Team 17

Voting System

Software Design Document

Carlos Alvarenga (alvar357)
Justin Koo (kooxx078)
Michael McLaughlin (mclau361)
Xiaochen Zhang (zhan4487)

10/28/2018

TABLE OF CONTENTS

INTRODUCTION **2**
- Purpose 2
- Scope 2
- Overview 2
- Reference Material 3
- Definitions and Acronyms 3

SYSTEM OVERVIEW **4**

SYSTEM ARCHITECTURE **5**
- Architectural Design 5
- Decomposition Descriptions 6
- Design Rationale 7

DATA DESIGN **7**
- Data Description 7
- Data Dictionary 8

COMPONENT DESIGN **18**

HUMAN INTERACTION DESIGN **18**
- Overview of User interface 18
- Screen Images 19
- Screen Objects and Actions 19

REQUIREMENTS MATRIX **20**

APPENDICES **21**

1. INTRODUCTION

1.1. Purpose

This software design document describes the architecture and the system design of the VS. This is accomplished through UML activity, sequence and class diagrams that encompass different aspects of the VS software. This document is intended for developers as well as project managers, users, testers, documentation writers and other staff with a base level knowledge of technical concepts. As an example, it is sufficient for the audience to know at a high level what a database is and why they are used, but not necessary to know the full details of the internal workings of one.

<Identify the purpose of this SDD and its intended audience. (e.g. "This software design document describes the architecture and system design of XX. ....").>

1.2. Scope

This SDD covers the entire scope of the VS, including any and all existing and future subsystems. The VS product is a program will be written in the Java programming language and will consist of Java source file(s) and class file(s) that implement said product.
This document describes the implementation details of the VS Software. The software's major function is to grant *election officials* the capability to perform IRV and OPLV to determine election winners in either voting system. Based on all the ballots received and gathered, this software product will streamline the *election officials'* process of tallying up the votes to find a winner when conducting an IRV or OPLV election. Doing so will allow the *elected officials* to reap organizational, logistical benefits by providing them an efficient, unbiased voting system to perform IRV and OPLV elections and determine the winning candidate, according to the public's expressed interest.

<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.>

1.3. Overview

This document is organized into 8 sections. Section 1 (this section) gives a brief introduction to the VS product as well as conventions used within the SDD itself. Section 2 provides an overall high-level system overview of the VS through the use of a activity diagram. Section 3 outlines the VS architectural description and description by featuring a context diagram and referencing the sequence diagram. Section 4 outlines and describes all the VS entities and their design in relation to the various data components of VS. Section 5 systematically outlines the various components of the VS by referencing the UML class diagram. Section 6 provides an overview of the VS's user interface that will be interacted with. Section 7 identifies which system components satisfy each of the functional requirements from the SRS. Finally, Section 8 contains appendix/appendices that provide supporting details that could aid in the understanding of the SDD.

<Provide an overview of this document and its organization.>

1.4. Reference Material

The description of the VS product, as well as any requirement specifications within this SRS, were initially specified by the customers found here: https://ay17.moodle.umn.edu/pluginfile.php/2780770/mod_assign/introattachment/0/Project1_Waterfall_VotingSRS_Fall2018V3_92518.pdf?forcedownload=1

The SRS for the VS product can be found here: https://github.umn.edu/umn-csci-5801-F18/repo-Team17/blob/master/RedoSRS/SRS_Redo_Team17.pdf

