Software Design Document (SDD) Template

Software design is a process by which the software requirements are translated into a representation of software components, interfaces, and data necessary for the implementation phase. The SDD shows how the software system will be structured to satisfy the requirements. It is the primary reference for code development and, therefore, it must contain all the information required by a programmer to write code. The SDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage, i.e. the detailed design stage, more detailed data structures are defined and algorithms are developed for the defined architecture.

This template is an annotated outline for a software design document adapted from the IEEE Recommended Practice for Software Design Descriptions. The IEEE Recommended Practice for Software Design Descriptions have been reduced in order to simplify this assignment while still retaining the main components and providing a general idea of a project definition report. For your own information, please refer to [IEEE Std 1016­1998](http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee-SDD.pdf)1 for the full IEEE Recommended Practice for Software Design Descriptions.

1<http://www.cs.concordia.ca/~ormandj/comp354/2003/Project/ieee>­SDD.pdf

# Team10

**Party List Voting System**

# Software Design Document

Name (s): Wong. A(wang8039), Olson. H (olso7397), Li. G(li000490), Koversky. A(kover002)

Date: (11/04/2019)

1. [INTRODUCTION 2](#_bookmark0)
   1. [Purpose 2](#_bookmark1)
   2. [Scope 2](#_bookmark2)
   3. [Overview 2](#_bookmark3)
   4. [Reference Material 2](#_bookmark4)
   5. [Definitions and Acronyms 2](#_bookmark5)
2. [SYSTEM OVERVIEW 2](#_bookmark6)
3. [SYSTEM ARCHITECTURE 2](#_bookmark7)
   1. [Architectural Design 2](#_bookmark8)
   2. [Decomposition Description 3](#_bookmark9)
   3. [Design Rationale 3](#_bookmark10)
4. [DATA DESIGN 3](#_bookmark11)
   1. [Data Description 3](#_bookmark12)
   2. [Data Dictionary 3](#_bookmark13)
5. [COMPONENT DESIGN 3](#_bookmark14)
6. [HUMAN INTERFACE DESIGN 4](#_bookmark15)
   1. [Overview of User Interface 4](#_bookmark16)
   2. [Screen Images 4](#_bookmark17)
   3. [Screen Objects and Actions 4](#_bookmark18)
7. [REQUIREMENTS MATRIX 4](#_bookmark19)
8. [APPENDICES 4](#_bookmark20)

### INTRODUCTION

## Purpose

Identify the purpose of this SDD and its intended audience. (e.g. “This software design document describes the architecture and system design of XX. ….”).

The purpose of this document is to describe the architecture and system design of Party List Voting System (PLVS). This document is for potential stakeholders, users, and developers of the system.

## Scope

Provide a description and scope of the software and explain the goals, objectives and benefits of your project. This will provide the basis for the brief description of your product.

This document contains a complete description of the design of PLVS. The intent of the PLVS is to serve as a ballot counting program that calculates Closed and Open Party Election results, given a file containing all ballots of the election. Implementing this program will significantly reduce the time it takes to perform ballot-counting compared to people doing it by hand, saving significant amounts of time and money.

The design implements a Model-View-Controller Architecture. A GUI is used as the View, allowing the system to display and read data to and from the user. The Model is the system itself. It does all computations with the data, unseen by the user. The Controller is also a part of the system, translating data to and from the user.

## Overview

Provide an overview of this document and its organization.

Section 2 is a System Overview that describes the functionality and context of the design. This gives more details to the purpose and implementation of the design.

Section 3 discusses the System Architecture and is divided into three sections. The first section is the Architectural Design breaks the design into a modular program structure and describes the relationships between the modules. Next is the Decomposition Description section, which provides a decomposition of the sub systems in the architectural design. The last part is the Design Rationale, which discusses the reasoning behind the chosen architecture and its pros/cons.

Section 4 is the Data Design which describes how system information is described as data structures. Additionally, this section contains a Data Dictionary of all system entities and major data elements and their descriptions.

Section 5 contains the Component Design which goes into detail about the components systematically.

Section 6 discusses the Human Interface Design, which involves detailing the GUI for this design

Section 7 includes the Requirements Matrix, which details the relationships between the components and data structures of the system with the requirements detailed in the Software Requirements Specification document.

Section 8 contains the Appendix.

