University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Secure Multiparty Computation for Internet Voting

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Joyce Anne L. Piscos

June 2015

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "Secure Multiparty Computation for Internet Voting" prepared and submitted by Joyce Anne L. Piscos in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Richard Bryann L. Chua, M.Sc.
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*candidate*)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Perlita E. Gasmen, M.Sc. (*candidate*)	_____	_____
4. Ma. Sheila A. Magboo, M.Sc.	_____	_____
5. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____
6. Bernie B. Terrado, M.Sc. (*candidate*)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Alex C. Gonzaga, Ph.D., Dr.Eng.
Dean
College of Arts and Sciences

Abstract

Internet voting sacrifices privacy especially when there is a trusted third party involved. Secure Multiparty Computation (SMC) eliminates the need of a trusted third party. It involves multiple participants that jointly perform an agreed computation of their inputs without revealing these inputs to each other. In this study, SMC is adopted in the Internet Voting System. It is a web-based application that provides protection of voter privacy and anonymity, as well as vote verifiability using Sharemind as the SMC framework.

Keywords: secure multiparty computation, internet voting, privacy, anonymity, verifiability, Sharemind

Contents

Acceptance Sheet i

Abstract ii

List of Figures iv

List of Tables v

I. Introduction 1

A. Background of the Study 1

B. Statement of the Problem 2

C. Objectives of the Study 2

D. Significance of the Project 4

E. Scope and Limitations 5

F. Assumptions 5

II. Review of Related Literature 7

III. Theoretical Framework 15

A. Internet Voting 15

B. Secure Multiparty Computation 17

C. Sharemind 17

IV. Design and Implementation 20

A. Use Cases 20

B. System Development 23

C. Database Design 25

D. Data Dictionary 27

E. System Architecture 30

F. Technical Architecture 31

V. Results **32**

VI. Discussions **54**

VII. Conclusions **57**

VIII. Recommendations **58**

IX. Bibliography **59**

X. Appendix **64**

A. Source Code 64

A..1 Sharemind 64

A..2 Web Application Components 67

XI. Acknowledgement **122**

List of Figures

1 Ideal World vs Real World 7
2 Sharemind's representative-based architecture 18
3 Top-level use case diagram of the Internet Voting System 20
4 View the result of the election use case diagram of the Internet Voting System 21
5 Cast a vote in an election use case diagram of the Internet Voting System 21
6 Setup an election use case diagram of the Internet Voting System . . 22
7 Setup the voter's list use case diagram of the Internet Voting System 22
8 Manage the voting official list use case diagram of the Internet Voting System 23
9 Layered Architecture of the System 24
10 Relational Database for Election Configuration Details 26
11 Sharemind Secret-Shared Database of Votes 27
12 System Architecture 30
13 Home page for unregistered users 32
14 Home page for voting officials 32
15 Home page for the administrator 33
16 View results of the elections 33
17 View results of the election selected 34
18 Export results as PDF 34
19 View bulletin board 35
20 Export results as PDF 35
21 View verification code tracker 36
22 Log in to the system 36
23 View account profile 37
24 Edit account details 37

25 Change password 38
26 Cast a vote 38
27 Review ballot 39
28 View elections 39
29 Add an election 40
30 Edit an election 40
31 Delete an election 41
32 View positions by election 41
33 Add a position 42
34 Edit a position 42
35 Delete a position 43
36 View parties by election 43
37 Add a party 44
38 Edit a party 44
39 Delete a party 45
40 View candidates by election 45
41 Add a candidate 46
42 Edit a candidate 46
43 Delete a candidate 47
44 View voter types by election 47
45 Add a voter type 48
46 Edit a voter type 48
47 Delete a voter type 49
48 View voters 49
49 Add a voter 50
50 Import voters 50
51 Edit a voter 51

52 Delete a voter 51
53 View voting officials 52
54 Add a voting official 52
55 Edit a voting official 53
56 Delete a voting official 53

List of Tables

1 user table 28

2 voter_type table 28

3 voter table 28

4 election table 28

5 voter_type_elections table 29

6 position table 29

7 party table 29

8 party_elections table 29

9 candidate table 29

10 abstain table 30

11 Example code for inserting a row in a database table with one column in Sharemind 2 and 3 58

I. Introduction

A. Background of the Study

The voting process of numerous countries still employ the traditional balloting system despite the emerging technologies in the present age. However, Internet voting and its development became a topic of research in the recent years. Instead of the conventional paper-based voting, Internet voting is utilized for its benefits such as accessibility, accuracy, and speed.

Web-based voting is implemented with different approaches in United States, United Kingdom, Ireland, Switzerland, Canada, France and Estonia. For instance, voters receive their access passwords through mail in Switzerland while in Estonia, advanced national identity cards with electronic chips are used to access the system [1].

Enforcing such voting system demands certain security requirements in order to gain the trust of the voters. These security requirements include voter privacy, wherein a vote must not be associated to the voter; eligibility, in which only registered users are allowed to vote; uniqueness, or only one vote per voter should be counted; fairness, wherein no partial tally is revealed before the end of the voting period; uncoercibility, or the inability of any coercer to coerce a voter to cast his vote; receipt-freeness, in which no confirmation of the vote is provided; accuracy, or correctly counting the votes; and individual verifiability, wherein the voter can validate that his vote is tabulated correctly [2]. By making sure that these security requirements are met, a voting scheme is considered to be secure. A web voting system should preserve the privacy of the voters, that is, only the results of the elections are shown publicly.

Adopting SMC in applications like web voting is made easy through numerous available frameworks such as Sharemind. Sharemind's secure computation is done by

the three dedicated miners and these miners will perform the secret sharing that will secure the inputs of the voters.

B. Statement of the Problem

Internet voting sacrifices privacy since it is done without supervision. A trivial solution to this security problem is creating a trusted third party to handle the votes. However, data security may still be compromised because the solution created a single point of attack where all the data can potentially be stolen [3]. Moreover, anonymity, and reliability are other primary concerns in applications involving trusted third parties.

Secure Multiparty Computation (SMC) eliminates the need of a trusted third party. It involves multiple participants that jointly perform an agreed computation of their inputs without revealing these inputs to each other [4]. A multiparty computation is considered secure if it satisfies two conditions: the correctness of the output and the privacy of the parties' inputs [5]. Thus, the idea of SMC can be implemented in Internet voting wherein a voter can only know his own vote and the result of the elections, in terms of accessing the database by an unauthorized user. Adopting SMC to Internet voting is another way to ensure that the security requirements are met. SMC will not just preserve privacy of vote, but will also guarantee the parties that the output is correct.

C. Objectives of the Study

1. To create a simple web voting system using Sharemind which has the following user roles and functionalities:

    (a) Voter

        i. Cast a vote in an election

ii. Download hashes of voted candidates

iii. View the results of the elections

iv. Download the results of the elections

v. View the bulletin board of votes

vi. Download the bulletin board of votes

vii. View account details

viii. Edit account details

(b) Public User

i. View the results of the elections

ii. Download the results of the elections

iii. View the bulletin board of votes

iv. Download the bulletin board of votes

(c) Voting Official

i. Setup an election

A. Manage positions

B. Manage parties

C. Manage candidates

D. Manage voter types

E. Manage election period

ii. Setup the voter's list

A. Upload voter's list that is either in .csv format or by using an interface where the voting official can input the details.

B. Update a voter

C. Delete a voter

iii. View the results of the elections

iv. Download the results of the elections

v. View the bulletin board of votes

vi. Download the bulletin board of votes

vii. View account details

viii. Edit account details

(d) Administrator

i. Manage the voting official list

D. Significance of the Project

A web-based voting system can significantly speed up the processing of results, as well as ensure tally accuracy. Voter turnout will also increase since people with disabilities or living abroad can still vote. Moreover, it can provide potential long-term cost savings compared to other voting schemes. However, efficient security measures should be achieved to prevent attacks and malicious behavior in this kind of system. The theory behind SMC can be applied in this type of scenario. An SMC-based voting system can provide protection of voter privacy and anonymity. Correctness of the tally is also ensured and only the result of the elections is made public.

Although secure multiparty computation is not widely used in the application standpoint, it is said that it can be a powerful tool in data encryption [6]. Goldwasser also emphasized that multiparty computation will become an "integral part of our computing reality" [7]. There are only few applications that utilize SMC. With that, this project may contribute to the growing technology of SMC.

E. Scope and Limitations

1. This project is a general hierarchical voting application. Voting is within a structural organization only.
2. This project will use the Sharemind framework with only three data miners.
3. There is no voter registration in the system. Voters are added via the voters' list setup by the voting official.
4. Weighted voting will not be considered.
5. Only means to verify authenticity of voters is purely through their email addresses.
6. Failure of election will not be considered.
7. There is no provision for certification of source codes.
8. Determination of nuisance candidates will not be considered.
9. Write-in votes will not be considered.

F. Assumptions

1. There is no way for miners to deviate from the protocol.
2. Voting official has valid email addresses of the voters.
3. Email addresses of voters are secured.
4. The voting official decides the acceptability of email addresses.
5. An abstain vote represents not voting for any candidate in a certain position.
6. The system will not be able to capture proxy voting/flying voters.

7. Election protests will not be entertained.

8. The total number of voters will not exceed the 32-bit integer.

II. Review of Related Literature

Trusting a third party to handle private inputs from parties is not practical for applications involving sensitive and confidential data. Secure multiparty computation eliminates this need of a trusted third party. It involves the communication of two or more parties, each of which has private inputs, and is implemented using an agreed computation without the use of a trusted party—thus resembling a real world model [8]. In contrast with the ideal world as shown in Figure 1, parties will not depend on a trusted third party to compute the result of their inputs [6].

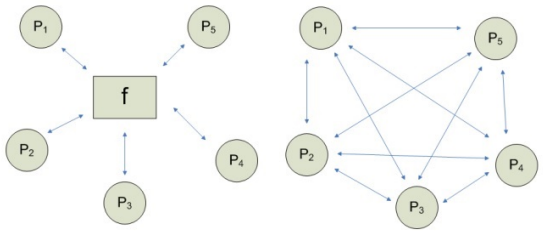

Figure 1: Ideal World vs Real World

In 1982, Yao [9] introduced the SMC problem by introducing the millionaire's problem—determining who among two millionaires is richer without revealing their salaries. He proposed theoretical solutions to the problem involving different types of protocols. Yao's garbled circuit evaluation protocol for two parties was one of them wherein one party generates the circuit and the other party evaluates that circuit. It was proven that any function computable in polynomial time can be computed with a logical circuit—this implies that any function computable in polynomial-time can be computed securely with polynomial communication and computation and a constant number of rounds [10]. Hence, the problem of SMC can be solved also in polynomial time, unlike other schemes such as fully homomorphic encryption [11].

Yao also suggested in [9] numerous privacy-preserving applications where SMC can be realized. It is not until 2008 that a practical application of SMC was developed. The first large-scale SMC application was a secure double auction used for trading

sugar beet contracts with Danish farmers to Danisco, a sugar beets processor [12]. Bids of farmers should be kept private because it could affect the whole bidding process if the auctioneer knows the economic status of the bidders. SMC was used to keep the bids private. This auction system provided a basis for the use of SMC in other applications.

In this regard, numerous SMC frameworks are now available for creating applications based on multiparty computation using different types of architecture—Fairplay [13] and FairplayMP [14] with circuit-based architectures, Sepia [15] and Sharemind [16] with representative-based architectures, and Canon-MPC [17] and VIFF [18] with purely SMC architectures. These frameworks provide developers the fundamental elements, i.e. an Application Programming Interface (API), in implementing multiparty computation applications.

Fairplay is the first practical SMC implementation based on Yao's garbled circuits for two-party computation. It was developed in 2004 by Malkhi et al. [13]. The framework used a high-level programming language that resembles the C language called Secure Function Definition Language (SFDL), for programming the secure computation; and a low-level language called Secure Hardware Definition Language (SHDL), for describing the boolean circuit [13]. In 2008, it was extended to a multiparty version called FairplayMP implemented using Java. Despite being the first SMC frameworks, there are no known real-world applications of Fairplay and FairplayMP. Some disadvantages were seen in this framework as it performed poorly in runtime because of the circuit evaluation running in a number of rounds.

Security through Private Information Aggregation or SEPIA, is also one of the available SMC frameworks. It is an open-source Java library that uses Shamir's secret sharing. SEPIA's representative architecture consists of input peers which performs the secret sharing, and privacy peers which performs the actual computation. SEPIA uses Java over C++ because of platform independence [15]. This framework

achieves high performance for parallel execution of private operations [19]. As of version 0.9, SEPIA is robust against input/privacy peers connection failure [19]. One application that used SEPIA as the SMC framework is a system for troubleshooting network outages by determining the root cause developed by Djatmiko et al. [20]. In analyzing the root cause of outages, intelligent correlation of measurements from different Internet locations is necessary. Traffic monitoring should be private, and SMC provides privacy preserving for near-real-time monitoring of outages [20].

Developed for privacy-preserving computations, Sharemind uses additive secret sharing scheme for securing the inputs [16]. In the framework's architecture, secure computation is done by the three dedicated miners. Controller applications in Sharemind are implemented using C++, while protocols are implemented using the SecreC language. Among the frameworks, Sharemind has the simplest and most efficient platform for creating and testing SMC applications. Using C++ as the implementation language has increased the speed of Sharemind applications [21]. In addition, Sharemind's SecreC, a C-like procedural language, is similar to how developers normally code basic operations. SecreC also offers flexibility since public and private data separation is explicitly done by the programmer. Sharemind uses vectorized operations for improved performance contrary to VIFF and Fairplay [21]. Although computation miners of Sharemind are limited to three unlike SEPIA, it is proven to be the most optimal case since adding parties increases the communication overhead quadratically [22, 23]. Configuration and setup is also much simpler compared to SEPIA. Moreover, the framework is being actively developed by the Cybernetica team in Estonia to update the current functionalities and improve their performance.

The first real-world Sharemind application was designed by Talviste [24] for the Estonian Association of Information Technology and Telecommunications (ITL). The ITL application collected basic economic indicators from the ITL members and utilized this information for benchmarking analysis. The indicators were necessary in

comparing themselves to other companies in the information and communication technology sector. These indicators include total return, number of employees, percentage of export, added value, labour costs, training costs, and profit. Privacy of the inputs is needed since the members do not want their information to be leaked to other members.

Since the Sharemind controller library does not have a web interface, Talviste used a web programming language outside of the Sharemind Software Development Kit to develop the web interface. He created a JavaScript controller library that performs the secret sharing in the web browser. The shares, together with a randomly generated 32-bit session identifier, were then sent to the web servers of each Sharemind miner and were saved to a buffer database. The miners also executed a proxy application that retrieved all the shares from the buffer database and transferred them to the miners' internal database. Execution of the proxy application can be done periodically or after the data collection has ended. The results were stored and were used to create reports such as tables, charts, spreadsheet or PDF document.

Another application using the Sharemind framework is the online survey prototype developed by Estonian companies Cybernetica, Quretec, and Software Technology and Applications Competence Centre (STACC) [16]. Similar to the previous application, survey results were collected to generate reports. It also used Talviste's JavaScript controller library.

Emerging as a new framework is the CAsual NON-interactive secure Multi-Party Computation or Canon-MPC[1]. The main objective of Canon-MPC is to run secure computation without any complicated setup [17]. By simply accessing a website, anyone can run SMC in their web browsers. Unlike FairplayMP, Canon-MPC is secure against malicious adversaries. The framework is implemented using Google's Native Client technology that enables running C or C++ code in the web browser.

[1] canon-mpc.org

However, this technology is limited only to the Google Chrome browser. Moreover, Canon-MPC is not built for running computations with sensitive data because of the non-usage of TLS in the transmission of data from client's browser to the server [17].

Another SMC framework is the Virtual Ideal Functionality Framework or VIFF. It is an open-source Python library that uses Shamir and pseudo-random secret sharing. The framework allows running operations in parallel and it was shown from the benchmark results that VIFF is reasonably fast [13]. The Danisco double auction system was successfully repeated using this framework [21]. VIFF was also used in performing AES encryption and decryption operations through SMC [25]. Another application using the VIFF framework is an author ranking system designed by Vegge [6] using Python's graphical user interface library. However, its complicated setup led Vegge to implement VIFF in a web voting application through the use of Java applets running in the client side. All of the security requirements of a voting protocol were satisfied in this system except individual verifiability.

One of the areas that will find secure multiparty computation useful is Internet voting. The first country to offer Internet voting nationally is Estonia [26]. Estonia offers several government web services through the use of e-ID cards for authentication. These cards are also used for Internet voting to guarantee voter eligibility. To ensure voter secrecy, their voting protocol implements the double envelope system wherein the inner envelope contains only the encrypted vote and the outer envelope contains the voter details that were digitally signed through the e-ID. These envelopes are later separated after the voting period. For the voter to verify his or her vote, he or she can scan the QR code generated after the casting of the vote. Seven days are allotted for the Internet voting period and three days for cross-checking before the actual election day. The election day utilizes the old paper voting for those who failed to vote online or wanted to change their vote. Security is one of the primary concerns in Internet voting. Although there is a cryptographic protocol in the Estonian voting

scheme, malicious attacks can obtain information through the voter's e-ID card, PIN, or QR code. In the security analysis of Springall et. al [26], they attempted to test the vulnerability of the system by creating a malware that captures the PINs and silently replaces the user's vote. They also created a malicious verification app that supports the previous malware by displaying whatever candidate is embedded in the QR code, rather than the candidate for whom the vote was actually cast.

Norway also followed suit as they launched the E-valg 2011 project which aimed to increase voter accessibility. The E-valg 2011 project focuses on ten municipalities with approximately 200,000 voters. Their implementation also utilizes the double envelope system. The difference of the Norwegian voting protocol is the use of a pre channel, through the use of postal mail; and a post channel, through the use of a mobile phone [27]. Each voter receives a postal mail containing the names of the candidates and their corresponding verification codes. After casting the vote, the voter will receive an SMS with the verification code. This code is then double-checked with the verification codes in the postal mail to verify if the vote is correct. Security issues are also present in this type of scheme. One of which is the possibility of SMS attacks that can be in the form of an SMS virus or SMS injection. Moreover, the postal system cannot be fully trusted since insiders may steal the code from the printing house [27].

On a smaller scale, Université catholique de Louvain (UCL) in Belgium also used an online voting application for their university president elections. They used a custom deployment of Helios, a web-based open-audit system. Voter coercion issue is not one of the primary concerns of Helios thus the system is only ideal for online software communities, local clubs, student government, and other environments [28]. UCL changed Helio's vote encryption from the use of mixnets to homomorphic tallying since they employed a vote weighting computation for tallying the results wherein more weight is assigned to the faculty voters than the researchers, administrative

staff, and the students. In addition, UCL's existing authentication system was used instead of Helio's default settings. A bulletin board of voter ids and voter hashes was also another feature at the audit phase for the verification of the votes.

Estonia, Norway and UCL all used a trusted third party in their Internet voting schemes. Their security architecture forces them to trust the servers which are used for forwarding, storage, and counting of the votes. In order to put more trust in Internet voting and to remove a point of single failure, the need for a trusted third party should be removed.

One of the Internet voting protocols that removes the need for a trusted third party is the zero-hacking protocol of Mishra et. al. [29]. It is divided into the several layers, namely, fragmentation, anonymous, and computation layers, with interfaces that serves as the communication of these layers. Votes were fragmented in the fragmentation layer and were distributed across the network to the anonymizers that hid the identity of the voter at the anonymous layer. As a result, votes from a particular location were not known. There is no way to tally the votes by district as there is only one tally, the tally for the national level. The votes were then collected by tally systems for counting in the computation layer. One of the tally systems was arbitrarily chosen to perform the necessary computation, called the master TTP. However, the zero-hacking protocol is not entirely SMC since only one TTP computes for the result of the function.

Another voting scheme is proposed by Gang [30] satisfying the voting protocol security requirements. A balloting scheme based on the matrix security summation in transmission was used in this study. A third party will generate random numbers for each voter. These random numbers are then added to each of the partitioned secure data of the original vote that voters will send to each other. The collector then computes for the result using SMC. Similar to the previous voting protocol, this study also has a representative-based architecture and proposed to split the data

before sending to the computation server.

III. Theoretical Framework

A. Internet Voting

Internet voting is a type of voting system wherein voters can use any personal computer with Internet connection to cast a vote [31]. By making voting as convenient as possible, Internet voting improves accessibility to absentee voters, voters in remote areas, and voters with mobility issues [32]. As a result, voter turnout could possibly increase. It also provides faster processing of results and more accurate tallying. However, making the voting process available online introduces security risks. These security risks include unauthorized voter casting, voter impersonation, ballot stuffing, voter privacy compromise, voter coercion, vote buying, vote modifying, and vote deletion [33]. Hence, voting schemes must satisfy a list of security requirements. Çetinkaya [2] listed down these requirements:

1. Voter Privacy - A vote must not be associated to a voter.
2. Eligibility - Only registered voters are allowed to vote.
3. Uniqueness - One vote per voter should be counted.
4. Fairness - No partial tally is revealed before the end of the voting period. The result must be published at the end of the election.
5. Uncoercibility - Any coercer should not be able to coerce a voter to cast his vote.
6. Receipt-freeness - To prevent vote buying or selling, the system must not provide a confirmation of the receipt of the vote that yields content of the voter's identity and his or her vote.
7. Accuracy - All valid votes should be counted correctly. In addition, all counted votes should satisfy eligibility and uniqueness.

8. Individual Verifiability - The voter should be able to check that his encrypted vote was counted and tabulated correctly.

Based and Mjølsnes [34] also listed some additional security requirements in voting schemes:

9. Integrity - A cast vote should not be altered or modified by any unauthorized party.
10. Robustness - All components of the voting scheme must be functional.
11. Soundness - An invalid vote should be discarded.
12. Completeness - All valid votes should be tallied.

To satisfy these security requirements, three classes of computing protocols were proposed: homomorphic encryption, mix net, and secure multiparty computation. Homomorphic encryption is a cryptosystem wherein algebraic operations are performed on ciphertext and getting ciphertext as the result [35]. Zhao et. al. [35] designed a system wherein voters send encrypted ballots to tellers who will add the votes in ciphertext using the additive homomorphic encryption algorithm. Tellers will send the encrypted results to the announcers whose job is to decrypt them and to reveal the result to the public. Another type of a cryptographic protocol used in Internet voting schemes is a mix net. Invented by David Chaum, a mix net shuffles and reorganizes encrypted data and decrypts it [31]. For a mix net-based voting scheme to satisfy voter verifiability, it has to prove that the correct operation is done [31]. The first provable mix net is the Sako-Kilian mix net, which is used in Helios 1.0, a web-based open-audit system [28]. Secure multiparty computation can also be utilized for secure web voting. SMC is implemented voting applications in [6], [30], and [36].

B. Secure Multiparty Computation

Secure multiparty computation is the communication of two or more parties, each of which has private inputs, and is implemented using an agreed computation without the use of a trusted party [8]. SMC is formally defined as the problem where n parties P_1, ..., P_n with inputs $x_1, ..., x_n$ jointly compute an agreed function $f(x_1, ..., x_n)$ to produce $(y_1, ..., y_n)$ such that party i is guaranteed to learn y_i [5].

Having all parties engage in SMC requires them to be online at the same time [3]. A representative-based approach is suggested wherein parties split the roles into groups—input servers, computation servers, and output servers [3]. A logical circuit-based approach is also another way to solve the SMC problem. In Yao's garbled circuit, the function f can be represented as a logical circuit which is securely evaluated, masking the inputs and outputs of each gates [37].

Security in protocols of a multiparty computation depend on the behavior of the players that participate in the computation. These players are classified into three types: honest, passive, and active adversaries [38]. Honest players follow the agreed computation without trying to cheat. Passive, honest-but-curious or semi-honest adversaries also follow the protocol, however, they may try to collude with other players to learn more information on the data. Active or malicious adversaries deviate from the protocol with the goal to abort the protocol, to produce incorrect results, and to obtain additional information from other players. These adversaries are considered in creating secure protocols in SMC. When majority of the participants are honest, it is possible to securely compute any function [10].

C. Sharemind

Sharemind[2] is an SMC framework designed for fast privacy-preserving computations developed by the Cybernetica team lead by Bogdanov et. al. in Estonia [39]. It has a

[2] https://sharemind.cyber.ee/

representative-based architecture wherein secure multiparty computation is done on the three miners, and is proven to be secure on a semi-honest corruption. The number of dedicated miners is considered optimal since increasing the number of miners will increase communication complexity quadratically [23]. The three miners perform the secret sharing of the data provided by the data donors, results are then published through data mining algorithms as shown in Figure 2.

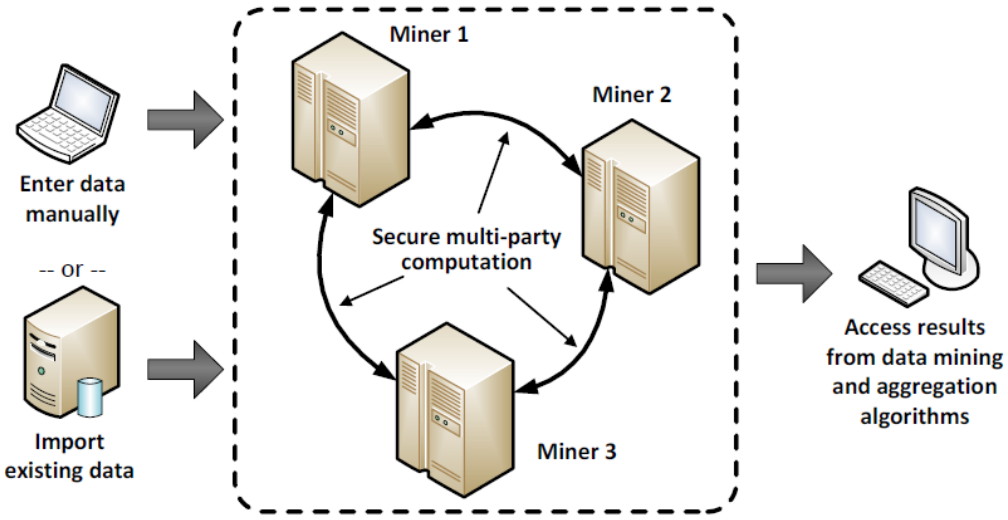

Figure 2: Sharemind's representative-based architecture

The framework uses additive secret sharing over the ring of 32-bit integers and provides basic mathematical operations. The operations supported are secure addition, multiplication, comparison, and bit extraction of 32-bit integers [21]. Sharemind also provides a simple and efficient platform for creating SMC applications through the use of a procedural language called SecreC. SecreC is a C-like language that is used to program privacy-preserving data mining algorithms to be executed by the three miners [40]. SecreC allows developers to specify public and private data, and can perform vectorized operations without the need of loops. When a SecreC program is compiled, each SMC operation is implemented as an assembly instruction which is interpreted by the miners [40]. Below is an example of a SecreC code from [40] for finding a needle in a haystack:

```
public int count (private int needle, private int[] haystack) {
public int n; n = getRowCount(haystack);

// An indicator vector of ones and zeroes
private bool[n] indicator;
indicator = (haystack == needle);

private int sum; sum = vecSum(indicator);
public int count; count = declassify(sum);
return count;
}
```

On the other hand, controller applications that accepts input from data donors, are developed using C++. To communicate with the miners, a controller library provided by Sharemind is used to develop these controller applications. The choice of C++ over Java increased the speed of operations [21]. Hence, Sharemind is considered to be faster compared to other SMC frameworks.

IV. Design and Implementation

A. Use Cases

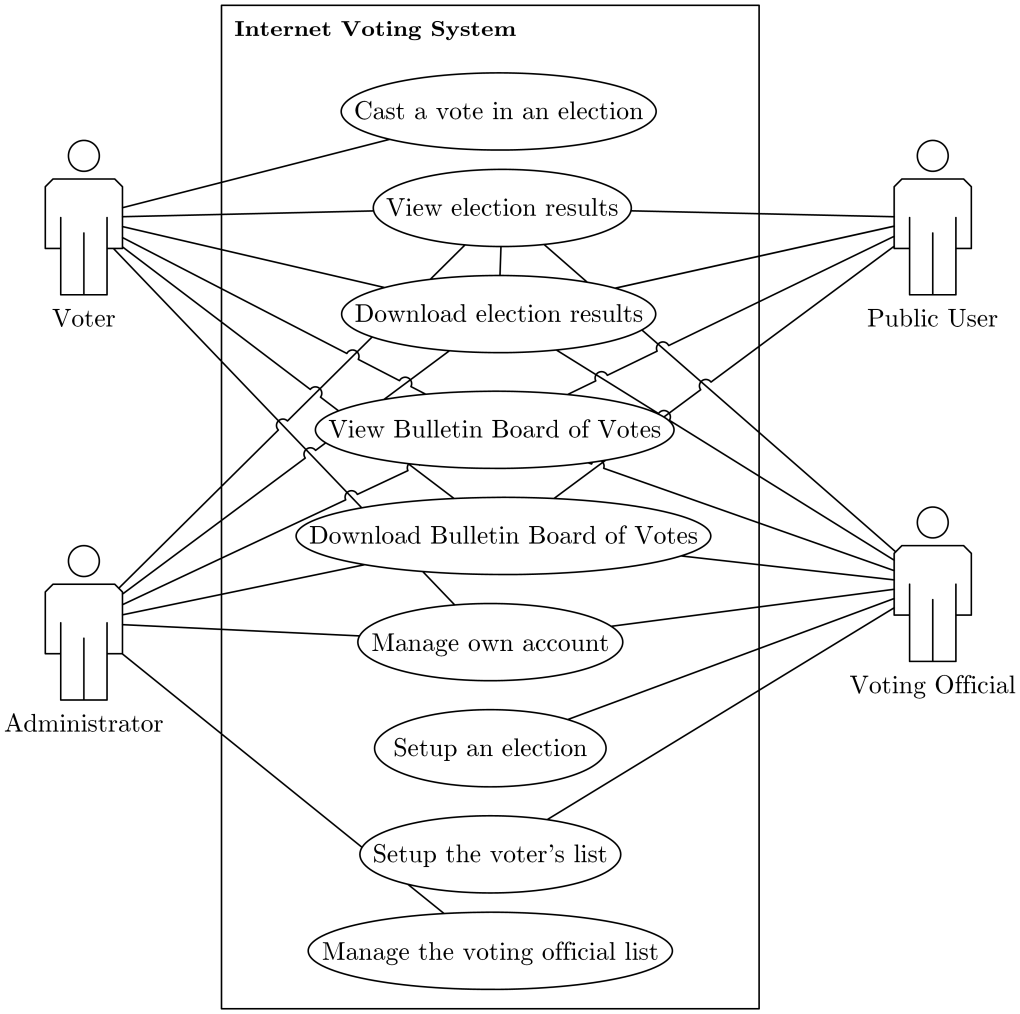

Figure 3: Top-level use case diagram of the Internet Voting System

The system consists three users: the voter, the voting official, and the administrator—the voter can cast a vote; the voting official is responsible for setting up the election and the voter's list; and the administrator will manage the voting official list. Once the election is done, all of the users can view the result of the election. In addition, any user can view the bulletin board of votes, where cast votes are displayed with their corresponding voter identification called aliases.