<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

Acronym	Long Form	Definition
SDD	Software Design Document	A description of a software design and architecture of the voting system. It shows how the voting software system will be structured to satisfy the requirement.
SRS	Software Requirements Specification	A description of a software system to be developed
IRV	Instant Runoff Voting	A plurality/majority voting system also known as the "majority preferential voting". One of the voting systems that needs to be implemented as part of the VS and whose requirements are specified within the SRS.
OPLV	Open Party List Voting	A proportional representative type of

		voting system that also needs to be implemented as part of the VS and whose requirements are specified within the SRS.
VS	Voting System	The voting system program that must be created and is capable of performing IRV and OPLV. This is the product being described and specified within this SRS
UML	Unified Modeling Language	Standardized modeling language that enables developers to specify, visualize, construct and document the artifacts of a software system.
GUI	Graphical User Interface	An interface that includes graphical elements for users to interact with
CSE	College of Science & Engineering	One of the colleges of the University of Minnesota in Minneapolis, Minnesota
TBD	To Be Determined	An acronym for “to be determined”
CLI	Command Line Interface	A form of sending commands to the computer via successive lines of text
MVC	Model View Controller	A software architectural pattern that divides a system into three parts: model, view, and controller.

<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.>

2. SYSTEM OVERVIEW

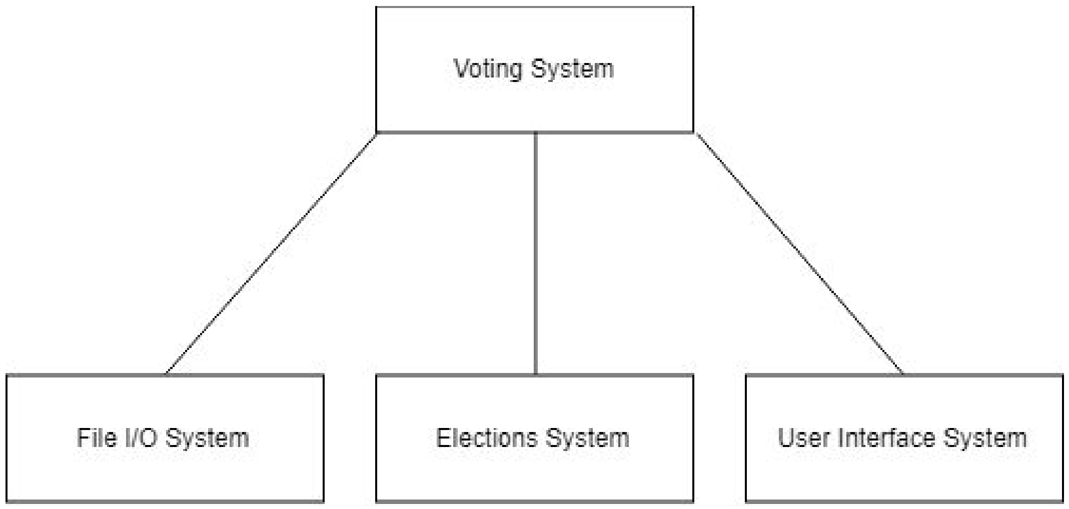
The major function of the VS is to grant the intended users, the *election officials*, the ability to calculate and display the election results of OPLV and IRLV in an efficient manner. A high level overview of the flow of control for the VS product is as follows: first users will provide the VS an input file containing all the necessary information needed to calculate the results of an election, next the VS system performs an algorithm to calculate the results of the election based on the input files’ parameters, finally the results are displayed to the user and an audit file is generated containing more details about the election results.

The VS product is modularized into three subsystems that each manage specific tasks necessary to fulfil all requirements. Each of the subsystems is introduced at a high level in the next section.

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

3. SYSTEM ARCHITECTURE

3.1. Architectural Design



The context diagram above shows how the VS is modularized into three distinct subsystems. Each subsystem is responsible for fulfilling different requirements in the SRS. An exhaustive account for how each requirement in the SRS is linked to each subsystem can be found in section 7. For now we give a brief overview of the roles/responsibilities of each subsystem, enough for the reader to gain a general understanding of how and why the VS was decomposed.

- File I/O System: this subsystem is tasked with handling any and all operations involving file input and output, that is, the transfer of data to and from the VS product. Specifically, the file I/O subsystem will perform two key operations:
    1. Read a ballot file located in the same directory as the VS program specified by the users. The structure of the ballot file is described in section 2.7 of the SRS.
    2. Write an audit file containing election results to the same directory as the VS program. The name of this file will be automatically generated.

