Team 4

Voting System Project
Software Design Document

Name (s):
Yangjiawen Xu (xuxx1262),
Yanjun Cui (cui00022),
Jerry Nie (nie00008),
Hanzhang Wu(wu000123)
Lab Section: Workstation:
Date: 11/4/2019

Table of Contents

1. Introduction
1.1 Purpose 2
1.2 Scope 2
1.3 Overview 2
1.4 Reference Material 2
1.5 Definitions and Acronyms 2
2. System Overview **3**
3. System Architecture
3.1 Architectural Design 3
3.2 Decomposition Description 5
3.3 Design Rationale 5
4. Data Design
4.1 Data Description 6
4.2 Data Dictionary 6
5. Component Design **9**
6. Human Interface Design
6.1 Overview of User Interface 12
6.2 Screen Images 13
6.3 Screen Objects and Actions 14
7. Requirements Matrix **14**
8. Appendices **16**

1. INTRODUCTION

1.1 Purpose

This Software Design Document provides the design details of Voting System. The expected audiences are designers and testers who will develop and maintain this Voting System.

1.2 Scope

This document contains a complete description of the design of Voting System.

The basic architecture is a Linux based software. Java will be the basic design and implement language.

Voting System is a tool that displays the candidates who are elected through the open party list ballot or the closed party list ballot. The result about the winners and information about the election will be displayed to the screen. Also, the result will be shared through the media personnel. The users can read in the file, display the results to the screen, and share the results with media personnel.

1.3 Overview

First, by authorization, officials can log into the system to do some operations such as uploading file and sorting. Second, ballot file records the result of election, which needs to be uploaded. Third, the system can analyze file, determine it is open party list or closed party list, and run the sorting algorithm. After sorting finishing, it will have a basic results page, which includes some buttons. In here, officials can choose to see winner list, generate audit files, or send the results to media. At the end, officials can log out the system.

1.4 Reference Material

[IEEE] The applicable IEEE standards are published in “IEEE Standards Collection,” 2001 edition.

1.5 Definitions and Acronyms

PR system: Proportional representation system.