## Reference Material

*This section is optional.*

List any documents, if any, which were used as sources of information for the test plan.

Wong. A, Olson. H, Li. G, Koversky. A. (2019). *Software Requirements Specification for Part List Voting System.* Unpublished Document, University of Minnesota.

Software Design Document (SDD) Template. Retrieved from: https://canvas.umn.edu/courses/ 134519/files?sort=created\_at&order=desc

## Definitions and Acronyms

*This section is optional.*

Provide definitions of all terms, acronyms, and abbreviations that might exist to properly interpret the SDD. These definitions should be items used in the SDD that are most likely not known to the audience.

1. Voter - A member of the electorate casting a ballot in an election.
2. Candidate - A member of a political party running for office
3. Political Party (Party) - An organization of candidates and government officials sharing common ideals and working together to actualize them.
4. Ballot - The record of an individual vote.
5. Tie - The event where multiple different candidates or parties have the same number of votes.
6. CSELabs - The University of Minnesota's Science of College and Engineering computer lab machines, as well as what can be emulated on https://vole.cse.umn.edu.
7. GB - Acronym representing Gigabyte
8. PLVS - Acronym representing Party List Voting System
9. Party-List Voting (PLV) - A voting algorithm allowing political parties to select a candidate for each electoral district and allows each voter to vote for a party. Political seats are then assigned based the proportion of the vote each party receives and the order of each party's candidates.
   1. Closed-Party-List Voting (CPLV) - A variant of the PLV algorithm in which parties decide the order of their candidates prior to election, and the voters vote for a party.
   2. Open-Party-List Voting (OPLV) - A variant of the PLV algorithm in which the order of a party's candidates is determined by the voters, who vote for a single candidate. This vote also counts towards that candidate's party.
10. Graphical User Interface (GUI) – A user interface that contains icons, images, and menus that the user can interact with to manipulate the system.

### SYSTEM OVERVIEW

Give a general description of the functionality, context and design of your project. Provide any background information if necessary.

The PLVS software seeks to improve the election process used in United States (US) elections by implementing party list voting. Currently US elections use a form of plurality algorithm is used in many US elections where each voter votes for a single candidate for each seat, and the candidate with the most total votes wins. Party list voting, on the other hand, has each voter for a party and then appoints the top X candidates of each party to seats, where X is the proportion of the total available seats relative to the proportion of the vote earned by each party. Two types of party list voting will be supported by the PLVS. "Closed-party-list" voting (CPLV) allows each party to decide the order of their candidates, and each voter to vote for a party. "Open-party-list" voting (OPLV) allows the voter to pick a candidate (and their party) and determines the order of the candidates from the results of the vote.

The PLVS software will have the functionality to read in a file containing all of the necessary information to conduct the election as well as the ballots, and the functionality to write out an audit log once an election has concluded. The system will, once a file has been inputted, be able to determine whether it should be processed as a closed or open party list election and should calculate the correct election results from the information in the input file. These results should then be displayed to the user, and the option to create a media output file should be presented.

The user classes of the PLVS software will consist of programmers, testers, and government officials. To accommodate these classes, the system will use a GUI with three separate screens; a file input screen, a process input screen, and a display results screen. It will be programed in the Java language and will require the Java runtime environment to operate.

The PLVS system will be designed according to the Model-View architecture, with the GUI providing the view and the vote processing algorithms, file input, and audit file/media report output providing the model. The model will be made up of a VoteProcessor interface with two child classes, ClosedPartyProcessor and OpenPartyProcessor. It will also contain AuditFile and MediaReport classes that perform all required functionality for the audit file and media report respectively. Finally, there will be a PLVS class that acts as a controller and transfers data between the model and view.

### SYSTEM ARCHITECTURE

## Architectural Design

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high level overview of how responsibilities of the system were partitioned and then assigned to subsystems. Identify each high level subsystem and the roles or responsibilities assigned to it. Describe how these subsystems collaborate with each other in order to achieve the desired functionality. Don’t go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together. Provide a diagram showing the major subsystems and data repositories and their interconnections. Describe the diagram if required.

## Decomposition Description

Provide a decomposition of the sub systems in the architectural design. Supplement with text as needed. You may choose to give a functional description or an object-­oriented description. For a functional description, put top ­level data flow diagram (DFD) and structural decomposition diagrams. For an OO description, put sub system model, object diagrams, generalization hierarchy diagram(s) (if any), aggregation hierarchy diagram(s) (if any), interface specifications, and sequence diagrams here.

