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/%7Eormandj/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](http://www.cs.concordia.ca/%7Eormandj/comp354/2003/Project/ieee)­SDD.pdf

Team-12

**Voting System**

Software Design Document

Author(s):

*Rex Zhu(zhu00100), Sunny Qin(qing0002),*

*Yingjin Zhang(zhan4943), Xiaohui Chao(chao0070)*

Workstation: CSE LAB MACHINE LIND40-02

Date: (11/01/2019)

**TABLE OF CONTENTS**

1. [**INTRODUCTION 2**](#_heading=h.gjdgxs)
   1. [Purpose 2](#_heading=h.30j0zll)
   2. [Scope 2](#_heading=h.1fob9te)
   3. [Overview 2](#_heading=h.3znysh7)
   4. [Reference Material 2](#_heading=h.2et92p0)
   5. [Definitions and Acronyms 2](#_heading=h.tyjcwt)
2. [**SYSTEM OVERVIEW 2**](#_heading=h.3dy6vkm)
3. [**SYSTEM ARCHITECTURE 2**](#_heading=h.1t3h5sf)
   1. [Architectural Design 2](#_heading=h.4d34og8)
   2. [Decomposition Description 3](#_heading=h.2s8eyo1)
   3. [Design Rationale 3](#_heading=h.17dp8vu)
4. [**DATA DESIGN 3**](#_heading=h.3rdcrjn)
   1. [Data Description 3](#_heading=h.26in1rg)
   2. [Data Dictionary 3](#_heading=h.lnxbz9)
5. [**COMPONENT DESIGN 3**](#_heading=h.35nkun2)
6. [**HUMAN INTERFACE DESIGN 4**](#_heading=h.1ksv4uv)
   1. [Overview of User Interface 4](#_heading=h.44sinio)
   2. [Screen Images 4](#_heading=h.2jxsxqh)
   3. [Screen Objects and Actions 4](#_heading=h.z337ya)
7. [**REQUIREMENTS MATRIX 4**](#_heading=h.3j2qqm3)
8. [**APPENDICES 4**](#_heading=h.1y810tw)

1. **INTRODUCTION**
   1. **Purpose**

The Software Requirement Specification comprises of concrete information about Voting System which includes its intended use, architecture design, deployment method, and other relevant resources. This document faces users and developers interacted with Voting System.

* 1. **Scope**

Countries and regions require a more and more transparent, efficient, and scientific method to hold elections. Under this circumstance, the Voting System is designed and put into use. The Voting System has encapsulated algorithms for two types of election: OPL (Open Party List) and CPL (Closed Party List).

* 1. **Overview**

This document was created based on the IEEE template for Software Design Documents. Reading the document, readers can follow the sequence of Chapter 1->Chapter 2->Chapter 4->Chapter 5-> Chapter 3->Chapter 6->Chapter 7->Chapter 8.

* 1. **Reference Material**

*Textbook: Software Engineering, Ian Sommerville, Tenth Edition, ISBN 10: 0-13-394303-8, ISBN 13: 978-0-13-394303-0*

*System Requirement Specification: https://github.umn.edu/umn-csci-5801-f19/repo-Team12/blob/master/Homework1/docs/CSci5801HW1\_Fall2019\_v1.pdf*

*Software Requirements Specification for Gephi: https://canvas.umn.edu/courses/134519/pages/software-requirements-specification-supporting-documents?module\_item\_id=2883887*

*Software Requirements Specification for Web App:*

*https://canvas.umn.edu/courses/134519/pages/software-requirements-specification-*

*supporting-documents?module\_item\_id=2883887*

*Priority Levels: http://www.jumpmind.com/services/support/priority-levels*

* 1. **Definitions and Acronyms**

*See Appendix A in chapter 8.*

1. **SYSTEM OVERVIEW**

The Voting System is a useful, self-contained product. It is designed to count the voting ballots and decide the winners and winning parties. This system is supposed to be used after the actual voting is finished and a voting file is provided to the election officials. It allows election officials to transfer the original voting file into a final election result with the election information and output the result. Also, this system helps the election officials send results to media and let the public know the election results. All the functions mentioned above can be accomplished by this system itself.

The Voting System is mainly constructed by the following components:



1. **SYSTEM ARCHITECTURE**

* 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.

* 1. **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 object­oriented description. For a functional description, put top­level 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.

* 1. **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.

1. **DATA DESIGN**

* 1. **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.

* 1. **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.

1. **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

1. **HUMAN INTERACTION DESIGN**

* 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.

* 1. **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.)

* 1. **Screen Objects and Actions**

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

1. **REQUIREMENT MATRIX**

Provide a cross­reference 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.

1. **APPENDICES**

# *Appendix A: Glossary*

* **Voting System**: Election Voting System, which is used by election officials during the election.
* **CSV**: a kind of file format, which can be generated by Excel
* **CPL**: stands for “closed party list”. In a closed party list system--the original form of party list voting--the party fixes the order in which the candidates are listed and elected, and the voter simply casts a vote for the party as a whole. Voters are not able to indicate their preference for any candidates on the list but must accept the list in the order presented by the party.
* **OPL**: stands for “open party list”. In an open party list system, voters are allowed to express a preference for particular candidates, not just parties. It is designed to give voters some say over the order of the list and thus which candidates get elected.
* **Allocate Seats**: After voting, each party receives the number of seats they win. There are lots of different formulas for the allocation of seats to the parties. One is called the “largest

remainder formula,” which is also used in this voting system. In this approach, the first step

is to calculate a quota, which is determined by taking the number of valid votes and dividing it by the number of seats. The party wins one seat for each whole number produced. After the first allocation of seats is complete, the remainder numbers for the parties are compared and the parties with the largest remainders are allocated the remaining seats. Finally, all the parties win the number of seats that as closely as possible approximate their percentage of the vote.

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