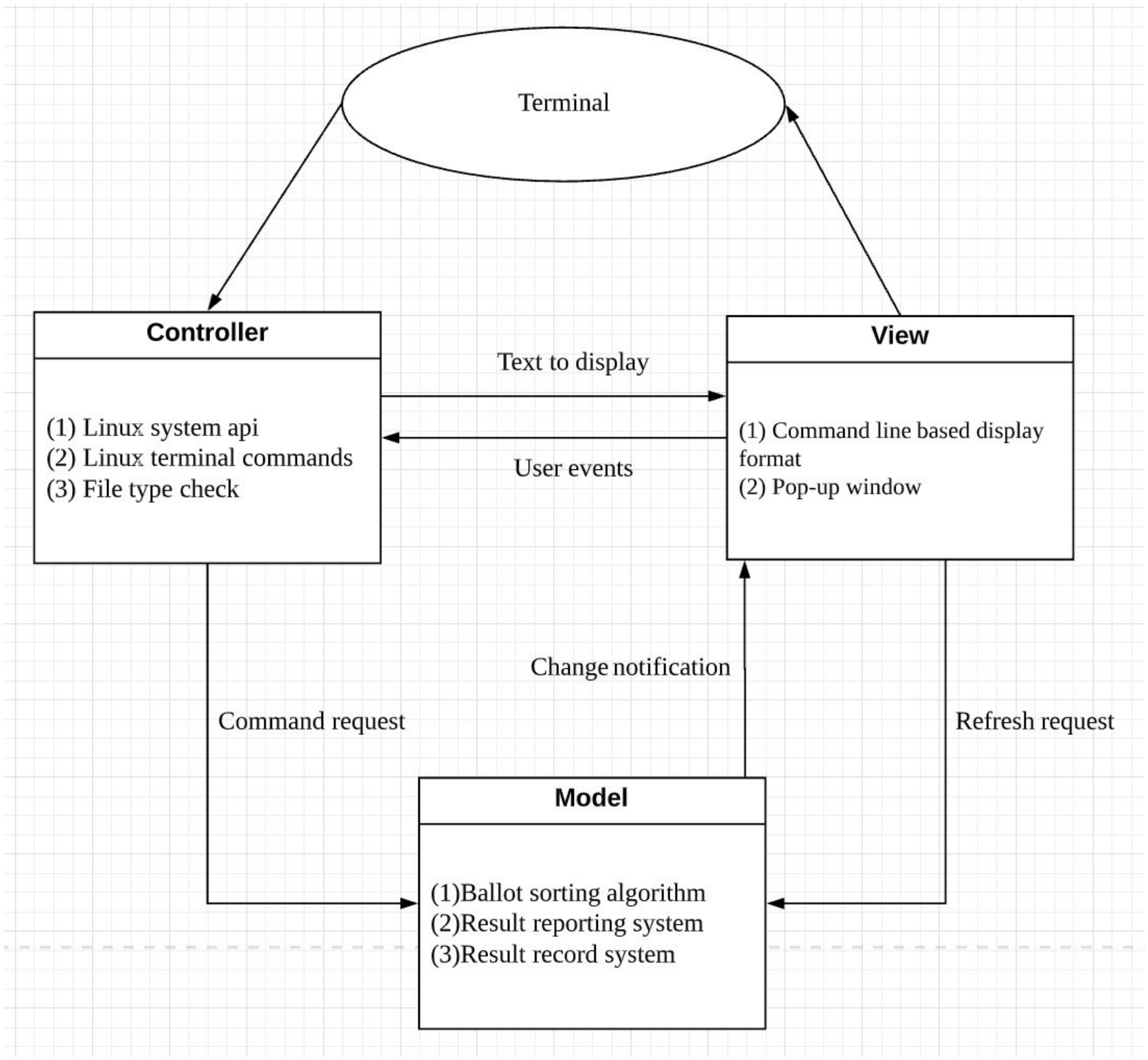
2. SYSTEM OVERVIEW

Party list voting systems are by far the most common form of proportional representation. Over 80% of the PR systems used worldwide are some form of party list voting. It remains the system used in most European democracies and in many newly democratized countries. Legislators are elected in large, multi-member districts. Each party puts up a list or slate of candidates equal to the number of seats in the district. Independent candidates may also run, and they are listed separately on the ballot as if they were their own party. In this system, according to the input ballot file, the system runs a sorting algorithm to determine the candidate's ranking. This system could also generate different type of results for different people to view.

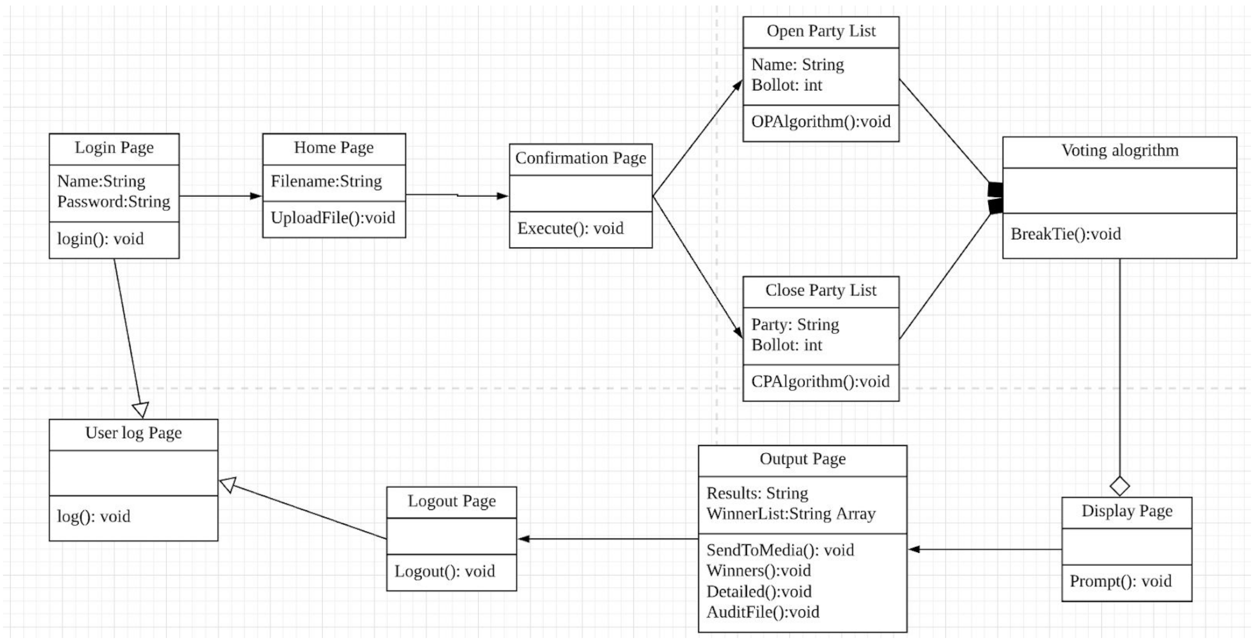
3. SYSTEM ARCHITECTURE

3.1 Architectural Design

This architectural structure shows the interaction of operating system. The voting system is controlled through terminal. By texting at terminal, since terminal has a connected Linux system API, the system can be operated by these commands. When our commands need to invoke some relevant functions, such as sorting algorithm or getting winners, it will request relevant models to run. After finished that part, a new notification will be sent to view page. Therefore, users can begin next operation, and the request will be refreshed to the new one.


