

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/ieeeSDD.pdf)¹ for the full IEEE Recommended Practice for Software Design Descriptions.

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

Team 21
Voting System
Software Design Document

Name (s): Conor Brown (brow4339), Jack Soderwall (soder365), Joe Cassidy (cassi083), Sean Carter (carte899)

Lab Section: N/A

Workstation: N/A

Date: 2/23/2021

TABLE OF CONTENTS

1. INTRODUCTION

- 1.1 Purpose
- 1.2 Scope
- 1.3 Overview
- 1.4 Reference Material
- 1.5 Definitions and Acronyms

2. SYSTEM OVERVIEW

3. SYSTEM ARCHITECTURE

- 3.1 Architectural Design
- 3.2 Decomposition Description
- 3.3 Design Rationale

4. DATA DESIGN

- 4.1 Data Description
- 4.2 Data Dictionary

5. COMPONENT DESIGN

6. HUMAN INTERFACE DESIGN

- 6.1 Overview of User Interface
- 6.2 Screen Images
- 6.3 Screen Objects and Actions

7. REQUIREMENTS MATRIX

8. APPENDICES

1. INTRODUCTION

1.1 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 provide a detailed description of the Voting System architecture and design.

This document is intended for the CSCI 5801 instruction staff at The University of Minnesota. It will also be referenced by our team when implementing the system and possibly making changes.

1.2 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.

The system will be accessed by programmers, testers and election officials. Programmers will write and test the code. Testers will test code intermittently throughout the design process. Election officials will use the final product to conduct vote tallying.

The goal is to produce election results in a clean and timely manner, while maintaining well written documentation and code. More specifically, the key objective is to process 100,000 ballots in under 8 minutes.

1.3 Overview

Provide an overview of this document and its organization.

This document details the architectural design our group decided upon to create a modular program structure. A UML for the system as a whole is provided along with UML activity diagrams for both Instant Runoff voting and Open Party Listing. A sequence diagram detailing Open Party Listing is provided. Data structures and all major data is described in section 4. Section 5 contains pseudocode or summaries of all algorithms and functions. In section 6 an overview of the user interface is provided. Finally, this document and the functionality it describes is cross referenced with our Software Requirement Specifications document to show where use cases are being addressed.

1.4 Reference Material

This section is optional.

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

1.5 Definitions and Acronyms

All diagrams are produced using UML specifications, where UML stands for Unified Modeling Language. Other than that, there should not be any acronyms that hinder the audience from interpreting this document.

2. SYSTEM OVERVIEW

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

The Voting System will be able to produce election results for the Instant Runoff voting model and Open Party Listing model. The Instant Runoff model asks voters to rank the candidates in order of preference. If a voter's top preference is eliminated by having the least number of votes, that voter's next best choice receives their vote. Candidates are eliminated until one candidate achieves a majority. In the Open party listing model a voter is able to vote once. That vote counts for one candidate and their party. In Open Party Listing there are a required number of seats to be populated by candidates. Based on the number of seats available and number of ballots cast determines the quota needing to be filled in order for a political party to acquire a seat. If seats remain after counting each party's acquired seats, the remaining seats are awarded to parties based on remaining votes that did not fulfill the quota. Party's seats are given to the candidates with the most votes in each corresponding party.

The election in question will have a corresponding ballots csv file. This file is provided through a command line argument in a terminal when the voting system is run by an election official. At termination a summary of the election will be displayed on screen. Additionally, an audit file with a unique name containing more detailed information about the ballot tallying process will be produced. A media file will be created to provide to media members, containing a summary of the election.

Our team's solution design will require only one program to be run at the terminal to compute both Instant Runoff and Open Party Listing results. The design will feature a modularized model to enhance functionality, readability and efficiency.

3. SYSTEM ARCHITECTURE

3.1 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.

3.2 Decomposition Description

Provide a decomposition of the subsystems 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 subsystem model, object diagrams, generalization hierarchy diagram(s) (if any), aggregation hierarchy diagram(s) (if any), interface specifications, and sequence diagrams here.

*Put class diagram and sequence diagram here

3.3 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.

4. DATA DESIGN

4.1 Data Description

Our system takes in a filename that starts the main process of our program. The crux of the system was designed around maintaining as much information within one structure as possible. In doing this, the main data structures developed were Candidate, Party, and Ballot classes that are all made use of within the larger ElectionData class. The ElectionData class is the primary container for storage of election information, and when files are parsed to generate election statistics and results,

4.2 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.