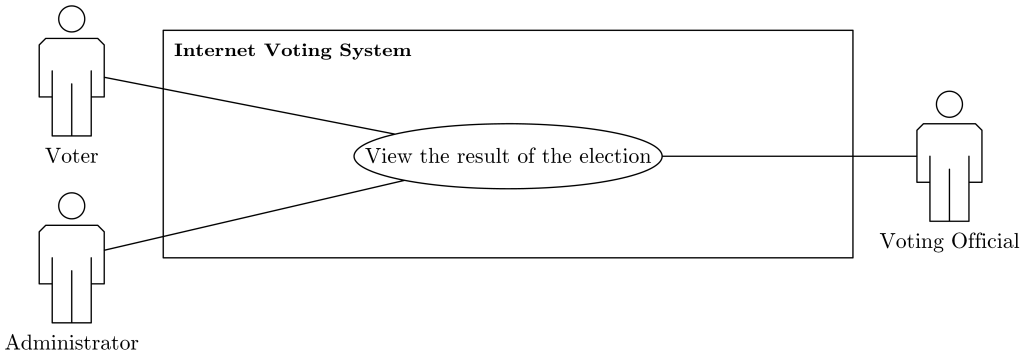

Figure 4: View the result of the election use case diagram of the Internet Voting System

The results can also be downloaded in a Portable Document Format (PDF).

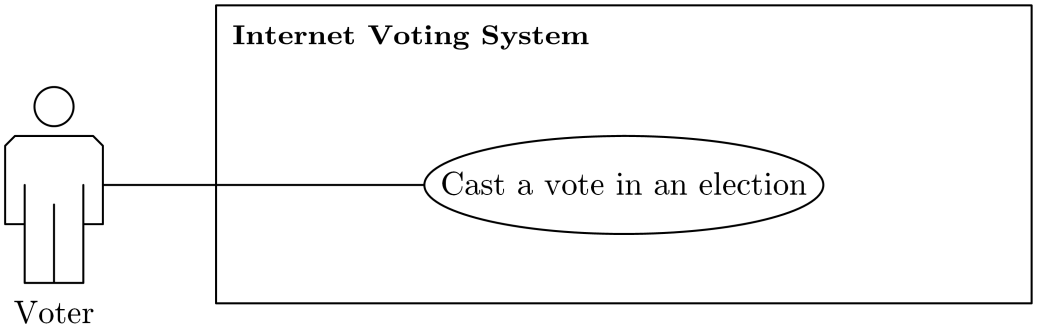

Figure 5: Cast a vote in an election use case diagram of the Internet Voting System

After a successful authentication, the voter can cast a vote in his/her assigned election/s. After submitting the votes, the corresponding hashes are displayed, and the voter can review the ballot before the final casting of the vote. The voter can download a text file containing these hashes to verify his/her votes in the bulletin board. The voter can also overwrite his/her previous votes once he/she already submitted his/her final votes.

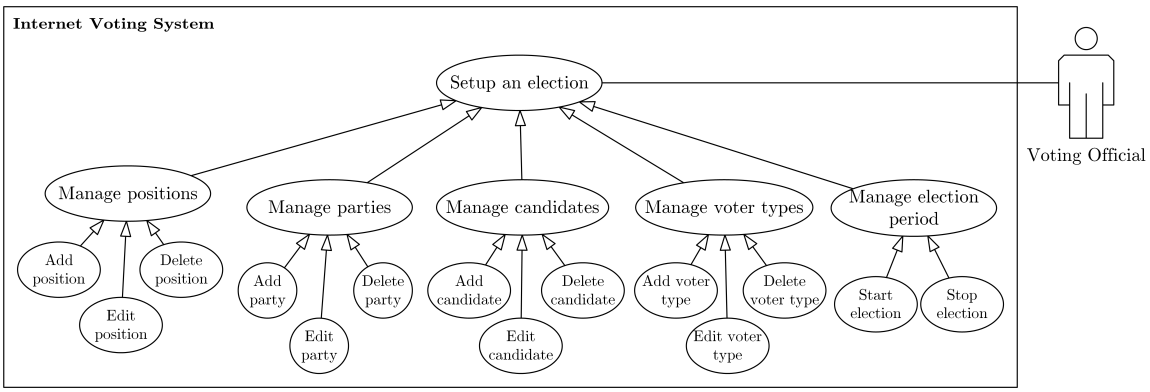

Figure 6: Setup an election use case diagram of the Internet Voting System

Setting up an election requires managing the positions, parties, and candidates. In addition, the voting official can also start and stop an election.

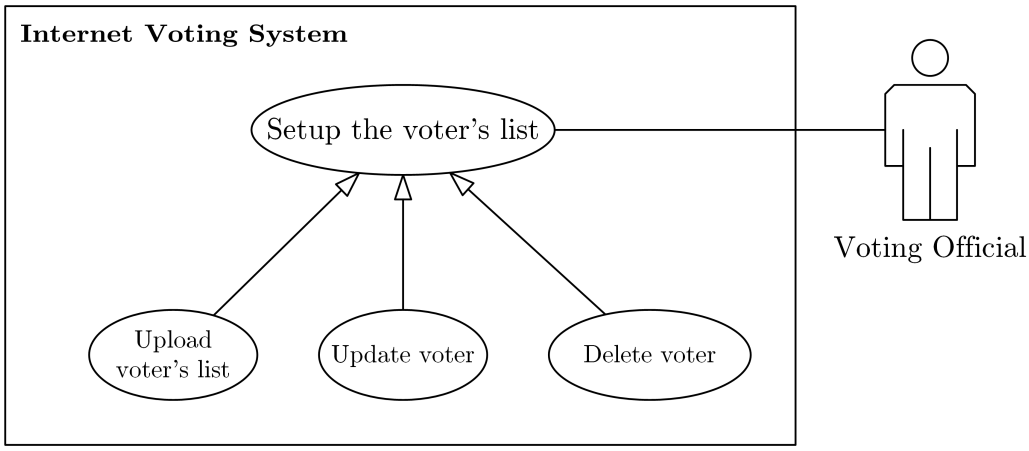

Figure 7: Setup the voter's list use case diagram of the Internet Voting System

The voting official can upload the voter's list via a web interface or via a .csv file. An email containing login credentials will be sent to each voters. A generated password will be used for the first login.

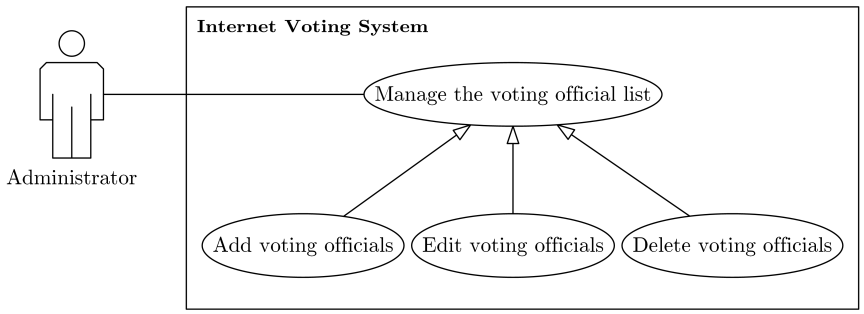

Figure 8: Manage the voting official list use case diagram of the Internet Voting System

The administrator is responsible for managing the voting official list. Similar to the voters, voting officials will receive the login credentials through email.

B. System Development

The system is implemented in the Java EE platform using the Spring framework, Hibernate object-relational mapping framework, and Thymeleaf template engine. The web application employs Spring's Controller-Service-Repository layered architecture—controllers that are responsible for handling user inputs, services that comprise all the business logic of the system, and repositories that communicate with the data storage.

For the data storage, two databases are used in the system, namely the relational database implemented in MySQL and Sharemind's database. All the election configuration and results are stored in a relational database, since these data do not need to be kept private. Spring JPA and Hibernate are used to handle all MySQL operations. On the other hand, votes are stored in Sharemind's secret-shared database. The votes are stored as integer values through the ids of the position and candidate that comprise each vote.

The system's business logic communicates with the repositories, as shown in Figure 9. Included in this layer is the execution of the Sharemind processes. During

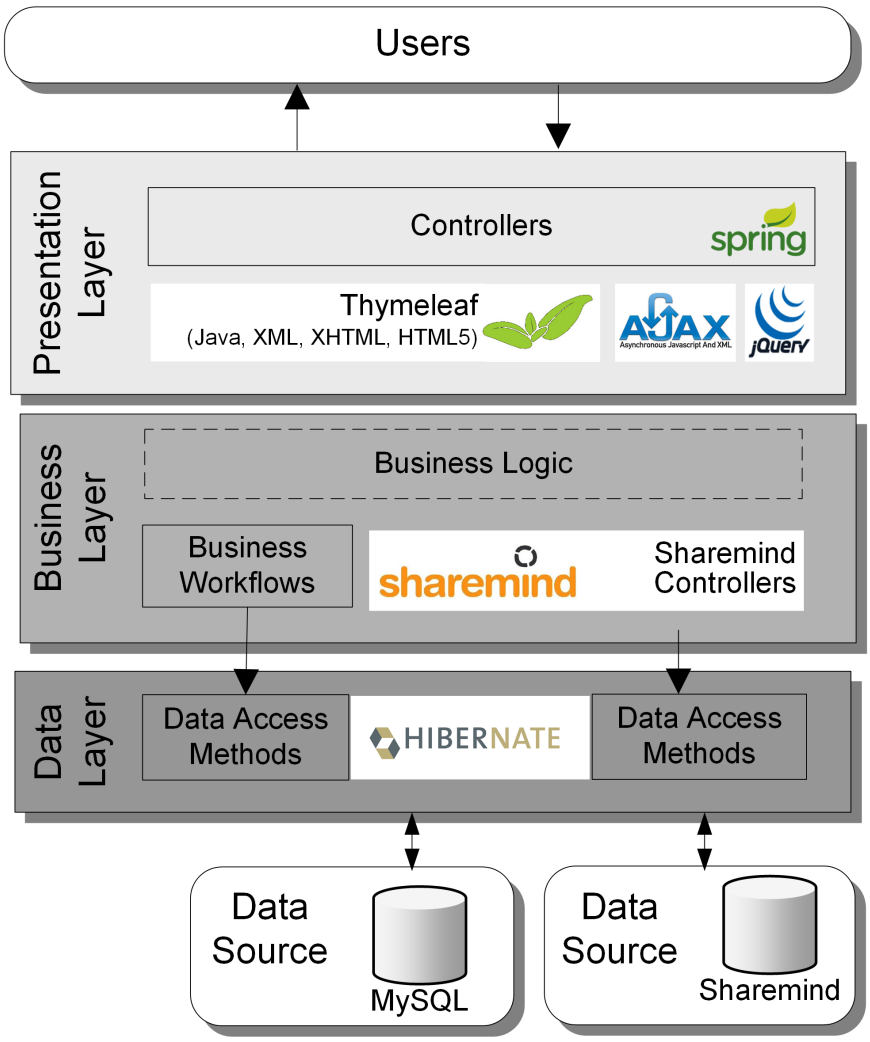

Figure 9: Layered Architecture of the System

the voting process, computation of results, and retrieval of bulletin board entries, Sharemind controllers are called by the system. To carry out these processes, these controllers must run the SecreC scripts that are intended for each process. To store a vote in the Sharemind database, a script for database entry is used. On the other hand, two scripts are used to overwrite and update the vote—one for deleting the previous votes and one for adding the new votes. To compute for the results of the computation, a script for counting the occurrences of the candidate ids is used. For retrieving the vote of a single user, a script returning the candidate ids is used. These SecreC scripts are responsible in performing secure multiparty computation by con-

necting to the three miners to obtain the results. The scripts used SecreC's built-in functions such as database operations like creating a private table, and inserting and deleting rows; vector operations such as element addition; and utility functions such as declassifying private data and publishing the results. These operations are all computed by the three miners through DevMiner.

In addition to the Sharemind processes, the processes for vote verifiability are also included in the business layer. Voters are assigned with salts that are appended to the voted candidate, and processed with a cryptographic hash function using the MD5 message digest algorithm. After submitting a vote, an alias and a verification code is produced and a JavaScript Object Notation (JSON) is generated. The JSON string consists of attribute-value pairs that is easy for any person to understand. The JSON includes the elections, positions in those elections, and the hashes of the voted candidates for those positions. All of this information are viewed in the bulletin board.

Lastly, controllers are used to render views and to handle inputs. These controllers communicate with the appropriate services for each client request. Spring MVC, Spring Security, and Thymeleaf template engine are used to manage these requests.

C. Database Design

The Internet Voting System consists of two databases, a relational database for storing the election configuration set up by the voting official, and a secret-shared Sharemind database for storing the votes.

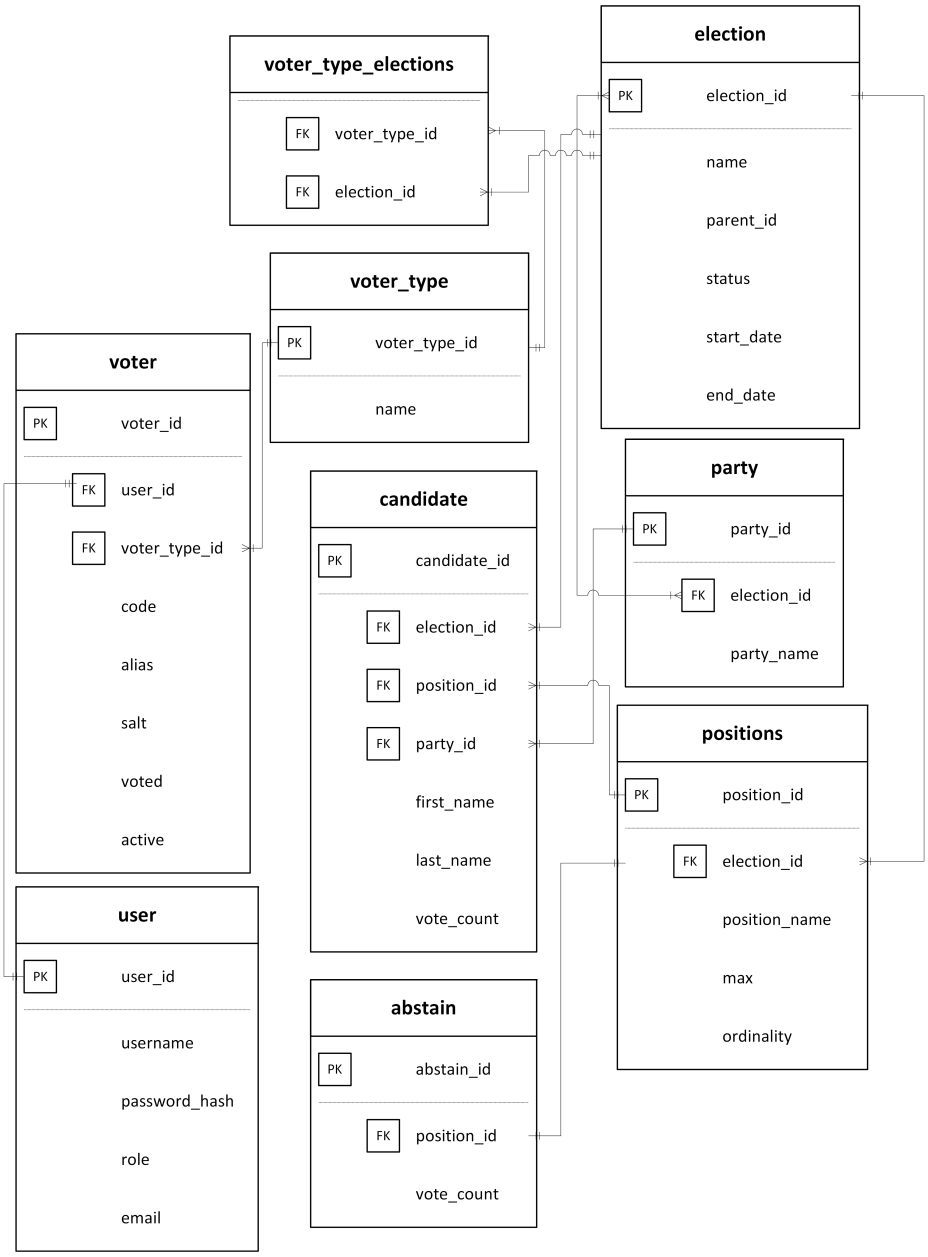

Figure 10: Relational Database for Election Configuration Details

The `position_id`, `candidate_id`, and `voter_id` fields are used for storing the votes in the Sharemind database. In addition, the `election_id` field in the position table is used to know which election a position belongs. For instance, a `position_id` of 1 is mapped to the university election with `election_id` of 1, while a `position_id` of 2 is mapped to the college election with `election_id` of 2, although these two positions are for the Chairperson position for both elections. This implementation is used to avoid confusion in the Sharemind database.

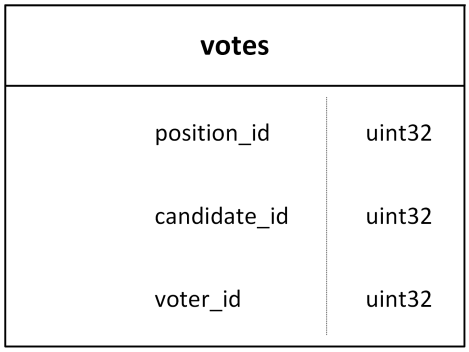

Figure 11: Sharemind Secret-Shared Database of Votes

Votes are stored in secret-shared form which means a share is stored in each miners, and not the original value. Each of the three miners has a copy of the secret-shared database. The columns are the ids of the position, the voted candidate, and voter who cast the vote. The `position_id` and `candidate_id` can be used to query the relational database to get other election information such as the election and party a position or candidate belongs. On the other hand, the `voter_id` is used for bulletin board entries.

D. Data Dictionary

Column	Type	Description
user_id	int(11)	ID of the user
username	varchar(20)	Username of the user
password_hash	varchar(255)	BCrypt-hashed password of the user
email	varchar(30)	Email of the user
role	enum('VOTER', 'VOTING_OFFICIAL', 'ADMIN')	Role of the user

Table 1: user table

Column	Type	Description
voter_type_id	int(11)	ID of the voter type
name	varchar(20)	Name of the voter type

Table 2: voter_type table

Column	Type	Description
voter_id	int(11)	ID of the voter
user_id	int(11)	User ID of the voter
voter_type_id	int(11)	Voter Type ID of the voter
alias	varchar(10)	Alias of the voter
salt	varchar(10)	Salt of the voter
code	varchar(50)	Verification code of the voter
voted	bit(1)	Voting status of the voter
active	bit(1)	Status of account

Table 3: voter table

Column	Type	Description
election_id	int(11)	ID of the election
name	varchar(20)	Name of the election
parent_id	int(11)	Election ID of the election's parent
status	bit(1)	Election period status of the election
start_date	datetime	Timestamp when the election is started
end_date	datetime	Timestamp when the election is ended

Table 4: election table

Column	Type	Description
voter_type_id	int(11)	ID of the voter type
election_id	int(11)	ID of the election associated with the voter type

Table 5: voter_type_elections table

Column	Type	Description
position_id	int(11)	ID of the position
name	varchar(20)	Name of the position
max	int(11)	Maximum number of elected candidates for the position
ordinality	int(11)	Order of the position in the ballot
election_id	int(11)	Election ID of the position

Table 6: position table

Column	Type	Description
party_id	int(11)	ID of the party
name	varchar(20)	Name of the party

Table 7: party table

Column	Type	Description
party_id	int(11)	ID of the party
election_id	int(11)	Election ID of the party

Table 8: party_elections table

Column	Type	Description
candidate_id	int(11)	ID of the candidate
first_name	varchar(30)	First name of the candidate
last_name	varchar(30)	Last name of the candidate
election_id	int(11)	Election ID of the candidate
position_id	int(11)	Position ID of the candidate
party_id	int(11)	Party ID of the candidate
vote_count	int(11)	Total number of votes of the candidate

Table 9: candidate table

Column	Type	Description
abstain_id	int(11)	ID of abstain
position_id	int(11)	ID of the position with abstain
vote_count	int(11)	Total number of abstain votes

Table 10: abstain table

E. System Architecture

The system and its components are all running on Sharemind's virtual machine (Linux Debian 6.06). Apache Tomcat was used as the system's web server, with MySQL as its database server. For SMC processes, the appropriate Sharemind controllers are called. The vote submission calls the controller for adding a database entry. The computation of the results calls the controller responsible for counting the occurrences of the candidate ids. The retrieving of bulletin board entries calls the controller for retrieving the votes of the voters. These controllers then connect with Sharemind's secret-shared database.

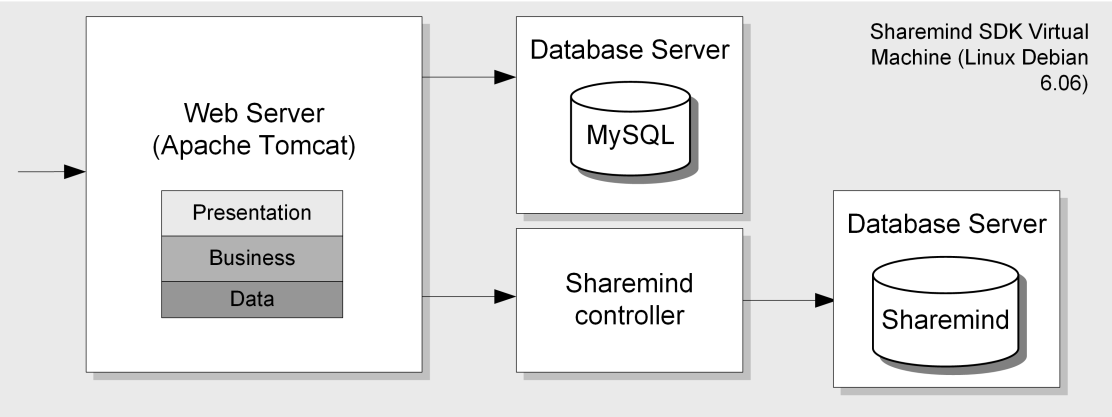

Figure 12: System Architecture

F. Technical Architecture

Using Oracle VM VirtualBox, the system should be running on the Sharemind virtual machine, installed with Tomcat, MySQL, and a JavaScript-enabled browser. The DevMiner is run locally due to limitations in the availability of Sharemind.

Host OS Requirements

- Windows 8.1, 64-bit
- Oracle VM VirtualBox 4.3.20

Guest OS Requirements

- Linux Debian 6.06, 64-bit
- 1.92GB RAM
- 1.80GHz processor
- Sharemind 2 SDK (DevMiner, SecreC)
- Apache Tomcat 7
- MySQL 5
- Any JavaScript-enabled browser

V. Results

The following are the home pages for each of the users of the system. The home page of the voter is redirected to the voting page. There are different navigation links depending on the role of the user. A user can log in to the system by clicking the Get Started button. For voting officials, clicking the Get Started button will redirect them to the elections page. For the administrator, clicking the Get Started button will redirect him/her to the voting officials page.

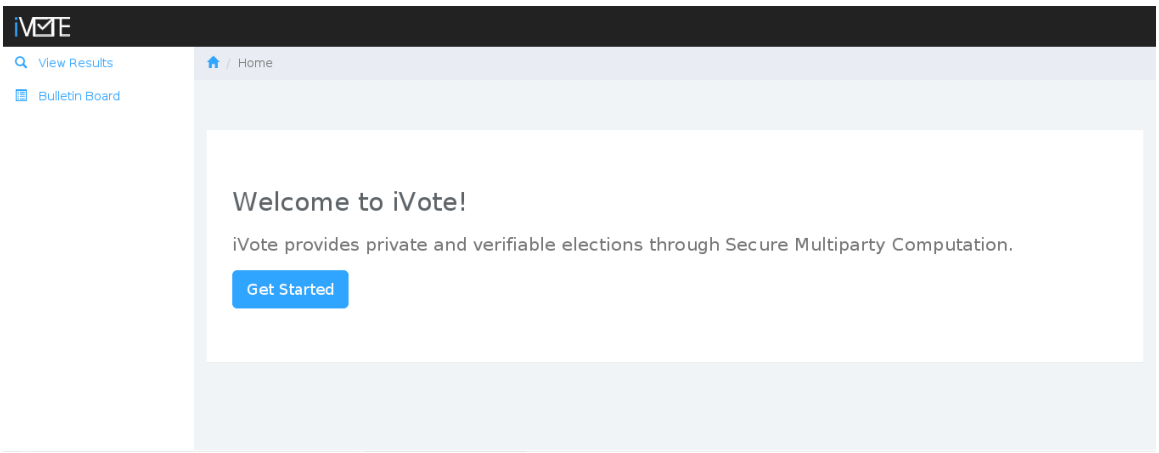

Figure 13: Home page for unregistered users

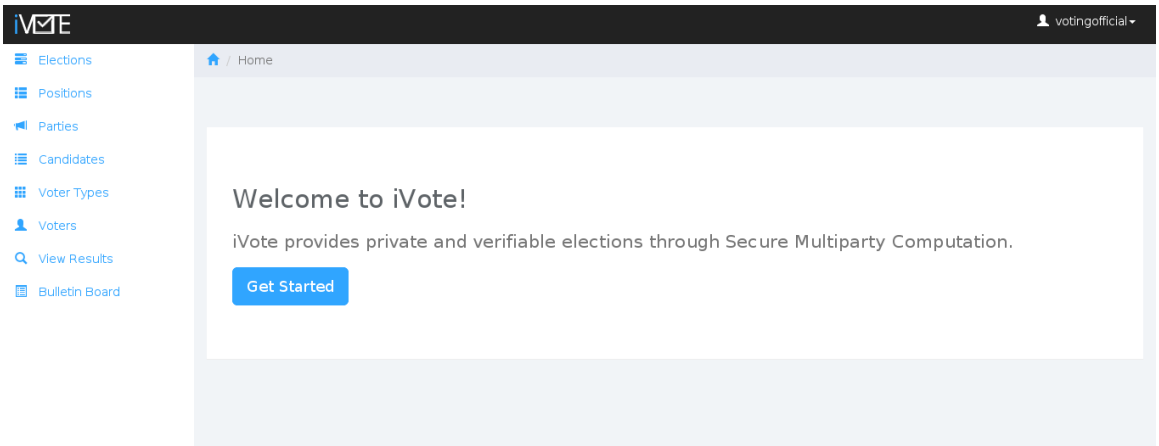

Figure 14: Home page for voting officials

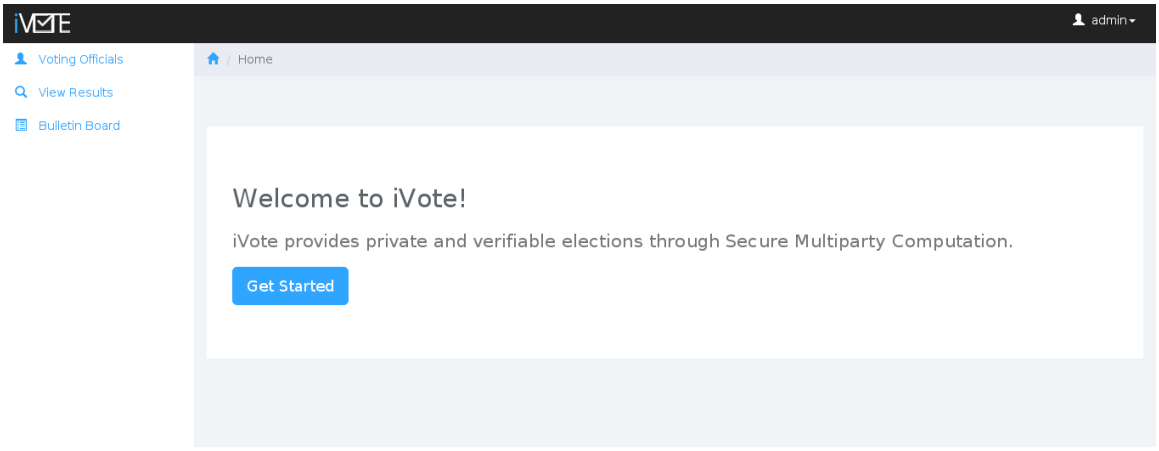

Figure 15: Home page for the administrator

The following screenshots show what all users (unregistered users, voters, voting officials, and administrators) are allowed to do in the system.

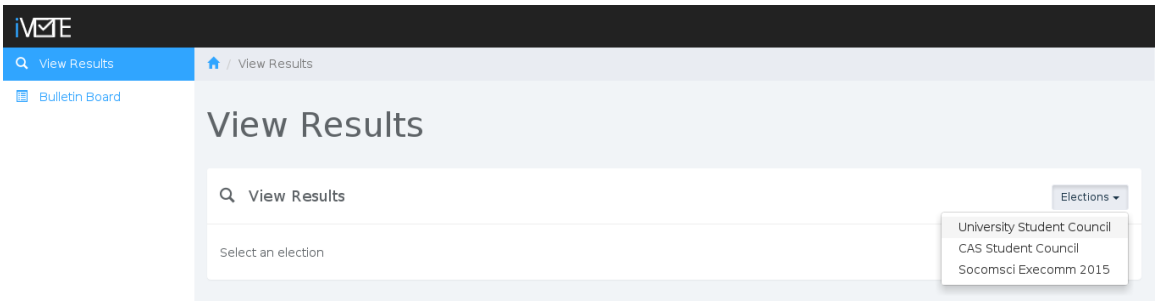

Figure 16: View results of the elections

The user selects the election he/she wants to view through the dropdown at the upper right part of the panel. A user can only view the results of the finished elections.

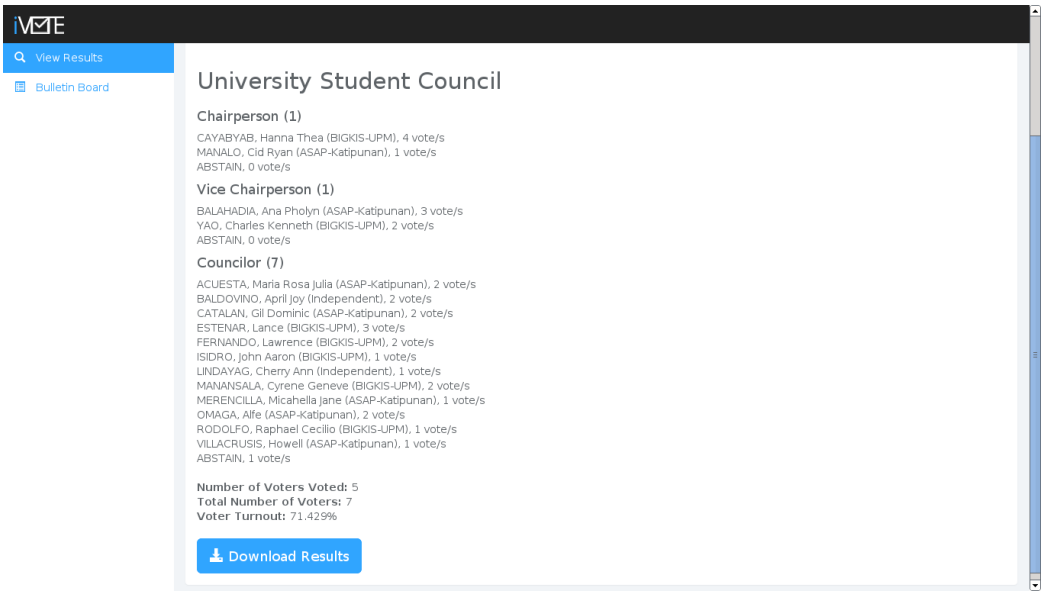

Figure 17: View results of the election selected

The corresponding tally of the votes per candidate is shown as well as the abstain votes. In addition, the number of cast votes, total number of voters, and the voter turnout are also displayed. The user can export these results in a PDF by clicking the Download Results button.