Terminal
Controller
(1) Linux system api
(2) Linux terminal commands
(3) File type check
Text to display
User events
View
(1) Command line based display format
(2) Pop-up window
Change notification
Command request
Refresh request
Model
(1)Ballot sorting algorithm
(2)Result reporting system
(3)Result record system

3.2 Decomposition Description



3.3 Design Rationale

Terminal is chosen because it is a platform on which we are very easy to implement functionalities, maneuver the system, and display results, especially in the linux operating system. It could, nevertheless, be difficult for users not familiar with computer science to use since the terminal is highly programmer-oriented. We include ballot sorting algorithm, result reporting system and result record system in our model according to the program specification. We considered to implement a database that manages the data of files and users. However, we found out that storing data in the memory might be a better choice because database is hard to build and not efficient in terms of the speed of access, although the data are not preservable for future use in this case.

4. DATA DESIGN

4.1 Data Description

The data is stored in cache and memory. The reason why we choose not applicable database is that the database is often stored in the hard disk, and the reading and writing speed of the hard disk is much slower than that of cache / memory.

For the open party list voting, we use dictionary/hashmap to record the ballots information for each candidate. In the dictionary/hashmap function, we will use candidates' name(string) as keys and number of ballots(integer) as values to create several key-value pairs. In the process of counting ballots, each time a candidate gets a ballot, the value of his corresponding K-V pair will be increased by one.

For the closed party list voting, we also use dictionary/hashmap. But we will use parties' names to replace the candidates' names. Counting the number of ballots every party has. These numbers are then used to determine how many seats each party has won.

4.2 Data Dictionary

Login

Name: Login

Type: Login

Description: The user will be asked to log in the system after the system is run.

Attributes: Name: String

Password: String

Resources: None

Operations:

Name: Login()

Arguments: None

Returns: No return value

Pre-condition: Connected to terminal

Post-condition: Input file

Exceptions: If username/password is wrong, login again.

Flow of Events:

1. User is presented with the login instructions (username and password)
2. User is asked to input a file.

Home Page

Name: Home Page

Type: Home Page
Description: After the user logs on the system, the user will be asked to upload a file.
Attributes: Filename: String
Resources: None
Operations:

Name:UploadFile()
Arguments: None
Returns: No return value
Pre-condition: The user logs on.
Post-condition: The system is run and the results are generated.
Exceptions: If uploading the file fails, uploading again.

Flow of Events:

1. User is asked to upload a file.
2. User uploads the file.
3. The file is successfully uploaded.

Confirmation Page
Name: Confirmation Page
Type: Confirmation Page
Description: After the user uploads the file, the user will be asked to run the system.
Attributes: None
Resources: None
Operations:

Name: Execute()
Arguments: None
Returns: No return value
Pre-condition: The user uploads the file.
Post-condition: The results are generated.
Exceptions: If running the system fails, rerun it.

Flow of Events:

1. User clicks the “confirm” button to begin to run.

Display page
Name: Display Page
Type: Display Page
Description: A window reminds user it is finished, and generate a simple result.
Attributes: None
Resources: None
Operations:

Name: Prompt()
Arguments: None
Returns: No return value
Pre-condition: The election algorithm finished.
Post-condition: It will go to output page.
Exceptions: If the display fails, rerun the it.

Flow of events:

1. Uses clicks confirm to go to output page.

Output page
Name: output page
Type: output page
Descriptions: After results generated, user can select what they want to do about these results, such as exporting audit file or sending the result to media.
Attributes:
Audit Filename: String
Result: String
WinnerList: String array
Resources: None
Operation:

Name: SendToMedia()
Arguments: None
Returns: No return value
Pre-condition: The results of election is generated.
Post-condition: The result will be sent to specific media.
Exceptions: If the sending fails, rerun the it.

Name:Winners()
Arguments: None
Returns: No return value
Pre-condition: The results of election is generated.
Post-condition: The winners will show on the screen.
Exceptions: If the showing fails, rerun the it.

Name: Detailed()
Arguments: None
Returns: No return value
Pre-condition: The results of election is generated.
Post-condition: The detailed results will show on the screen.

