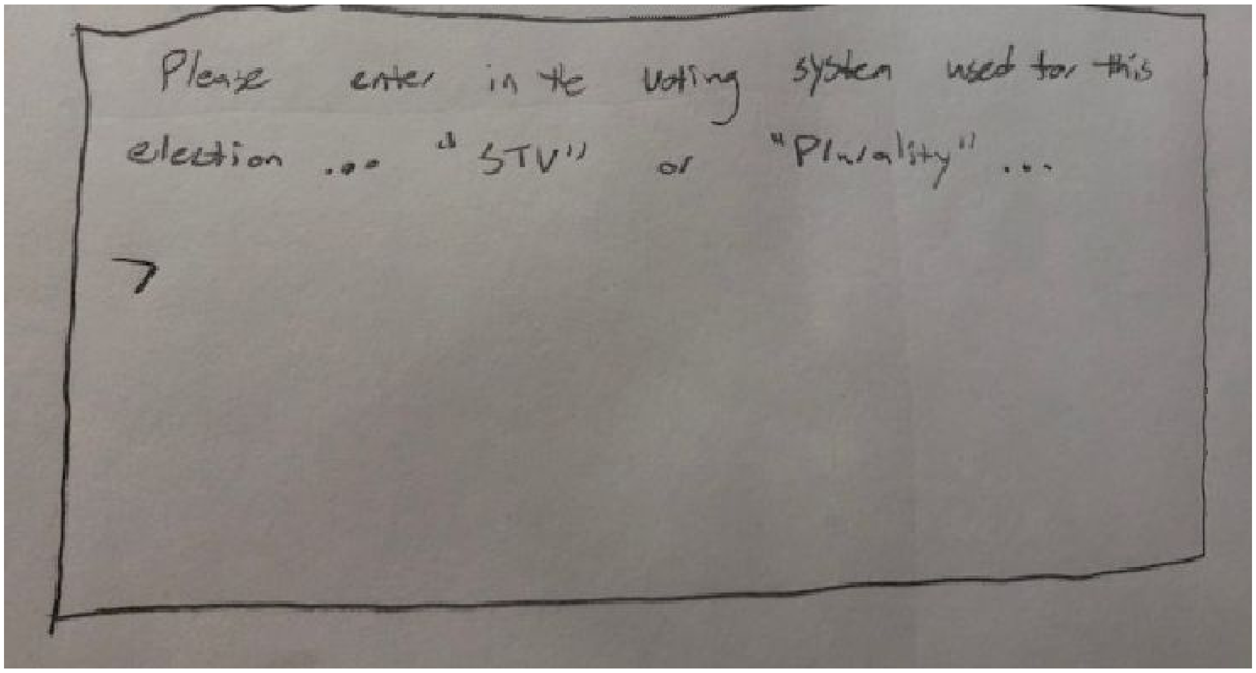
Textual File Location



Please Enter the file Location of the ballots you wish to run an election on...