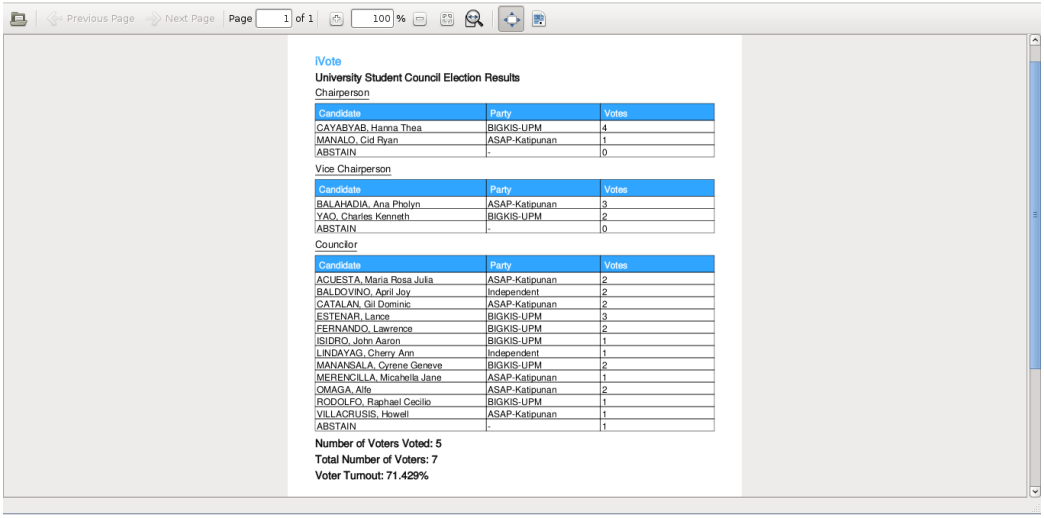

Figure 18: Export results as PDF

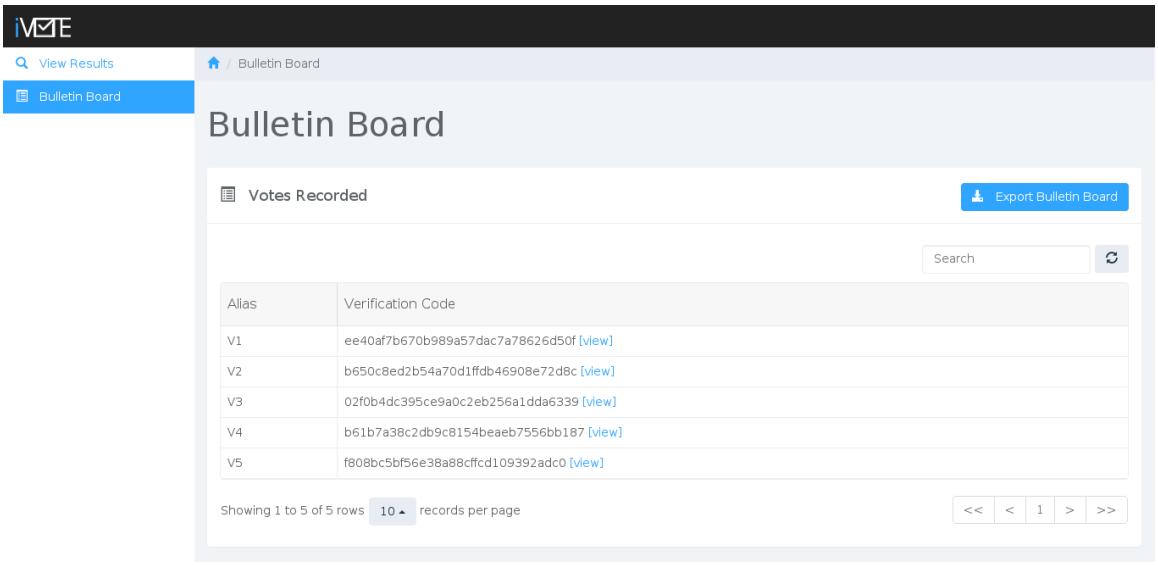

Figure 19: View bulletin board

Voter aliases and their corresponding verification codes are shown in the bulletin board. Clicking the view link for a certain voter verification code will display the JSON string consisting of the hashes of his/her voted candidates. The user can search the alias or verification code by typing the search query in the text box above the table. The user can export the bulletin board in a PDF by clicking on the Export Bulletin Board button at the upper right of the panel.

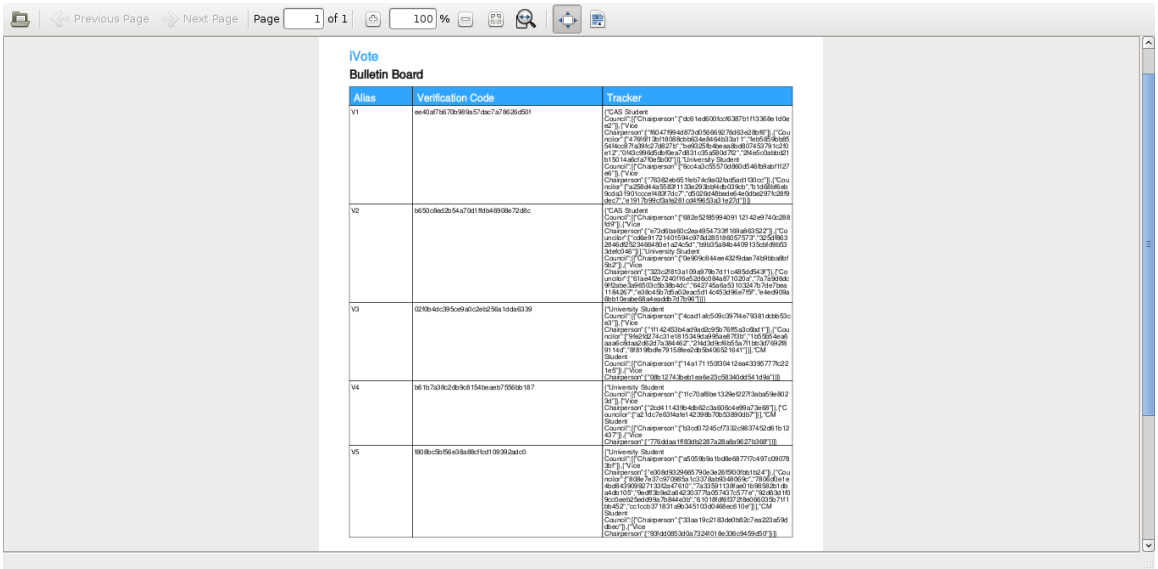

Figure 20: Export results as PDF

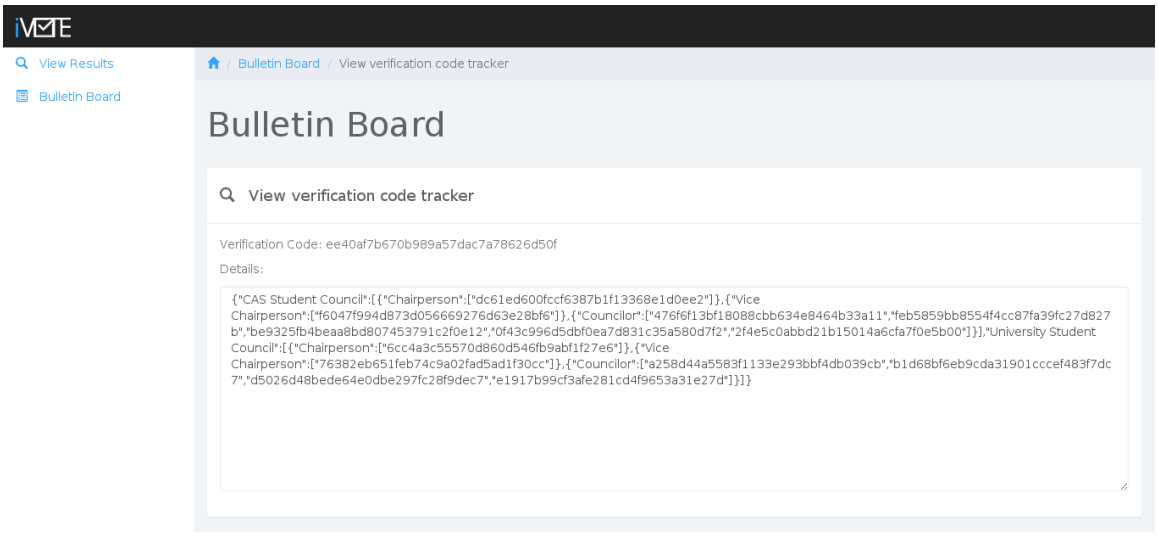

Figure 21: View verification code tracker

The following screenshots show what only registered users are allowed to do in the system.

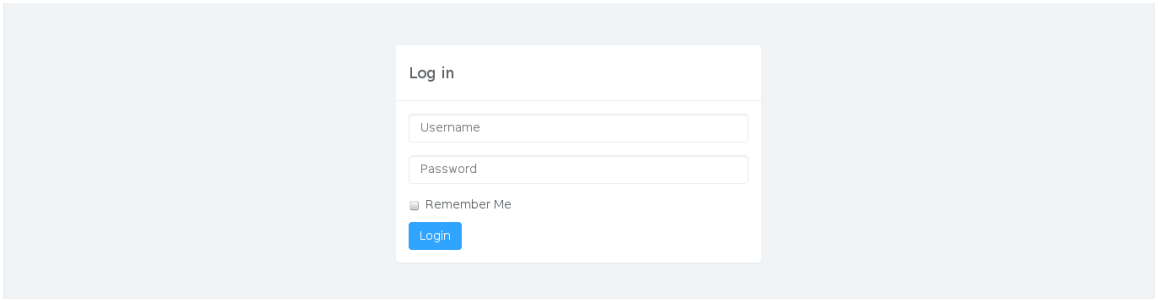

Figure 22: Log in to the system

Users can log in to the system by using the log in credentials sent via their emails. A user is prompted if a log in error has occurred. A successful login will redirect the user to the home page depending on his/her role.

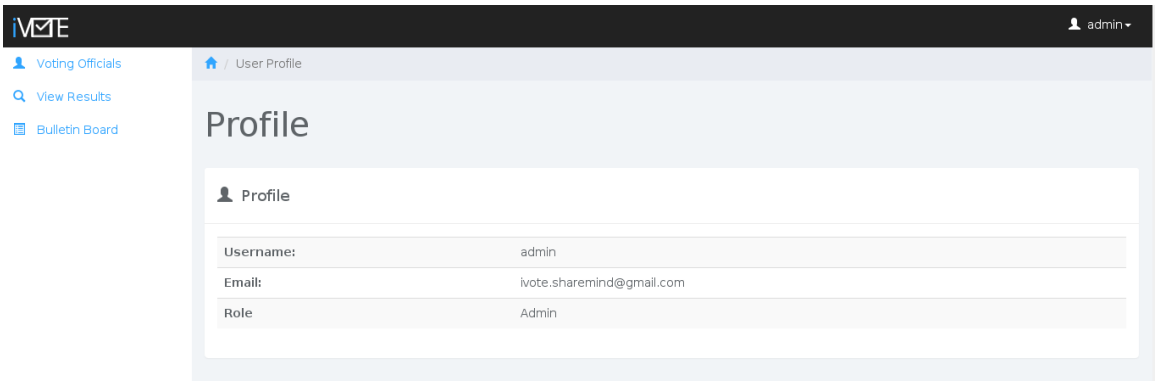

Figure 23: View account profile

Authenticated users can view their account profile by clicking on their usernames at the top right of the header then clicking Profile. They can also edit their account details by clicking on their usernames at the the top right of the header then clicking Settings.

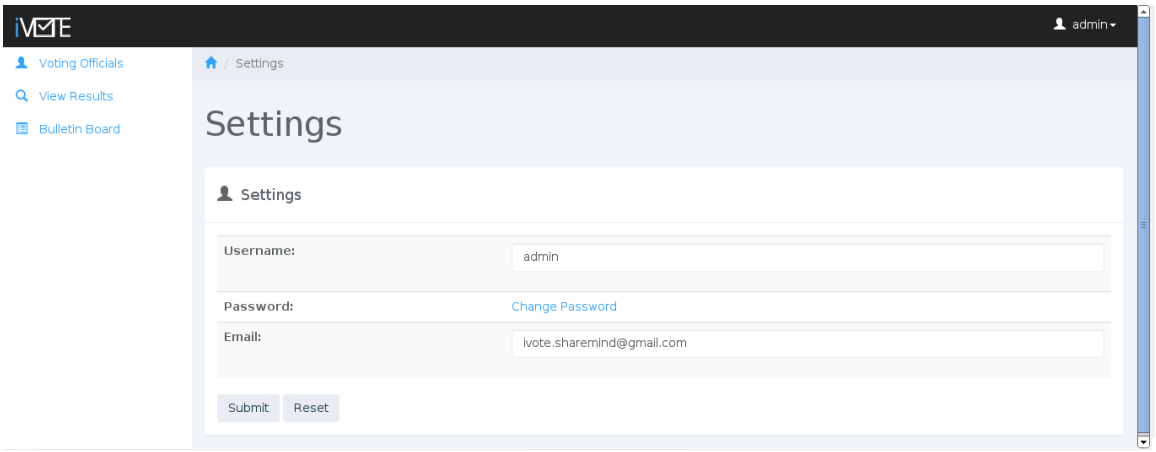

Figure 24: Edit account details

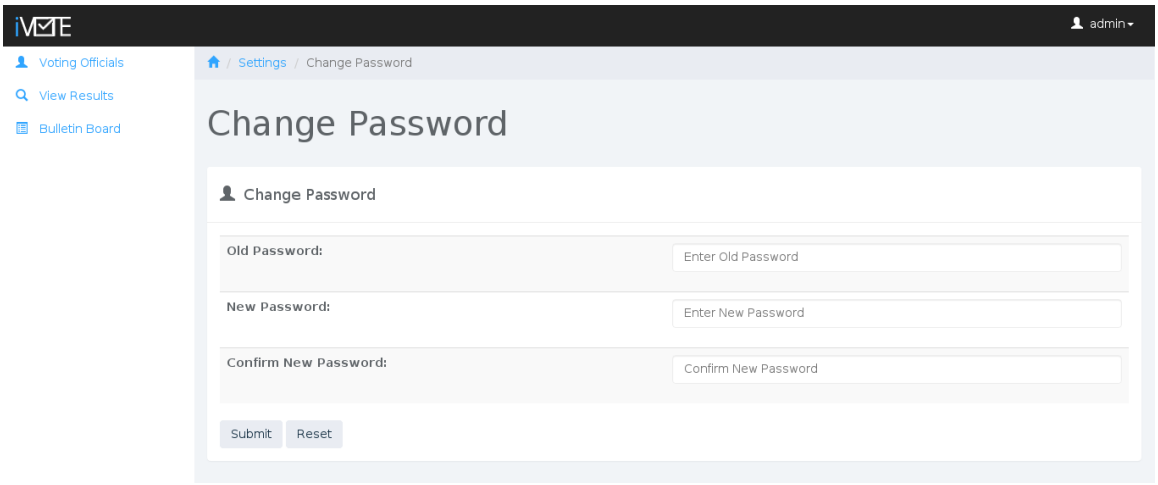

Figure 25: Change password

In the settings page, the user can also change his/her password. For a successful password change, the old password should be correct, and the new password should match the repeated new password. The user can only change his/her own password.

The following screenshots show what only voters are allowed to do in the system.

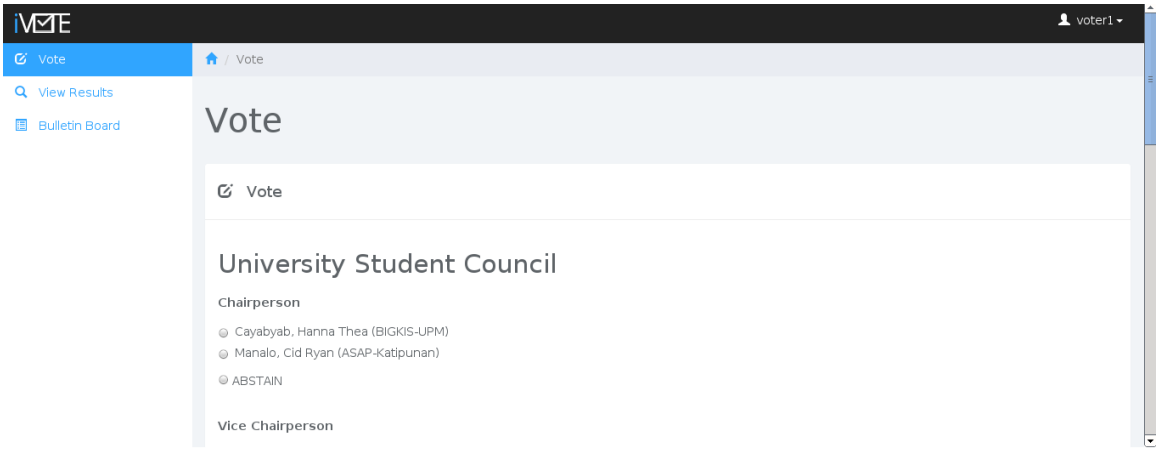

Figure 26: Cast a vote

The voter selects the candidate he/she wants to vote. The voter is prompted if he/she exceeded the number of votes required for a certain position.

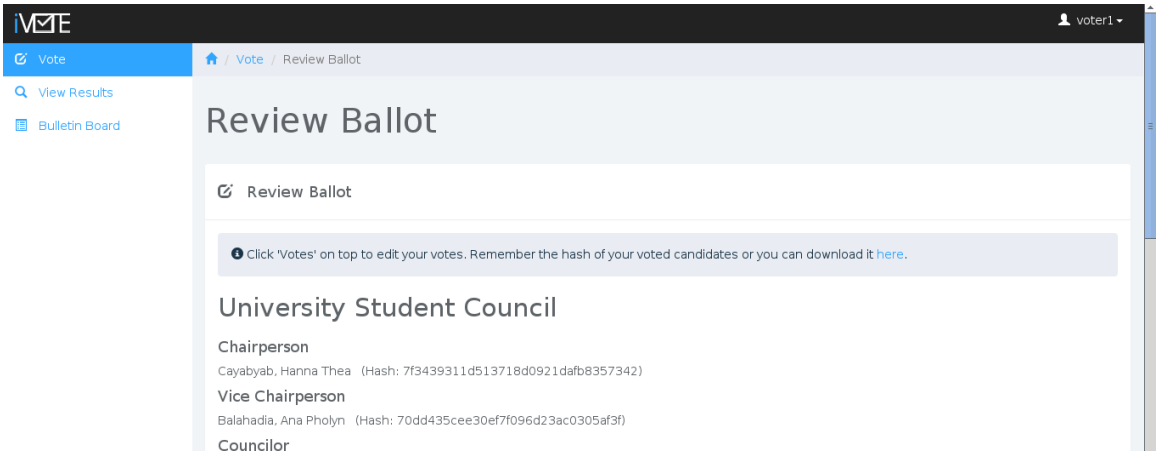

Figure 27: Review ballot

After the voter has cast his/her vote, a review ballot page is shown with all the voted candidates and their corresponding hashes. A voter can download a text file containing these hashes. The voter can also go back to the vote page by clicking the Vote link on top.

The following screenshots show what only voting officials are allowed to do in the system.

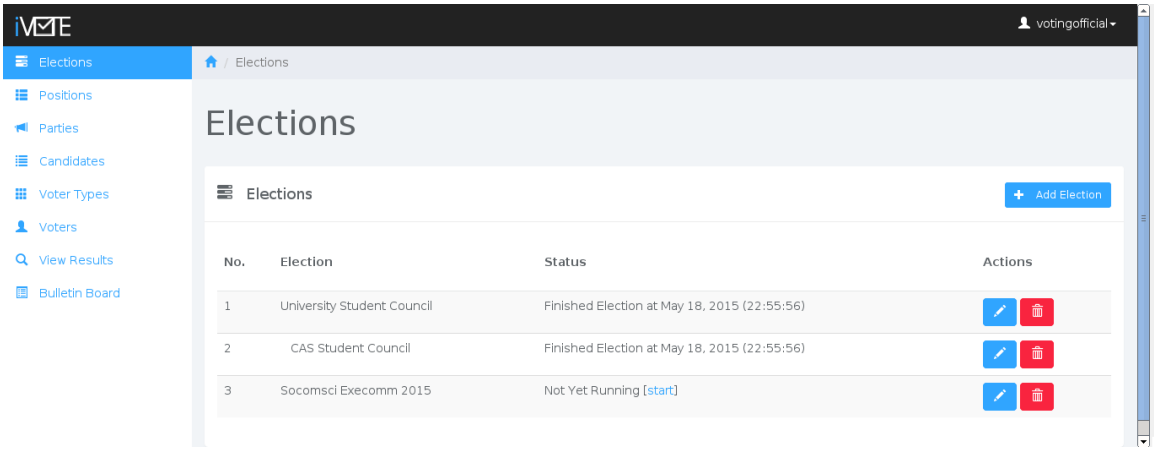

Figure 28: View elections

The names of the elections, their statuses, and actions are shown in this page. A start link will be displayed in elections that are not yet running, and a stop link will be displayed for running elections. Start and end dates are also shown. To add an

election, the voting official can click the Add Election button at the upper right of the panel. Edit and delete actions can be done in an election by clicking the pencil and trash icons, respectively.

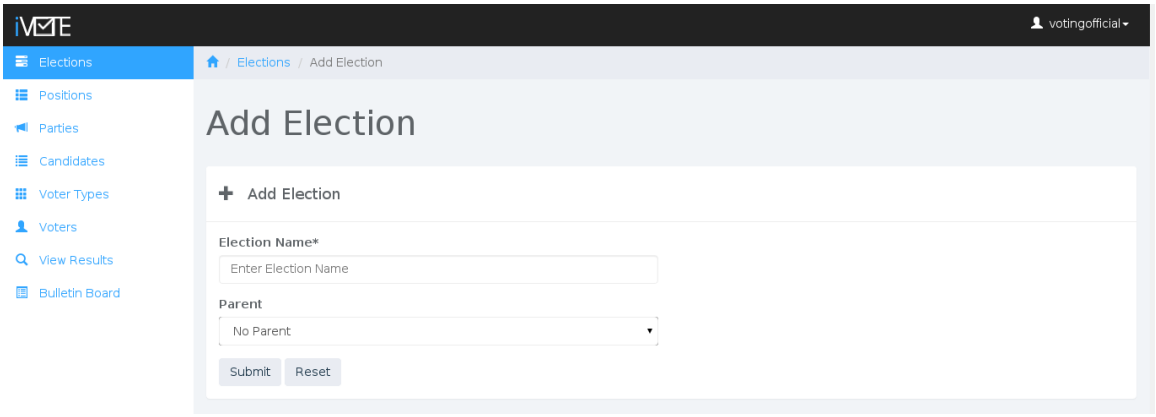

Figure 29: Add an election

Before adding positions, parties, candidates, and voter types, the voting official must add an election first. The voting official can add a child election by specifying its parent. Starting or stopping a parent election will also start or stop its child elections.

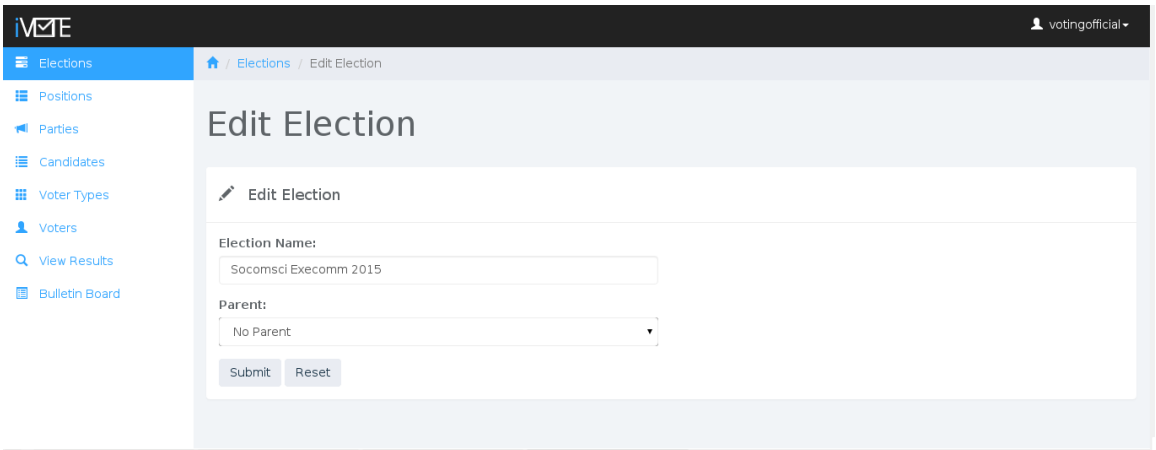

Figure 30: Edit an election

The voting official cannot edit a running or finished elections, or elections in use by positions, parties, candidates, and voter types.

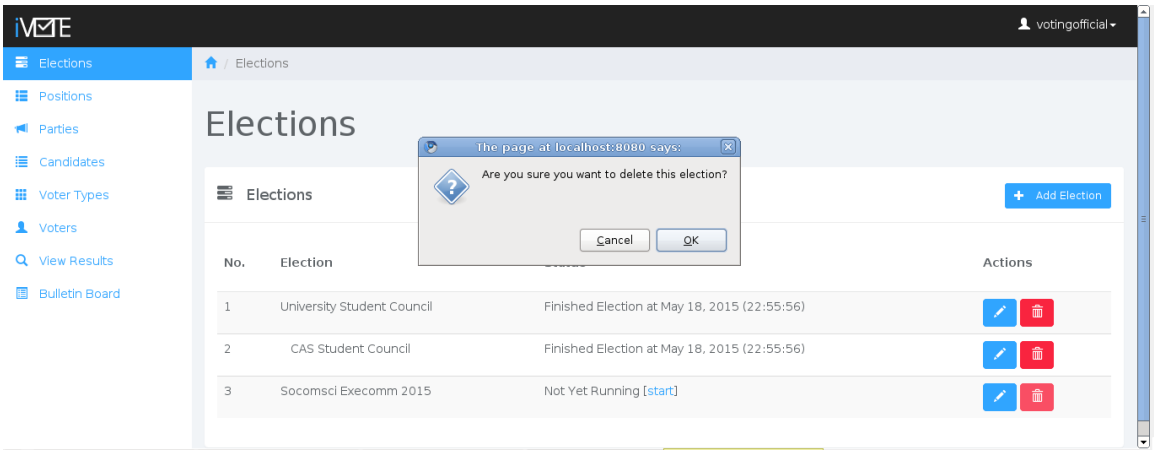

Figure 31: Delete an election

The voting official also cannot delete a running or finished elections, or elections in use by positions, parties, candidates, and voter types.

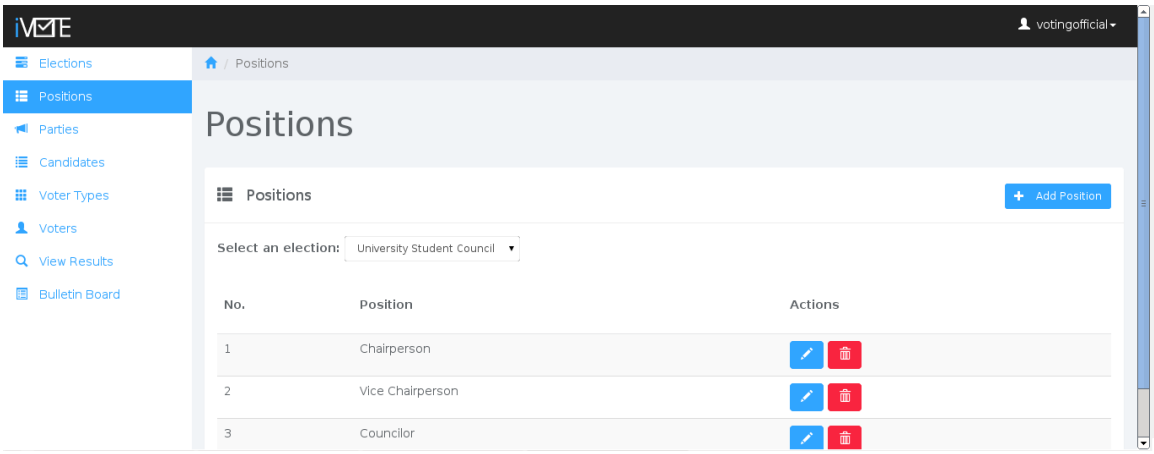

Figure 32: View positions by election

The names of the positions in the election selected and their actions are shown in this page. Edit and delete actions can be done in a position by clicking the pencil and trash icons, respectively.

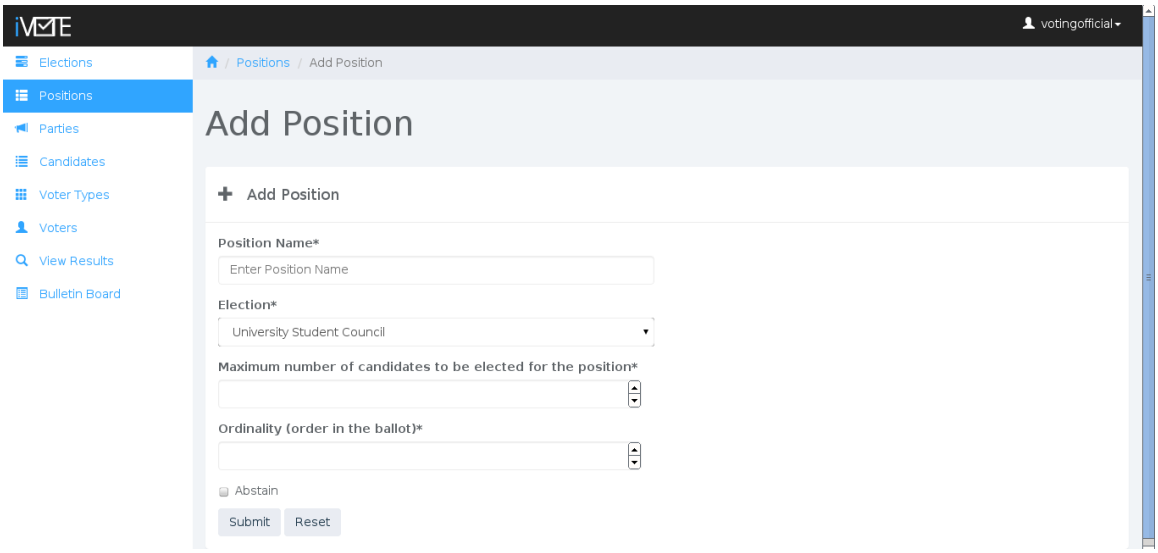

Figure 33: Add a position

The voting official must input the name of the position, the election it belongs, the maximum number of candidates elected in that position, and the order of the position in the ballot. In addition, he/she can specify if the position allows abstain votes.