Exceptions: If showing fails, rerun the it.

Name:Auditfile()
Arguments: None
Returns: No return value
Pre-condition: The results of election is generated.
Post-condition: The audit file is generated, and user can save it on disk.
Exceptions: If the file generated fails, rerun the it.

Flow of event:

1. Users click the button to choose what they want to do.
2. Execute that relevant functions.
3. Go back to output page, to select if they want to do another operation.
4. If no more operation user wants to do, then log out the system.

Logoff
Name: Logoff
Type: Logoff
Descriptions: After the system is executed and the result is printed to the screen, the audit file is generated and the result is sent to media, the user will be asked to log out the system.
Attributes: None
Resources: None
Operations:

Name: Logout()
Arguments: None
Returns: No return value
Pre-condition: The result
Post-condition: None
Exceptions: None

Flow of Events:

1. User will be asked if he/she wants to log off the system.

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.

```
public void login(username, password){
  if username in name_list:
  if case 1:enter username again
  if case 2:exit to user log page
  if hashmap(username) != password:
  if case 1: go back to enter username
  if case 2: enter password again
  if case 3: exit to user log page
  else, then redirect to upload_file function
}

public void upload_file(char* filename){
  file = open(filename)
}

public void execute(){
  if file.type == open:
  redirect to open_party_algorithm()
  else if if file.type == close:
  redirect to close_party_algorithm()
  else:
  exit with an error
}

public void open_party_algorithm(){
  while(read(file,buffer) != NULL){
   buffer = buffer.delimiter()
   cand_index = get_name_index(buffer.name)
   list[cand_index].key = buffer.name
   list[cand_index].value += buffer.ballot
  }
  list.sort()
}

public void close_party_algorithm(){
  while(read(file,buffer) != NULL){
   buffer = buffer.delimiter()
   party_index = get_party_index(buffer.name)
   list[party_index].key = buffer.name
   list[party_index].value += buffer.ballot
  }
  list.sort()
}

public void break_tie(){
  int i = randint(0,1);
  if i == 0 then winner = cand0;
  else winner = cand1;
}

public void prompt_page(){
  for (int i = 0; i < len(list); i++){
   print(list[i].key);
  }
}

public void send_to_media(){
  connect_to_server();
  send_to_server(list);
  disconnet_from_server();
}

public void get_winner(){
  print(cand_list[0].key);
  print(":");
  print(cand_list[0].value);
  print("\n");
}

public void detail_info(){
  for (int i = 0; i < len(cand_list); i++){
   print(list[i].key);
   print(":");
   print(list[i].value);
   print("\n");
  }
}
```

```
public void audit_file(){
  file = open("audit.txt", "w");
  for (int i = 0; i < len(cand_list); i++){
   file.write(list[i].key);
   file.write(":");
   file.write(list[i].value);
   file.write("\n");
  }
  close("audit.txt")
}

public void login(username, password){
  redirect to log page
}
```