>

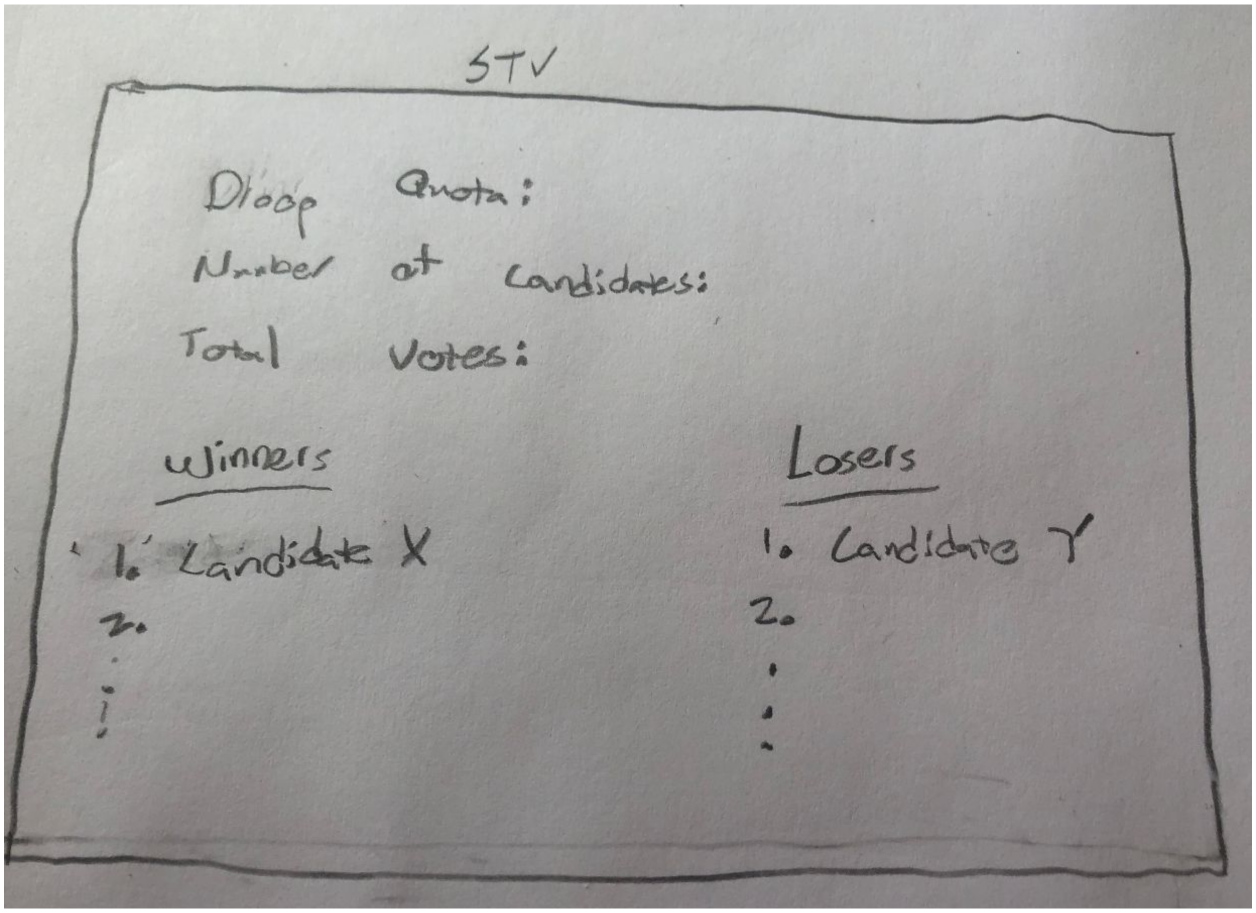
Textual Voting Type



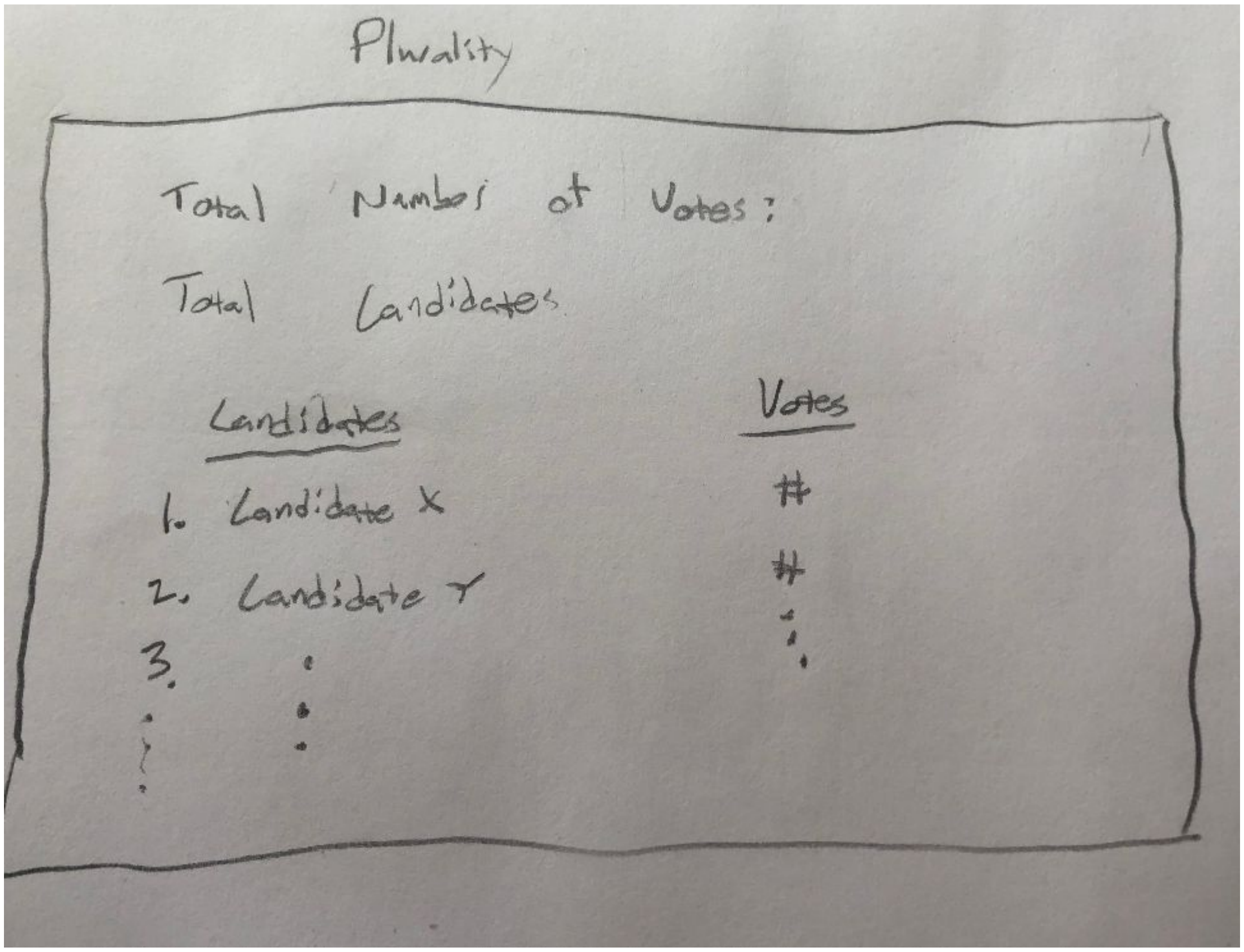
Please enter in the voting system used for this election ... "STV" or "Plurality" ...