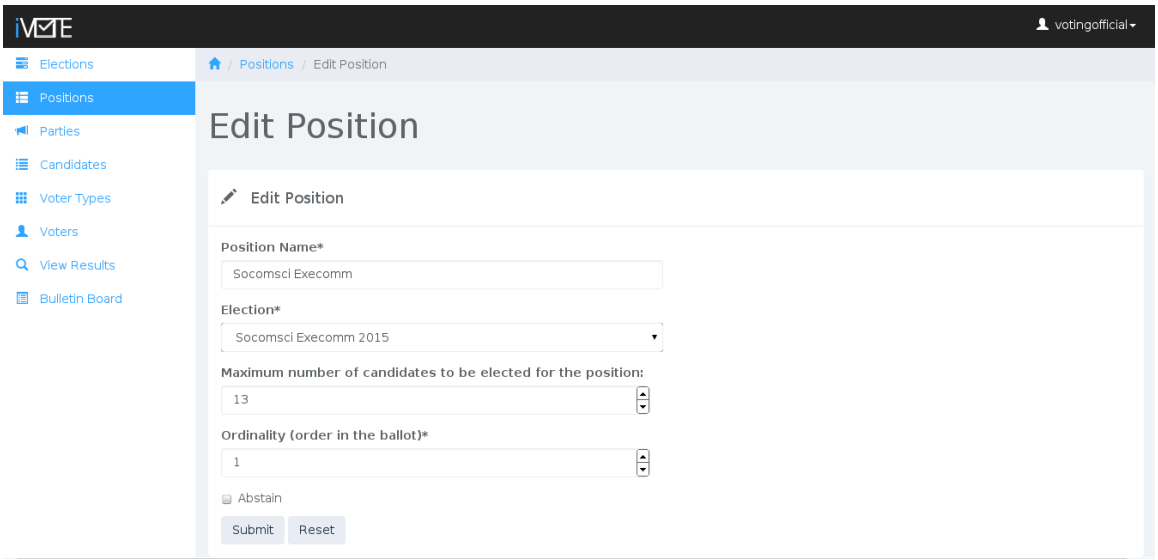

Figure 34: Edit a position

The voting official cannot edit a position in a running or finished elections, or positions in use by candidates.

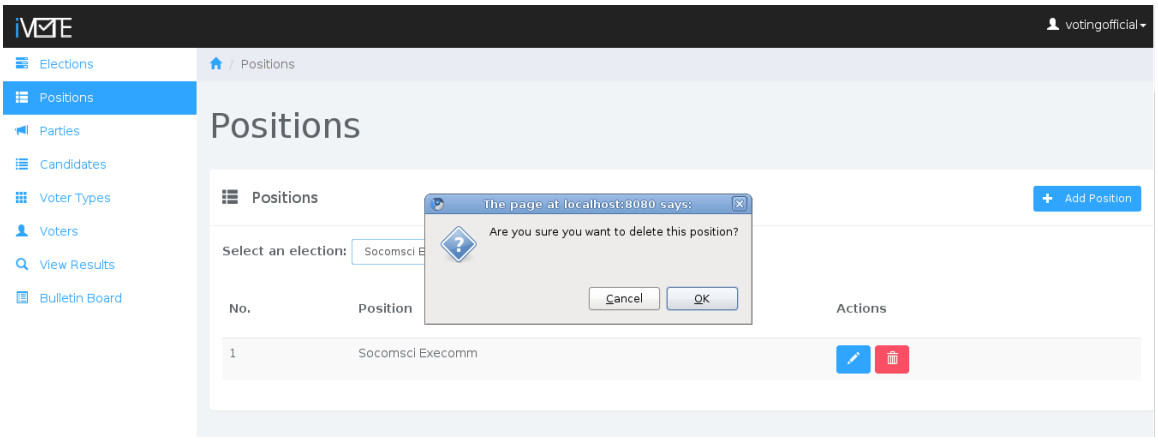

Figure 35: Delete a position

The voting official also cannot delete a position in a running or finished elections, or positions in use by candidates.

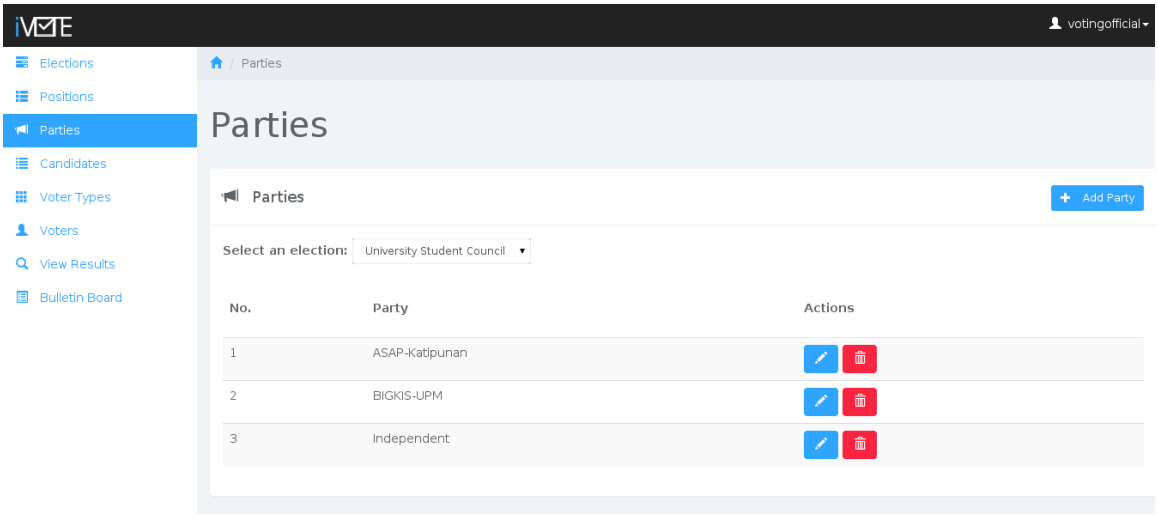

Figure 36: View parties by election

The names of the parties in the election selected and their actions are shown in this page. Edit and delete actions can be done in a party by clicking the pencil and trash icons, respectively.

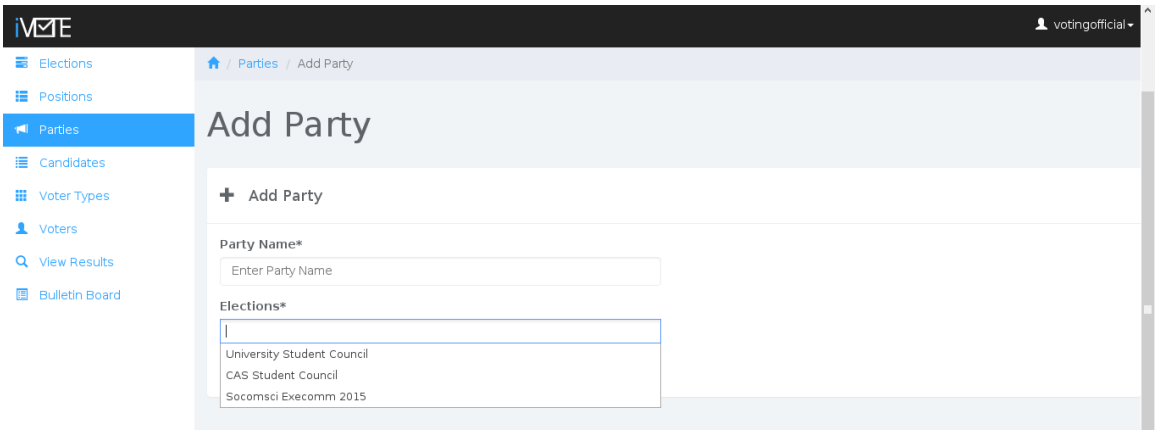

Figure 37: Add a party

The voting official must input the name of the party, and the elections it belongs. The voting official can select multiple elections by clicking on the text box of the elections.

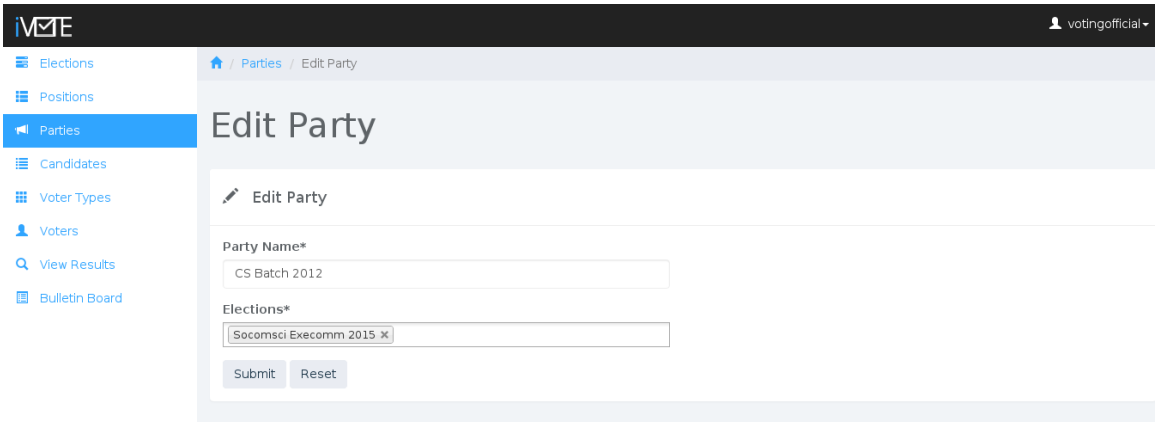

Figure 38: Edit a party

To remove an election from the list of the elections of the party, the x button is clicked. The voting official cannot edit a party in a running or finished elections, or parties in use by candidates.

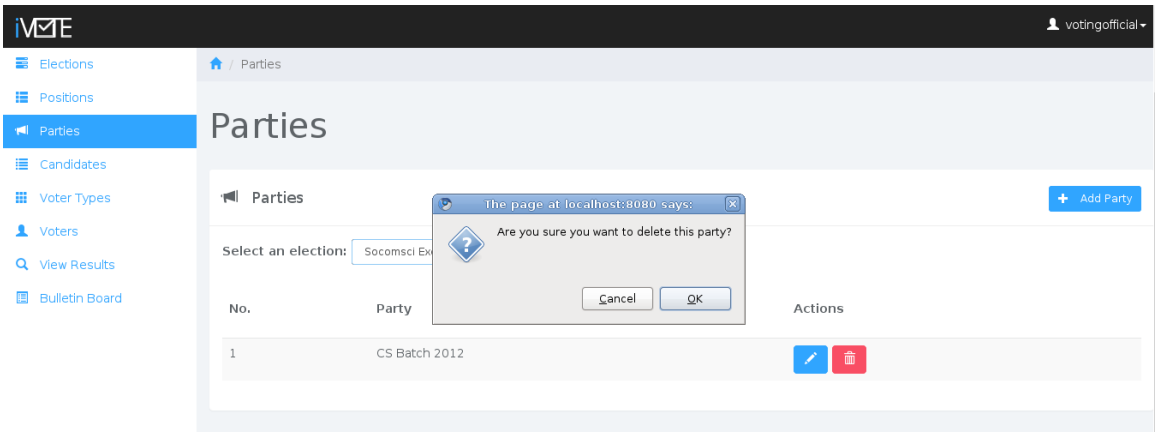

Figure 39: Delete a party

The voting official also cannot delete a party in a running or finished elections, or parties in use by candidates.

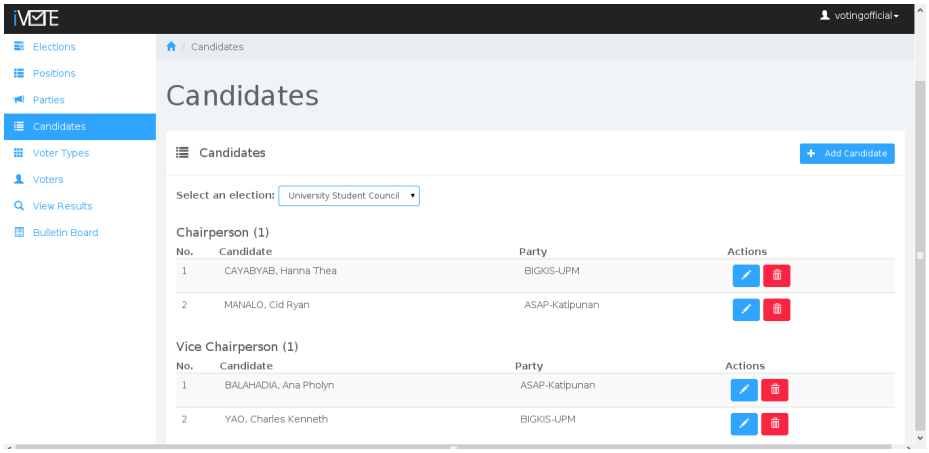

Figure 40: View candidates by election

The names of the candidates in the election selected, their parties, and their actions are shown in this page. The candidates are grouped with their corresponding positions. Edit and delete actions can be done in a candidate by clicking the pencil and trash icons, respectively.

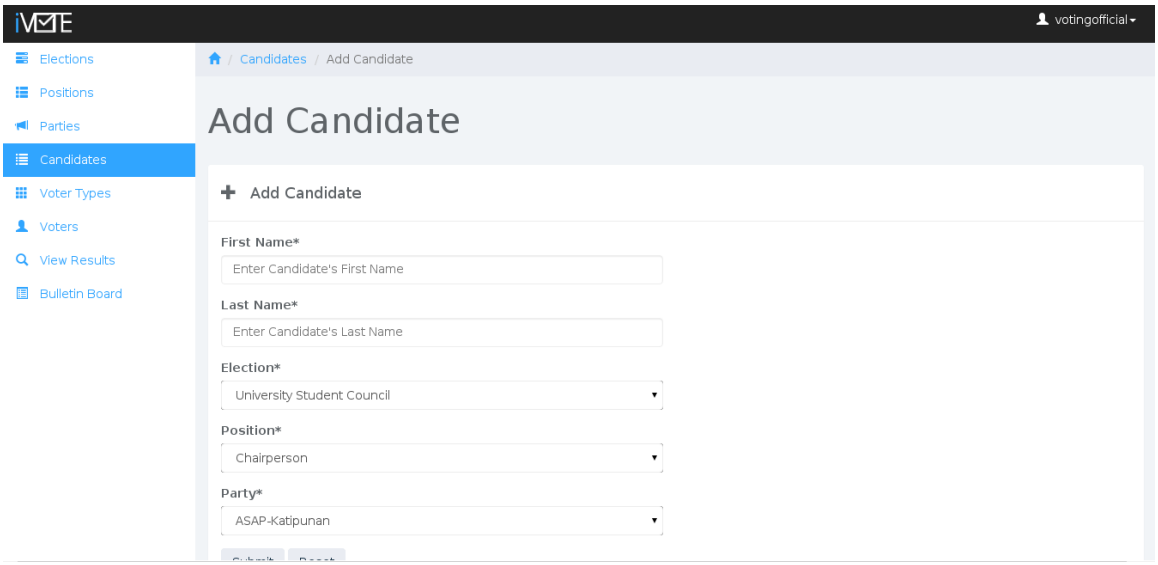

Figure 41: Add a candidate

Before adding a candidate, the voting official must add an election, position, and party. The voting official must input the first and last names of the candidate, and the election, position, and party it belongs.

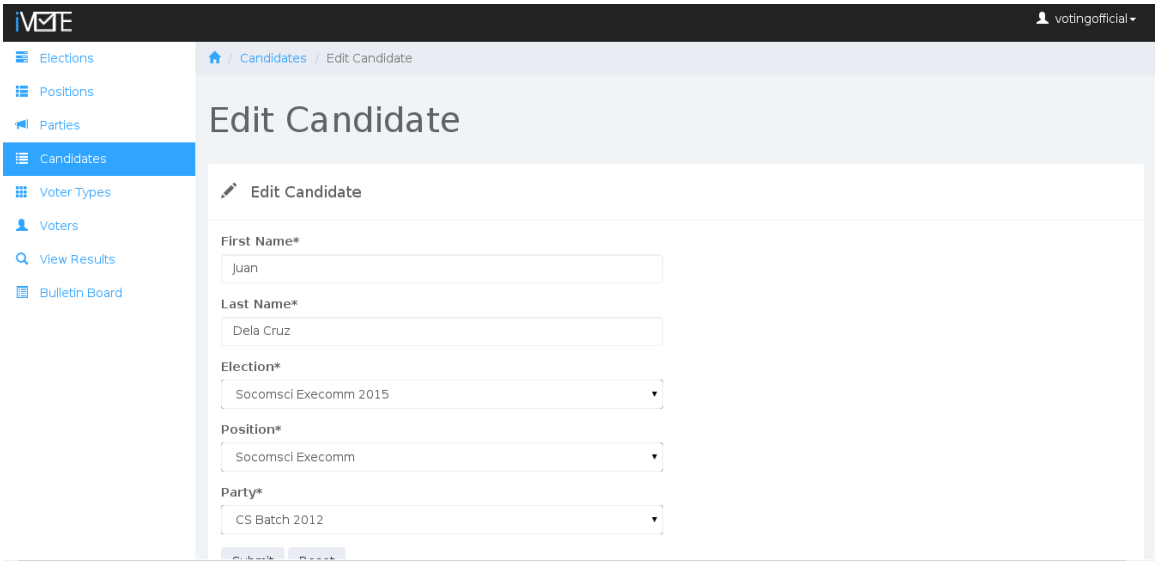

Figure 42: Edit a candidate

The voting official cannot edit a candidate in a running or finished elections.

Figure 43: Delete a candidate

The voting official also cannot delete a candidate in a running or finished elections.

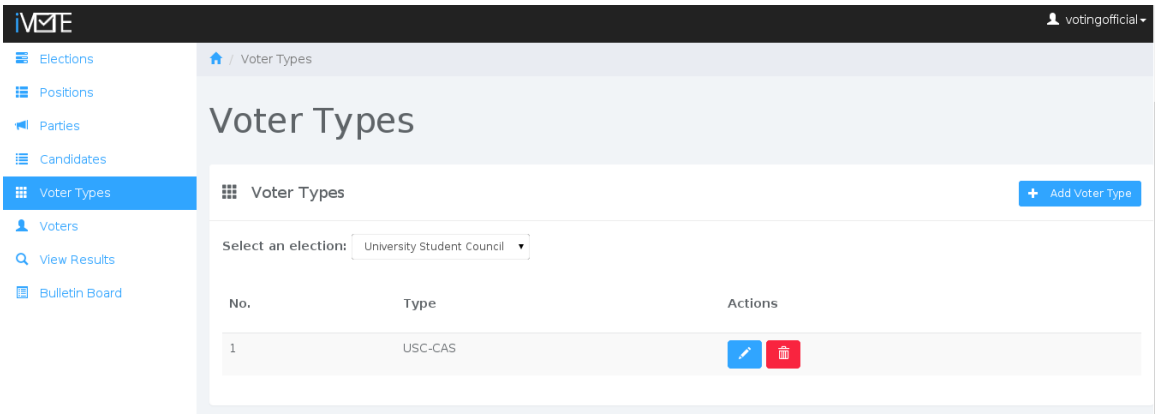

Figure 44: View voter types by election

The names of the voter types in the election selected, and their actions are shown in this page. Edit and delete actions can be done in a voter type by clicking the pencil and trash icons, respectively.

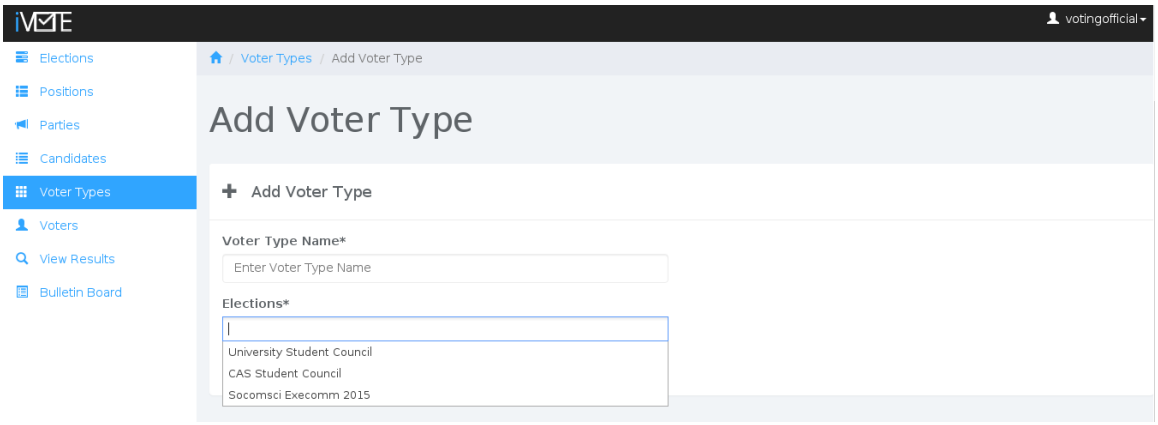

Figure 45: Add a voter type

The voting official must input the name of the voter type, and the elections it belongs. He/she can select multiple elections.

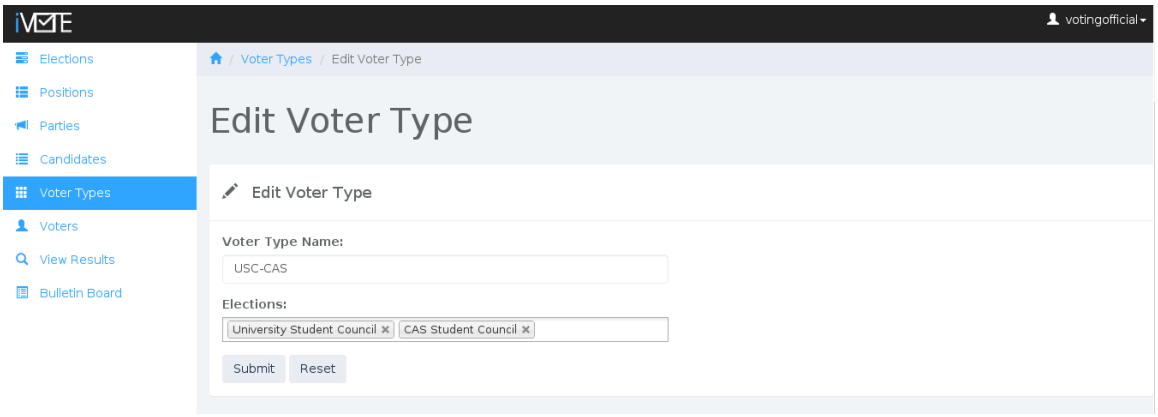

Figure 46: Edit a voter type

To remove an election from the list of the elections of the voter type, the x button is clicked. The voting official cannot edit a voter type in a running or finished elections.

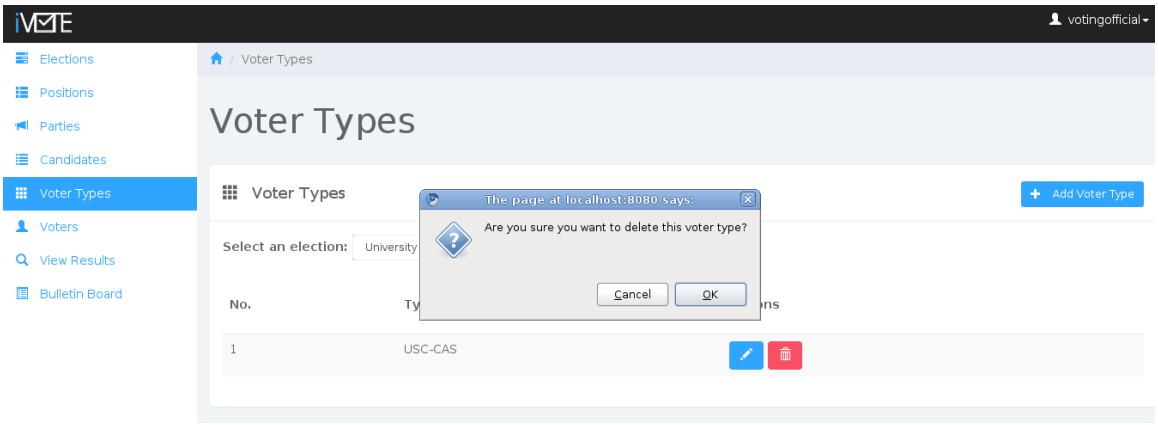

Figure 47: Delete a voter type

The voting official cannot delete a voter type in a running or finished elections.

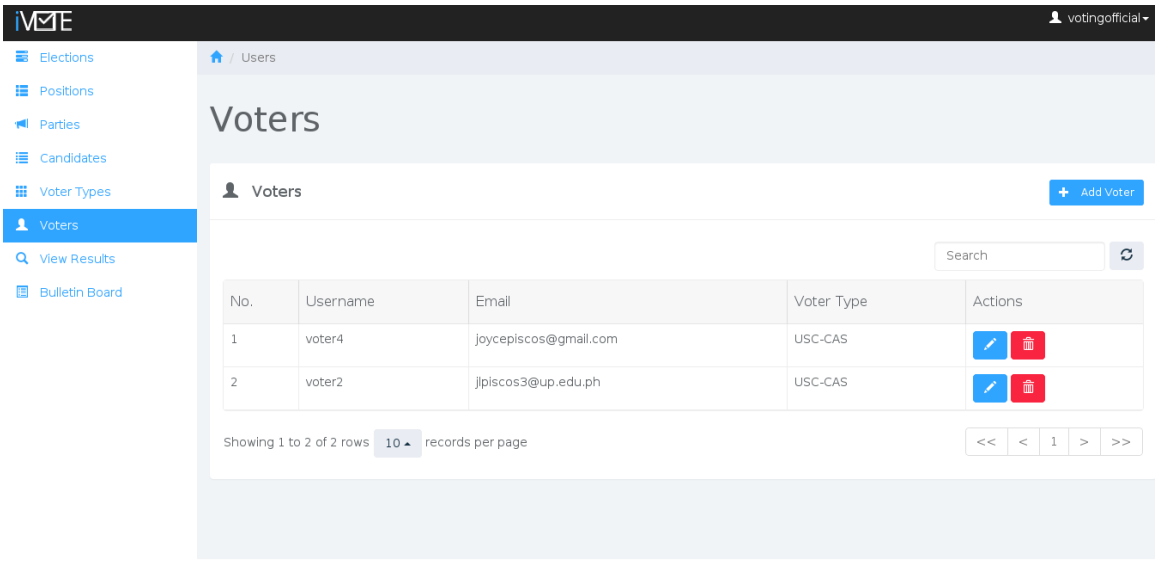

Figure 48: View voters

The usernames of the voters, their emails, their voter type, and the actions are shown in this page. The voting official can search any voter information by typing the search query in the text box above the table. Edit and delete actions can be done in a voter by clicking the pencil and trash icons, respectively.

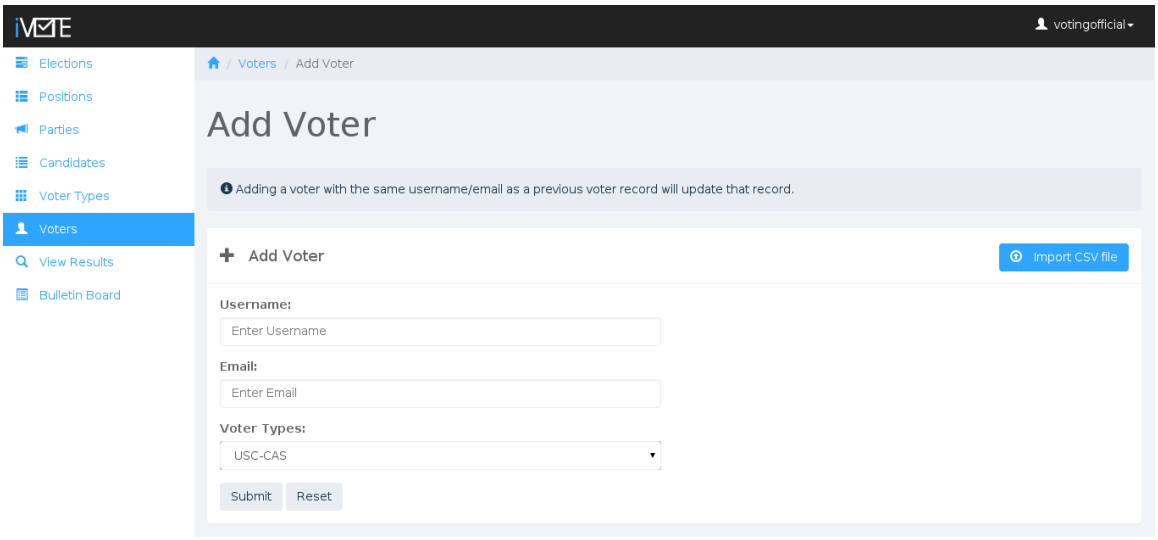

Figure 49: Add a voter

Before adding a voter, the voting official must add a voter type. He/she must input the username, email and voter type of the voter. A password will be automatically generated and will be sent via email together with the username of the voter. The voting official can also import voters via a CSV file by clicking the Import CSV file button.

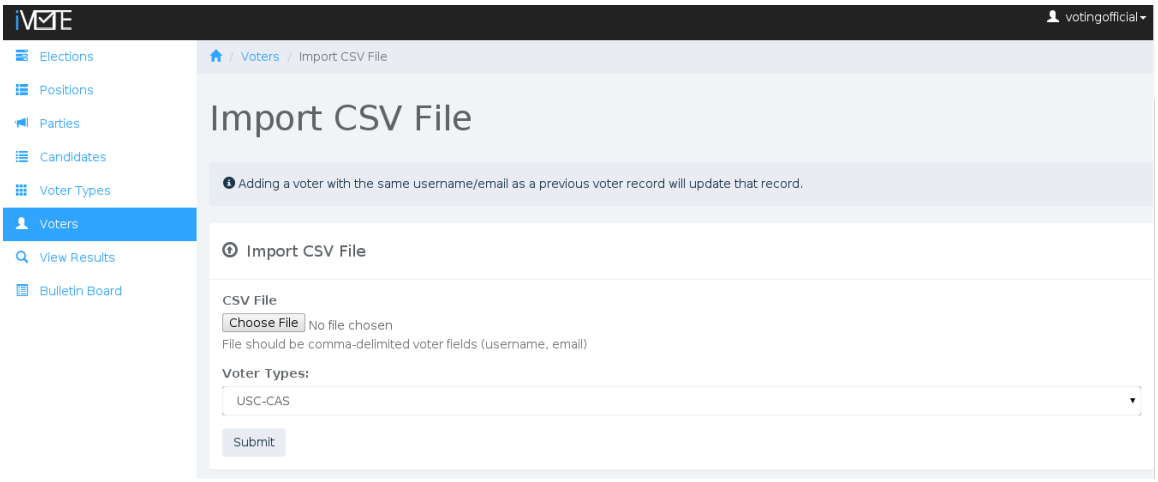

Figure 50: Import voters

Comma-delimited file containing the username and email must be uploaded with size not exceeding 128KB. The voter type of the voters must also be specified.