6. Human Interface Design

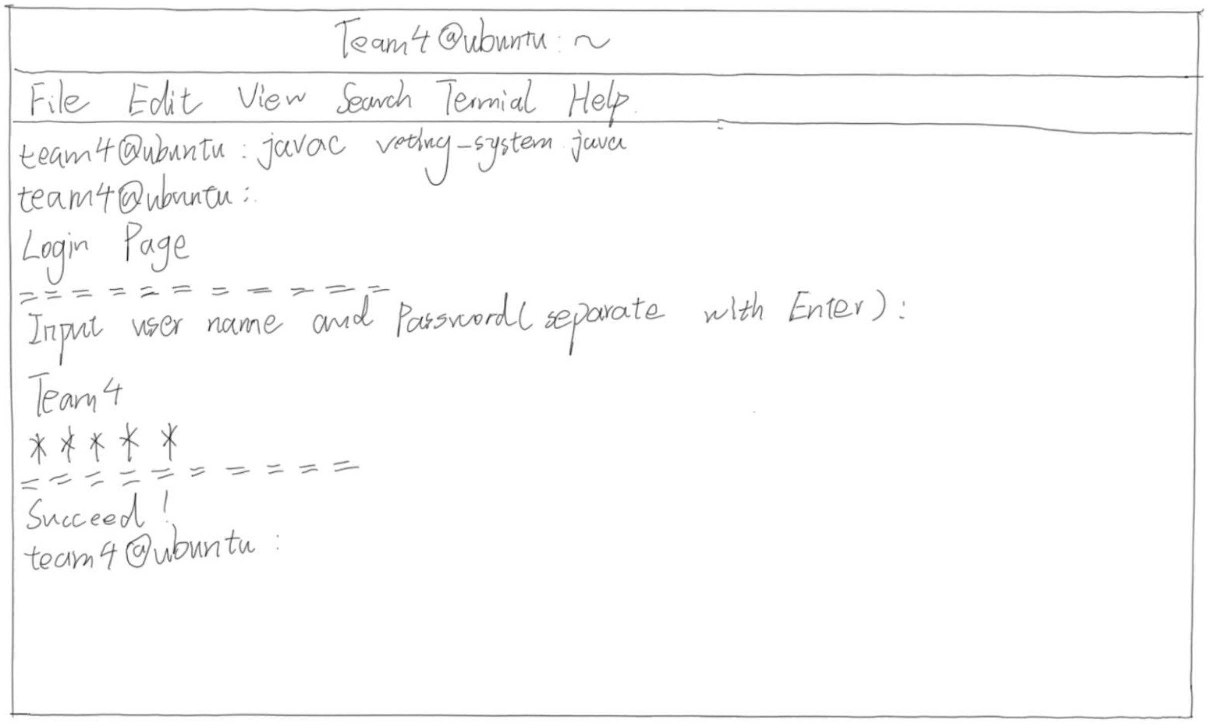
6.1 Overview of User Interface

All the interfaces are displayed in the terminal. The user interface will provide a relatively simple design for user to interact. The first page is login page. After login, user will use commands to manipulate the voting system, which contains other important functional calls.

After login successfully, user should upload the ballot information file as input first. After the system has confirmed the input file is correct, user could use command “Rank” to execute the ranking algorithm. After system finishing the ranking process, if system met a tie, the system will inform the user and ask the user to run BreakTie command to flip a coin inorder to break the tie. When proper election results are generated, the system will prompt a pop-up window showing simple election results and jump to the output page. In the end, there are several other commands for user to achieve their task like check audit file, send election results to media, etc.

6.2 Screen Images

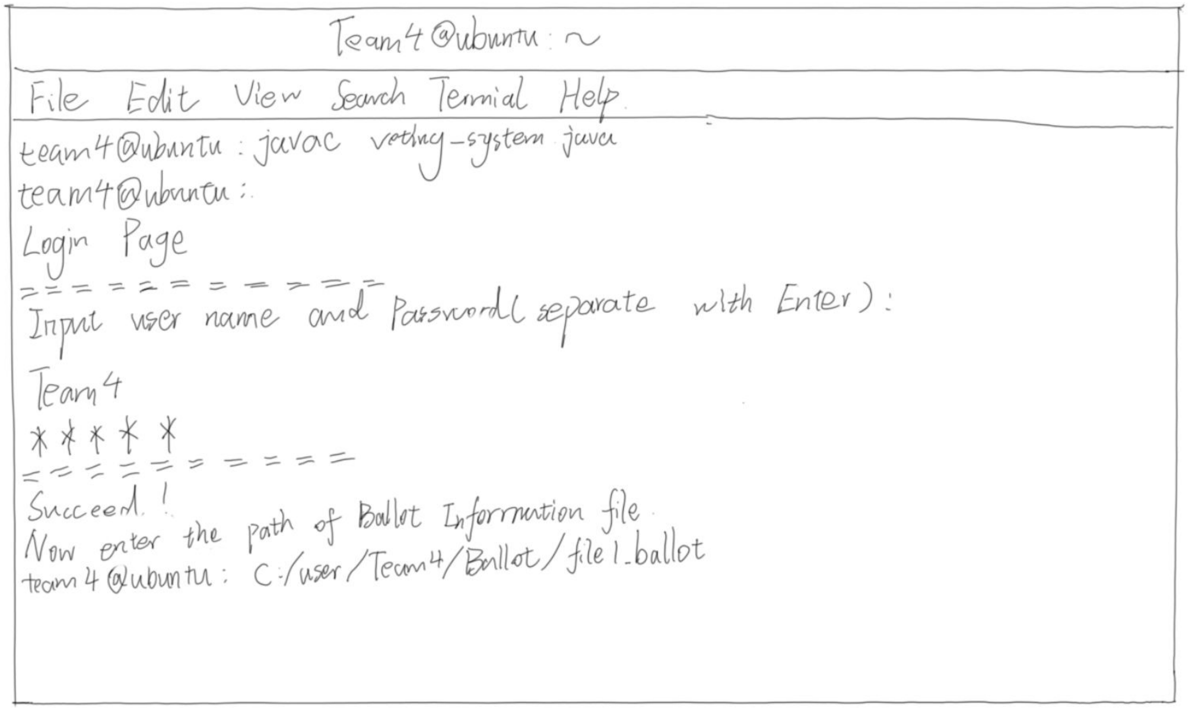
Successfully login:



```
team4@ubuntu : javac voting-system java
team4@ubuntu :
Login Page
=============
Input user name and Password( separate with Enter ):

Team4
*****
============
Succeed !
team4@ubuntu :
```

