Software Design Document

Voting System

Yuanli Wang
Yizhe Wang
Dengyuan Wang

Oct. 28, 18

Contents

1 Introduction

1.1 Purpose

This software design document describes the architecture and system design of the Voting System. The Voting System is designed to perform two types of voting: Instant Runoff Voting and Open Party List Voting.

1.2 Scope

Voting System is a tool that can perform two types of voting: Instant Runoff Voting and Open Party List Voting. Election officials could use it to get the result of the election from the input file (e.g. winners, type of election, number of seats). These will be displayed on the screen. A detailed audit file about the progress of the election will also be generated. Advanced Users, such as programmers and testers could use it to do unit test and system test. So the reliability of the system will be ensured. This document will not specify any testing of the software.

1.3 Overview

The remaining chapters and their contents are listed below:

Section 2 is an overview of the voting system. Section 3 is the architectural design that specifies the design entities that collaborate to perform all the functions included in the system. Section 4 shows the data structure Design. Section 5 defines the pseudocode of some components. Section 6 discusses the user interface design.

2 System Overview

2.1 Description of the system

The voting system so far will be the stand-alone system. The long-term goal is for this system to be part of an integrated online voting system. The voting system is designed to handle both of the Instant Runoff Voting and Open Party List Voting. It will take the csv file and output the audit file. The system could run 100,000 ballots within 8 minutes. There is no communication protocols information from the requirement and there is no special safety or security requirement.

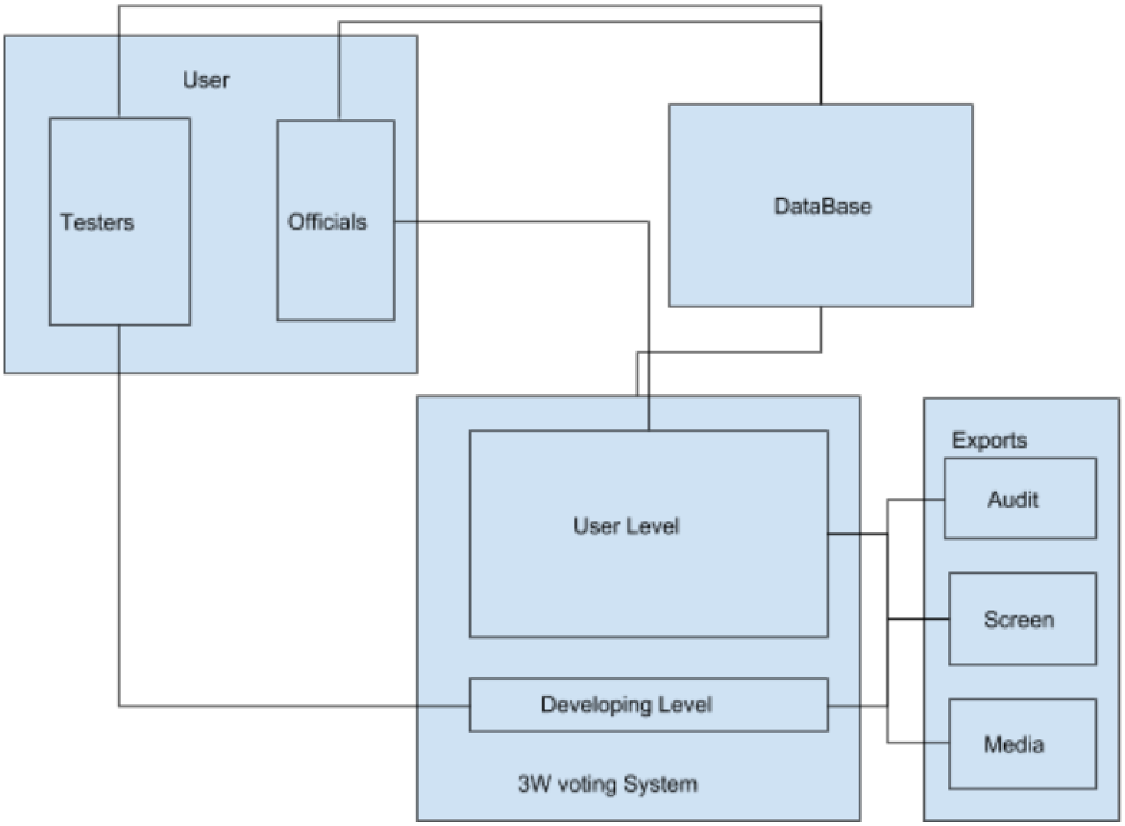
2.2 Technologies Used

The voting system is developed in Java, it has been built on top of the Eclipse Platform. It could be run on every CSE lab machine which has Ubuntu 18.04.1 LTS installed. And it could be run either at the command prompt or through Eclipse.

The Voting System requires Java to be installed on the system, more specifically java version “1.8.0_151" or for its latest release. Additional information can be found on section 2.7 of this document. Also, the Voting System can be connected with a MySQL or SQLite database to distinguish users’ level by loading users’ data. The Voting System does not require an internet connection to perform. All actions will be done in the local machine.