## Design Rationale

Discuss the rationale for selecting the architecture described in 3.1 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn’t choose them.

### DATA DESIGN

## Data Description

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed and organized. List any databases or data storage items.

## Data Dictionary

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description in Section 3.2, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

### COMPONENT DESIGN

In this section, we take a closer look at what each component does in a more systematic way. If you gave a functional description in section 3.2, provide a summary of your algorithm for each function listed in 3.2 in procedural description language (PDL) or pseudocode. If you gave an OO description, summarize each object member function for all the objects listed in 3.2 in PDL or pseudocode. Describe any local data when necessary.

### HUMAN INTERFACE DESIGN

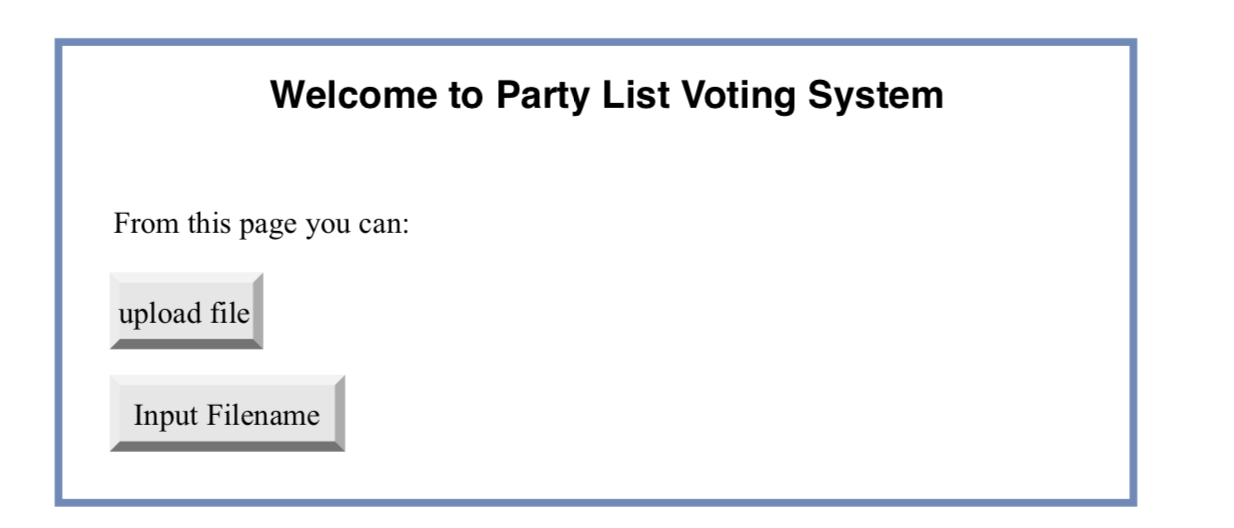
## Overview of User Interface

Describe the functionality of the system from the user’s perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

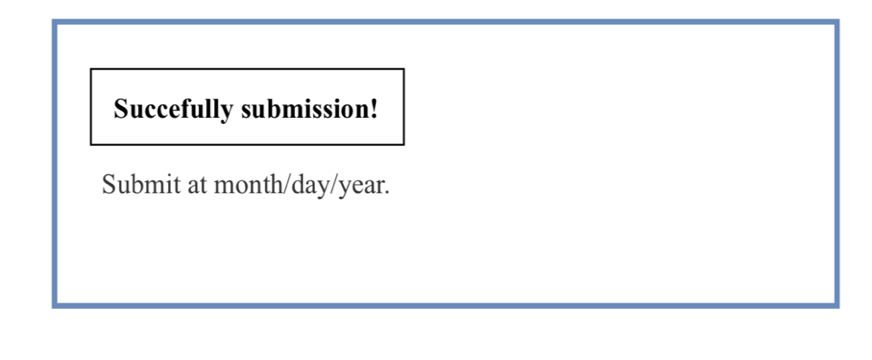
The interface will use the same color and design schema used on the vote process software to create a continuous effect. Users will begin by launching the application from the command prompt, after which it will open a GUI window to prompt the user to input a filename which they want to open. Once the user is satisfied with the input and clicks the OK button, the system starts read the data in the file and process it to output an audit file. Once processed, the system displays to the screen the winner(s) and election information.