Select ballot information file :



```
team4@ubuntu : javac voting-system java
team4@ubuntu :
Login Page
=============
Input user name and Password( separate with Enter) :
Team4
*****
===========
Succeed. !
Now enter the path of Ballot Information file.
team4@ubuntu : C:/user/Team4/Ballot/file1.ballot
```

6.3 Screen Objects and Actions

The main page of Voting system has three basic process: Input ballot information file, executing ranking algorithm and result display. Process "Input ballot information file" is used to upload election file, Process "executing ranking algorithm" is used to begin ranking, while process " result display" is used to generate and display results, which users can see the winners and generate the audit file. After generating results, if needed, the system can produce an overview and send it to the media.

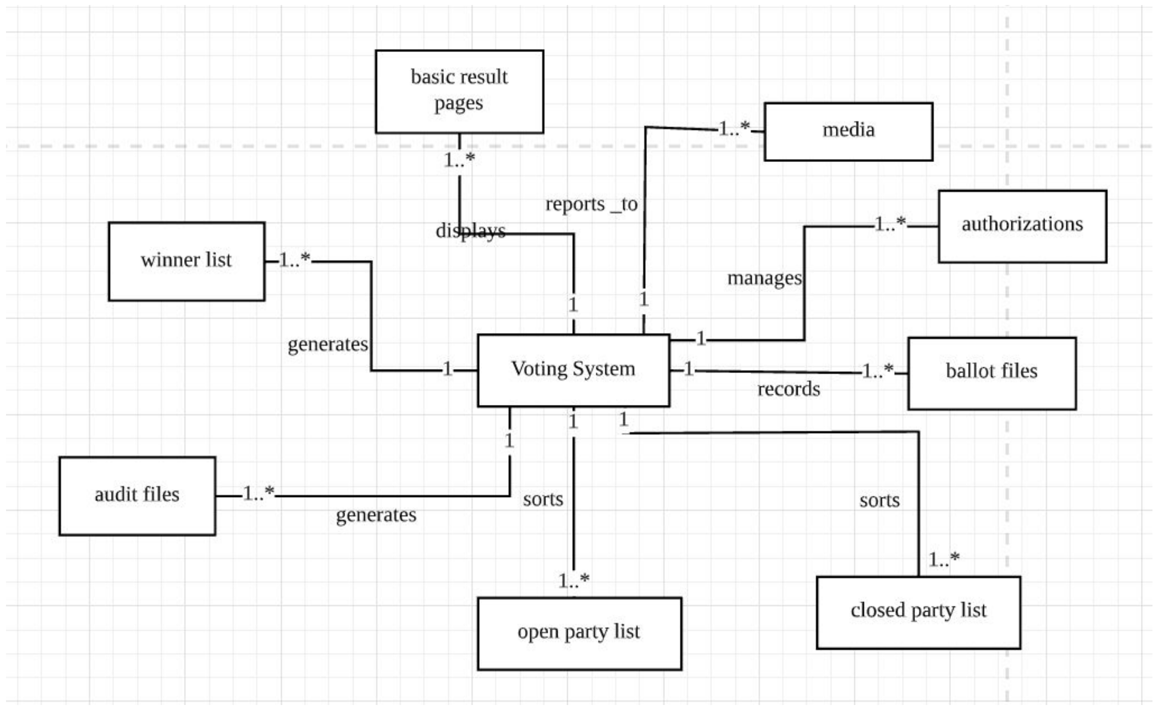
7. Requirements Matrix

UseCase ID	UseCase Name	System components
UC_001	Read in a file	Home page. People can select a file to upload.
UC_002	Click to begin the election	Confirmation page. After confirming the file, election algorithm can be executed.