3 System Architecture

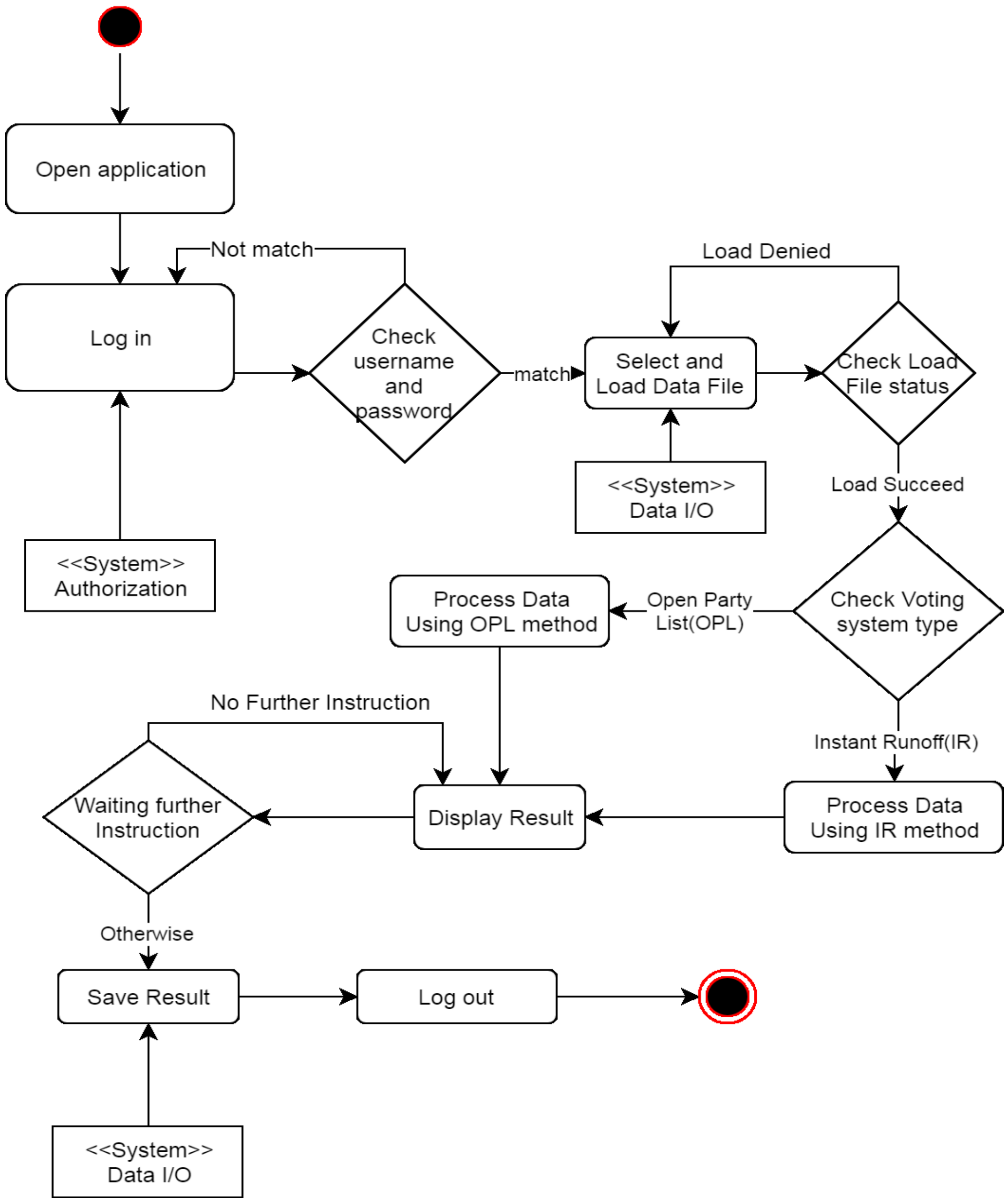
3.1 Architectural Design



The voting system is composed by 4 parts. The user part, Voting system itself, the export output part and the database. User part has two subsystems, testers and officials. They will connect to the voting system by sign in with database. Database will track their user level to access different interfaces. In voting system, both user level and developing level have export functions which support export audit files, print on screens and Media file. Voting System has the status controller to monitor User info, Voting system, Data Loading, interface status, process status, Data_IO and connect to Export part.