- Elections System: this subsystem is tasked with carrying out the algorithms necessary to calculate election results. The two key algorithms are tied to the two types of elections the VS can handle:
    1. An algorithm to calculate the results of an IRV ballot file.
    2. An algorithm to calculate the results of an OPLV ballot file.
- User Interface System: this subsystem is tasked with providing the users of the VS product a clean, logical, interface through the CLI. Its two main responsibilities are:
    1. Provide a textual prompt, so that users may have the necessary context and directions to use the VS product.
    2. Perform error checking on the users' inputs and inform them of any erroneous inputs.

<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 Descriptions

3.2.1. File I/O Subsystem
See attached UML activity, sequence and class diagrams.

3.2.2. Elections Subsystem
See attached UML activity, sequence and class diagrams.

3.2.3. User Interface Subsystem
See attached UML activity, sequence and class diagrams.

<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.>

3.3. Design Rationale

The architecture of the VS as outlined in section 3.1 was chosen due to the (as of now) relatively small complexity of the VS product, ease of understanding for end users, and the ability to group each of the requirements in the SRS under at least one of the subsystems. An MVC architecture was considered; however, due to the lack of (as of now) for the VS product to have a backend model, this architecture was disregarded as being unfit to accurately portray the organization of the product. In the future, if and when complexity increases, and end users or their requirements change, revisions to the architecture may be needed to accommodate.

<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

Candidates are represented by a Candidate object class.
Parties are represented by a Party object class.
IRV and OPLV elections each have an object class, IRElection and OPLElection, respectively.
IRElection and OPLElection each contains an array (list) candidates (instances of the Candidate class).
The OPLV class has an array (list) of parties (instances of the Party class) to keep track of the number of votes each party has received and distribute seats accordingly.
Instances of the candidate class when running an IRV election contain an array of votes where each *ith* index of the array corresponds to the number of votes that the respective candidate has received as the *i+1* option. For example, index 0 of candidate will contain the value 5 if that candidate has received 5 first place votes, index 1 will contain the value 10 if the candidate has received 10 second place votes, etc.

<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.>

4.2. Data Dictionary

See attached class diagram document (*ClassDiagram17.pdf*) for a complete overview of the entire system.

4.2.1. AuditFile Class

4.2.1.1. Attributes

Name	Type	Description
name	String	Name of the audit file
output_stream	FileOutputStream	File stream of audit file

4.2.1.2. Methods

String getName()	
Input	None
Output	Name of the audit file
Description	Method to get the name of the audit file

FileOutputStream getOutputStream()	
Input	None
Output	File output stream of the audit file
Description	Method that returns the output stream of the audit file

void setName(String name)	
Input	Name of the audit file
Output	None
Description	Method to change the name

	of the audit file

4.2.2. BallotFile Class

4.2.2.1. Attributes

Name	Type	Description
name	String	Name of the ballot file
input_stream	FileInputStream	File input stream of the ballot file

4.2.2.2. Methods

String getName()	
Input	None
Output	Name of the ballot file
Description	Method to get the name of the ballot file

FileInputStream getInputStream()	
Input	None
Output	File input stream of the ballot file
Description	Method that returns the input stream of the ballot file

void setName(String name)	
Input	Name of the ballot file
Output	None
Description	Method to change the name of the ballot file

4.2.3. Candidate Class

4.2.3.1. Attributes

Name	Type	Description
name	String	Candidate's name
num_votes	int	Number of votes that candidate has
party	int	Candidate's party

4.2.3.2. Methods

String getName()	
Input	None
Output	Name of the candidate
Description	Returns the name of the candidate

String getParty()	
Input	None
Output	Name of the party
Description	Returns the name of the political party

int getNumVotes(int place)	
Input	The place of the candidate in the polls
Output	The number of votes that the candidate has
Description	Returns the number of votes the candidate in the *ith* place (denoted by the parameter)

	has received. Used for IRV elections.