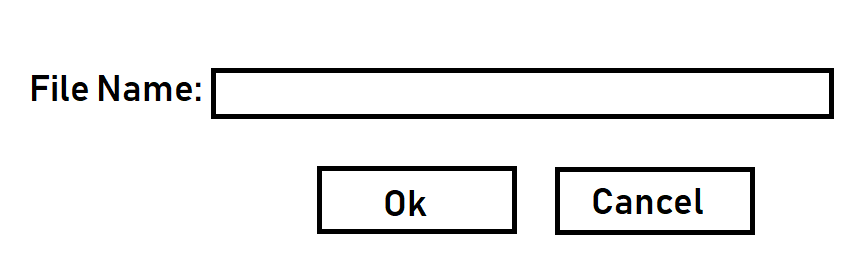
## Screen Images

Display screenshots showing the interface from the user’s perspective. These can be hand­ drawn or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)



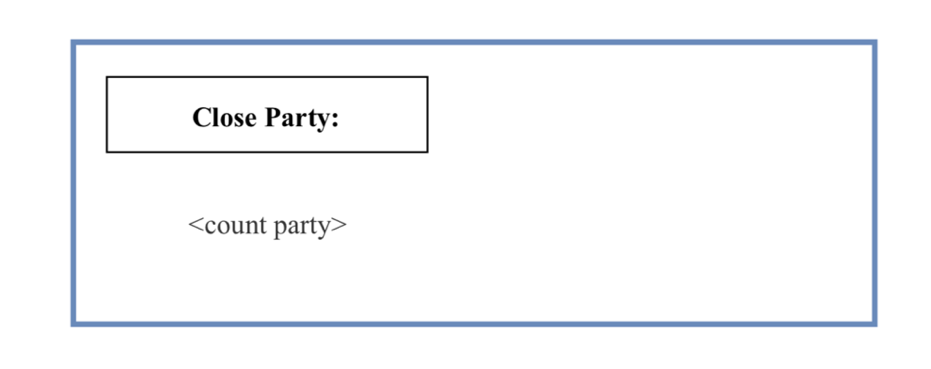
**Figure** Welcome page



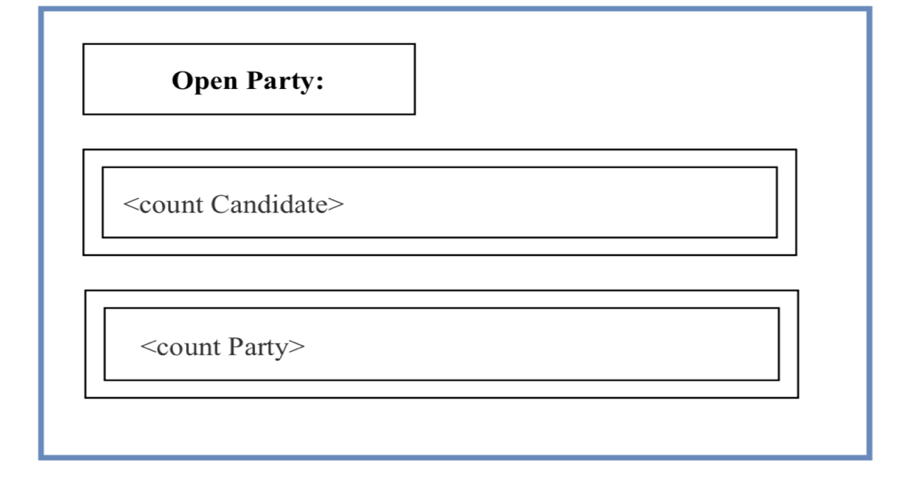
**Figure** Submission page

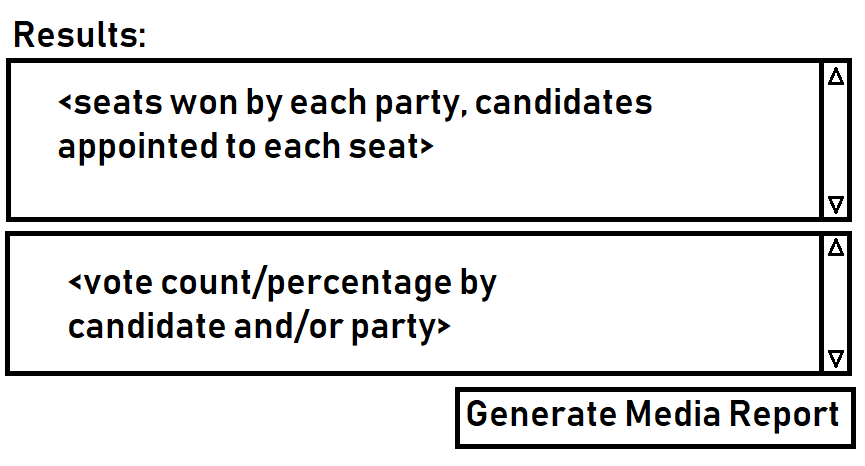
## Figure Prompt user to enter a filename.

## Figure Prompt to start processing the data in the input file.

****

* Figure Processing Close Party

****

* Figure Processing Open Party.
* Figure Display results.

## 

## Figure Choose Media File Type.

## Screen Objects and Actions

A discussion of screen objects and actions associated with those objects.

* + - **User open the software**
    - **Software let users choose input filename or upload a file**

----Check for the exist.

* + - **System reads the file**

----Get voting type of the file (Closed or Open Vote type).

----Get Parties.

----Get candidates.

* + - **System process the voting result**

----For the open party list, account the candidates and the parties.

----For the closed party, only account for the parties.

* + - **System determines winner**

---- Top X candidates of each party where x is the proportion of the vote captures.

* + - **Display results**

---- Prompt for the media summary.

* + - **Choose media file type**

---- Choose generate file with type txt.

### REQUIREMENTS MATRIX

Provide a cross­ reference that traces components and data structures to the requirements in your SRS document.

Use a tabular format to show which system components satisfy each of the functional requirements from the SRS. Refer to the functional requirements by the numbers/codes that you gave them in the SRS.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  | |
| Requirements Traceability Matrix | | | | | |
| Project Name: | Waterfall Project ( Voting System ) | | | | |
| Request No. | Req Desc | SRS Requirements | | SDD Requirements | |
| REQ-1 | The system needs to read in the input file. | 4.1, 4.2 | |  | |