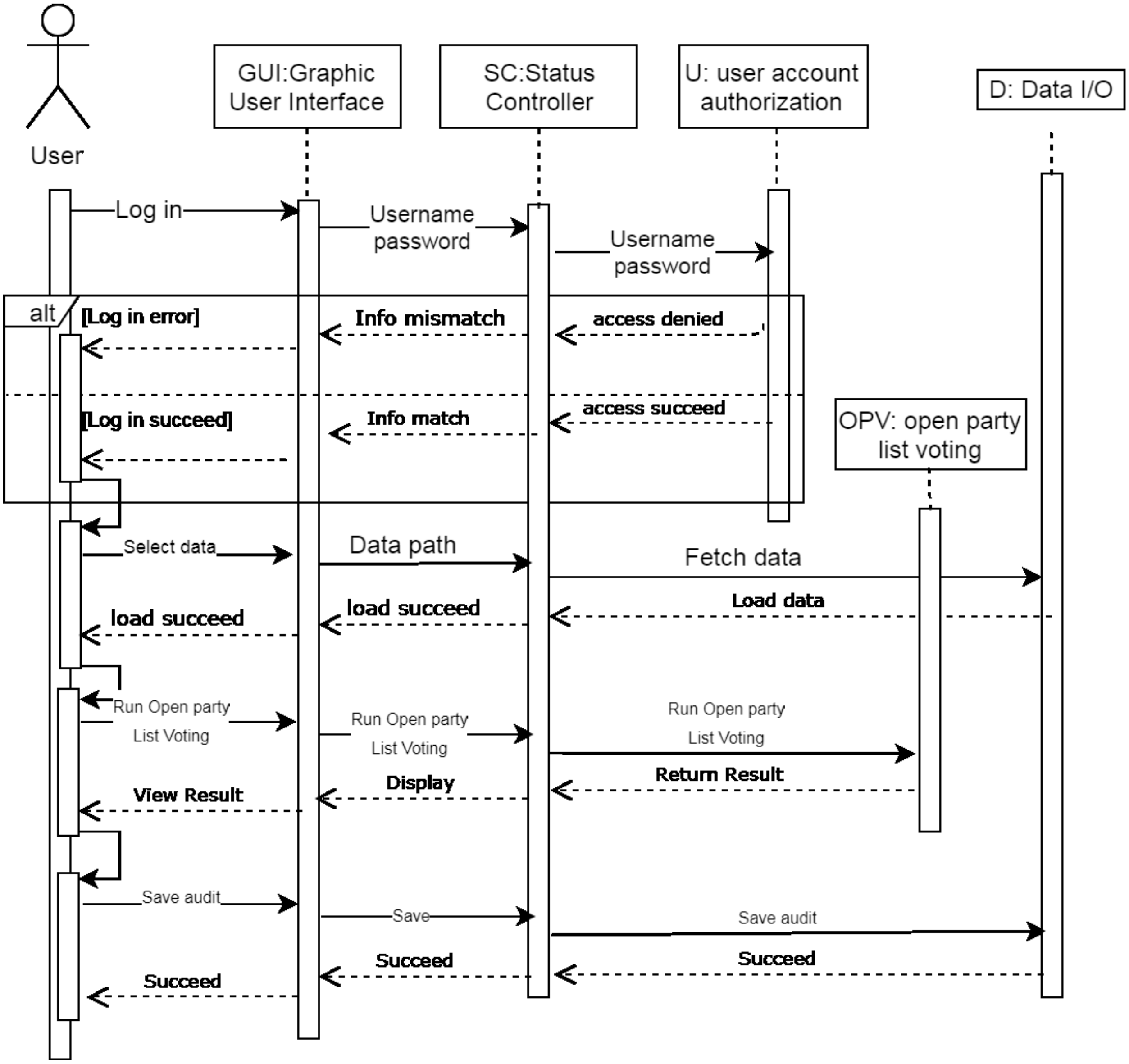
3.2 Decomposition Description

Activity diagram of the voting system



The activity diagram shows the process model for instant runoff voting and open party list voting. Firstly, the user should log into the system. The system will check the username and password. If they are matched, the user could enter into the system. Then the system will request the user to load an input file. After the user selecting the file, the system will check the load file status. If it is successful, the system will check the voting type defined in the file, and process data in an open party list method or instant runoff method accordingly. Finally, the system will display the result, and waiting for further instruction from the user. The system could save the result file if user requests.