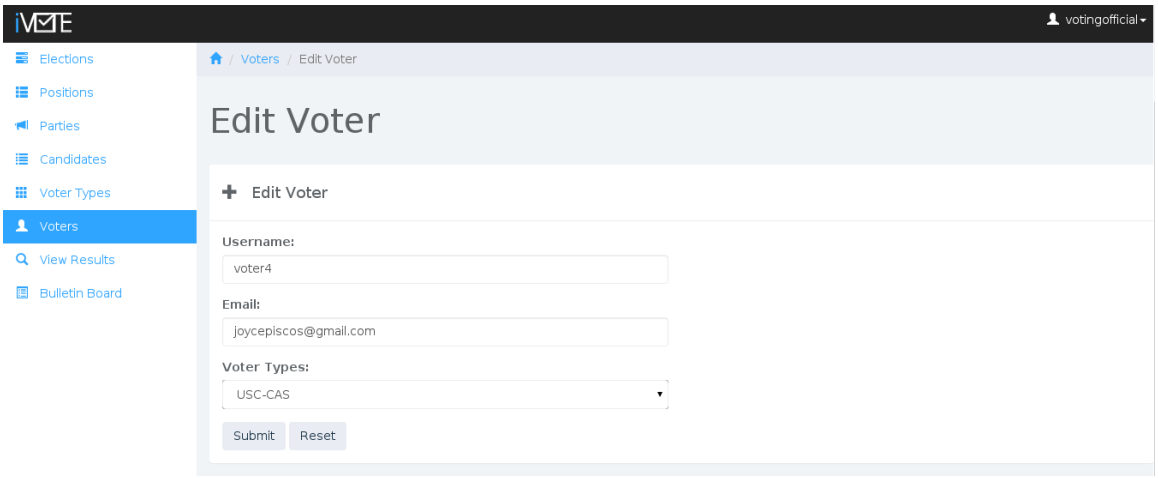

Figure 51: Edit a voter

The voting official cannot edit a voter in a running or finished elections, and if the voter has already voted.

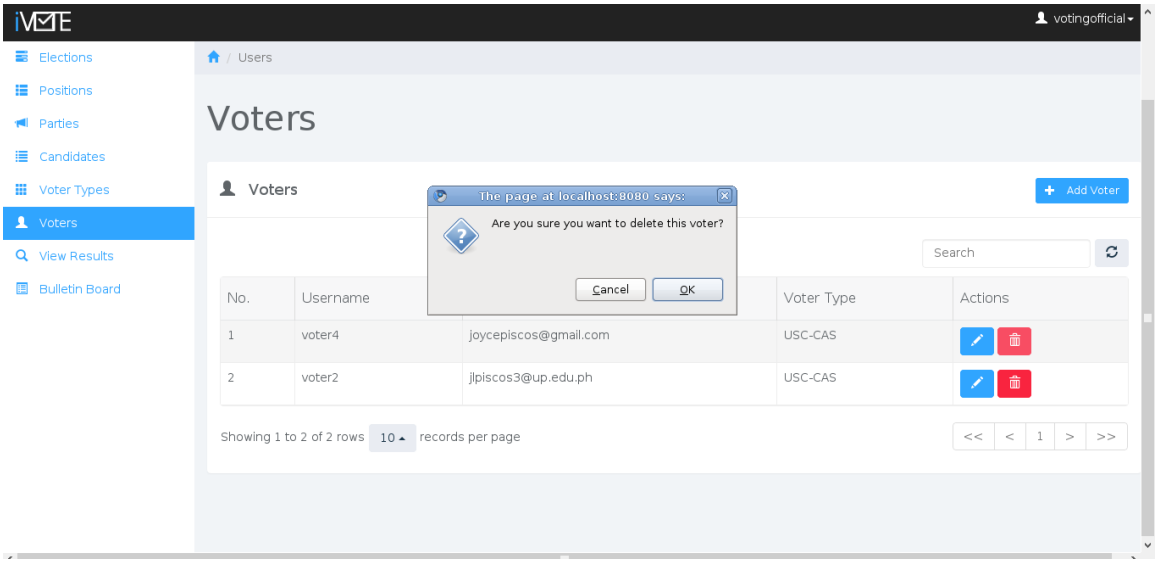

Figure 52: Delete a voter

The voting official also cannot delete a voter in a running or finished elections, and if the voter has already voted.

The following screenshots show what only administrators are allowed to do in the system.

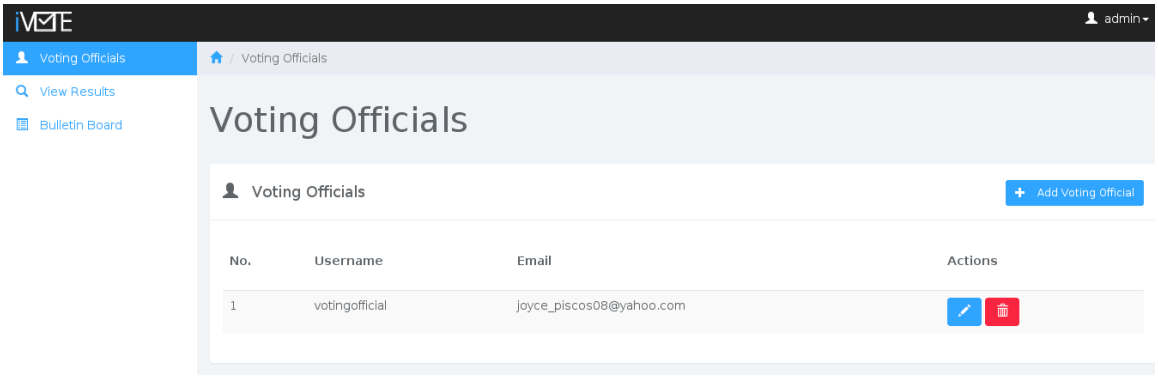

Figure 53: View voting officials

The usernames of the voting officials, their emails, and the actions are shown in this page. Edit and delete actions can be done in a voting official by clicking the pencil and trash icons, respectively.

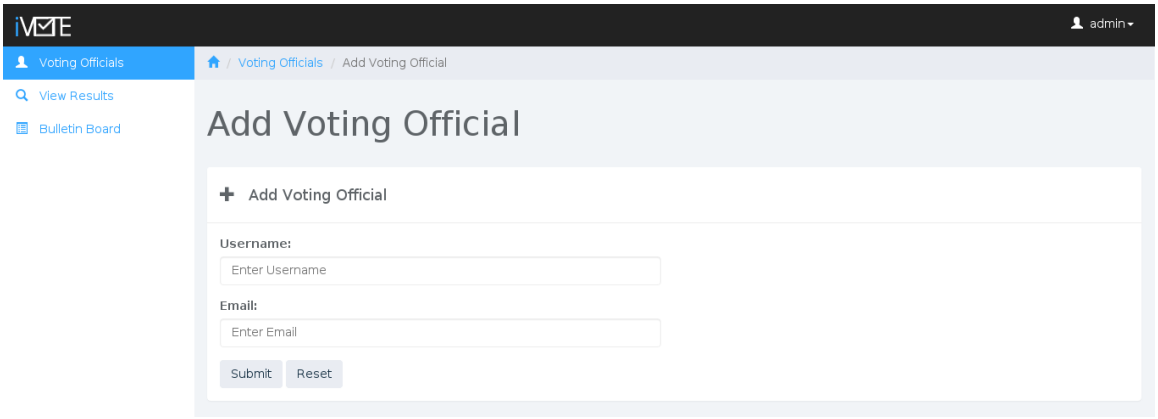

Figure 54: Add a voting official

Similar with adding a voter, the administrator must input the username and the email of the voting official. A password will be automatically generated and will be sent via email together with the username of the voting official.

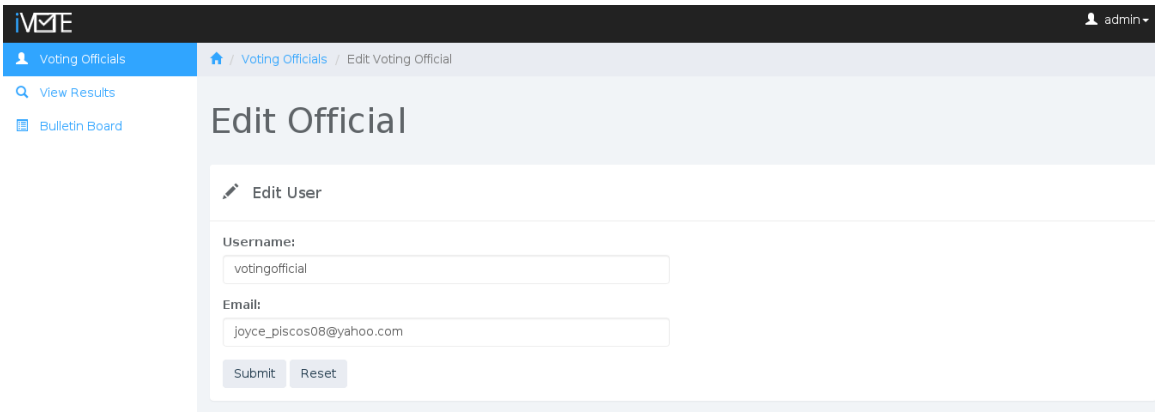

Figure 55: Edit a voting official

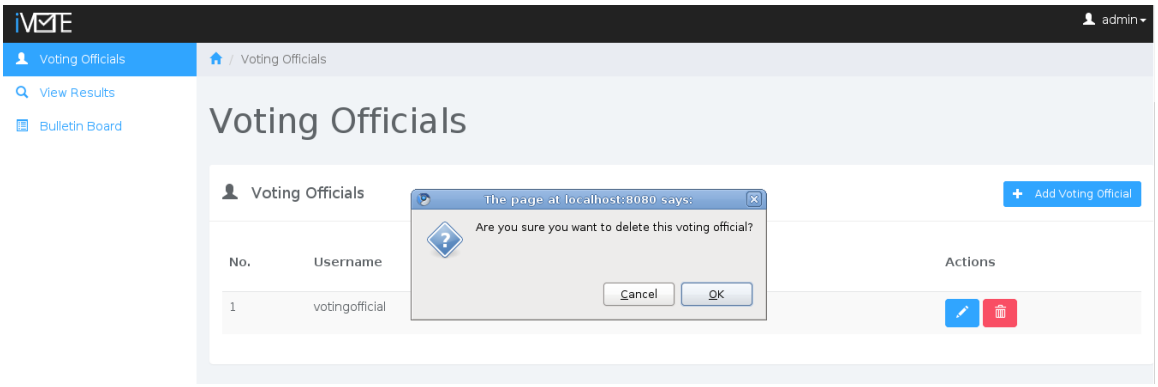

Figure 56: Delete a voting official

VI. Discussions

The Internet Voting System is a web-based application that provides protection of voter privacy and anonymity, as well as vote verifiability through Secure Multiparty Computation. Sharemind is used to handle SMC processes such as storing votes, computation of results, and retrieval of bulletin board entries. The system has three roles: voter, voting official, and administrator.

The system allows the voters to cast a vote securely. The voters can also verify their votes through viewing of the bulletin board, and view the results of the elections since these two functionalities are publicly available. The system also allows the voting officials to configure the election information. Voting officials can manage elections, positions, parties, candidates, voter types, and voters. Aside from the provided web interface, a voting official can also add voters by uploading a CSV file that contains the usernames and email addresses of the voters. On the other hand, the system provides the administrator the functionality to manage the voting official list. In addition, all users of the system has the privilege to view and update their account information.

By using the secret-shared database of Sharemind, the stored votes are considered secure. A user cannot directly view the database without running DevMiner and using a SecreC script. A user also cannot view the contents of the votes stored in each of the miners since these contents are encrypted and are not the complete values, but only a share. Sharemind uses additive secret sharing scheme for securing the inputs and is proven to be secure on a semi-honest corruption [16]. With this reason, this SMC-based voting system can provide protection of voter privacy and anonymity, two of the main properties of an Internet voting system.

Another property is the vote verifiability wherein a voter can check that his/her encrypted vote was counted and tabulated correctly. The system generates hashes for each of the candidates, wherein these hashes are unique per voter. The voter

can verify his/her votes by checking the bulletin board if the hashes of the candidate he/she voted is the same with what was posted.

For the development of the application, the Spring framework provided an ease of programming since there are numerous documentation for references. This is not the case for the Sharemind framework which only has a limited documentation and few developers using the framework. In addition, since the Sharemind database is secret-shared by design, its performance will be slow depending on the operation. For instance, a single comparison operation for 32-bit integers takes 1.25 seconds [39].

The main issue encountered is the slow running time when performing a secure multiparty computation in Sharemind. Listed below are the time periods for each SMC computation classified according to the number of voters with 17 sample candidates.

SMC Processes	Time Period (seconds)				
	10 Voters	20 Voters	50 Voters	70 Voters	100 Voters
Vote Submission	4	4	4	4	4
Viewing of Bulletin Board (During the elections)	25	52	141	214	330
Stopping of Elections (Computing Results)	10	10	10	10	10

Increasing the number of candidates will also increase the time for all the SMC processes. Among the processes, viewing the bulletin board entries took the longest time since Sharemind is slowly retrieving each votes of the voter. Sharemind maps every position and voter and returns the voted candidates. Retrieving all of these information caused the slow performance of viewing the bulletin board with increasing number of voters and candidates.

To optimize the loading time of the results of the elections, the computation of results is only done once. Computation of results is done when the voting official stops the election. These results are stored in the MySQL database for faster loading.

Similarly for the bulletin board entries, the process is done when there are new voters that are not yet published in the bulletin board. The entries are also stored in the MySQL database.

VII. Conclusions

The Internet Voting System is a web-based application that preserves voter anonymity and privacy through Secure Multiparty Computation. The system will not just speed up the voting process, but also ensure that the required security properties such as verifiability, anonymity, and privacy are met.

The system provides the functionality to cast a vote and view the results for voters, to manage the election configuration and the voters list for voting officials, and to manage the voting officials list for the administrator.

Sharemind provides the framework to perform the required SMC processes namely the storage of votes, computation of results, and retrieving votes for bulletin board entries. The system can guarantee to the voter that his/her vote is secured since it is stored in the Sharemind database. The correctness of the results are also guaranteed if the databases of the miners are not corrupted.

VIII. Recommendations

It is highly suggested that the performance of the SMC processes be further optimized. These processes are have slow loading times especially the viewing of the bulletin board when the election is running. The Sharemind 3 can be used to develop controllers and SecreC scripts for these processes. However, migrating from Sharemind 2 to Sharemind 3 is difficult due to the huge differences in the syntax for the SecreC scripts and controllers.

```
// Sharemind 2
private uint32[1] data;
data[0] = value;

dbInsertRow('table_name'
   , data);
```

```
// Sharemind 3
uint32 colSize =
   getColumnSize('
   column_name');
if (colSize == 0)
  putColumn('column_name',
   val);
else
  appendColumn('column_name'
   , val);
```

Table 11: Example code for inserting a row in a database table with one column in Sharemind 2 and 3

In addition, the system only focused on properties of an Internet voting system namely voter privacy, anonymity, and vote verifiability. Other properties can also be achieved like uncoercibility. The system only provides low-coercion elections wherein a voter can overwrite his/her votes.

IX. Bibliography

[1] C. Uscatu, "Electronic universal voting," *Informatica Economica*, vol. 4, no. 4, pp. 130–135, 2008.

[2] O. Çetinkaya, "Analysis of security requirements for cryptographic voting protocols (extended abstract)," in *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pp. 1451–1456, March 2008.

[3] R. Cramer, I. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing - An Information Theoretic Approach*. 2013.

[4] D. Mishra, R. Pathak, S. Joshi, and A. Ludhiyani, "Secure multi-party computation for statistical computations using virtual parties on a token ring network," in *Wireless And Optical Communications Networks (WOCN), 2010 Seventh International Conference On*, pp. 1–6, Sept 2010.

[5] R. Cramer and I. Damgård, "Multiparty computation, an introduction," in *Contemporary Cryptology*, Advanced Courses in Mathematics - CRM Barcelona, pp. 41–87, Birkhäuser Basel, 2005.

[6] H. Vegge, "Realizing secure multiparty computations," Graduate Thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2009.

[7] S. Goldwasser, "Multi party computations: Past and present," in *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '97, (New York, NY, USA), pp. 1–6, ACM, 1997.

[8] A. B. David, A. Jarrous, and T. Reinman, "Secure multi-party protocols - tools, implementations, and applications." http://www.pinkas.net/mpc.html. Accessed: 2014-10-18.

[9] A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, pp. 160–164, Nov 1982.

[10] K. B. Frikken, "Algorithms and theory of computation handbook," ch. Secure Multiparty Computation, pp. 14–14, Chapman & Hall/CRC, 2010.

[11] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology - Crypto 2012*, vol. 7417 of *Lecture Notes in Computer Science*, pp. 643–662, Springer, 2012.

[12] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft, "Financial cryptography and data security," ch. Secure Multiparty Computation Goes Live, pp. 325–343, Berlin, Heidelberg: Springer-Verlag, 2009.

[13] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay – a secure two-party computation system," in *In USENIX Security Symposium*, pp. 287–302, 2004.

[14] A. Ben-David, N. Nisan, and B. Pinkas, "Fairplaymp: A system for secure multiparty computation," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, (New York, NY, USA), pp. 257–266, ACM, 2008.

[15] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "Sepia: Privacy-preserving aggregation of multi-domain network events and statistics," in *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, (Berkeley, CA, USA), pp. 15–15, USENIX Association, 2010.

[16] D. Bogdanov, *Sharemind: programmable secure computations with practical applications*. PhD thesis, University of Tartu, Tartu, Estonia, 2013.

[17] A. Jarrous and B. Pinkas, "Canon-mpc, a system for casual non-interactive secure multi-party computation using native client," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, (New York, NY, USA), pp. 155–166, ACM, 2013.

[18] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen, "Asynchronous multiparty computation: Theory and implementation," in *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC '09*, Irvine, (Berlin, Heidelberg), pp. 160–179, Springer-Verlag, 2009.

[19] M. Burkhart, D. Many, and M. Widmer, *Sepia User Manual and Developers Guide*.

[20] M. Djatmiko, D. Schatzmann, X. Dimitropoulos, A. Friedman, and R. Boreli, "Collaborative network outage troubleshooting with secure multiparty computation," *Communications Magazine, IEEE*, vol. 51, pp. 78–84, November 2013.

[21] M. Geisler, *Cryptographic Protocols: Theory and Implementation*. PhD thesis, Aarhus University, Denmark, 2010.

[22] R. Talviste, "Web-based data entry in privacy-preserving applications," Undergraduate Thesis, University of Tartu, Tartu, Estonia, 2009.

[23] D. Bogdanov, R. Jagomägis, and S. Laur, "A universal toolkit for cryptographically secure privacy-preserving data mining," in *Proceedings of the 2012 Pacific Asia Conference on Intelligence and Security Informatics*, PAISI'12, (Berlin, Heidelberg), pp. 112–126, Springer-Verlag, 2012.

[24] R. Talviste, *Deploying secure multiparty computation for joint data analysisa case study*. PhD thesis, University of Tartu, Tartu, Estonia, 2011.

[25] I. Damgård and M. Keller, "Secure multiparty aes," in *Proceedings of the 14th International Conference on Financial Cryptography and Data Security*, FC'10, (Berlin, Heidelberg), pp. 367–374, Springer-Verlag, 2010.

[26] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J. A. Halderman, "Security analysis of the estonian internet voting system," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 703–715, ACM, 2014.

[27] M. J. M. Chowdhury, "Comparison of e-voting schemes: Estonian and norwegian solutions," *International Journal of Applied Information Systems*, vol. 6, pp. 60–66, September 2013. Published by Foundation of Computer Science, New York, USA.

[28] B. Adida, "Helios: Web-based open-audit voting," in *Proceedings of the 17th Conference on Security Symposium*, SS'08, (Berkeley, CA, USA), pp. 335–348, USENIX Association, 2008.

[29] V. Mishra, P. Trivedi, and D. K. Mishra, "Secure internet voting protocol using secure multiparty computational methods," *Journal of Technology and Engineering Sciences*, vol. 1, 2009.

[30] C. Gang, "An electronic voting scheme based on secure multi-party computation," in *Computer Science and Computational Technology, 2008. ISCSCT '08. International Symposium on*, vol. 1, pp. 292–294, Dec 2008.

[31] M. Stenbro, "A survey of modern electronic voting technologies," Graduate Thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2010.

[32] K. P. Kaliyamurthie, R. Udayakumar, D. Parameswari, and S. N. Mugunthan, "Highly secured online voting system over network," *Indian Journal of Science and Technology*, vol. 6, 2013.

[33] J. Puiggalí, J. Chóliz, and S. Guasch, "Best practices in internet voting,"

[34] M. Based and S. Mjølsnes, "A secure internet voting scheme," in *Algorithms and Architectures for Parallel Processing* (Y. Xiang, A. Cuzzocrea, M. Hobbs, and W. Zhou, eds.), vol. 7017 of *Lecture Notes in Computer Science*, pp. 141–152, Springer Berlin Heidelberg, 2011.

[35] Y. Zhao, Y. Pan, S. Wang, and J. Zhang, "An anonymous voting system based on homomorphic encryption," in *Communications (COMM), 2014 10th International Conference on*, pp. 1–4, May 2014.

[36] S. F. MJØLSNES and M. B. STENBEK, "Multiparty ballot counting in an internet voting scheme," *Acta Electrotechnica et Informatica*, vol. 10, no. 4, pp. 10–15, 2010.

[37] P. Snyder, "Yao's garbled circuits : Recent directions and implementations," 2014.

[38] T. I. Reistad, *A General Framework for Multiparty Computations*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2012.

[39] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ESORICS '08, (Berlin, Heidelberg), pp. 192–206, Springer-Verlag, 2008.

[40] R. Jagomögis, "Secrec: a privacy-aware programming language with applications in data mining," Graduate Thesis, University of Tartu, Tartu, Estonia, 2010.

X. Appendix

A. Source Code

A..1 Sharemind

1. SecreC Scripts

1.1. CreateVoteDatabase.sc

```
void main(public string db, public string tbl, public string
      column1, public string column2, public string column3) {
    public bool exists;
    exists = dbExists(db);
    if (!exists) {
        dbCreate(db);
    }

    dbLoad(db);

    exists = dbTableExists(tbl);
    if (exists) {
        dbTableDelete(tbl);
    }

    dbTableCreatePrivate(tbl);

    private uint32[0] empty;
    dbInsertColumn(tbl, column1, empty);
    dbInsertColumn(tbl, column2, empty);
    dbInsertColumn(tbl, column3, empty);
}
```

1.2. VoteDatabaseEntry.sc

```
void main(public string db, public string tbl, private uint32
      value1, private uint32 value2, private uint32 value3) {
    dbLoad(db);

    private uint32[3] data;
    data[0] = value1;
    data[1] = value2;
    data[2] = value3;

    string tableName = "votes";
    public uint32 rows; rows = dbRowCount(tableName);
    dbInsertRow(tbl, "r" + rows, data);
}
```

1.3. UpdateVoteEntry.sc

```
void main(private uint32 positionId, private uint32 voterId){
    string dbName = "VotesDB";
    string tableName = "votes";
    string positionColumn = "position_id";
    string voterColumn = "voter_id";

    dbLoad(dbName);
    public uint32 rows; rows = dbRowCount(tableName);

    private uint32[rows] positionData;
    positionData = dbReadColumnPrivateUint32(tableName,
          positionColumn);
    private uint32[rows] voterData;
    voterData = dbReadColumnPrivateUint32(tableName,
          voterColumn);

    private uint32[rows] copies;
    private uint32[rows] copies2;
    private bool[rows] eq;
    private uint32[rows] eq2;
    public uint32[rows] eq3;

    copies = positionId;
    copies2 = voterId;
    eq = (voterData == copies2) * (positionData == copies);
    eq2 = boolToInt(eq);
    eq3 = declassify(eq2);

    public uint32 i;
    public uint32 j;
    public uint32 length; length = vecLength(eq3);
    public uint32 m; m = 0;
    public uint32 n;
    for(i = 0; i < length; i = i+1){

        if(eq3[i] == 1){
            if(m==0){
                n = i;
                print("deleted= r" + n);
                dbDeleteRow(tableName, "r"+ n);
            } else {
                print("deleted= r" + n);
                dbDeleteRow(tableName, "r"+ n);
            }
            for(j = i+1; j<length; j=j+1){
                private uint32[3] rowAfter;
                print("r"+(j-m));
                rowAfter = dbReadRowPrivateUint32(tableName, "
                      r"+(j-m));
                public uint32 k; k=(j-m)-1;
                print("r" + k);
                dbDeleteRow(tableName, "r" + (j-m));
                dbInsertRow(tableName, "r" + k, rowAfter);
            }
            m = m+1;
        }
    }
}
```

1.4. count.sc

```
void main(public uint32 positionId, public uint32 candidateId)
     {
    string dbName = "VotesDB";
    string tableName = "votes";
    string positionColumn = "position_id";
    string candidateColumn = "candidate_id";

    dbLoad(dbName);
    public uint32 rows; rows = dbRowCount(tableName);

    private uint32[rows] positionData;
    positionData = dbReadColumnPrivateUint32(tableName,
          positionColumn);
    private uint32[rows] candidateData;
    candidateData = dbReadColumnPrivateUint32(tableName,
          candidateColumn);
    private uint32[rows] copies;
    private uint32[rows] copies2;
    private bool[rows] eq;
    private uint32 presult;
    public uint32 result;

    copies = positionId;
    copies2 = candidateId;
    eq = (positionData == copies) * (candidateData == copies2)
        ;
    presult = vecSum(eq);
    result = declassify(presult);
    publish("count", result);
}
```

1.5. check-voter.sc

```
void main(public uint32 positionId, public uint32 voterId) {
    string dbName = "VotesDB";
    string tableName = "votes";
    string positionColumn = "position_id";
    string voterColumn = "voter_id";

    dbLoad(dbName);
    public uint32 rows; rows = dbRowCount(tableName);

    private uint32[rows] positionData;
    positionData = dbReadColumnPrivateUint32(tableName,
          positionColumn);
    private uint32[rows] voterData;
    voterData = dbReadColumnPrivateUint32(tableName,
          voterColumn);
    private uint32[rows] copies;
    private uint32[rows] copies2;
    private bool[rows] eq;
    public bool[rows] eq2;
    private uint32 presult;
    public uint32 result;

    copies = positionId;
    copies2 = voterId;
    eq = (positionData == copies) * (voterData == copies2);
    eq2 = declassify(eq);
    public uint32 length;
    length = vecLength(eq);
    private uint32 count;
    count = vecSum(eq);
    public uint32 countP;
```

```
    countP = declassify(count);
    public uint32 i;
    public uint32 j = 0;
    public uint32[countP] candidates;
    for(i = 0; i < length; i = i+1){
        if(eq2[i]){
            private uint32 val; public uint32 pubVal;
            val = dbReadValuePrivateUint32(tableName, i, "
                  candidate_id");
            pubVal = declassify(val);
            candidates[j] = pubVal;
            j = j+1;
        }
    }
    publish("candidates", candidates);
}
```

2. Sharemind Controllers

2.1. CreateVoteDatabase.cpp

```
/*
 * Controller for creating the votes database
 *
 * Adapted from the sample codes of
 * Roman Jagomgis and Karl-Aksel Puulmann
 *
 */
#include "controller/ControllerLibrary.h"
#include <string>
#include <iostream>
#include <sstream>
using namespace std;

const string scriptName = "CreateVoteDatabase.sa";

ControllerInterface* StartConnection() {
    ControllerInterface *controller = new ControllerInterface(
          "CreateVoteDatabase.log");
    controller->loadConfiguration ("controller.cfg");
    if (!controller->connect ("testuser")) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Failed to
              connect to the miners!")
        delete controller;
        controller = NULL;
    }
    return controller;
}

void EndConnection(ControllerInterface *controller) {
    controller->shutdown();
    delete (controller);
}

bool CreateDatabase (const string& db, const string& tbl,
      const string& column1, const string& column2, const
      string& column3,
    ControllerInterface* const controller) {
    WRITE_LOG_DEBUG (controller->getConsole(), "Running script
          '" << scriptName << "' on the miners.")

    vector<RegisterDescriptor> parameters;
    RegisterDescriptor db_param(PUBLIC_STRING, "db");
    db_param.setValueAsString(db);
    parameters.push_back(db_param);
    RegisterDescriptor tbl_param(PUBLIC_STRING, "tbl");
    tbl_param.setValueAsString(tbl);
    parameters.push_back(tbl_param);
    RegisterDescriptor column1_param(PUBLIC_STRING, "column1")
          ;
    column1_param.setValueAsString(column1);

    parameters.push_back(column1_param);
    RegisterDescriptor column2_param(PUBLIC_STRING, "column2")
          ;
    column2_param.setValueAsString(column2);
    parameters.push_back(column2_param);
    RegisterDescriptor column3_param(PUBLIC_STRING, "column3")
          ;
    column3_param.setValueAsString(column3);
    parameters.push_back(column3_param);

    vector<RegisterDescriptor> results;
    if (!controller->runScript (scriptName, parameters,
          results)) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Error:
              Failed to run the script!")
        return false;
    }

    return true;
}

int main(int argc, char* argv[])
{
    string dbName = "VotesDB";
    string tableName = "votes";
    string columnName1 = "position_id";
    string columnName2 = "candidate_id";
    string columnName3 = "voter_id";

    ControllerInterface *controller = StartConnection();
    if (controller == NULL) {
        return EXIT_FAILURE;
    }

    if (!CreateDatabase(dbName, tableName, columnName1,
          columnName2, columnName3, controller)) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Error
              creating table named " << tableName << ". -
              FAILED!")
        EndConnection(controller);
        return EXIT_FAILURE;
    }

    WRITE_LOG_NORMAL (controller->getConsole(), "Successfully
          created table `" << tableName << "'.")

    EndConnection(controller);
    return EXIT_SUCCESS;
}
```