<u>Attribute Name</u>	<u>Attribute Type</u>	<u>Attribute Size</u>

5. 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.

*Provide English summary of functions here

6. HUMAN INTERFACE DESIGN

6.1 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.

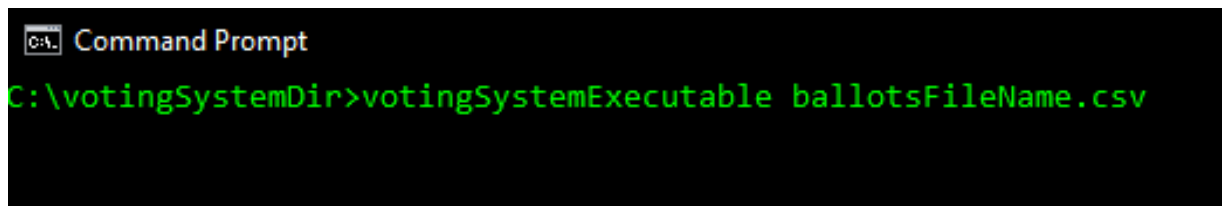
Election officials and testers will be able to run the Voting System from a terminal on a Linux, Mac or Windows machine with the specified hardware and software described in the System

Requirements Specifications. The user should navigate to the correct directory where the program and ballots file are stored. To run the program they will type in the executable name and provide the ballots file name as a command line argument. A confirmation message will be provided if the ballots tallying has successfully begun. If not an error message will appear asking them to re-enter the file name and to confirm it is within the same directory as the program. While ballots are being tallied, the user may suspend or terminate the system with a Ctrl + C or Ctrl + Z command within the terminal. Signal handling will be handled gracefully, providing a response message with process details prior to the signal. If there is an error within the ballots file itself or any unexpected error is encountered, an error message will be printed to the terminal describing the error and the process will terminate. Upon successful completion of the program a summary of the election results will be printed to the terminal. Additionally, an audit file containing a detailed description of the tallying process will be produced with a unique filename based on the time the program was executed. A media file will be generated to be distributed to media outlets by an election official with information about the election. The user can find the audit file within the same directory as the executable and the ballots file.

6.2 Screen Images

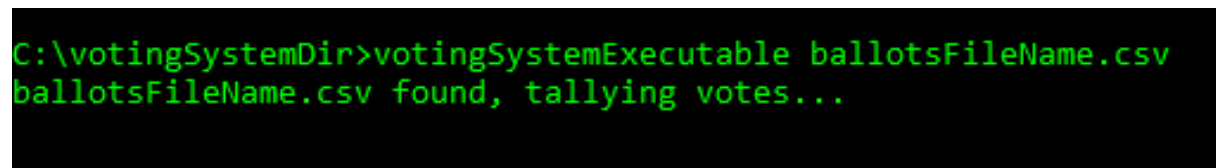
Display screenshots showing the interface from the user's perspective. These can be hand drawn or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)

Figure 1



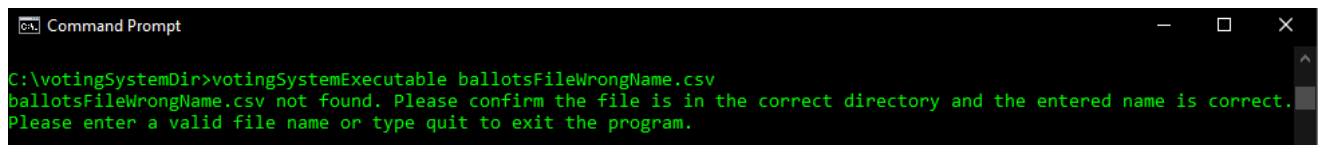
```
Command Prompt
C:\votingSystemDir>votingSystemExecutable ballotsFileName.csv
```

Figure 2



```
C:\votingSystemDir>votingSystemExecutable ballotsFileName.csv
ballotsFileName.csv found, tallying votes...
```

Figure 3



```
Command Prompt
C:\votingSystemDir>votingSystemExecutable ballotsFileWrongName.csv
ballotsFileWrongName.csv not found. Please confirm the file is in the correct directory and the entered name is correct.
Please enter a valid file name or type quit to exit the program.
```