>

STV Results Screen



Plurality Results Screen



6.3
Screen Objects and Actions

Action 1 - Start or help

The first screen presented. If the user presses “h” then the help class is kicked off. If the user presses enter, the information gathering begins where the software looks to fill the objects for the voting system and file location.

Object interactions

These textual UI actions are done within the driver class. If the user inputs the string “h” - then the help function will be kicked off showing the user the help window. If the user presses enter, the system will continue down the linear path of the driver class to the next function to obtain data for an election.

Action 2 - Return to menu

While the help class is running, if the user presses the backspace key, returns the user to the main screen to start over again.

Object Interactions

The help function is ended and the system returns to the very first reporter class function of asking a user whether it needs the help screen.

Action 3 - File Location enter

The user will be prompted to enter in the file location. This information once entered will first be checked to determine if a file is in this location, once it has been found it will be added to the election class for the location to import data from.

Object Interactions

The filename will be added to the string in the object driver to find the .CSV file. This file is used in the BallotBox class to use the data within the .CSV file.

Action 4 - Voting Selection

When a file has been selected, the user is asked which specific voting system they want to use. Based on their selection, the algorithm for counting the file is kicked off. No object is changed with this action but a specific class to run (STV or popular vote) is chosen.

Object Interactions

When a voting type has been selected from the reporter object, it sets the voting system and starts that specific election inherited class.

7. Requirements Matrix

Requirement #	Description	System Object	Description of use
4.1.1	User inputs a file address for the software to find and begin counting votes of the file	Driver Class BallotBox Class	Driver class requests the file location from the user and is sent to the ballot box class
4.1.2	The software will notify the user if a	Driver Class	Driver class has the function to get vote file

	wrong file has been selected (such as not a CSV file or a CSV file without the proper		locations and sets it in an array of strings within the driver class. This function will detect if a file cannot find a file and alert the user.
4.1.3	The software allows the user to return to the main screen if the file entered was incorrect	Driver Class	This same function will have a sub function to return the user to the initial screen.
4.2.1	Users must be able to select which voting system they would like - plurality voting being one option	Driver Class Election Class	The system uses the set test type function in the driver class to ask the user to input which test should be performed. The system sends this information to the election class
4.2.2	The software must allow the user to run plurality voting which is done with an algorithm that counts every vote and has the winners with the most votes displayed to the user	Election Class Plurality SubClass BallotBox Class Candidate Class	Within the election class, two inherited classes for counting the votes are used. The Plurality class calls a ballotbox and the candidates to count the total votes and then send the results back to the election and driver class.
4.3.1	Users must be able to select which voting system they would like - STV voting being one option	Driver Class Election Class STV Class	The system uses the SetTestType function to get which function should be used. This value is used in the election class to determine which subclass to use for counting.
4.3.2	The software must allow the user to run an STV voting system. This process will show winners and losers of the election based on the STV algorithm	STV Class Election Class BallotBox Class Candidates Class Candidate Class	The system uses the Counting function within the election-STV class to use a BallotBox object and candidates object to assign Candidate objects to winners and losers. This is displayed in the driver class to the user.

4.3.3	This process is documented so that users understand how exactly a winner was determined	Driver Class Election Class Reporter Class	When an election has been completed, the driver class calls the reporter class with the results from election class to paste to a file.
4.4.1	The software must be running and be in the initial screen	Driver Class	The initial function that is run within the driver class shows a welcome screen along with choosing between help and starting an election.
4.4.2	The software must include a help option in order to access the help window	Driver Class	The driver class contains a displayhelp function to display the help option
4.4.3	The software must be able to display a help window after the help option is selected	Driver Class	This displayhelp function is kicked off by pressing the “h” key.
4.5.1	The software must be able to have access to a text file and print to it.	Driver Class Reporter Class	The reporter class is made in order to access a file for printing and print the results of an election to it
4.5.2	The software must be able to run the two voting algorithms and be able to print to text file at the same time.	Election Class Reporter Class Driver Class	The election results are made with a 2D array that can take multiple ballot boxes and produce multiple results. This can all be taken by the reporter class and print at the same time.