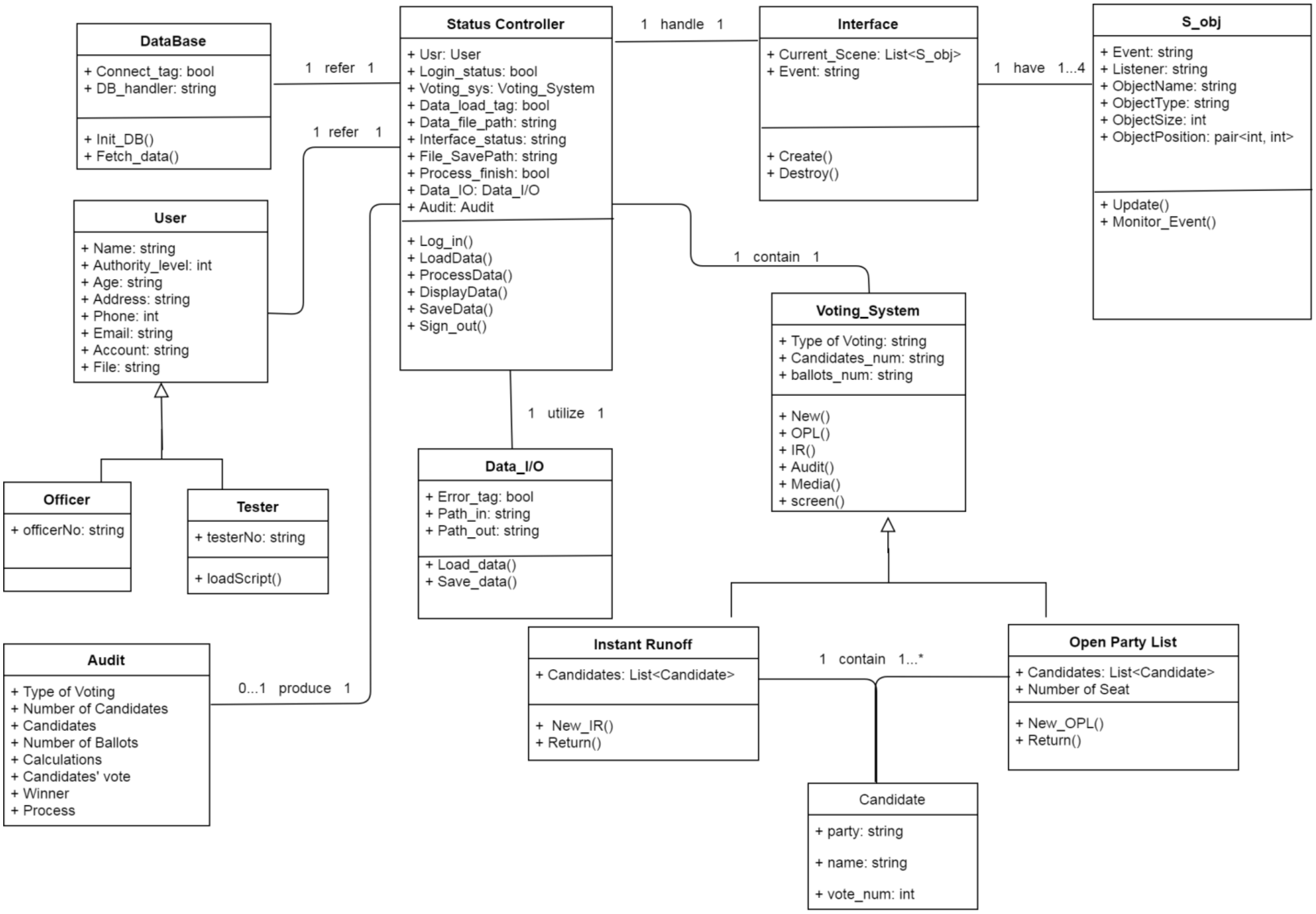
2. Sequence diagram of open party list voting



The sequence diagram shows the sequence of interactions happened in open party list voting.

When running open party list voting, the user will log into the system firstly. The user account authorization system will check the username and password. If these are not matched, the system will show the login error, otherwise, the user will enter into the system. Then the user will select an input data. The operation will be done in the data I/O module. Then the open party list voting will be run. The open party list voting module will process the data, and return the result to the user interface. If user clicks save audit file button, the data I/O module will save the result into a file.

3. Class diagram of the voting system



The class diagram shows the classes in the system and the association between these classes. The status controller means the core of the system, which defines the key variables and functions. The interface takes charge of the user interface. The Voting_System defines variables and functions that will be used when processing voting data, and it has two sub-class: one for instant runoff voting and another for open party list voting. Data I/O defines operations for opening and saving data. The user class is defined in order to save user information. The DataBase class will define database operations. The audit class defines variables for saving audit file.

4 Data Design

4.1 Data Description

Data in the Voting system will be stored and documented in two places. Users' AccoutName, Password, and user-level will be stored in the MySQL database. And dynamic information for the user will be stored by the status controller.

4.2 Data Dictionary

DataBase:

Attributes:

+Connect_tag,
+DB_handler

Method(s):

+Init_DB()
+Fetch_Data(string Username, string password)

Data I/O:

Attributes:

+Error_tag
+Path_in
+Path_out

Method(s):

+Load_data()
+Save_data(Audit audit)

Interface:

Attributes:

+Current_Scene

Method(s):

+Create()
+Destroy()

Status Controller:

Attributes:

+Usr,
+Login_status,
+Voting_sys,
+Data_load_tag,
+Data_file_path,
+Interface_status,
+File_SavePath,
+Process_finish
+Data_IO
+Audit

Method(s):

+Log_in(string username, string password)

+LoadData()
+ProcessData()
+DisplayData()
+SaveData()

S_obj:

Attributes:

+Event
+Listener
+ObjectName
+ObjectType
+ObjectSize
+ObjectPosition

Method(s):

+Update()
+Monitor_Event(string event_name)

Voting_System:(abstract)

Attributes:

+Type_of_Voting,
+Candidates_num
+Ballots_num,

Method(s):

+New()
+OPL()
+IR()
+Audit()
+Media()
+Screen()
+New_IR()
+New_OPL()

Instant Runoff:

Attributes:

+Candidates: List<Candidate>

Method(s):

+New()
+OPL()
+IR()
+Audit()
+Media()
+Screen()
+New_IR()
+New_OPL()

Open Party List:

Attributes:

+ Candidates: List<Candidate>
+ Number of Seat

Method(s):

+New()
+OPL()

+IR()
+Audit()
+Media()
+Screen()
+New_IR()
+New_OPL()

5 Component Design

In this section, we take a closer look at what each component does in a more systematic way and summarize each object member function for all the objects listed in 3.2 in pseudocode.