2.2. VoteDatabaseEntry.cpp

```
/*
 * Controller for inserting a row in the votes database
 *
 * Adapted from the sample codes of
 * Roman Jagomgis and Karl-Aksel Puulmann
 *
 */
#include "controller/ControllerLibrary.h"
#include <string>
#include <iostream>
#include <sstream>
using namespace std;

const string scriptName = "VoteDatabaseEntry.sa";

ControllerInterface* StartConnection() {
    ControllerInterface *controller = new ControllerInterface(
          "VoteDatabaseEntry.log");
    controller->loadConfiguration ("/home/sharemind/DevTools/
          bin/controller.cfg");
    if (!controller->connect ("testuser")) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Failed to
              connect to the miners!")
        delete controller;
        controller = NULL;
    }
    return controller;
}

void EndConnection(ControllerInterface *controller) {
    controller->shutdown();
    delete (controller);
}

bool SaveValue (const string& db, const string& tbl, const
      val_t value1, const val_t value2, const val_t value3,
    ControllerInterface* const controller) {
    WRITE_LOG_DEBUG (controller->getConsole(), "Running script
          '" << scriptName << "' on the miners.")

    vector<RegisterDescriptor> parameters;
    RegisterDescriptor db_param(PUBLIC_STRING, "db");
    db_param.setValueAsString(db);
    parameters.push_back(db_param);

    RegisterDescriptor tbl_param(PUBLIC_STRING, "tbl");
    tbl_param.setValueAsString(tbl);
    parameters.push_back(tbl_param);
    RegisterDescriptor value1_param(PRIVATE_INT, "value1");
    value1_param.setValueAsInteger(value1);
    parameters.push_back(value1_param);
    RegisterDescriptor value2_param(PRIVATE_INT, "value2");
    value2_param.setValueAsInteger(value2);
    parameters.push_back(value2_param);
    RegisterDescriptor value3_param(PRIVATE_INT, "value3");
    value3_param.setValueAsInteger(value3);
    parameters.push_back(value3_param);

    vector<RegisterDescriptor> results;
    if (!controller->runScript (scriptName, parameters,
          results)) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Error:
              Failed to run the script!")
        return false;
    }

    return true;
}

int main(int argc, char* argv[])
{
    string inputStr;
    val_t value1;
    val_t value2;
    val_t value3;
    val_vector_t allValues;

    string dbName = "VotesDB";
    string tableName = "votes";

    ControllerInterface *controller = StartConnection();
    if (controller == NULL) {
        return EXIT_FAILURE;
    }

    stringstream(argv[1]) >> value1;
    stringstream(argv[2]) >> value2;
    stringstream(argv[3]) >> value3;
```

```
    allValues.push_back(value1);
    allValues.push_back(value2);
    allValues.push_back(value3);

    if (!SaveValue(dbName, tableName, value1, value2, value3,
          controller)) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Error
              inserting value in the table `" << tableName <<
              "'.")
```

2.3. UpdateVoteEntry.cpp

```
/*
 * Controller for updating votes of a voter
 *
 * Adapted from the sample codes of
 * Roman Jagomgis and Karl-Aksel Puulmann
 *
 */
#include "controller/ControllerLibrary.h"
#include <string>
#include <iostream>
#include <sstream>
using namespace std;

const string scriptName = "UpdateVoteEntry.sa";

ControllerInterface* StartConnection() {
    ControllerInterface *controller = new ControllerInterface(
          "UpdateVoteEntry.log");
    controller->loadConfiguration ("/home/sharemind/DevTools/
          bin/controller.cfg");
    if (!controller->connect ("testuser")) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Failed to
              connect to the miners!")
        delete controller;
        controller = NULL;
    }
    return controller;
}

void EndConnection(ControllerInterface *controller) {
    controller->shutdown();
    delete (controller);
}

bool SaveValue (const val_t value1, const val_t value2,
      ControllerInterface* const controller) {
    WRITE_LOG_DEBUG (controller->getConsole(), "Running script
          '" << scriptName << "' on the miners.")

    vector<RegisterDescriptor> parameters;

    RegisterDescriptor value1_param(PRIVATE_INT, "positionId")
          ;
    value1_param.setValueAsInteger(value1);
    parameters.push_back(value1_param);

    RegisterDescriptor value2_param(PRIVATE_INT, "voterId");
    value2_param.setValueAsInteger(value2);
    parameters.push_back(value2_param);

    vector<RegisterDescriptor> results;
    if (!controller->runScript (scriptName, parameters,
          results)) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Error:
              Failed to run the script!")
        return false;
    }

    return true;
}

int main(int argc, char* argv[])
{
    string inputStr;
    val_t value1;
    val_t value2;
    val_vector_t allValues;

    ControllerInterface *controller = StartConnection();
    if (controller == NULL) {
        return EXIT_FAILURE;
    }

    stringstream(argv[1]) >> value1;
    stringstream(argv[2]) >> value2;

    allValues.push_back(value1);
    allValues.push_back(value2);

    if (!SaveValue(value1, value2, controller)) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Error
              inserting value in the table.")
        EndConnection(controller);
        return EXIT_FAILURE;
    }

    WRITE_LOG_NORMAL (controller->getConsole(), "Thank you!")

    EndConnection(controller);
    return EXIT_SUCCESS;
}
```

2.4. ElectionResults.cpp

```
/*
 * Controller for counting the occurrences of a candidate
 *
 * Adapted from the sample codes of
 * Roman Jagomgis and Karl-Aksel Puulmann
 *
 */
#include "controller/ControllerLibrary.h"
#include <string>
#include <iostream>
#include <sstream>
using namespace std;

string scriptName = "count.sa";
string returnValueName = "count";

ControllerInterface* StartConnection() {
    ControllerInterface *controller = new ControllerInterface(
          "ElectionResults.log");
    controller->loadConfiguration ("controller.cfg");
    if (!controller->connect ("testuser")) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Failed to
              connect to the miners!")
        delete controller;
        controller = NULL;
    }
    return controller;
}

void EndConnection(ControllerInterface *controller) {
    controller->shutdown();
    delete (controller);
}

bool CountValue(val_t &positionId, val_t &candidateId, uint32
      &count, ControllerInterface *controller) {
    WRITE_LOG_DEBUG (controller->getConsole(), "Running script
          '" << scriptName << "' on the miners.")
    vector<RegisterDescriptor> parameters;

    RegisterDescriptor pos_param(PUBLIC_INT, "positionId");
    pos_param.setValueAsInteger(positionId);
    parameters.push_back(pos_param);

    RegisterDescriptor can_param(PUBLIC_INT, "candidateId");
    can_param.setValueAsInteger(candidateId);
    parameters.push_back(can_param);

    vector<RegisterDescriptor> results;
    if (!controller->runScript (scriptName, parameters,
          results)) {
        WRITE_LOG_NORMAL (controller->getConsole(), "Error:
              Failed to run the script!")
        return false;
    }

    if (results.size() != 1 || results[0].getName() !=
          returnValueName) {
        WRITE_LOG_NORMAL(controller->getConsole(), "Error:
              Script returned wrong result!");
        return false;
    }

    if (!results[0].getValueAsInteger(count)) {
        WRITE_LOG_NORMAL(controller->getConsole(), "Error:
              Failed to read result returned by script!");
        return false;
    }

    return true;
}

int main(int argc, char* argv[])
{
    val_t positionId;
    val_t candidateId;
    uint32 count;

    ControllerInterface *controller = StartConnection();
    if (controller == NULL) {
        return EXIT_FAILURE;
    }

    bool uploadResult = controller->uploadScript(scriptName,
          scriptName);
    if (!uploadResult) {
        WRITE_LOG_DEBUG(controller->getConsole(), "Error:
              Failed to upload script!")
        EndConnection(controller);
        return EXIT_FAILURE;
    }

    stringstream(argv[1]) >> positionId;
    stringstream(argv[2]) >> candidateId;

    if (!CountValue(positionId, candidateId, count, controller
```

```
            )){
            WRITE_LOG_NORMAL(controller->getConsole(), "FAILED")
            EndConnection(controller);
            return EXIT_FAILURE;
        } else {
            WRITE_LOG_NORMAL(controller->getConsole(), "Count=" <<
                    count)
        }

        EndConnection(controller);
        return EXIT_SUCCESS;
    }
```

2.5. CheckVoter.cpp

```
    /*
     * Controller for finding the voted candidates of a voter
     *
     * Adapted from the sample codes of
     * Roman Jagomgis and Karl-Aksel Puulmann
     *
     */
    #include "controller/ControllerLibrary.h"
    #include <string>
    #include <iostream>
    #include <sstream>
    using namespace std;

    string scriptName = "check-voter.sa";
    string returnValueName = "candidates";

    ControllerInterface* StartConnection() {
        ControllerInterface *controller = new ControllerInterface(
              "CheckVoter.log");
        controller->loadConfiguration ("controller.cfg");
        if (!controller->connect ("testuser")) {
            WRITE_LOG_NORMAL (controller->getConsole(), "Failed to
                  connect to the miners!")
            delete controller;
            controller = NULL;
        }
        return controller;
    }

    void EndConnection(ControllerInterface *controller) {
        controller->shutdown();
        delete (controller);
    }

    bool CountValue(val_t &positionId, val_t &voterId,
          val_vector_t &candidates, ControllerInterface *
          controller) {
        WRITE_LOG_DEBUG (controller->getConsole(), "Running script
              '" << scriptName << "' on the miners.")
        vector<RegisterDescriptor> parameters;

        RegisterDescriptor pos_param(PUBLIC_INT, "positionId");
        pos_param.setValueAsInteger(positionId);
        parameters.push_back(pos_param);

        RegisterDescriptor can_param(PUBLIC_INT, "voterId");
        can_param.setValueAsInteger(voterId);
        parameters.push_back(can_param);

        vector<RegisterDescriptor> results;
        if (!controller->runScript (scriptName, parameters,
              results)) {
            WRITE_LOG_NORMAL (controller->getConsole(), "Error:
                  Failed to run the script!")
            return false;
        }

        if (results.size() != 1 || results[0].getName() !=
              returnValueName) {
            WRITE_LOG_NORMAL(controller->getConsole(), "Error:
                  Script returned wrong result!");
            return false;
        }

        if (!results[0].getValueAsIntegerVector(candidates)) {
            WRITE_LOG_NORMAL(controller->getConsole(), "Error:
                  Failed to read result returned by script!");
            return false;
        }

        return true;
    }

    int main(int argc, char* argv[])
    {
        val_t positionId;
        val_t voterId;
        val_vector_t candidates;

        ControllerInterface *controller = StartConnection();
        if (controller == NULL) {
            return EXIT_FAILURE;
        }

        bool uploadResult = controller->uploadScript(scriptName,
              scriptName);
        if (!uploadResult) {
            WRITE_LOG_DEBUG(controller->getConsole(), "Error:
                  Failed to upload script!")
            EndConnection(controller);
            return EXIT_FAILURE;
        }

        stringstream(argv[1]) >> positionId;
        stringstream(argv[2]) >> voterId;

        if (!CountValue(positionId, voterId, candidates,
              controller)){
            WRITE_LOG_NORMAL(controller->getConsole(), "FAILED")
            EndConnection(controller);
            return EXIT_FAILURE;
        } else {
            uint32 i;
            for(i = 0; i < candidates.size(); i++){
                WRITE_LOG_NORMAL(controller->getConsole(), "
                      Candidates=" << candidates[i])
            }
        }

        EndConnection(controller);
        return EXIT_SUCCESS;
    }
```

A..2 Web Application Components

1. IvoteApplication.java

```
package ph.edu.upm.agila.japiscos;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.
      SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder
      ;
import org.springframework.boot.context.web.
      SpringBootServletInitializer;

@SpringBootApplication
public class IvoteApplication extends
      SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(
          SpringApplicationBuilder application) {
        return application.sources(IvoteApplication.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(IvoteApplication.class, args);
    }

}
```

2. Application Configuration

2.1. AsyncConfig.java

```
    package ph.edu.upm.agila.japiscos.config;

    import java.util.concurrent.Executor;

    import org.springframework.aop.interceptor.
          AsyncUncaughtExceptionHandler;
    import org.springframework.context.annotation.Configuration;
    import org.springframework.scheduling.annotation.
          AsyncConfigurer;
    import org.springframework.scheduling.annotation.EnableAsync;
    import org.springframework.scheduling.concurrent.
          ThreadPoolTaskExecutor;

    @Configuration
    @EnableAsync
    public class AsyncConfig implements AsyncConfigurer {

        @Override
        public Executor getAsyncExecutor() {
            ThreadPoolTaskExecutor executor = new
                  ThreadPoolTaskExecutor();
            executor.setCorePoolSize(5);
            executor.setMaxPoolSize(40);
            executor.setQueueCapacity(100);
            executor.setThreadNamePrefix("MyExecutor-");
            executor.initialize();
            return executor;
        }

        @Override
        public AsyncUncaughtExceptionHandler
              getAsyncUncaughtExceptionHandler() {
            return null;
        }

    }
```

2.2. ResolverConfig.java

```
package ph.edu.upm.agila.japiscos.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.ViewResolver;
import org.springframework.web.servlet.config.annotation.
      WebMvcConfigurerAdapter;
import org.springframework.web.servlet.view.
      ResourceBundleViewResolver;
import org.thymeleaf.spring4.view.ThymeleafViewResolver;

@Configuration
public class ResolverConfig extends WebMvcConfigurerAdapter {

    @Bean
    public ResourceBundleViewResolver
          resourceBundleViewResolver() {
        ResourceBundleViewResolver resolver = new
              ResourceBundleViewResolver();
        resolver.setBasename("views");
        resolver.setOrder(0);
        return resolver;
    }

    @Bean
    public ViewResolver viewResolver() {
        ThymeleafViewResolver viewResolver = new
              ThymeleafViewResolver();
        viewResolver.setApplicationContext(
              resourceBundleViewResolver().
              getApplicationContext());
        return viewResolver;
    }
}
```

2.3. SecurityConfig.java

```
package ph.edu.upm.agila.japiscos.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.security.
      SecurityProperties;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;
import org.springframework.security.config.annotation.
      authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.
      builders.HttpSecurity;
import org.springframework.security.config.annotation.web.
      builders.WebSecurity;
import org.springframework.security.config.annotation.web.
      configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.
      UserDetailsService;
import org.springframework.security.crypto.bcrypt.
      BCryptPasswordEncoder;

@Configuration
@Order(SecurityProperties.ACCESS_OVERRIDE_ORDER)
public class SecurityConfig extends
      WebSecurityConfigurerAdapter {

    @Autowired
    private UserDetailsService userDetailsService;

    @Override
    protected void configure(HttpSecurity http) throws
          Exception {
        http.authorizeRequests()
                .antMatchers("/home", "/login", "/bulletin-
                      board/**", "/results/**")
                .permitAll().antMatchers("/elections/**",  "/
                      positions/**", "/parties/**",
                    "/candidates/**", "/voter-types/**")
                .hasAuthority("VOTING_OFFICIAL").anyRequest()
                .fullyAuthenticated().antMatchers("/users/**")
                .hasAuthority("ADMIN").anyRequest()
                .fullyAuthenticated().antMatchers("/vote/**")
                .hasAuthority("VOTER").anyRequest()
                .fullyAuthenticated().and().formLogin().
                      loginPage("/login")
                .failureUrl("/login?error").defaultSuccessUrl(
                      "/home")
                .usernameParameter("username").permitAll().and
                      ()
                .logout().logoutUrl("/logout").
                      logoutSuccessUrl("/login").permitAll();
    }

    @Override
    public void configure(AuthenticationManagerBuilder auth)
          throws Exception {
        auth.userDetailsService(userDetailsService).
              passwordEncoder(new BCryptPasswordEncoder());
    }

    @Override
    public void configure(WebSecurity web) throws Exception {
        web.ignoring().antMatchers("/css/**", "/fonts/**", "/
              img/**", "/js/**", "/tables/**");
    }
}
```

3. Controllers

3.1. BulletinBoardController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.
      RedirectAttributes;

import ph.edu.upm.agila.japiscos.exception.SharemindException;
import ph.edu.upm.agila.japiscos.model.Voter;
import ph.edu.upm.agila.japiscos.service.VoterService;

@Controller
@RequestMapping("/bulletin-board")
public class BulletinBoardController {

    @Autowired
    private VoterService voterService;

    @RequestMapping(method = RequestMethod.GET)
    public String bulletinBoard(Model model) {
        model.addAttribute("activeSettingsBulletinBoard", "
              active");
        List<Voter> voters = voterService.getVotersVoted();
        if (!voters.isEmpty()) {
            List<Voter> recordedVoters;
            try {
                recordedVoters = voterService.
                      checkRecordedVoters(voters);
            } catch (SharemindException e) {
                model.addAttribute("errorMessage", "Sharemind
                      has encountered an error. " +
                    "Please run DevMiner if you have not run
                          it yet.");
                return "bulletin-board";
            }
            model.addAttribute("recordedVoters",
                  recordedVoters);
        } else {
            model.addAttribute("recordedVoters", new ArrayList
                  <>());
        }
        return "bulletin-board";
    }

    @RequestMapping(value = "/view-tracker/{alias}", method =
          RequestMethod.GET)
    public String viewTracker(@PathVariable String alias,
            RedirectAttributes redirectAttributes, Model model
                  ) {
        Voter voter = voterService.getVoterByAlias(alias);
        if (voter == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "There is no voter with that alias!");
            return "redirect:/bulletin-board";
        }
        String verificationCode = voter.getVerificationCode();
        model.addAttribute("verificationCode",
              verificationCode);
        model.addAttribute("json", voter.getTracker());
        return "view-tracker";
    }

    @RequestMapping(value = "/export", method = RequestMethod.
          GET)
    public ModelAndView exportBulletinBoard() {
        List<Voter> publishedVoters = voterService.
              getPublishedVoters();
        return new ModelAndView("pdfBulletinBoardView", "
              publishedVoters", publishedVoters);
    }
}
```

3.2. CandidateController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.
      RedirectAttributes;

import ph.edu.upm.agila.japiscos.formbacker.CandidateBacker;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.service.CandidateService;
import ph.edu.upm.agila.japiscos.service.ElectionService;
import ph.edu.upm.agila.japiscos.service.PartyService;
import ph.edu.upm.agila.japiscos.service.PositionService;
import ph.edu.upm.agila.japiscos.validator.
      CandidateFormValidator;

@Controller
@RequestMapping("/candidates")
public class CandidateController {

    @Autowired
    private ElectionService electionService;

    @Autowired
    private CandidateService candidateService;

    @Autowired
    private PartyService partyService;

    @Autowired
    private PositionService positionService;

    @RequestMapping(method = RequestMethod.GET)
    public String manageCandidatePage(Model model) {
        model.addAttribute("activeSettingsCandidate", "active"
            );
        List<Election> electionList = electionService.
            getAllElections();
        model.addAttribute("electionList", electionList);
        return "candidates";
    }

    @RequestMapping(value = "/get-candidates", method =
          RequestMethod.GET)
    public @ResponseBody List<Candidate> getCandidates(
            @RequestParam("electionId") long id) throws
                Exception {
        Election election = electionService.findById(id);
        return candidateService.getCandidatesByElection(
            election);
    }

    @RequestMapping(value = "/add", method = RequestMethod.GET
          )
    public String addCandidatePage(Model model,
            RedirectAttributes redirectAttributes) {
        model.addAttribute("activeSettingsCandidate", "active"
            );
        List<Election> electionList = electionService.
            getAllElections();
        if (electionList.isEmpty()) {
            redirectAttributes.addFlashAttribute("errorMessage
                ",  "You cannot add a candidate with no
                elections.");
            return "redirect:/candidates";
        }
        CandidateBacker candidateBacker = new CandidateBacker
            ();
        model.addAttribute("electionList", electionList);
        if (!model.containsAttribute("candidateBacker")) {
            model.addAttribute("candidateBacker",
                candidateBacker);
        }
        return "add-candidate";
    }

    @RequestMapping(value = "/add", method = RequestMethod.
          POST)
    public String addCandidate(@Valid CandidateBacker
          candidateBacker,
            BindingResult result, RedirectAttributes
                redirectAttributes) {
        CandidateFormValidator candidateFormValidator = new
            CandidateFormValidator();
        candidateFormValidator.validate(candidateBacker,
            result);
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                candidateBacker", result);
            redirectAttributes.addFlashAttribute("
                candidateBacker", candidateBacker);
            return "redirect:add";
        } else {
            Election election = electionService.findById(
                candidateBacker.getElectionId());
            if (election.isStatus()) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "You cannot add a
                    candidate in a running election.");
                return "redirect:add";
            }
            if (election.getEndDate() != null) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "You cannot add a
                    candidate in a finished election.");
                return "redirect:add";
            }
            candidateService.add(candidateBacker);
            redirectAttributes.addFlashAttribute("
                successMessage", "Successfully added " +
                candidateBacker.getLastName().toUpperCase() +
                    ", " + candidateBacker.getFirstName());
            return "redirect:add";
        }
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.GET)
    public String editCandidatePage(@PathVariable long id,
          Model model,
            RedirectAttributes redirectAttributes) {
        model.addAttribute("activeSettingsCandidate", "active"
            );
        Candidate candidate = candidateService.findById(id);
        if (candidate == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "There is no candidate with that id!");
            return "redirect:/candidates";
        }
        if (candidate.getElection().isStatus()) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "A candidate in a running election cannot
                 be edited.");
            return "redirect:/candidates";
        }
        if (candidate.getElection().getEndDate() != null) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "A candidate in a finished election
                cannot be edited.");
            return "redirect:/candidates";
        }
        List<Election> electionList = electionService.
            getAllElections();
        model.addAttribute("electionList", electionList);
        if (!model.containsAttribute("candidateBacker")) {
            CandidateBacker candidateBacker = new
                CandidateBacker();
            candidateBacker.setId(candidate.getId());
            candidateBacker.setFirstName(candidate.
                getFirstName());
            candidateBacker.setLastName(candidate.getLastName
                ());
            candidateBacker.setElectionId(candidate.
                getElection().getId());
            candidateBacker.setPositionId(candidate.
                getPosition().getId());
            candidateBacker.setPartyId(candidate.getParty().
                getId());
            model.addAttribute("candidateBacker",
                candidateBacker);
        }
        return "edit-candidate";
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.POST)
    public String editCandidate(@PathVariable long id,
            @Valid CandidateBacker candidateBacker,
                BindingResult result,
            RedirectAttributes redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                candidateBacker", result);
            redirectAttributes.addFlashAttribute("
                candidateBacker", candidateBacker);
            return "redirect:/candidates/edit/" + id;
        } else {
            candidateService.edit(candidateBacker);
            redirectAttributes.addFlashAttribute("
                successMessage", "Successfully edited " +
                candidateBacker.getLastName().toUpperCase() +
                    ", " + candidateBacker.getFirstName());
            return "redirect:/candidates/edit/" + id;
        }
    }

    @RequestMapping(value = "/delete/{id}", method =
          RequestMethod.GET)
    public String deleteCandidate(@PathVariable long id,
            RedirectAttributes redirectAttributes) {
        Candidate candidate = candidateService.findById(id);
        if (candidate == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "There is no candidate with that id!");
            return "redirect:/candidates";
        }
        if (candidate.getElection().isStatus()) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "A candidate in a running election cannot
                 be deleted.");
```

```
            return "redirect:/candidates";
        }
        if (candidate.getElection().getEndDate() != null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A candidate in a finished election
                  cannot be deleted.");
            return "redirect:/candidates";
        }
        String candidateName = candidate.getLastName().
              toUpperCase() + ", " + candidate.getFirstName();
        candidateService.delete(candidate);
        redirectAttributes.addFlashAttribute("successMessage",
              "Successfully deleted " + candidateName);
        return "redirect:/candidates";
    }
}
```

3.3. ElectionController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.io.IOException;
import java.util.List;

import javax.servlet.http.HttpSession;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.
      RedirectAttributes;

import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.service.CandidateService;
import ph.edu.upm.agila.japiscos.service.ElectionService;
import ph.edu.upm.agila.japiscos.service.VoteService;
import ph.edu.upm.agila.japiscos.service.VoterService;

@Controller
@RequestMapping("/elections")
public class ElectionController {

    @Autowired
    private ElectionService electionService;

    @Autowired
    private CandidateService candidateService;

    @Autowired
    private VoteService voteService;

    @Autowired
    private VoterService voterService;

    @RequestMapping(method = RequestMethod.GET)
    public String manageElectionPage(Model model) {
        model.addAttribute("activeSettingsElection", "active")
              ;
        List<Election> electionList = electionService.
              getElectionsByLevel();
        model.addAttribute("electionList", electionList);
        return "elections";
    }

    @RequestMapping(value = "/get-elections", method =
          RequestMethod.GET)
    public @ResponseBody List<Election> getElections() throws
          Exception {
        List<Election> allElections = electionService.
              getAllElections();
        return allElections;
    }

    @RequestMapping(value = "/add", method = RequestMethod.GET
          )
    public String addElectionPage(Model model) {
        model.addAttribute("activeSettingsElection", "active")
              ;
        List<Election> electionList = electionService.
              getAllElections();
        model.addAttribute("electionList", electionList);
        if (!model.containsAttribute("election")) {
            model.addAttribute("election", new Election());
        }
        return "add-election";
    }

    @RequestMapping(value = "/add", method = RequestMethod.
          POST)
    public String addElection(@Valid Election election,
          BindingResult result,
            RedirectAttributes redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                  springframework.validation.BindingResult.
                  election", result);
            redirectAttributes.addFlashAttribute("election",
                  election);
            return "redirect:add";
        } else {
            electionService.save(election);
            redirectAttributes.addFlashAttribute("
                  successMessage", "Successfully added " +
                  election.getName());
            return "redirect:add";
        }

    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.GET)
    public String editElectionPage(@PathVariable long id,
          Model model,
            RedirectAttributes redirectAttributes) {
        model.addAttribute("activeSettingsElection", "active")
              ;
        Election election = electionService.findById(id);
        if (election == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "There is no election with that id!");
            return "redirect:/elections";
        }
        if (election.getEndDate() != null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A finished election cannot be edited.");
            return "redirect:/elections";
        }
        if (election.isStatus()) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A running election cannot be edited.");
            return "redirect:/elections";
        }
        List<Election> electionList = electionService.
              getAllElections();
        model.addAttribute("electionList", electionList);
        if (!model.containsAttribute("election")) {
            model.addAttribute("election", election);
        }
        return "edit-election";
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.POST)
    public String editElection(@PathVariable long id, @Valid
          Election election,
            BindingResult result, RedirectAttributes
                  redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                  springframework.validation.BindingResult.
                  election", result);
            redirectAttributes.addFlashAttribute("election",
                  election);
            return "redirect:/elections/edit/" + id;
        } else {
            electionService.edit(election);
            redirectAttributes.addFlashAttribute("
                  successMessage", "Successfully edited " +
                  election.getName());
            return "redirect:/elections/edit/" + id;
        }
    }

    @RequestMapping(value = "/delete/{id}", method =
          RequestMethod.GET)
    public String deleteElection(@PathVariable long id,
            RedirectAttributes redirectAttributes) {
        Election election = electionService.findById(id);
        if (election == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "There is no election with that id!");
            return "redirect:/elections";
        }
        if (electionService.electionInUse(election)) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "An election in use cannot be deleted.");
            return "redirect:/elections";
        }
        if (election.getEndDate() != null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A finished election cannot be deleted.")
                  ;
            return "redirect:/elections";
        }
        if (election.isStatus()) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A running election cannot be deleted.");
            return "redirect:/elections";
        }
        String deletedElection = election.getName();
        electionService.delete(election);
        redirectAttributes.addFlashAttribute("successMessage",
              "Successfully deleted " + deletedElection);
        return "redirect:/elections";
    }

    @RequestMapping(value = "/change-status/{id}", method =
          RequestMethod.GET)
    public String changeStatus(@PathVariable long id,
            RedirectAttributes redirectAttributes, HttpSession
                  session) {
        Election election = electionService.findById(id);
        if (election == null) {
            redirectAttributes.addFlashAttribute("errorMessage
```

```
                ", "There is no election with that id!");
            return "redirect:/elections";
        }
        if (election.getEndDate() != null) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "You cannot start a finished election.");
            return "redirect:/elections";
        }
        if (!election.isStatus()) {
            List<Candidate> candidates = candidateService.
                getCandidatesByElection(election);
            if (candidates.isEmpty()) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "You attempted to start
                    an election with no candidates.");
                return "redirect:/elections";
            }
        }
        electionService.changeStatus(election);
        if (!election.isStatus()) {
            List<Candidate> candidates = candidateService.
                getCandidatesByElectionWithChildren(election
                );
            try {
                voteService.processResults(candidates);
            } catch (IOException | InterruptedException e) {
                e.printStackTrace();
            }
        }
        redirectAttributes.addFlashAttribute("successMessage",
            "Successfully changed election status");
        return "redirect:/elections";
    }
}
```

3.4. LoginController.java

```
package ph.edu.upm.agila.japiscos.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import ph.edu.upm.agila.japiscos.service.UserService;

@Controller
public class LoginController {

    @Autowired
    private UserService userService;

    @RequestMapping(value = "/login", method = RequestMethod.
        GET)
    public String loginPage() {
        return "login";
    }

}
```

3.5. MainController.java

```
package ph.edu.upm.agila.japiscos.controller;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.mvc.support.
      RedirectAttributes;

import ph.edu.upm.agila.japiscos.model.CurrentUser;
import ph.edu.upm.agila.japiscos.model.Voter;
import ph.edu.upm.agila.japiscos.service.VoterService;

@Controller
public class MainController {

    @Autowired
    private VoterService voterService;