int getNumVotes()	
Input	None
Output	The number of votes that the candidate has
Description	Returns the number of votes that the candidate has received. Used for OPLV elections.

void setName(String name)	
Input	Name of the candidate
Output	None
Description	Changes the name of the candidate to the specified name in the parameter

void setNumVotes(int place, int votes)	
Input	The place of the candidate and the number of votes
Output	None
Description	Changes the votes of the *ith* place votes (As denoted by the *place* parameter) to the new *votes* value specified in the parameter.

void addVotes(int place)	

Input	The *ith* place of the candidate
Output	None
Description	Increments the vote counter for *ith* place votes of the candidate by one

4.2.4. Election Class

4.2.4.1. Attributes

N/A

4.2.4.2. Methods

void runElection(String filename)	
Input	Name of ballot file
Output	None
Description	Method to calculate the voting result of the election given the ballot file

4.2.5. IRElection Class

4.2.5.1. Attributes

Name	Type	Description
election	String	Election type
num_candidates	Int	Number of candidates
num_ballots	Int	Number of ballots
candidates	Candidate[]	List of candidates

4.2.5.2. Methods

void sortCandidates(Candidate[] candidates)	
Input	List of candidates
Output	None

Description	Sorts the candidates based on the number of first place votes that they have received

4.2.6. OPLElection Class

4.2.6.1. Attributes

Name	Type	Description
election	String	Election type (OPL or IR)
num_candidates	int	Number of candidates
num_ballots	int	Number of ballots
candidates	Candidate[]	List of candidates
parties	Party[]	List of parties
num_seats	int	Number of seats available

4.2.6.2. Methods

void sortCandidates(Candidate[] candidates)	
Input	List of candidates
Output	None
Description	Sorts the candidates based on the total number of votes that they have received

void sortParties(Party[] parties)	
Input	List of parties
Output	None
Description	Sorts the political parties based on the total number of votes that they have received

4.2.7. Party Class

4.2.7.1. Attributes

Name	Type	Description
name	String	Name of the party
num_candidates	int	Number of candidates within the party
num_seats	int	Number of seats that party has
num_votes	int	Number of votes that party has
num_rem_votes	int	Number of remaining votes that party has

4.2.7.2. Methods

String getName()	
Input	None
Output	Name of the political party
Description	Returns the name of the political party

String getNumSeats()	
Input	None
Output	Name of seats that the party has
Description	Returns the number of seats that the political party currently has

int getNumCandidates()	

Input	None
Output	The number of candidates that the political party has
Description	Returns the number of candidates that are a part of the political party

int getNumVotes()	
Input	None
Output	The number of votes that the party has received
Description	Returns the number of votes that the political party has received

int getNumRemainingVotes()	
Input	None
Output	The number of remaining votes that the party has
Description	Returns the number of remaining votes that the political party has received

void setName(String name)	
Input	Name of the party
Output	None
Description	Changes the name of the party

void setNumSeats(int num_seats)	

Input	Number of seats
Output	None
Description	Changes the number of seats of the party to the one specified

void setNumCandidates(int num_cand)	
Input	Number of candidates
Output	None
Description	Changes the number of candidates that the party consists of

void setNumVotes(int num_votes)	
Input	Number of votes
Output	None
Description	Changes the number of votes that the party has received

void setNumRemainingVotes(int num_votes)	
Input	Number of votes
Output	None
Description	Changes the number of remaining votes that the party has received

void addVote()	
Input	None

Output	None
Description	Increments the number of votes that the party has received by one

4.2.8. UserInterface Class

4.2.8.1. Attributes

N/A

4.2.8.2. Methods

void requestBallotFile()	
Input	None
Output	None
Description	Methods that obtains the ballot file from the user

void displayAuditFileName(String name)	
Input	Name of audit file
Output	None
Description	Method where the audit file name is displayed in the terminal window to the user

void displayElectionResults(Election election)	
Input	Instance of Election object
Output	None
Description	Method where the winner and other pertinent election information is outputted in the terminal window.