1. Status Controller

```
interface.create()
while(Log_in())
{
    LoadData()
    ProcessData()
    DisplayData()
    SaveData()
    Sign_out()
}
interface.destroy()
```

2. Log_in

```
if(init_DB()=true)
//if Log_in info match, return User data and set log_in status to true, otherwise, set status to false
    fetch_data()
Else
    Alart: Username/password mismatch
```

3. LoadData   //precondition: User press Load data button in UI

```
if(Load_data()){
    Data_load_tag = true
    Fill Data into Voting system attributes
}
Else
    Data_load_tag = false
    Alart: File can not load
```

4. ProcessData

```
if(Data_load_tag = true){
    New()  //this function is belonged to Voting_System
    if(file.firstline = "IR")
        {
            Audit = IR()
            Process_finish = true
        }
`   else if(file.firstline = "OPL"){
            Audit = OPL()
            Process_finish = true
        }
}
```

5. Display Data

```
if(Process_finish = true){
    Interface = interface.create()
    while(true){
        if(monitor_event()==audit){
            Return(Audit())
        }
        Else if(monitor_event()==screen){
            Return(Screen())
        }
        Else if(monitor_event()==media){
            Return(Media())
        }
        }
}
```

6. Save Data

```
if(monitor_event()==save)
{
        save_data()
}
```

7. Sign out

```
sign_out()
```

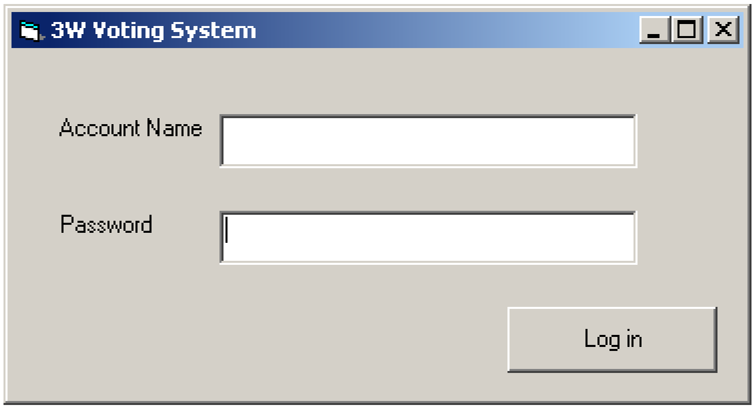
6 Human Interface Design

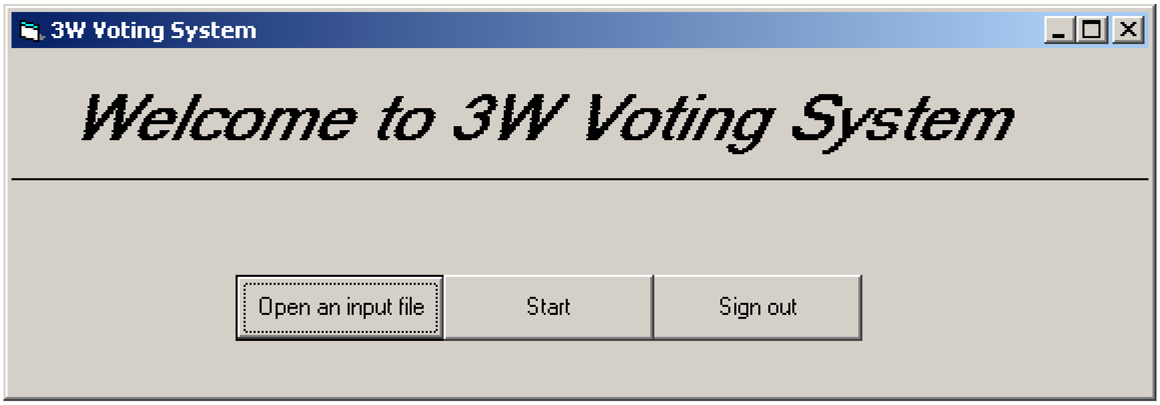
6.1 Overview of User Interface

The Voting System will have a graphic user interface for the user to interact with the system. It will contain buttons, labels, text boxes, and file operation dialogs. The user could type in the text boxes, and click on the buttons. Information will be shown directly in the windows.