| REQ-2 | The system needs to determine that the election is uses CPLV from the input file. | 4.1, 4.2 | |  | |
| REQ-3 | The system needs to determine that the election is uses OPLV from the input file. | 4.2 | |  | |
| REQ-4 | The system needs to read in the number of seats, parties, ballots, candidates' names and affiliations, and the ballots themselves from the input file. | 4.1, 4.2 | |  | |
| REQ-5 | The system needs to tally the vote for each party as each ballot is processed | 4.1, 4.2 | |  | |
| REQ-6 | The system needs to update the audit log as each ballot is processed (see TBD) | 4.1, 4.2 | |  | |
| REQ-7 | The system needs to determine and display the results of the election once all ballots are processed | 4.1, 4.2 | |  | |
| REQ-8 | Ties should be broken using random number generation | 4.2 | |  | |
| REQ-9 | An editable text file needs to be produced. | 4.3 | |  | |
| REQ-10 | The new text needs to be labelled “Audit.txt” | 4.3 | |  | |
| REQ-11 | Logs from processes 4.1 and 4.2 need to be detailed in new lines of the audit file. | 4.4 | |  | |
| REQ-12 | Log entries must be separated by a new line. | 4.4 | |  | |
| REQ-13 | The system receives order to create media summary | 4.5 | |  | |
| REQ-14 | The system needs to determine that the election in the input file uses OPL or CPL | 4.5 | |  | |
| REQ-15 | The system needs to read seats, parties, votes, candidate names and affiliations, and the number of votes themselves from the input file. | 4.5 | |  | |
| REQ-16 | The system will determine the final list of candidates based on different types of election rules | 4.5 | |  | |
| REQ-17 | The system creates final poll result. | 4.5 | |  | |
| REQ-18 | The system needs to open the input file. | 4.6 | |  | |
| REQ-19 | The system needs to check that the file name is that of a file that exists. | 4.7 | |  | |

### APPENDICES

*This section is optional.*

Appendices may be included, either directly or by reference, to provide supporting details that could aid in the understanding of the Software Design Document.