    @RequestMapping(value = "/home", method = RequestMethod.
          GET)
    public String homePage(HttpServletRequest request,
            Authentication authentication, RedirectAttributes
                redirectAttributes) {
        if (request.isUserInRole("VOTER")) {
            CurrentUser currentUser = (CurrentUser)
                authentication.getPrincipal();
            Voter voter = voterService.findVoterByUser(
                currentUser.getUser());
            if (voter.isActive()) {
                return "redirect:/vote";
            } else {
                try {
                    request.logout();
                } catch (ServletException e) {
                    e.printStackTrace();
                }
                redirectAttributes.addFlashAttribute("
                    errorMessage", "Your account is no
                    longer active.");
                return "redirect:/login";
            }
        } else {
            return "home";
        }
    }
}
```

3.6. PartyController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.
      RedirectAttributes;

import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Party;
import ph.edu.upm.agila.japiscos.service.ElectionService;
import ph.edu.upm.agila.japiscos.service.PartyService;

@Controller
@RequestMapping("/parties")
public class PartyController {

    @Autowired
    private ElectionService electionService;

    @Autowired
    private PartyService partyService;

    @RequestMapping(method = RequestMethod.GET)
    public String managePartyPage(Model model) {
        model.addAttribute("activeSettingsParty", "active");
        List<Election> electionList = electionService.
            getAllElections();
        model.addAttribute("electionList", electionList);
        return "parties";
    }

    @RequestMapping(value = "/get-parties", method =
          RequestMethod.GET)
    public @ResponseBody List<Party> getParties(
            @RequestParam("electionId") long id) throws
                Exception {
        Election election = electionService.findById(id);
        List<Party> allParties = partyService.
            getPartiesByElection(election);
        return allParties;
    }

    @RequestMapping(value = "/add", method = RequestMethod.GET
        )
    public String addPartyPage(Model model,
            RedirectAttributes redirectAttributes) {
        model.addAttribute("activeSettingsParty", "active");
        List<Election> electionList = electionService.
            getAllElections();
        if (electionList.isEmpty()) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "You cannot add a party with no elections
                .");
            return "redirect:/parties";
        }
        model.addAttribute("electionList", electionList);
        if (!model.containsAttribute("party")) {
            model.addAttribute("party", new Party());
        }
        return "add-party";
    }

    @RequestMapping(value = "/add", method = RequestMethod.
        POST)
    public String addParty(@Valid Party party, BindingResult
          result,
            RedirectAttributes redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                party", result);
            redirectAttributes.addFlashAttribute("party",
                party);
            return "redirect:add";
        } else {
            for (Election election : party.getElections()) {
                if (election.isStatus()) {
                    redirectAttributes.addFlashAttribute("
                        errorMessage", "You cannot add a
                        party in a running election.");
                    return "redirect:add";
                }
                if (election.getEndDate() != null) {
```

```
                    redirectAttributes.addFlashAttribute("
                          errorMessage", "You cannot add a
                          party in a finished election.");
                    return "redirect:add";
                }
            }
            partyService.save(party);
            redirectAttributes.addFlashAttribute("
                  successMessage", "Successfully added " +
                  party.getName());
            return "redirect:add";
        }
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.GET)
    public String editPartyPage(@PathVariable long id, Model
          model,
            RedirectAttributes redirectAttributes) {
        model.addAttribute("activeSettingsParty", "active");
        Party party = partyService.findById(id);
        if (party == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "There is no party with that id!");
            return "redirect:/parties";
        }
        List<Election> elections = party.getElections();
        for (Election election : elections) {
            if (election.isStatus()) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "A party in a running
                      election cannot be edited.");
                return "redirect:/parties";
            }
            if (election.getEndDate() != null) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "A party in a finished
                      election cannot be edited.");
                return "redirect:/parties";
            }
        }
        model.addAttribute("activeSettingsParty", "active");
        List<Election> electionList = electionService.
              getAllElections();
        model.addAttribute("electionList", electionList);
        if (!model.containsAttribute("party")) {
            model.addAttribute("party", party);
        }
        return "edit-party";
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.POST)
    public String editParty(@PathVariable long id, @Valid
          Party party,
            BindingResult result, RedirectAttributes
                  redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                  springframework.validation.BindingResult.
                  party", result);
            redirectAttributes.addFlashAttribute("party",
                  party);
            return "redirect:/parties/edit/" + id;
        } else {
            partyService.save(party);
            redirectAttributes.addFlashAttribute("
                  successMessage", "Successfully edited " +
                  party.getName());
            return "redirect:/parties/edit/" + id;
        }
    }

    @RequestMapping(value = "/delete/{id}", method =
          RequestMethod.GET)
    public String deleteParty(@PathVariable long id,
            RedirectAttributes redirectAttributes) {
        Party party = partyService.findById(id);
        if (party == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "There is no party with that id!");
            return "redirect:/parties";
        }
        List<Election> elections = party.getElections();
        for (Election election : elections) {
            if (election.isStatus()) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "A party in a running
                      election cannot be deleted.");
                return "redirect:/parties";
            }
            if (election.getEndDate() != null) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "A party in a finished
                      election cannot be deleted.");
                return "redirect:/parties";
            }
        }
        if (!partyService.partyInUse(party)) {
            partyService.delete(party);
            redirectAttributes.addFlashAttribute("
                  successMessage", "Successfully deleted " +
                  party.getName());
        } else {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A party in use cannot be deleted.");
        }
        return "redirect:/parties";
    }
}
```

3.7. PositionController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.
      RedirectAttributes;

import ph.edu.upm.agila.japiscos.formbacker.PositionBacker;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Position;
import ph.edu.upm.agila.japiscos.service.ElectionService;
import ph.edu.upm.agila.japiscos.service.PositionService;

@Controller
@RequestMapping("/positions")
public class PositionController {

    @Autowired
    private ElectionService electionService;

    @Autowired
    private PositionService positionService;

    @RequestMapping(method = RequestMethod.GET)
    public String managePositionPage(Model model) {
        model.addAttribute("activeSettingsPosition", "active")
              ;
        List<Election> electionList = electionService.
              getAllElections();
        model.addAttribute("electionList", electionList);
        return "positions";
    }

    @RequestMapping(value = "/get-positions", method =
          RequestMethod.GET)
    public @ResponseBody List<Position> getPositions(
            @RequestParam("electionId") long id) throws
                  Exception {
        Election election = electionService.findById(id);
        List<Position> allPositions = positionService.
              getPositionsByElection(election);
        return allPositions;
    }

    @RequestMapping(value = "/add", method = RequestMethod.GET
          )
    public String addPositionPage(Model model,
            RedirectAttributes redirectAttributes) {
        model.addAttribute("activeSettingsPosition", "active")
              ;
        List<Election> elections = electionService.
              getAllElections();
        if (elections.isEmpty()) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "You cannot add a position with no
                  elections.");
            return "redirect:/positions";
        }
        model.addAttribute("electionList", elections);
        if (!model.containsAttribute("positionBacker")) {
            model.addAttribute("positionBacker", new
                  PositionBacker());
        }
        return "add-position";
    }

    @RequestMapping(value = "/add", method = RequestMethod.
          POST)
    public String addPosition(@Valid PositionBacker
          positionBacker,
            BindingResult result, RedirectAttributes
                  redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                  springframework.validation.BindingResult.
                  positionBacker", result);
            redirectAttributes.addFlashAttribute("
                  positionBacker", positionBacker);
            return "redirect:add";
        } else {
            Election election = electionService.findById(
                  positionBacker.getElectionId());
            if (election.isStatus()) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "You cannot add a
                      position in a running election.");
                return "redirect:add";
```

```
            }
            if (election.getEndDate() != null) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "You cannot add a
                      position in a finished election.");
                return "redirect:add";
            }
            positionService.add(positionBacker);
            redirectAttributes.addFlashAttribute("
                  successMessage", "Successfully added " +
                  positionBacker.getName());
            return "redirect:add";
        }
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.GET)
    public String editPositionPage(@PathVariable long id,
          Model model,
            RedirectAttributes redirectAttributes) {
        Position positionToUpdate = positionService.findById(
              id);
        if (positionToUpdate == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "There is no position with that id!");
            return "redirect:/positions";
        }
        if (positionToUpdate.getElection().isStatus()) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A position in a running election cannot
                  be edited.");
            return "redirect:/positions";
        }
        if (positionToUpdate.getElection().getEndDate() !=
              null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A position in a finished election cannot
                   be edited.");
            return "redirect:/positions";
        }
        model.addAttribute("activeSettingsPosition", "active")
              ;
        List<Election> electionList = electionService.
              getAllElections();
        model.addAttribute("electionList", electionList);
        if (!model.containsAttribute("positionBacker")) {
            PositionBacker positionBacker = new PositionBacker
                  ();
            positionBacker.setId(positionToUpdate.getId());
            positionBacker.setName(positionToUpdate.getName())
                  ;
            positionBacker.setElectionId(positionToUpdate.
                  getElection().getId());
            positionBacker.setMaxElected(positionToUpdate.
                  getMaxElected());
            positionBacker.setOrdinality(positionToUpdate.
                  getOrdinality());
            positionBacker.setAbstain(positionToUpdate.
                  isAbstain());
            model.addAttribute("positionBacker",
                  positionBacker);
        }
        return "edit-position";
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.POST)
    public String editPosition(@PathVariable long id,
            @Valid PositionBacker positionBacker,
                  BindingResult result,
            RedirectAttributes redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                  springframework.validation.BindingResult.
                  positionBacker", result);
            redirectAttributes.addFlashAttribute("
                  positionBacker", positionBacker);
            return "redirect:/positions/edit/" + id;
        } else {
            positionService.edit(positionBacker);
            redirectAttributes.addFlashAttribute("
                  successMessage", "Successfully edited " +
                  positionBacker.getName());
            return "redirect:/positions/edit/" + id;
        }
    }

    @RequestMapping(value = "/delete/{id}", method =
          RequestMethod.GET)
    public String deletePosition(@PathVariable long id,
            RedirectAttributes redirectAttributes) {
        Position position = positionService.findById(id);
        if (position == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "There is no position with that id!");
            return "redirect:/positions";
        }
        if (positionService.positionInUse(position)) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A position in use cannot be deleted.");
            return "redirect:/positions";
        }
        if (position.getElection().isStatus()) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A position in a running election cannot
                  be deleted.");
            return "redirect:/positions";
        }
        if (position.getElection().getEndDate() != null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "A position in a finished election cannot
                   be deleted.");
            return "redirect:/positions";
        }
        String deletedPosition = position.getName();
        positionService.delete(position);
        redirectAttributes.addFlashAttribute("successMessage",
               "Successfully deleted " + deletedPosition);
        return "redirect:/positions";
    }
}
```

3.8. ResultsController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.
      RedirectAttributes;

import ph.edu.upm.agila.japiscos.model.Abstain;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Position;
import ph.edu.upm.agila.japiscos.model.Voter;
import ph.edu.upm.agila.japiscos.model.VoterType;
import ph.edu.upm.agila.japiscos.service.AbstainService;
import ph.edu.upm.agila.japiscos.service.CandidateService;
import ph.edu.upm.agila.japiscos.service.ElectionService;
import ph.edu.upm.agila.japiscos.service.PositionService;
import ph.edu.upm.agila.japiscos.service.VoterService;
import ph.edu.upm.agila.japiscos.service.VoterTypeService;

@Controller
@RequestMapping("/results")
public class ResultsController {

    @Autowired
    private ElectionService electionService;

    @Autowired
    private PositionService positionService;

    @Autowired
    private CandidateService candidateService;

    @Autowired
    private VoterService voterService;

    @Autowired
    private VoterTypeService voterTypeService;

    @Autowired
    private AbstainService abstainService;

    @RequestMapping(method = RequestMethod.GET)
    public String viewResultsPage(Model model, Authentication
          authentication,
            HttpSession session) {
        model.addAttribute("activeSettingsResults", "active");
        boolean notRunning = false;
        List<Election> electionList = electionService.
              getElectionsByStatus(notRunning);
        if (!electionList.isEmpty()) {
            model.addAttribute("electionList", electionList);
        }
        return "view-results";
    }

    @RequestMapping(value = "/get-positions", method =
          RequestMethod.GET)
    public @ResponseBody List<Position> getPositions(
            @RequestParam("electionId") long id) throws
                  Exception {
        Election election = electionService.findById(id);
        List<Position> allPositions = positionService.
              getPositionsByElection(election);
        return allPositions;
    }

    @RequestMapping(value = "/get-abstain", method =
          RequestMethod.GET)
    public @ResponseBody int getAbstainVoteCount(
            @RequestParam("positionId") long id) throws
                  Exception {
        Position position = positionService.findById(id);
        Abstain abstain = abstainService.findByPosition(
```

```
                position);
        return abstain.getVoteCount();
    }

    @RequestMapping(value = "/get-candidates-results", method
          = RequestMethod.GET)
    public @ResponseBody List<Candidate> getCandidatesResults(
            @RequestParam("electionId") long id) throws
                  Exception {
        Election election = electionService.findById(id);
        List<Candidate> candidates = candidateService.
              getCandidatesByElection(election);
        return candidates;
    }

    @RequestMapping(value = "/get-voters", method =
          RequestMethod.GET)
    public @ResponseBody List<Voter> getVoters(
            @RequestParam("electionId") long id) throws
                  Exception {
        Election election = electionService.findById(id);
        List<VoterType> voterTypes = voterTypeService.
              getTypesByElection(election);
        List<Voter> voters = voterService.
              getVotersByVoterTypes(voterTypes);
        return voters;
    }

    @RequestMapping(value = "/download-results/{id}", method =
          RequestMethod.GET)
    public String downloadPDF(@PathVariable long id,
        RedirectAttributes redirectAttributes, Model model
              ) {
        Election election = electionService.findById(id);
        if (election == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "There is no election with that id!");
            return "redirect:/results";
        }
        List<Candidate> candidates = candidateService.
              getCandidatesByElection(election);
        List<Position> positions = positionService.
              getCandidatePositions(election);
        List<Abstain> abstains = new ArrayList<>();
        for (Position position : positions) {
            if (position.isAbstain()) {
                abstains.add(abstainService.findByPosition(
                      position));
            }
        }
        List<VoterType> voterTypes = voterTypeService.
              getTypesByElection(election);
        List<Voter> voters = voterService.
              getVotersByVoterTypes(voterTypes);
        model.addAttribute("positions", positions);
        model.addAttribute("candidates", candidates);
        model.addAttribute("abstains", abstains);
        model.addAttribute("voters", voters);
        return "pdfView";
    }
}
```

3.9. UserController.java

```
package ph.edu.upm.agila.japiscos.controller;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.mvc.support.
      RedirectAttributes;

import ph.edu.upm.agila.japiscos.exception.
      EmailExistsException;
import ph.edu.upm.agila.japiscos.exception.
      UsernameExistsException;
import ph.edu.upm.agila.japiscos.formbacker.ChangePasswordForm
      ;
import ph.edu.upm.agila.japiscos.formbacker.UserEditForm;
import ph.edu.upm.agila.japiscos.model.CurrentUser;
import ph.edu.upm.agila.japiscos.model.Role;
import ph.edu.upm.agila.japiscos.model.User;
import ph.edu.upm.agila.japiscos.service.UserService;
import ph.edu.upm.agila.japiscos.validator.PasswordValidator;

@Controller
@RequestMapping("/user")
public class UserController {

    @Autowired
    private UserService userService;

    @RequestMapping(value = "{id}", method = RequestMethod.GET
          )
    public String viewUser(@PathVariable long id,
            Authentication authentication, Model model) {
        CurrentUser currentUser = (CurrentUser) authentication
              .getPrincipal();
        if (currentUser.getUser().getId() == id) {
            User user = userService.findById(id);
            model.addAttribute("user", user);
        } else {
            model.addAttribute("errorMessage", "You cannot
                  view the profile of this user");
        }
        return "user-profile";
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.GET)
    public String editUserPage(@PathVariable long id,
            Authentication authentication, Model model) {
        CurrentUser currentUser = (CurrentUser) authentication
              .getPrincipal();
        if (currentUser.getUser().getId() == id) {
            if (!model.containsAttribute("userEditForm")) {
                User user = userService.findById(id);
                UserEditForm userEditForm = new UserEditForm()
                      ;
                userEditForm.setUsername(user.getUsername());
                userEditForm.setRole(user.getRole());
                userEditForm.setEmail(user.getEmail());
                model.addAttribute("userEditForm",
                      userEditForm);
            }
        } else {
            model.addAttribute("errorMessage", "You cannot
                  view the profile of this user");
        }
        return "settings";
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.POST)
    public String editUser(@PathVariable long id,
            @Valid UserEditForm userEditForm, BindingResult
                  result,
            RedirectAttributes redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                  springframework.validation.BindingResult.
                  userEditForm", result);
            redirectAttributes.addFlashAttribute("userEditForm
                  ", userEditForm);
            return "redirect:/user/edit/" + id;
        } else {
            userEditForm.setId(id);
            User newUser;
            try {
                newUser = userService.edit(userEditForm);
            } catch (UsernameExistsException e) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "The username " + e.
                      getUsername() + " exists!");
                return "redirect:/user/edit/" + id;
            } catch (EmailExistsException e) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "The email " + e.getEmail
                      () + " exists!");
                return "redirect:/user/edit/" + id;
            }
            redirectAttributes.addFlashAttribute("
                  successMessage", "Successfully edited " +
                  newUser.getUsername());
            return "redirect:/user/edit/" + id;
        }
    }

    @RequestMapping(value = "/change-password/{id}", method =
          RequestMethod.GET)
    public String changePasswordPage(@PathVariable long id,
            Authentication authentication, Model model) {
        CurrentUser currentUser = (CurrentUser) authentication
              .getPrincipal();
        if (currentUser.getUser().getRole() == Role.VOTER) {
            model.addAttribute("errorMessage", "You cannot
                  change your password.");
        } else if (currentUser.getUser().getId() == id) {
            if (!model.containsAttribute("changePasswordForm")
                  ) {
                model.addAttribute("changePasswordForm", new
                      ChangePasswordForm());
            }
        } else {
            model.addAttribute("errorMessage", "You cannot
                  change the password of this user.");
        }
        return "change-password";
    }

    @RequestMapping(value = "/change-password/{id}", method =
          RequestMethod.POST)
    public String changePassword(@PathVariable long id,
            @Valid ChangePasswordForm changePasswordForm,
                  BindingResult result,
            Authentication authentication, RedirectAttributes
                  redirectAttributes) {
        CurrentUser currentUser = (CurrentUser) authentication
              .getPrincipal();
        PasswordValidator passwordValidator = new
              PasswordValidator(currentUser.getUser());
        passwordValidator.validate(changePasswordForm, result)
              ;
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
```

```
                springframework.validation.BindingResult.
                changePasswordForm", result);
            redirectAttributes.addFlashAttribute("
                changePasswordForm", changePasswordForm);
            return "redirect:/user/change-password/" + id;
        } else {
            changePasswordForm.setId(id);
            userService.changePassword(changePasswordForm);
            redirectAttributes.addFlashAttribute("
                successMessage", "Successfully changed
                password");
            return "redirect:/user/change-password/" + id;
        }
    }
}
```

3.10. VoteController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletResponse;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.mvc.support.
    RedirectAttributes;

import ph.edu.upm.agila.japiscos.exception.SharemindException;
import ph.edu.upm.agila.japiscos.formbacker.Ballot;
import ph.edu.upm.agila.japiscos.formbacker.Vote;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.CurrentUser;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Position;
import ph.edu.upm.agila.japiscos.model.Voter;
import ph.edu.upm.agila.japiscos.model.VoterType;
import ph.edu.upm.agila.japiscos.service.CandidateService;
import ph.edu.upm.agila.japiscos.service.ElectionService;
import ph.edu.upm.agila.japiscos.service.PositionService;
import ph.edu.upm.agila.japiscos.service.VoteService;
import ph.edu.upm.agila.japiscos.service.VoterService;
import ph.edu.upm.agila.japiscos.service.VoterTypeService;
import ph.edu.upm.agila.japiscos.validator.BallotValidator;

@Controller
@RequestMapping("/vote")
public class VoteController {

    @Autowired
    private ElectionService electionService;

    @Autowired
    private PositionService positionService;

    @Autowired
    private CandidateService candidateService;

    @Autowired
    private VoteService voteService;

    @Autowired
    private VoterService voterService;

    @Autowired
    private VoterTypeService voterTypeService;

    @RequestMapping(method = RequestMethod.GET)
    public String votePage(Model model, Authentication
        authentication) {
        model.addAttribute("activeSettingsVote", "active");
        CurrentUser currentUser = (CurrentUser) authentication
            .getPrincipal();
        Voter voter = voterService.findVoterByUser(currentUser
            .getUser());
        VoterType voterType = voter.getVoterType();
        List<Election> voterElections = voterType.getElections
            ();
        List<Position> positions = new ArrayList<>();
        List<Candidate> candidates = new ArrayList<>();
        List<Election> runningElections = electionService.
            getRunningElections(voterElections);
        if (!runningElections.isEmpty()) {
            for (Election election : runningElections) {
                positions.addAll(positionService.
                    getCandidatePositions(election));
                candidates.addAll(candidateService.
                    getCandidatesByElection(election));
            }
        }
        if (voter.isVoted()) {
            model.addAttribute("voted", true);
        }
        if (!positions.isEmpty()) {
            model.addAttribute("positions", positions);
        }
        if (!candidates.isEmpty()) {
            model.addAttribute("candidates", candidates);
        }
        if (!model.containsAttribute("ballot")) {
            model.addAttribute("ballot", new Ballot());
        }
        return "vote";
    }

    @RequestMapping(value = "/confirm", method = RequestMethod
        .POST)
    public String vote(@Valid Ballot ballot, BindingResult
        result,
            RedirectAttributes redirectAttributes, Model model
                ) {
        BallotValidator ballotValidator = new BallotValidator
            ();
        ballotValidator.validate(ballot, result);
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                ballot", result);
            redirectAttributes.addFlashAttribute("ballot",
                ballot);
            return "redirect:/vote";
        } else {
            redirectAttributes.addFlashAttribute("ballot",
                ballot);
            return "redirect:review";
        }
    }

    @RequestMapping(value = "/review", method = RequestMethod.
        GET)
    public String reviewVotePage(Model model, Authentication
        authentication) {
        model.addAttribute("activeSettingsVote", "active");
        Ballot ballot = (Ballot) model.asMap().get("ballot");
        CurrentUser currentUser = (CurrentUser) authentication
            .getPrincipal();
        Voter voter = voterService.findVoterByUser(currentUser
            .getUser());
        List<Vote> votes = ballot.getVotes();
        List<Candidate> votedCandidates = candidateService.
            getVotedCandidates(votes, voter);
        List<Position> positions = new ArrayList<>();
        List<Election> elections = voter.getVoterType().
            getElections();
        for (Election election : elections) {
            positions.addAll(positionService.
                getCandidatePositions(election));
        }
        model.addAttribute("ballot", ballot);
        model.addAttribute("voter", voter);
        model.addAttribute("votedCandidates", votedCandidates)
            ;
        model.addAttribute("positions", positions);
        return "review-ballot";
    }

    @RequestMapping(value = "/review", method = RequestMethod.
        POST)
    public String vote(Ballot ballot, Authentication
        authentication,
            RedirectAttributes redirectAttributes) {
        CurrentUser currentUser = (CurrentUser) authentication
            .getPrincipal();
        Voter voter = voterService.findVoterByUser(currentUser
            .getUser());
        try {
            if (!voter.isVoted()) {
                voteService.addVotes(ballot.getVotes(), voter)
                    ;
            } else {
                voteService.updateVotes(ballot.getVotes(),
                    voter);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (SharemindException e) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "Sharemind has encountered an error. " +
                "Please run DevMiner if you have not run it
                    yet. ");
            redirectAttributes.addFlashAttribute("ballot",
                ballot);
            return "redirect:/vote/review";
        }
        redirectAttributes.addFlashAttribute("successMessage",
            "Vote submitted! <br/> Alias: " + voter.
            getAlias()
            + "<br/> Verification code: " + voter.
                getVerificationCode()
            + "<br/> Remember your alias and save the
                verification code to check if your vote is
                correctly recorded.");
        return "redirect:/vote";
    }

    @RequestMapping(value = "/ivote-candidate-hashes", method
        = RequestMethod.POST)
    public void export(Ballot ballot, Authentication
        authentication,
            HttpServletResponse response) throws IOException {
        List<Vote> votes = ballot.getVotes();
        CurrentUser currentUser = (CurrentUser) authentication
```

```
            .getPrincipal();
        Voter voter = voterService.findVoterByUser(currentUser
            .getUser());
        List<Candidate> votedCandidates = candidateService.
            getVotedCandidates(votes, voter);
        response.setContentType("text/plain");
        response.setHeader("Content-Disposition","attachment;
            filename=ivote-candidate-hashes.txt");
        ServletOutputStream out = response.getOutputStream();
        String electionName = "";
        for (Candidate candidate : votedCandidates) {
            if (electionName.equals("") || !electionName.
                equals(candidate.getElection().getName())) {
                out.println(candidate.getElection().getName())
                    ;
            }
            out.println(candidate.getPosition().getName());
            if (candidate.getId() != 0) {
                out.println(candidate.getLastName() + ", "
                        + candidate.getFirstName() + " (Hash:
                            "
                        + candidate.getHash() + ")");
            } else {
                out.println("ABSTAIN (Hash: " + candidate.
                    getHash() + ")");
            }
            out.println();
            electionName = candidate.getElection().getName();
        }
        out.flush();
        out.close();
    }
}
```

3.11. VoterController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.io.IOException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.multipart.MultipartException;
import org.springframework.web.servlet.
    HandlerExceptionResolver;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.
    RedirectAttributes;

import ph.edu.upm.agila.japiscos.exception.
    EmailExistsException;
import ph.edu.upm.agila.japiscos.exception.
    IncorrectCSVFormatException;
import ph.edu.upm.agila.japiscos.exception.
    InvalidEmailException;
import ph.edu.upm.agila.japiscos.exception.
    UsernameExistsException;
import ph.edu.upm.agila.japiscos.formbacker.ImportVoterForm;
import ph.edu.upm.agila.japiscos.formbacker.VoterCreateForm;
import ph.edu.upm.agila.japiscos.formbacker.VoterEditForm;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Voter;
import ph.edu.upm.agila.japiscos.model.VoterType;
import ph.edu.upm.agila.japiscos.service.BatchInsertService;
import ph.edu.upm.agila.japiscos.service.UserService;
import ph.edu.upm.agila.japiscos.service.VoterService;
import ph.edu.upm.agila.japiscos.service.VoterTypeService;
import ph.edu.upm.agila.japiscos.validator.FileValidator;

@Controller
@RequestMapping("/voters")
public class VoterController implements
    HandlerExceptionResolver {

    @Autowired
    private VoterService voterService;

    @Autowired
    private VoterTypeService voterTypeService;

    @Autowired
    private UserService userService;

    @Autowired
    private BatchInsertService batchInsertService;

    @RequestMapping(method = RequestMethod.GET)
    public String votersPage(Model model) {
        model.addAttribute("activeSettingsVoter", "active");
        List<Voter> voters = voterService.getAllActiveVoters()
            ;
        model.addAttribute("voterList", voters);
        return "voters";
    }

    @RequestMapping(value = "/add", method = RequestMethod.GET
        )
    public String addVoterPage(Model model,
            RedirectAttributes redirectAttributes) {
        model.addAttribute("activeSettingsVoter", "active");
        List<VoterType> voterTypes = voterTypeService.
            getAllVoterTypes();
        if (voterTypes.isEmpty()) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "You cannot add a voter with no voter
                types.");
            return "redirect:/voters";
        }
        model.addAttribute("voterTypeList", voterTypes);
        if (!model.containsAttribute("voterCreateForm")) {
            model.addAttribute("voterCreateForm", new
                VoterCreateForm());
        }
        return "add-voter";
    }

    @RequestMapping(value = "/add", method = RequestMethod.
        POST)
    public String addVoter(@Valid VoterCreateForm
        voterCreateForm,
            BindingResult result, RedirectAttributes
                redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                voterCreateForm", result);
            redirectAttributes.addFlashAttribute("
                voterCreateForm", voterCreateForm);
            return "redirect:add";
        } else {
            try {
                voterService.add(voterCreateForm);
            } catch (UsernameExistsException e) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "The username " + e.
                    getUsername() + " exists!");
                return "redirect:add";
            } catch (EmailExistsException e) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "The email " + e.getEmail
                    () + " exists!");
                return "redirect:add";
            }
            redirectAttributes.addFlashAttribute("
                successMessage", "Successfully added " +
                voterCreateForm.getUsername());
            return "redirect:add";
        }
    }

    @RequestMapping(value = "/import", method = RequestMethod.
        GET)
    public String importCsvFilePage(Model model,
            RedirectAttributes redirectAttributes) {
        model.addAttribute("activeSettingsVoter", "active");
        List<VoterType> voterTypes = voterTypeService.
            getAllVoterTypes();
        if (voterTypes.isEmpty()) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "You cannot import voters with no voter
                types.");
            return "redirect:/voters";
        }
        model.addAttribute("voterTypeList", voterTypes);
        if (!model.containsAttribute("importVoterForm")) {
            model.addAttribute("importVoterForm", new
                ImportVoterForm());
        }
        return "import-voters";
    }

    @RequestMapping(value = "/import", method = RequestMethod.
        POST)
    public String importCsvFile(@Valid ImportVoterForm
        importVoterForm,
            BindingResult result, RedirectAttributes
                redirectAttributes) {
        FileValidator fileValidator = new FileValidator();
        fileValidator.validate(importVoterForm, result);
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                importVoterForm", result);
            redirectAttributes.addFlashAttribute("
                importVoterForm", importVoterForm);
            return "redirect:import";
        } else {
            try {
                batchInsertService.processFile(importVoterForm
                    );
            } catch (IOException e) {
                e.printStackTrace();
            } catch (InvalidEmailException e) {
                redirectAttributes.addFlashAttribute("
                    errorMessage",
                    "The email " + e.getEmail() + " of " + e.
                        getUsername() + " is invalid!");
                return "redirect:/voters/import";
            } catch (UsernameExistsException e) {
                redirectAttributes.addFlashAttribute("
```

```
                    errorMessage", "The username " + e.
                    getUsername() + " exists!");
                return "redirect:/voters/import";
            } catch (EmailExistsException e) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "The email " + e.getEmail
                    () + " exists!");
                return "redirect:/voters/import";
            } catch (IncorrectCSVFormatException e) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "Error parsing file!
                    Please follow the correct format.");
                return "redirect:/voters/import";
            }
            redirectAttributes.addFlashAttribute("
                successMessage", "Successfully imported
                voters!");
            return "redirect:/voters/import";
        }
    }

    @Override
    public ModelAndView resolveException(HttpServletRequest
          request,
            HttpServletResponse response, Object handler,
                Exception exception) {
        Map<String, Object> model = new HashMap<String, Object
            >();
        if (exception instanceof MultipartException) {
            model.put("errorMessage", "The file you upload is
                too large! (Maximum upload size is 128KB)");
        }
        model.put("importVoterForm", new ImportVoterForm());
        List<VoterType> voterTypes = voterTypeService.
            getAllVoterTypes();
        model.put("voterTypeList", voterTypes);
        return new ModelAndView("import-voters", model);
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.GET)
    public String editVoterPage(@PathVariable long id,
            RedirectAttributes redirectAttributes, Model model
                ) {
        model.addAttribute("activeSettingsVoter", "active");
        if (!model.containsAttribute("voterEditForm")) {
            Voter voter = voterService.findById(id);
            if (voter == null) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "There is no voter with
                    that id!");
                return "redirect:/voters";
            }
            VoterEditForm voterEditForm = new VoterEditForm();
            voterEditForm.setId(voter.getId());
            voterEditForm.setUsername(voter.getUser().
                getUsername());
            voterEditForm.setEmail(voter.getUser().getEmail())
                ;
            voterEditForm.setVoterType(voter.getVoterType());
            model.addAttribute("voterEditForm", voterEditForm)
                ;
        }
        List<VoterType> voterTypes = voterTypeService.
            getAllVoterTypes();
        model.addAttribute("voterTypeList", voterTypes);
        return "edit-voter";
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.POST)
    public String editVoter(@PathVariable long id,
            @Valid VoterEditForm voterEditForm, BindingResult
                result,
            RedirectAttributes redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                voterEditForm", result);
            redirectAttributes.addFlashAttribute("
                voterEditForm", voterEditForm);
            return "redirect:/voters/edit/" + id;
        } else {
            voterService.edit(voterEditForm);
            redirectAttributes.addFlashAttribute("
                successMessage", "Successfully edited " +
                voterEditForm.getUsername());
            return "redirect:/voters/edit/" + id;
        }
    }

    @RequestMapping(value = "/delete/{id}", method =
          RequestMethod.GET)
    public String deleteVoter(@PathVariable long id,
            RedirectAttributes redirectAttributes) {
        Voter voter = voterService.findById(id);
        if (voter == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "There is no voter with that id!");
            return "redirect:/voters";
        }
        String voterToDelete = voter.getUser().getUsername();
        if (voter.isVoted()) {
            List<Election> voterElections = voter.getVoterType
                ().getElections();
            for (Election election : voterElections) {
                if (election.isStatus()) {
                    redirectAttributes.addFlashAttribute("
                        errorMessage", "A voter in a running
                         election cannot be deleted.");
                    return "redirect:/voters";
                }
            }
            redirectAttributes.addFlashAttribute("errorMessage
                ", "A voter in use cannot be deleted.");
            return "redirect:/voters";
        }
        voterService.delete(voter);
        redirectAttributes.addFlashAttribute("successMessage",
            "Successfully deleted " + voterToDelete);
        return "redirect:/voters";
    }
}
```