6.2 Screen Images

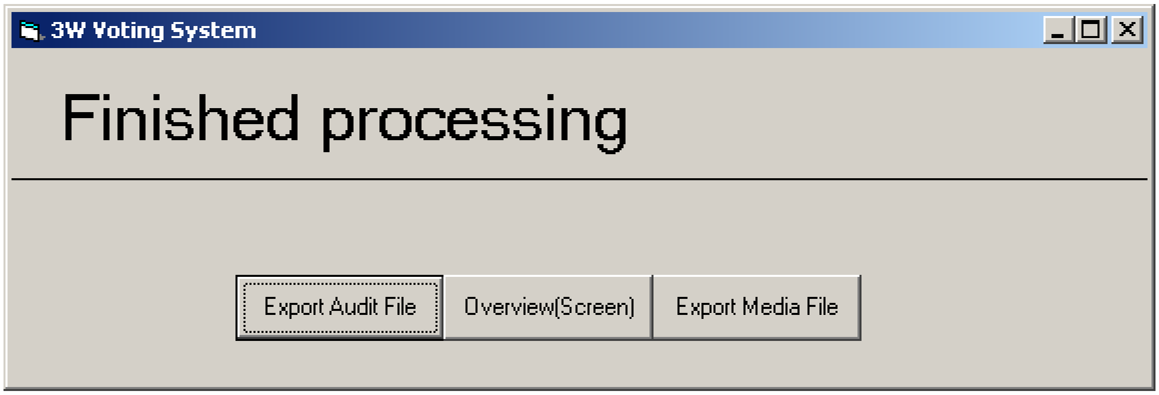
1.Sign In



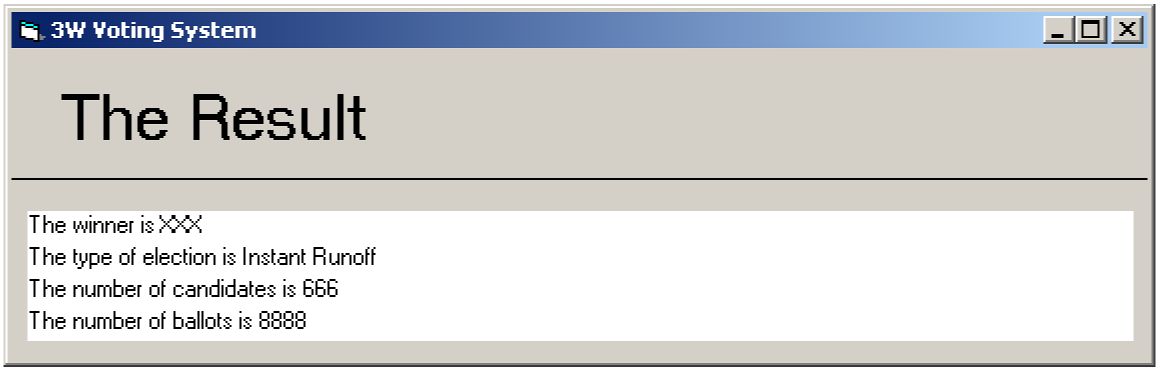

Welcome to 3W Voting System

2. Load Data File

3. Process Finished


Finished processing

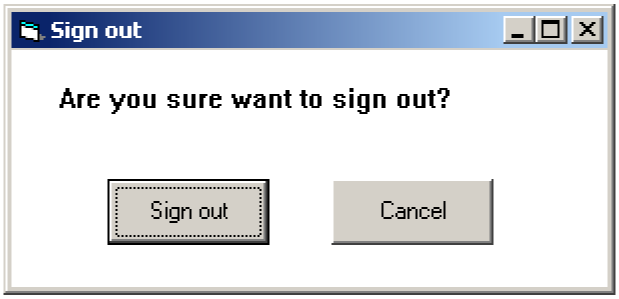
4. Overview(Screen)