UC_003	Produce an audit file	Output page. Click button to generate the audit file.
UC_004	Display election information to screen	Display page. After election finished, the result will automatically display.
UC_005	List the winners	Output page. Click button to list winners.
UC_006	Break tie	After obtaining results from OPL or CPL, the system will check whether there is a tie. If has, this function will run.
UC_007	Polish and share the results with media personnel	Output page. Click button to send the result to media.
UC_008	Prompt user to choose	Home page. If user forgets to choose an input file, the system will remind the user to upload a file.
UC_009	Ranking algorithm from OPL	Open party list algorithm
UC_0010	Ranking algorithm from CPL	Closed party list algorithm

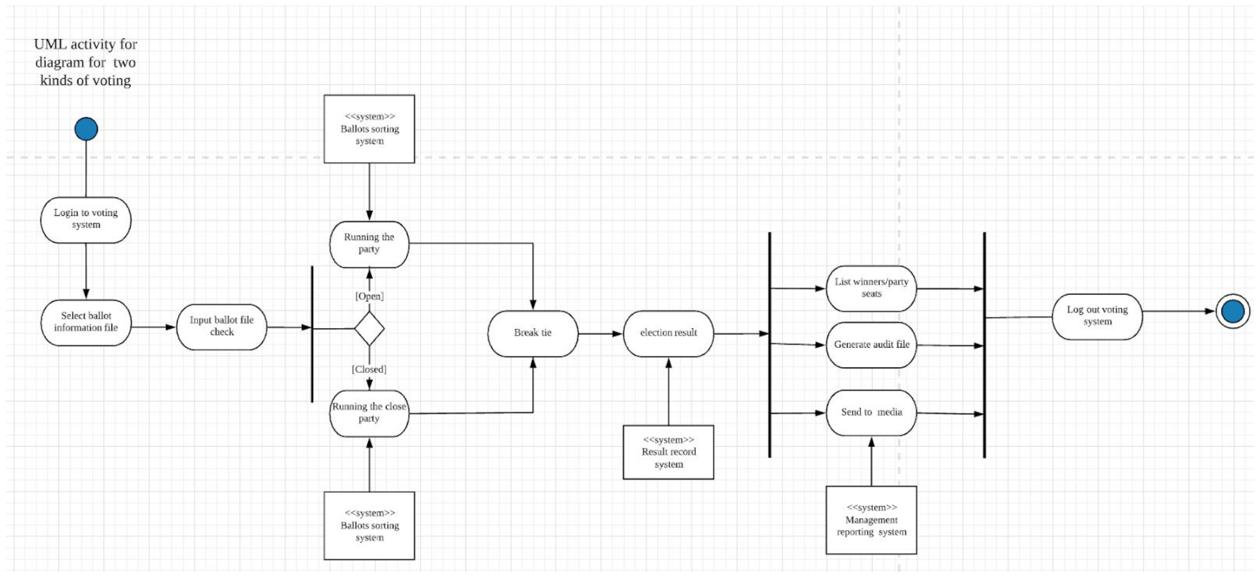
8. Appendices

UML class diagram:



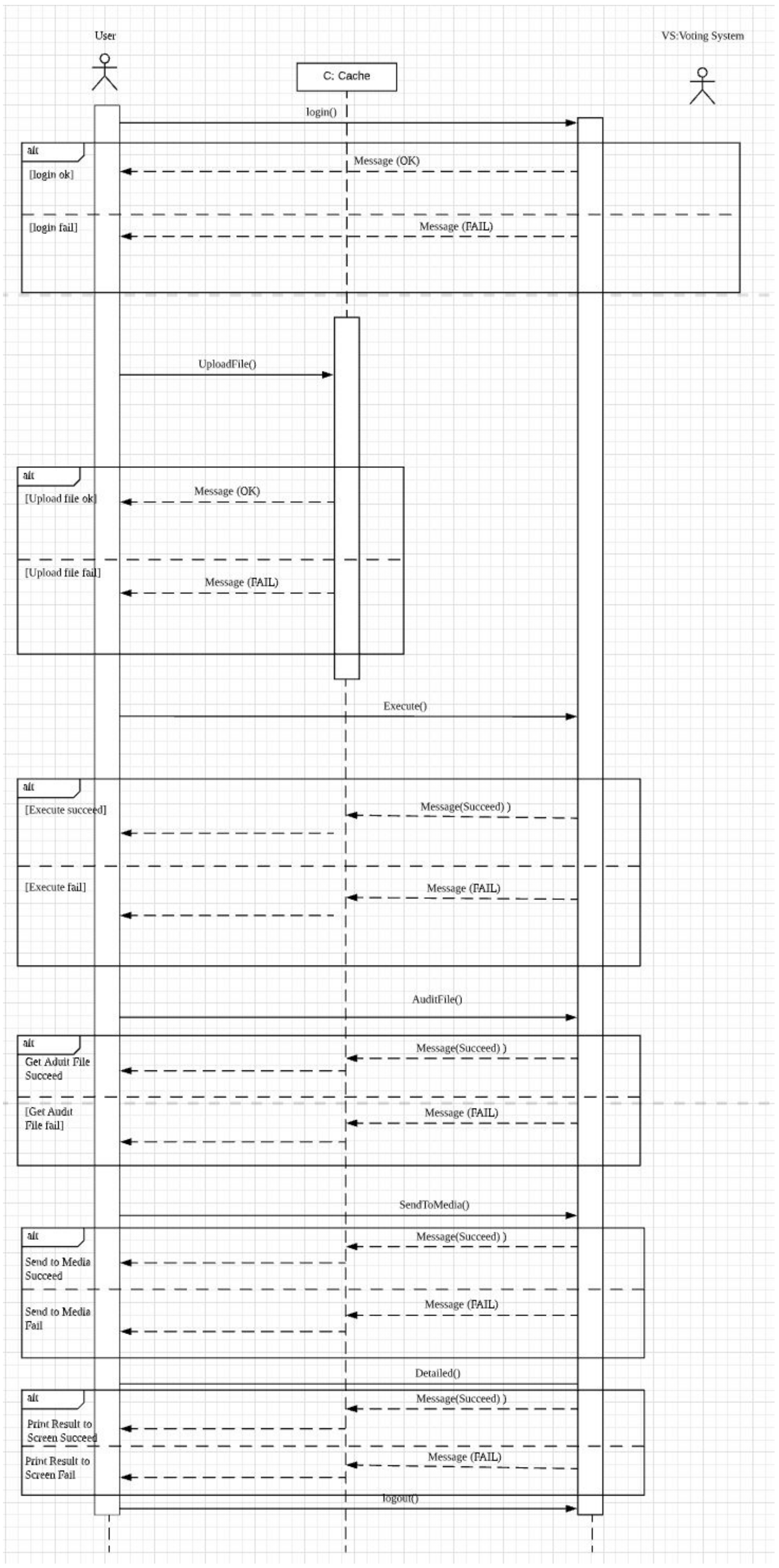
The voting system (VS) mainly have the following classes:
First, by authorization, officials can log into the system to do some operations such as uploading file and sorting, and at the end, officials can log out. Second, ballot file records the result of election, which needs to be uploaded. Third, the system can analyze file, determine it is open party list or closed party list, and run the sorting algorithm. After sorting finishing, it will have a basic results page, which includes some buttons. In here, officials can choose to see winner list, generate audit files, or send the results to media.

UML activity diagram:


UML activity for diagram for two kinds of voting
Login to voting system
Select ballot information file
Input ballot file check
<<system>> Ballots sorting system
Running the party
[Open]
[Closed]
Running the close party
<<system>> Ballots sorting system
Break tie
election result
<<system>> Result record system
List winners/party seats
Generate audit file
Send to media
<<system>> Management reporting system
Log out voting system

At first, the user logs on the system. Then, he/she chooses the file. The file is uploaded to the system. Based on the format of the file, the system can identify which voting system for this execution. After the result is generated, the system will identify whether there is a tie. If there is a tie, breaks it. Otherwise, go to the next step. Theuser will choose the output methods. The result can be directly printed to the screen, produce an audit file or sent to media. Finally, the user logs off the system.

UML sequence diagram:



The user logs on to the VS. Two options are available. If the user successfully logs on the system, the system will send a message to the user, notifying OK. If the user fails to log on the system, the system will send a message to the user, notifying FAIL. After the user logs on the system, the user is asked to upload file. Two options are available. If the user successfully uploads the file, the system will send a message to the user, notifying OK. If the user fails to upload the file, the system will send a message to the user, notifying FAIL. After the file is uploaded, the user is asked to execute the software. Two options are available. If the system is successfully executed, the system will send a message to the user, notifying OK. If the system fails to execute, the system will send a message to the user, notifying FAIL.

After the system is executed, the user is asked to generate the results. If the user chooses to get the audit file, two options are available. If the system successfully generates the audit file, the system will send a message to the user, notifying OK. If the system fails to generate the audit file, the system will send a message to the user, notifying FAIL. If the user chooses to print the result to screen, two options are available. If the system successfully prints the result to screen, the system will send a message to the user, notifying OK. If the system fails to print the result to screen, the system will send a message to the user, notifying FAIL. If the user chooses to send to media, two options are available. If the system successfully send the result to media, the system will send a message to the user, notifying OK. If the system fails to send the result to media, the system will send a message to the user, notifying FAIL.

Finally, the user logs off the VS.