<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.>

5. COMPONENT DESIGN

See section 4.2 for listings and descriptions of each method for all classes. Additionally, please refer to the class diagram provided in the attached ClassDiagram_Team17 document for complete visualization of the system and its entities, attributes and methods and their relationship to the different VS subsystems (i.e. Election, File I/O, User Interface).

<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.>

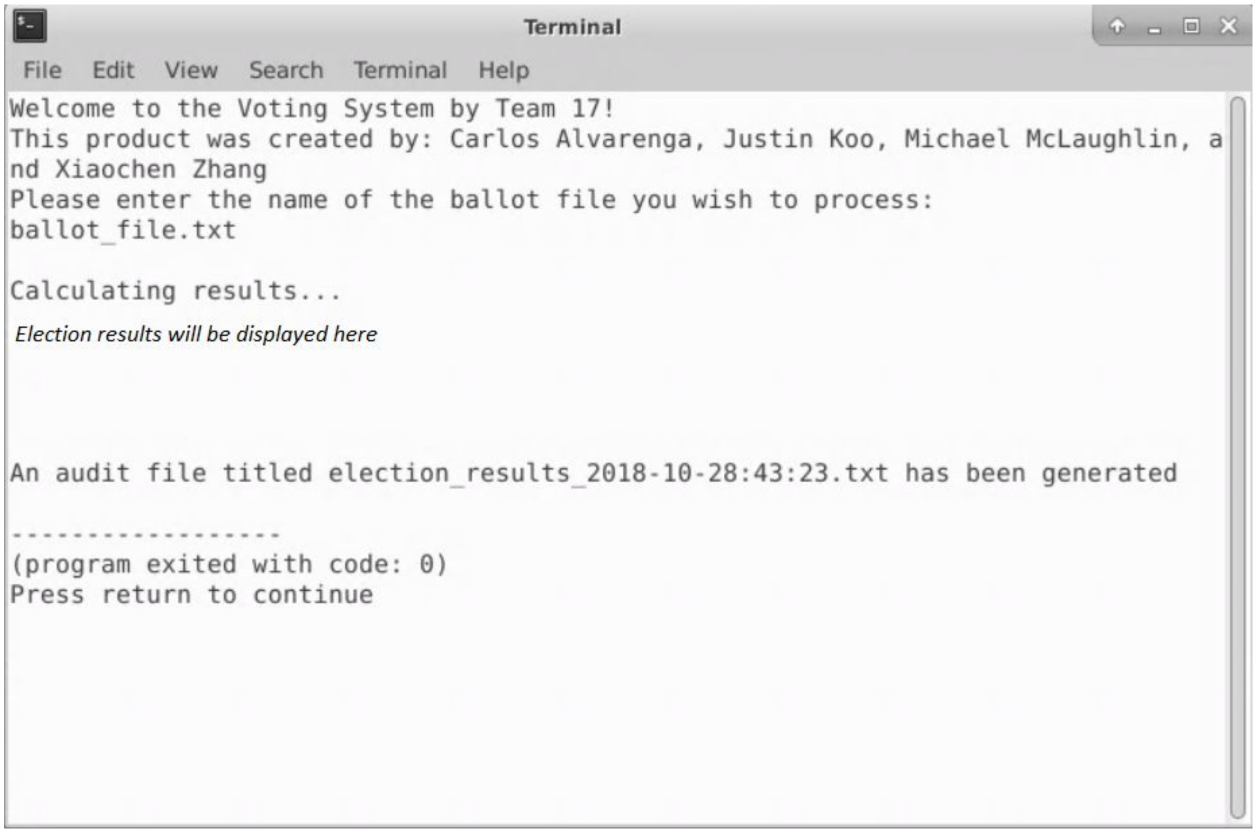
6. HUMAN INTERACTION DESIGN

6.1. Overview of User interface

The user will be able to conduct OPLV and IRV elections to determine winning candidates by simply providing correctly ballots to the programs. The user will be able to run the OPLV and IRV programs on a linux terminal window in any CSE lab machine. The name of the ballot file will be provided and as long as the ballot file is in the same directory as the program being run, the program will output the election winner and any pertinent information about the election, as well as an audit file containing more detailed information about the election.