The Result

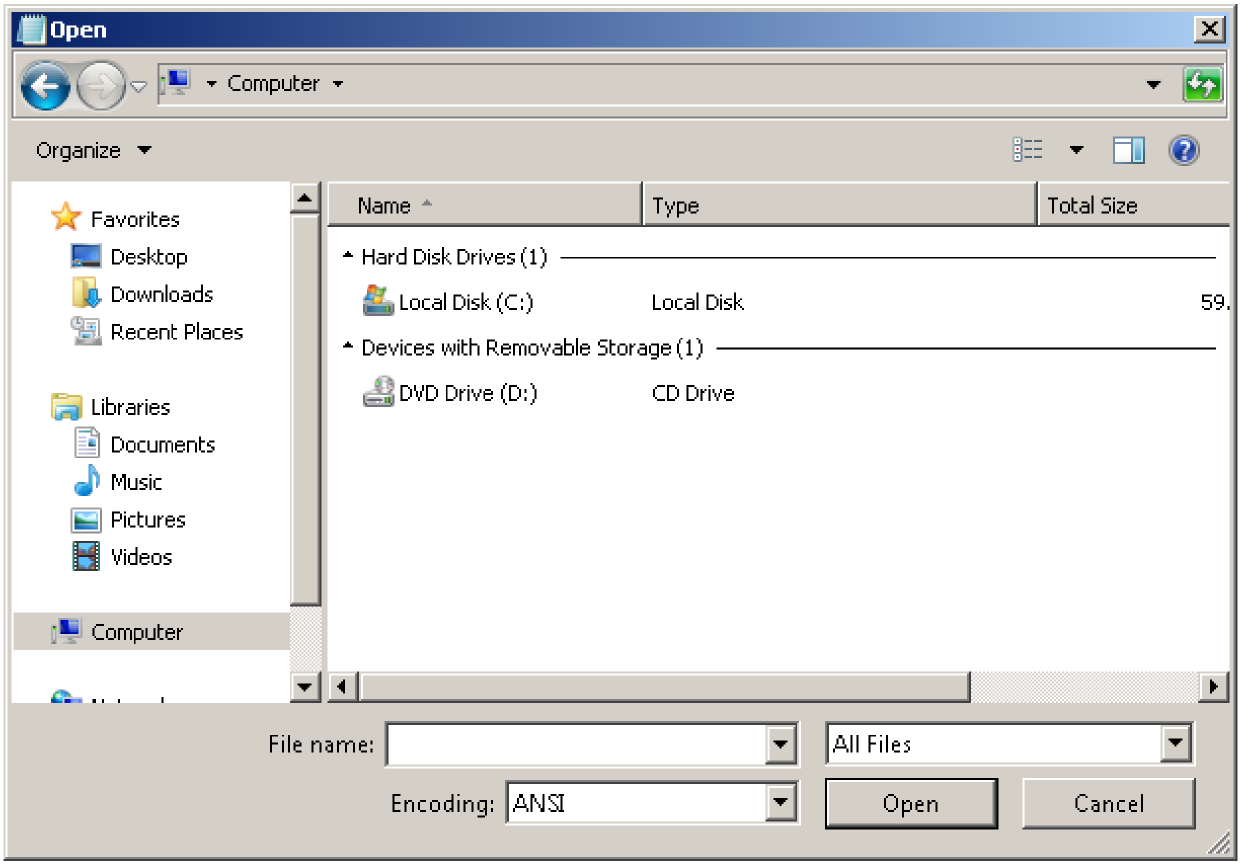
The winner is XXX
The type of election is Instant Runoff
The number of candidates is 666
The number of ballots is 8888

5. Sign out



Are you sure want to sign out?