Figure 4

```
C:\votingSystemDir>votingSystemExecutable ballotsFileName.csv
ballotsFileName.csv found, tallying votes...
^Z
SIGSTOP encountered, current election information:
Election Type: Instant Runoff
Candidates Eliminated: George Washington, Teddy Roosevelt
Total ballots: 55,000
Current standings: Franklin D. Roosevelt: 47%, Bill Clinton: 20%, Ronald Reagan 23%
Program terminating...
```

Figure 5

```
C:\votingSystemDir>votingSystemExecutable ballotsFileName.csv
ballotsFileName.csv found, tallying votes...
ERROR: IMPROPER FORMATTING in ballotsFileName.csv at line 10,697
"4,,8,15,,,,16,,,,23,,42"
PROGRAM TERMINATING
```

Figure 6

```
C:\votingSystemDir>votingSystemExecutable ballotsFileName.csv
ballotsFileName.csv found, tallying votes...
Program completed successfully.

Election Summary
Election Type: Instant Runoff
Winner: Joe Biden, with 78 % of the vote
Elimination order: (1) Jimmy Carter, (2) Woodrow Wilson, (3) Andrew Jackson
Total Ballots Counted: 99,999

An audit file with the name 2/24/2021-1:20AM has been produced in votingSystemDir
```

Figure 7

Election Type: Instant Runoff
Number of ballots: 99,999
Number of Candidates: 5
Candidates: Joe Biden, Woodrow Wilson, Andrew Jackson, George Bush, Jimmy Carter

No majority

Elim 1:
Joe Biden: 30,000 votes
Woodrow Wilson: 9,999 votes
Andrew Jackson: 30,000 votes
George Bush: 30,000 votes
Jimmy Carter: 0 votes

Jimmy Carter Eliminated, redistributing votes...

Joe Biden: 30,000 + 0 votes
Woodrow Wilson: 9,999 + 0 votes
Andrew Jackson: 30,000 + 0 votes
George Bush: 9,999 + 0 votes

No majority

Elim 2:
Joe Biden: 30,000 votes
Woodrow Wilson: 9,999 votes
Andrew Jackson: 30,000 votes
George Bush: 30,000 votes

Woodrow Wilson Eliminated, redistributing votes...

Joe Biden: 30,000 + 9,998 votes
Andrew Jackson: 30,000
George Bush: 30,000 + 1 vote

No majority

Elim 3:
Joe Biden: 39,998 votes
Andrew Jackson: 30,000 votes
George Bush: 30,001 votes

Andrew Jackson Eliminated, redistributing votes...

Joe Biden: 39,998 + 30,000 votes
George Bush: 30,001 + 0 votes

Majority found, Winner declared:

Joe Biden: 69,998 votes
George Bush 30,001 votes

Figure 8

2/25/2021 Instant Runoff Election Results

Candidates (5): Joe Biden, Woodrow Wilson, Andrew Jackson, George Bush, Jimmy Carter

Number of ballots counted: 99,999

Winner: Joe Biden with majority of 51% of the vote

Elimination order

-----|

(1) Woodrow Wilson, (2) Andrew Jackson, (3) Jimmy Carter

Final votes garnered by each remaining candidate

Joe Biden: 51,000

George Bush: 48,999

6.3 Screen Objects and Actions

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

Figure 1 shows the basic execution of the program in a Windows Terminal. This is how the user starts the program.

Figure 2 shows the initial message produced upon the program successfully finding the file. This confirms to the user that the file was found and tallying has begun. No immediate action is needed.

Figure 3 shows an error message if the file name could not be found. The user should now confirm that the file is in the correct directory and they have entered the correct file name. The user can either enter another file name or type quit to terminate the program.

Figure 4 shows an interaction between the program and a signal stop. If the user decides to stop the program using Ctrl + Z, the program will display the results it found and gracefully terminate. Data will be lost and the program may now be run again from the command line using the initial startup command.

Figure 5 shows an error message printed to the terminal if an error occurs within the ballots file. The user must now fix the file to be able to complete the program and receive election results.

Figure 6 shows successful termination of the program. The user may now take note of the results and inspect the audit file.

Figure 7 is an example of an audit file. The user may inspect the audit file for any computation errors or other discrepancies.

Figure 8 is an example of a media file that can be distributed to various media outlets by an election official.

7. 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.

8. 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.