<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.>

6.2. Screen Images



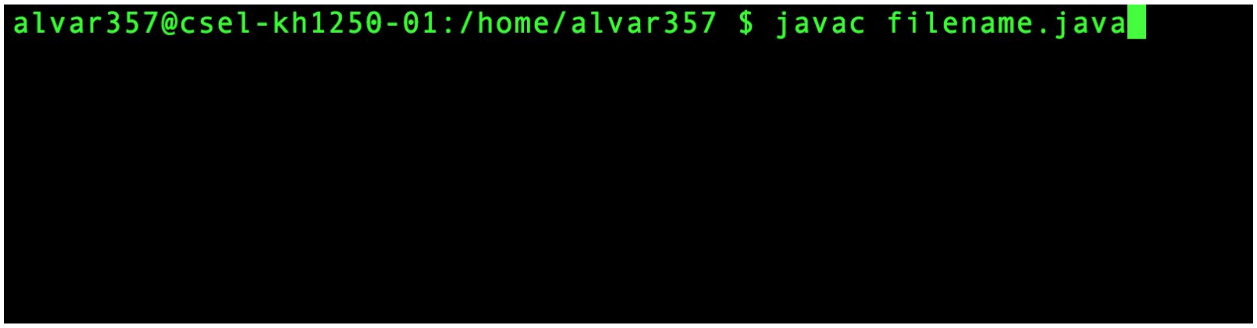
```
Welcome to the Voting System by Team 17!
This product was created by: Carlos Alvarenga, Justin Koo, Michael McLaughlin, a
nd Xiaochen Zhang
Please enter the name of the ballot file you wish to process:
ballot_file.txt

Calculating results...
Election results will be displayed here



An audit file titled election_results_2018-10-28:43:23.txt has been generated

------------------
(program exited with code: 0)
Press return to continue
```

The above screenshot shows a rough example of the user interface. End results may vary.

<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.)>

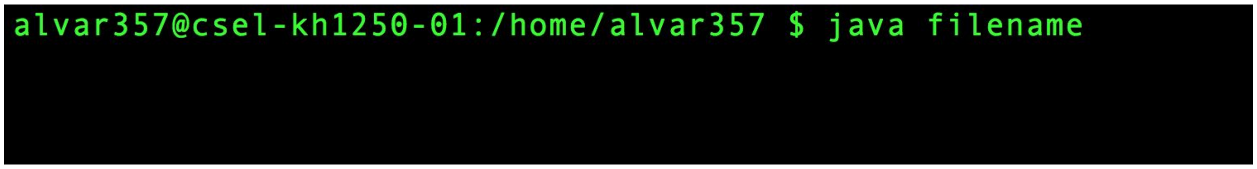
6.3. Screen Objects and Actions

Assuming that the user is operating in a linux terminal in a CSE lab machine, the following actions run the OPLV and IRV program:



```
alvar357@csel-kh1250-01:/home/alvar357 $ javac filename.java
```

The java program file can be compiled with the command *javac <filename>.java* where *filename* is the name of the program file.



```
alvar357@csel-kh1250-01:/home/alvar357 $ java filename
```

The java program file can be run with the command *java <filename>* where *filename* is the name of the program file.

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

7. REQUIREMENTS MATRIX

Requirement	Description	Design Reference
UC_001	Run IRV election	Section 3
UC_002	Run OPLV election	Section 3
UC_003	Input	Section 3&4
UC_004	Output results	Section 3&6
UC_005	View audit	Section 3&4

<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.>