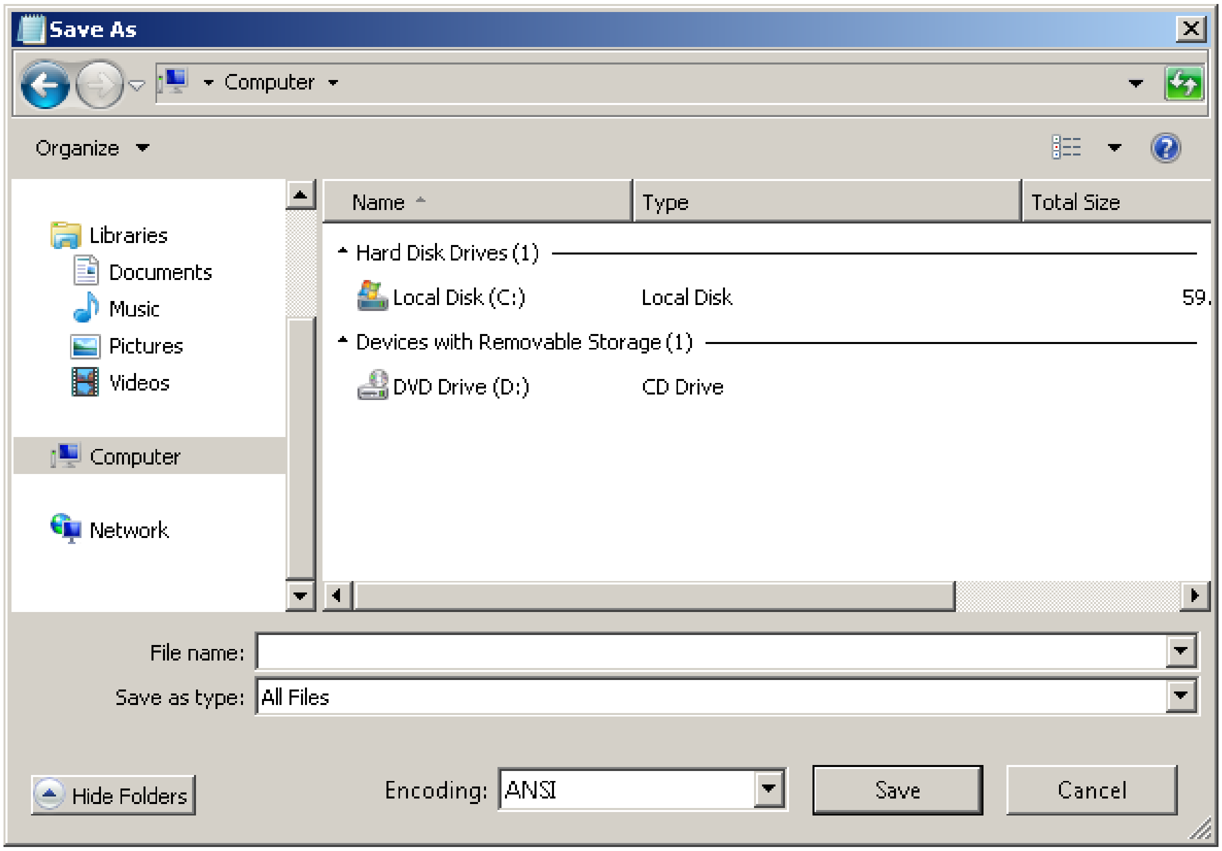
6. Open input file



Encoding: ANSI

All Files

7. Export Audit File



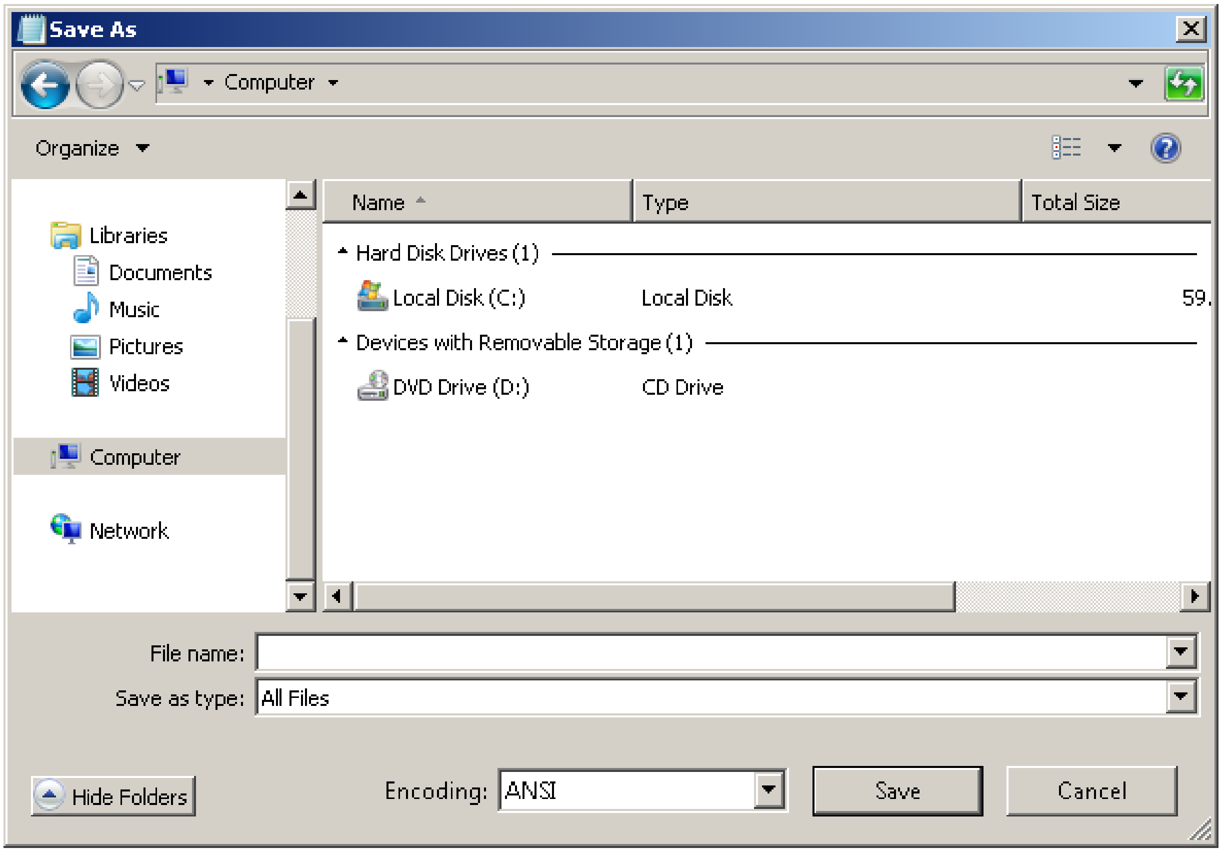
Name	Type	Total Size
Hard Disk Drives (1)		
Local Disk (C:)	Local Disk	59.
Devices with Removable Storage (1)		
DVD Drive (D:)	CD Drive	

File name:

Save as type: All Files

Encoding: ANSI

8. Export Media File



Name	Type	Total Size
Hard Disk Drives (1)		
Local Disk (C:)	Local Disk	59.
Devices with Removable Storage (1)		
DVD Drive (D:)	CD Drive	

File name:

Save as type: All Files

Encoding: ANSI

6.3 Screen Objects and Actions

In the sign in interface, users will type in their username and password. If they are matched, then load data interface will be shown, where user could choose to open an input file, start voting or sign out. If user choose open an input file, then a system dialog will be shown, and user can choose an input file. After choosing the input, user can click start to start processing. When processing finished, the finished processing interface will be shown. User could click overview to see the result, or click export audit file to save audit file, or export media file. If user choose to sign out, then a dialog will be shown to confirm the operation.