3.12. VoterTypeController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.mvc.support.
    RedirectAttributes;

import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.VoterType;
import ph.edu.upm.agila.japiscos.service.ElectionService;
import ph.edu.upm.agila.japiscos.service.VoterTypeService;

@Controller
@RequestMapping("/voter-types")
public class VoterTypeController {

    @Autowired
    private ElectionService electionService;

    @Autowired
    private VoterTypeService voterTypeService;

    @RequestMapping(method = RequestMethod.GET)
    public String voterTypePage(Model model) {
        model.addAttribute("activeSettingsVoterType", "active"
            );
        List<Election> electionList = electionService.
            getAllElections();
        model.addAttribute("electionList", electionList);
        return "voter-types";
    }

    @RequestMapping(value = "/get-voter-types", method =
          RequestMethod.GET)
    public @ResponseBody List<VoterType> getVoterTypes(
            @RequestParam("electionId") String electionId)
                throws Exception {
        long id = Integer.parseInt(electionId);
        Election election = electionService.findById(id);
        List<VoterType> voterTypes = voterTypeService.
            getTypesByElection(election);
        return voterTypes;
    }

    @RequestMapping(value = "/get-all-voter-types", method =
          RequestMethod.GET)
    public @ResponseBody List<VoterType> getVoterTypes()
            throws Exception {
        List<VoterType> voterTypes = voterTypeService.
            getAllVoterTypes();
        return voterTypes;
    }

    @RequestMapping(value = "/add", method = RequestMethod.GET
        )
    public String addVoterTypePage(Model model,
            RedirectAttributes redirectAttributes) {
        model.addAttribute("activeSettingsVoterType", "active"
            );
        List<Election> electionList = electionService.
            getAllElections();
        model.addAttribute("electionList", electionList);
        if (electionList.isEmpty()) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "You cannot add a voter type with no
                elections.");
            return "redirect:/voter-types";
        }
        if (!model.containsAttribute("voterType")) {
            model.addAttribute("voterType", new VoterType());
        }
        return "add-voter-type";
    }

    @RequestMapping(value = "/add", method = RequestMethod.
```

```
        POST)
    public String addVoterType(@Valid VoterType voterType,
            BindingResult result, RedirectAttributes
                redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                voterType", result);
            redirectAttributes.addFlashAttribute("voterType",
                voterType);
            return "redirect:add";
        } else {
            voterTypeService.save(voterType);
            redirectAttributes.addFlashAttribute("
                successMessage", "Successfully added " +
                voterType.getName());
            return "redirect:add";
        }
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.GET)
    public String editVoterTypePage(@PathVariable long id,
          Model model,
            RedirectAttributes redirectAttributes) {
        VoterType voterType = voterTypeService.findById(id);
        if (voterType == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "There is no voter type with that id!");
            return "redirect:/voter-types";
        }
        List<Election> voterElections = voterType.getElections
            ();
        for (Election election : voterElections) {
            if (election.isStatus()) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "A voter type in a
                    running election cannot be edited.");
                return "redirect:/voter-types";
            }
        }
        List<Election> electionList = electionService.
            getAllElections();
        model.addAttribute("electionList", electionList);
        model.addAttribute("activeSettingsVoterType", "active"
            );
        if (!model.containsAttribute("voterType")) {
            model.addAttribute("voterType", voterType);
        }
        return "edit-voter-type";
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.POST)
    public String editVoterType(@PathVariable long id,
            @Valid VoterType voterType, BindingResult result,
            RedirectAttributes redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                voterType", result);
            redirectAttributes.addFlashAttribute("voterType",
                voterType);
            return "redirect:/voter-types/edit/" + id;
        } else {
            voterTypeService.save(voterType);
            redirectAttributes.addFlashAttribute("
                successMessage", "Successfully edited " +
                voterType.getName());
            return "redirect:/voter-types/edit/" + id;
        }
    }

    @RequestMapping(value = "/delete/{id}", method =
          RequestMethod.GET)
    public String deleteVoterType(@PathVariable long id,
            RedirectAttributes redirectAttributes) {
        VoterType voterType = voterTypeService.findById(id);
        if (voterType == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                ", "There is no voter type with that id!");
            return "redirect:/voter-types";
        }
        List<Election> voterElections = voterType.getElections
            ();
        for (Election election : voterElections) {
            if (election.isStatus()) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "A voter type in a
                    running election cannot be deleted.");
                return "redirect:/voter-types";
            }
        }
        voterTypeService.delete(voterType);
        redirectAttributes.addFlashAttribute("successMessage",
            "Successfully deleted " + voterType.getName());
        return "redirect:/voter-types";
    }
}
```

3.13. VotingOfficialController.java

```
package ph.edu.upm.agila.japiscos.controller;

import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.mvc.support.
      RedirectAttributes;

import ph.edu.upm.agila.japiscos.exception.
      EmailExistsException;
import ph.edu.upm.agila.japiscos.exception.
      UsernameExistsException;
import ph.edu.upm.agila.japiscos.formbacker.UserCreateForm;
import ph.edu.upm.agila.japiscos.formbacker.UserEditForm;
import ph.edu.upm.agila.japiscos.model.Role;
import ph.edu.upm.agila.japiscos.model.User;
import ph.edu.upm.agila.japiscos.service.UserService;

@Controller
@RequestMapping("/voting-officials")
public class VotingOfficialController {

    @Autowired
    private UserService userService;

    @RequestMapping(method = RequestMethod.GET)
    public String votingOfficialsPage(Model model) {
        model.addAttribute("activeSettingsVotingOfficial", "
            active");
        List<User> votingOfficials = userService.
            getUsersByRole(Role.VOTING_OFFICIAL);
        model.addAttribute("votingOfficialList",
            votingOfficials);
        return "voting-officials";
    }

    @RequestMapping(value = "/add", method = RequestMethod.GET
        )
    public String addVotingOfficialPage(Model model) {
        model.addAttribute("activeSettingsVotingOfficial", "
            active");
        if (!model.containsAttribute("userCreateForm")) {
            UserCreateForm userCreateForm = new UserCreateForm
                ();
            userCreateForm.setRole(Role.VOTING_OFFICIAL);
            model.addAttribute("userCreateForm",
                userCreateForm);
        }
        return "add-voting-official";
    }

    @RequestMapping(value = "/add", method = RequestMethod.
        POST)
    public String addUser(@Valid UserCreateForm userCreateForm
        ,
            BindingResult result, RedirectAttributes
                redirectAttributes) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                springframework.validation.BindingResult.
                userCreateForm", result);
            redirectAttributes.addFlashAttribute("
                userCreateForm", userCreateForm);
            return "redirect:add";
        } else {
            try {
                userService.add(userCreateForm);
            } catch (UsernameExistsException e) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "The username " + e.
                    getUsername() + " exists!");
                return "redirect:add";
            } catch (EmailExistsException e) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "The email " + e.getEmail
                    () + " exists!");
                return "redirect:add";
            }
            redirectAttributes.addFlashAttribute("
                successMessage", "Successfully added " +
                userCreateForm.getUsername());
            return "redirect:add";
        }
    }

    @RequestMapping(value = "/edit/{id}", method =
          RequestMethod.GET)
    public String editUserPage(@PathVariable long id,
            RedirectAttributes redirectAttributes, Model model
                ) {
        model.addAttribute("activeSettingsVotingOfficial", "
            active");
        if (!model.containsAttribute("userEditForm")) {
            User user = userService.findById(id);
            if (user == null) {
                redirectAttributes.addFlashAttribute("
                    errorMessage", "There is no user with
                    that id!");
                return "redirect:/voting-officials";
            }
            UserEditForm userEditForm = new UserEditForm();
            userEditForm.setId(user.getId());
```

```
            userEditForm.setUsername(user.getUsername());
            userEditForm.setRole(user.getRole());
            userEditForm.setEmail(user.getEmail());
            model.addAttribute("userEditForm", userEditForm);
        }
        return "edit-voting-official";
    }

    @RequestMapping(value = "/edit/{id}", method =
            RequestMethod.POST)
    public String editUser(@PathVariable long id,
            @Valid UserEditForm userEditForm, BindingResult
                  result,
            RedirectAttributes redirectAttributes, Model model
                  ) {
        if (result.hasErrors()) {
            redirectAttributes.addFlashAttribute("org.
                  springframework.validation.BindingResult.
                  userEditForm", result);
            redirectAttributes.addFlashAttribute("userEditForm
                  ", userEditForm);
            return "redirect:/voting-officials/edit/" + id;
        } else {
            User newUser;
            try {
                newUser = userService.edit(userEditForm);
            } catch (UsernameExistsException e) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "The username " + e.
                      getUsername() + " exists!");
                return "redirect:/voting-officials/edit/" + id
                      ;
            } catch (EmailExistsException e) {
                redirectAttributes.addFlashAttribute("
                      errorMessage", "The email " + e.getEmail
                      () + " exists!");
                return "redirect:/voting-officials/edit/" + id
                      ;
            }
            redirectAttributes.addFlashAttribute("
                  successMessage", "Successfully edited " +
                  newUser.getUsername());
            return "redirect:/voting-officials/edit/" + id;
        }
    }

    @RequestMapping(value = "/delete/{id}", method =
            RequestMethod.GET)
    public String deleteUser(@PathVariable long id,
            RedirectAttributes redirectAttributes) {
        User user = userService.findById(id);
        if (user == null) {
            redirectAttributes.addFlashAttribute("errorMessage
                  ", "There is no user with that id!");
            return "redirect:/voting-officials";
        }
        String username = user.getUsername();
        userService.delete(user);
        redirectAttributes.addFlashAttribute("successMessage",
                "Successfully deleted " + username);
        return "redirect:/voting-officials";
    }
}
```

4. Services

4.1. AbstainService.java

```
package ph.edu.upm.agila.japiscos.service;

import ph.edu.upm.agila.japiscos.model.Abstain;
import ph.edu.upm.agila.japiscos.model.Position;

public interface AbstainService {

    Abstain findByPosition(Position position);

}
```

4.2. AbstainServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.
      Transactional;

import ph.edu.upm.agila.japiscos.dao.AbstainDao;
import ph.edu.upm.agila.japiscos.model.Abstain;
import ph.edu.upm.agila.japiscos.model.Position;

@Service("abstainService")
@Transactional
public class AbstainServiceImpl implements AbstainService {

    @Autowired
    private AbstainDao abstainDao;

    @Override
    public Abstain findByPosition(Position position) {
        return abstainDao.findByPosition(position);
    }

}
```

4.3. BatchInsertService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.io.IOException;

import ph.edu.upm.agila.japiscos.exception.
      EmailExistsException;
import ph.edu.upm.agila.japiscos.exception.
      IncorrectCSVFormatException;
import ph.edu.upm.agila.japiscos.exception.
      InvalidEmailException;
import ph.edu.upm.agila.japiscos.exception.
      UsernameExistsException;
import ph.edu.upm.agila.japiscos.formbacker.ImportVoterForm;

public interface BatchInsertService {

    void processFile(ImportVoterForm importVoterForm) throws
          IOException, InvalidEmailException,
            UsernameExistsException, EmailExistsException,
                  IncorrectCSVFormatException;

}
```

4.4. BatchInsertServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.io.IOException;
import java.io.InputStreamReader;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.List;
import java.util.regex.Pattern;

import javax.persistence.EntityManagerFactory;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.
      BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import ph.edu.upm.agila.japiscos.dao.UserDao;
import ph.edu.upm.agila.japiscos.dao.VoterDao;
import ph.edu.upm.agila.japiscos.exception.
      EmailExistsException;
import ph.edu.upm.agila.japiscos.exception.
      IncorrectCSVFormatException;
import ph.edu.upm.agila.japiscos.exception.
      InvalidEmailException;
import ph.edu.upm.agila.japiscos.exception.
      UsernameExistsException;
import ph.edu.upm.agila.japiscos.formbacker.ImportVoterForm;
import ph.edu.upm.agila.japiscos.formbacker.VoterImport;
import ph.edu.upm.agila.japiscos.formbacker.
      VoterLoginCredentials;
import ph.edu.upm.agila.japiscos.model.Role;
import ph.edu.upm.agila.japiscos.model.User;
import ph.edu.upm.agila.japiscos.model.Voter;
import au.com.bytecode.opencsv.CSVReader;
import au.com.bytecode.opencsv.bean.
      ColumnPositionMappingStrategy;
import au.com.bytecode.opencsv.bean.CsvToBean;

@Service("batchInsertService")
public class BatchInsertServiceImpl implements
      BatchInsertService {

    @Autowired
    private SendMailService sendMailService;

    @Autowired
    private UserDao userDao;

    @Autowired
    private VoterDao voterDao;

    @Autowired
    private HashService hashService;

    @Autowired
    private PasswordService passwordService;

    private SessionFactory sessionFactory;

    @Autowired
    public BatchInsertServiceImpl(EntityManagerFactory factory
          ) {
        if (factory.unwrap(SessionFactory.class) == null) {
            throw new NullPointerException("factory is not a
                  hibernate factory");
        }
        this.sessionFactory = factory.unwrap(SessionFactory.
              class);
    }

    @Override
    @SuppressWarnings({ "rawtypes", "unchecked" })
    public void processFile(ImportVoterForm form) throws
```

```
   IOException,
    InvalidEmailException, UsernameExistsException,
    EmailExistsException, IncorrectCSVFormatException
          {
CsvToBean csv = new CsvToBean();
CSVReader csvReader = new CSVReader(new
      InputStreamReader(form.getFile().getInputStream
      ()));
List<VoterImport> voters = csv.parse(setColumnMapping
      (), csvReader);
List<VoterLoginCredentials> importedVoters = new
      ArrayList<>();

Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();
for (int i = 0; i < voters.size(); i++) {
    VoterImport voterImport = voters.get(i);
    if (voterImport.getUsername() == null ||
          voterImport.getUsername().isEmpty()
            || voterImport.getEmail() == null ||
                voterImport.getEmail().isEmpty()) {
        throw new IncorrectCSVFormatException();
    }

    Pattern ptr = Pattern
        .compile("^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0
              -9-]+)*@[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)
              *(\\.[A-Za-z]{2,})$");

    if (!ptr.matcher(voterImport.getEmail()).matches()
          ) {
        throw new InvalidEmailException(voterImport.
              getUsername(), voterImport.getEmail());
    }

    boolean voterExist = voterExists(voterImport);
    User user = null;
    if (userDao.findByUsername(voterImport.getUsername
          ()) != null) {
        user = userDao.findByUsername(voterImport.
              getUsername());
        if (user.getRole() != Role.VOTER) {
            throw new UsernameExistsException(user.
                  getUsername());
        }
    } else if (userDao.findByEmail(voterImport.
          getEmail()) != null) {
        user = userDao.findByEmail(voterImport.
              getEmail());
        if (user.getRole() != Role.VOTER) {
            throw new EmailExistsException(user.
                  getEmail());
        }
    } else {
        user = new User();
    }
    user.setUsername(voterImport.getUsername());
    String generatedPassword = passwordService.
          generatePassword(10);
    user.setPasswordHash(new BCryptPasswordEncoder()
          .encode(generatedPassword));
    user.setEmail(voterImport.getEmail());
    user.setRole(Role.VOTER);
    session.save(user);

    Voter voter = null;
    if (!voterExist) {
        voter = new Voter();
    } else {
        voter = voterDao.findVoterByUser(user);
        voter.setVoted(false);
        voter.setPublished(false);
        voter.setActive(true);
    }
    voter.setUser(user);
    voter.setVoterType(form.getVoterType());
    String salt = "";
    try {
        salt = hashService.getSalt();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    voter.setSalt(salt);
    session.save(voter);
    VoterLoginCredentials newVoter = new
          VoterLoginCredentials();
    newVoter.setUsername(voterImport.getUsername());
    newVoter.setGeneratedPassword(generatedPassword);
    newVoter.setEmail(voterImport.getEmail());
    importedVoters.add(newVoter);
    if (i % 20 == 0) {
        session.flush();
        session.clear();
    }

}
tx.commit();
session.close();
emailVoters(importedVoters);
}

private void emailVoters(List<VoterLoginCredentials>
      importedVoters) {
    for (VoterLoginCredentials newVoter : importedVoters)
          {
        String mailMessage = "Please log in to iVote with
              the following credentials: \n\n"
                + "Username: " + newVoter.getUsername()
                + "\nPassword: " + newVoter.
                      getGeneratedPassword() + "\n";
        sendMailService.sendMail("ivote.sharemind@gmail.
              com", newVoter.getEmail(),
            "[iVote Sharemind] Login Credentials",
                mailMessage);
    }
}

@SuppressWarnings({ "rawtypes", "unchecked" })
private static ColumnPositionMappingStrategy
      setColumnMapping() {
    ColumnPositionMappingStrategy strategy = new
          ColumnPositionMappingStrategy();
    strategy.setType(VoterImport.class);
    String[] columns = new String[] { "username", "email"
          };
    strategy.setColumnMapping(columns);
    return strategy;
}

private boolean voterExists(VoterImport voterImport) {
    if (userDao.findByUsername(voterImport.getUsername())
          != null
        || userDao.findByEmail(voterImport.getEmail()) !=
              null) {
        return true;
    } else {
        return false;
    }
}

}
```

4.5. CandidateService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.List;

import ph.edu.upm.agila.japiscos.formbacker.CandidateBacker;
import ph.edu.upm.agila.japiscos.formbacker.Vote;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Voter;

public interface CandidateService {

    Candidate findById(long id);

    void add(CandidateBacker candidateBacker);

    void edit(CandidateBacker candidateBacker);

    void delete(Candidate candidate);

    List<Candidate> getCandidatesByElection(Election election)
          ;

    List<Candidate> getCandidatesByElectionWithChildren(
          Election election);

    List<Candidate> getVotedCandidates(List<Vote> votes, Voter
          voter);

}
```

4.6. CandidateServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.
      Transactional;

import ph.edu.upm.agila.japiscos.dao.CandidateDao;
import ph.edu.upm.agila.japiscos.dao.ElectionDao;
import ph.edu.upm.agila.japiscos.dao.PartyDao;
import ph.edu.upm.agila.japiscos.dao.PositionDao;
import ph.edu.upm.agila.japiscos.formbacker.CandidateBacker;
import ph.edu.upm.agila.japiscos.formbacker.Vote;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Party;
import ph.edu.upm.agila.japiscos.model.Position;
import ph.edu.upm.agila.japiscos.model.Voter;

@Service("candidateService")
@Transactional
public class CandidateServiceImpl implements CandidateService
      {

    @Autowired
    private CandidateDao candidateDao;

    @Autowired
    private PositionDao positionDao;

    @Autowired
```

```
    private PartyDao partyDao;

    @Autowired
    private ElectionDao electionDao;

    @Autowired
    private HashService hashService;

    @Override
    public Candidate findById(long id) {
        return candidateDao.findById(id);
    }

    @Override
    public void add(CandidateBacker candidateBacker) {
        Candidate candidate = new Candidate();
        candidate.setFirstName(candidateBacker.getFirstName())
                ;
        candidate.setLastName(candidateBacker.getLastName());
        Election election = electionDao.findById(
                candidateBacker.getElectionId());
        Party party = partyDao.findById(candidateBacker.
                getPartyId());
        Position position = positionDao.findById(
                candidateBacker.getPositionId());
        candidate.setElection(election);
        candidate.setParty(party);
        candidate.setPosition(position);
        save(candidate);
    }

    @Override
    public void edit(CandidateBacker candidateBacker) {
        Candidate candidate = candidateDao.findById(
                candidateBacker.getId());
        Election election = electionDao.findById(
                candidateBacker.getElectionId());
        Position position = positionDao.findById(
                candidateBacker.getPositionId());
        Party party = partyDao.findById(candidateBacker.
                getPartyId());
        candidate.setElection(election);
        candidate.setPosition(position);
        candidate.setParty(party);
        candidate.setFirstName(candidateBacker.getFirstName())
                ;
        candidate.setLastName(candidateBacker.getLastName());
        save(candidate);
    }

    private Candidate save(Candidate candidate) {
        return candidateDao.save(candidate);
    }

    @Override
    public void delete(Candidate candidate) {
        candidateDao.delete(candidate);
    }

    @Override
    public List<Candidate> getCandidatesByElection(Election
          election) {
        return candidateDao.
              getCandidatesByElectionOrderByLastNameAsc(
              election);
    }

    @Override
    public List<Candidate> getCandidatesByElectionWithChildren
          (Election election) {
        List<Candidate> candidates = new ArrayList<>(
              candidateDao
            .getCandidatesByElectionOrderByLastNameAsc(
                  election));
        if (election.getParentElectionId() == 0) {
            List<Election> childrenElections = electionDao.
                  getElectionsByParentElectionId(election.
                  getId());
            for (Election childElection : childrenElections) {
                if (!childElection.isStatus()) {
                    candidates.addAll(candidateDao.
                          getCandidatesByElectionOrderByLastNameAsc
                          (childElection));
                }
            }
        }
        return candidates;
    }

    @Override
    public List<Candidate> getVotedCandidates(List<Vote> votes
          , Voter voter) {
        List<Candidate> candidates = new ArrayList<>();
        for (Vote vote : votes) {
            long[] candidateIds = vote.getCandidateIds();
            for (int i = 0; i < candidateIds.length; i++) {
                Candidate candidate;
                if (candidateIds[i] != 0) {
                    candidate = candidateDao.findById(
                          candidateIds[i]);
                } else {
                    candidate = new Candidate();
                    candidate.setId(0);
                    Position position = positionDao.findById(
                          vote.getPositionId());
                    candidate.setPosition(position);
                    candidate.setElection(position.getElection
                          ());
                }
                String hash = hashService.getHash(String.
                      valueOf(candidate.getId()), voter.
                      getSalt());
                candidate.setHash(hash);
                candidates.add(candidate);
            }
        }
        return candidates;
    }

}
```

4.7. CustomUserDetailsService.java

```
package ph.edu.upm.agila.japiscos.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.
      UserDetailsService;
import org.springframework.security.core.userdetails.
      UsernameNotFoundException;
import org.springframework.stereotype.Service;

import ph.edu.upm.agila.japiscos.model.CurrentUser;
import ph.edu.upm.agila.japiscos.model.User;

@Service
public class CustomUserDetailsService implements
  UserDetailsService {

    @Autowired
    private UserService userService;

    @Override
    public CurrentUser loadUserByUsername(String username)
          throws UsernameNotFoundException {
        User user = userService.findByUsername(username);
        return new CurrentUser(user);
    }

}
```

4.8. ElectionService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.List;

import ph.edu.upm.agila.japiscos.model.Election;

public interface ElectionService {

    Election save(Election election);

    Election findById(long id);

    void edit(Election election);

    void delete(Election election);

    List<Election> getAllElections();

    List<Election> getElectionsByLevel();

    void changeStatus(Election election);

    boolean electionInUse(Election election);

    List<Election> getElections(long id);

    List<Election> getElectionsByStatus(boolean status);

    List<Election> getRunningElections(List<Election>
          elections);

    List<Election> getFinishedElections(List<Election>
          elections);

}
```

4.9. ElectionServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.
      Transactional;

import ph.edu.upm.agila.japiscos.dao.CandidateDao;
import ph.edu.upm.agila.japiscos.dao.ElectionDao;
import ph.edu.upm.agila.japiscos.dao.PartyDao;
import ph.edu.upm.agila.japiscos.dao.PositionDao;
import ph.edu.upm.agila.japiscos.model.Election;

@Service("electionService")
```

```
@Transactional
public class ElectionServiceImpl implements ElectionService {

    @Autowired
    private ElectionDao electionDao;

    @Autowired
    private PartyDao partyDao;

    @Autowired
    private PositionDao positionDao;

    @Autowired
    private CandidateDao candidateDao;

    @Override
    public Election save(Election election) {
        return electionDao.save(election);
    }

    @Override
    public Election findById(long id) {
        return electionDao.findById(id);
    }

    @Override
    public void edit(Election election) {
        Election electionToUpdate = findById(election.getId())
                ;
        electionToUpdate.setName(election.getName());
        electionToUpdate.setParentElectionId(election.
                getParentElectionId());
        save(electionToUpdate);
    }

    @Override
    public void delete(Election election) {
        electionDao.delete(election);
    }

    @Override
    public List<Election> getAllElections() {
        return (List<Election>) electionDao.findAll();
    }

    @Override
    public List<Election> getElectionsByLevel() {
        List<Election> parentElections = getElections(0);
        List<Election> result = new ArrayList<>();
        for (Election parentElection : parentElections) {
            result.add(parentElection);
            List<Election> childElections = getElections(
                    parentElection.getId());
            if (!childElections.isEmpty()) {
                for (Election childElection : childElections)
                        {
                    result.add(childElection);
                }
            }
        }
        return result;
    }

    @Override
    public void changeStatus(Election election) {
        boolean electionStatus = election.isStatus();

        if (!electionStatus) {
            election.setStartDate(new Date());
        } else {
            election.setEndDate(new Date());
        }
        if (election.getParentElectionId() == 0) {
            List<Election> childElections = getElections(
                    election.getId());
            if (!childElections.isEmpty()) {
                for (Election childElection : childElections)
                        {
                    if (!electionStatus) {
                        childElection.setStartDate(new Date())
                                ;
                    } else {
                        childElection.setEndDate(new Date());
                    }
                    childElection.setStatus(!electionStatus);
                    save(childElection);
                }
                election.setStatus(!electionStatus);
                save(election);
            } else {
                election.setStatus(!electionStatus);
                save(election);
            }
        } else {
            election.setStatus(!electionStatus);
            save(election);
        }
    }

    @Override
    public boolean electionInUse(Election election) {
        List<Election> elections = new ArrayList<>();
        elections.add(election);
        boolean electionUsedByParty = !partyDao.
                getPartiesByElections(elections).isEmpty();
        boolean electionUsedByPosition = !positionDao.
                getPositionsByElection(election).isEmpty();
        boolean electionUsedByCandidate = !candidateDao.
                getCandidatesByElection(election).isEmpty();
        return electionUsedByParty || electionUsedByPosition
                || electionUsedByCandidate;
    }

    @Override
    public List<Election> getElections(long id) {
        return electionDao.getElectionsByParentElectionId(id);
    }

    @Override
    public List<Election> getElectionsByStatus(boolean status)
            {
        return electionDao.getElectionsByStatus(status);
    }

    @Override
    public List<Election> getRunningElections(List<Election>
            elections) {
        List<Election> runningElections = new ArrayList<>();
        for (Election election : elections) {
            if (election.getParentElectionId() == 0) {
                Election parent = election;
                List<Election> childElections = getElections(
                        parent.getId());
                if (!childElections.isEmpty()) {
                    for (Election childElection :
                            childElections) {
                        if (childElection.isStatus() && parent
                                .isStatus()) {
                            if (!runningElections.contains(
                                    election)) {
                                runningElections.add(election)
                                        ;
                            }
                            if (elections.contains(
                                    childElection)) {
                                runningElections.add(
                                        childElection);
                            }
                        }
                    }
                } else {
                    if (parent.isStatus()) {
                        runningElections.add(election);
                    }
                }
            } else {
                if (election.isStatus() && !runningElections.
                        contains(election)) {
                    runningElections.add(election);
                }
            }
        }
        return runningElections;
    }

    @Override
    public List<Election> getFinishedElections(List<Election>
            elections) {
        List<Election> finishedElections = new ArrayList<>();
        for (Election election : elections) {
            if (!election.isStatus()) {
                finishedElections.add(election);
            }
        }
        return finishedElections;
    }
}
```

4.10. HashService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.security.NoSuchAlgorithmException;

public interface HashService {

    String getSalt() throws NoSuchAlgorithmException;

    String getHash(String text, String salt);

}
```

4.11. HashServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

import org.springframework.stereotype.Service;

@Service("hashService")
public class HashServiceImpl implements HashService {

    @Override
    public String getSalt() throws NoSuchAlgorithmException {
        SecureRandom sr = SecureRandom.getInstance("SHA1PRNG")
                ;
        byte[] salt = new byte[16];
```

```
        sr.nextBytes(salt);
        return salt.toString();
    }

    @Override
    public String getHash(String text, String salt) {
        MessageDigest md = null;
        try {
            md = MessageDigest.getInstance("MD5");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
```

4.12. PartyService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.List;

import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Party;

public interface PartyService {

    Party save(Party party);
```

4.13. PartyServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.
      Transactional;

import ph.edu.upm.agila.japiscos.dao.CandidateDao;
import ph.edu.upm.agila.japiscos.dao.PartyDao;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Party;

@Service("partyService")
@Transactional
public class PartyServiceImpl implements PartyService {

    @Autowired
    private PartyDao partyDao;

    @Autowired
    private CandidateDao candidateDao;

    @Override
    public Party save(Party party) {
        return partyDao.save(party);
```

4.14. PasswordService.java

```
package ph.edu.upm.agila.japiscos.service;

public interface PasswordService {
```

4.15. PasswordServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.security.SecureRandom;

import org.springframework.stereotype.Service;

@Service("passwordService")
public class PasswordServiceImpl implements PasswordService {

    @Override
    public String generatePassword(int len) {
        String AB = "0123456789
```

4.16. PositionService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.List;

import ph.edu.upm.agila.japiscos.formbacker.PositionBacker;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Position;

public interface PositionService {

    Position findById(long id);

    void add(PositionBacker positionBacker);
```

4.17. PositionServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.
      Transactional;

import ph.edu.upm.agila.japiscos.dao.AbstainDao;
import ph.edu.upm.agila.japiscos.dao.CandidateDao;
import ph.edu.upm.agila.japiscos.dao.ElectionDao;
import ph.edu.upm.agila.japiscos.dao.PositionDao;
import ph.edu.upm.agila.japiscos.formbacker.PositionBacker;
```

```
        md.update(salt.getBytes());
        byte[] digest = md.digest(text.getBytes());
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < digest.length; i++) {
            sb.append(Integer.toString((digest[i] & 0xff) + 0
                  x100, 16).substring(1));
        }
        return sb.toString();
    }
}
```

```
    Party findById(long id);

    void delete(Party party);

    List<Party> getPartiesByElection(Election election);

    boolean partyInUse(Party party);
}
```

```
    }

    @Override
    public Party findById(long id) {
        return partyDao.findById(id);
    }

    @Override
    public void delete(Party party) {
        partyDao.delete(party);
    }

    @Override
    public List<Party> getPartiesByElection(Election election)
          {
        List<Election> elections = new ArrayList<>();
        elections.add(election);
        return partyDao.getPartiesByElectionsOrderByNameAsc(
              elections);
    }

    @Override
    public boolean partyInUse(Party party) {
        return candidateDao.getCandidatesByParty(party).
              isEmpty() ? false : true;
    }
}
```

```
    String generatePassword(int len);

}
```

```
            ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
            ";
        SecureRandom rnd = new SecureRandom();
        StringBuilder sb = new StringBuilder(len);
        for (int i = 0; i < len; i++) {
            sb.append(AB.charAt(rnd.nextInt(AB.length())));
        }
        return sb.toString();
    }

}
```

```
    void edit(PositionBacker positionBacker);

    void delete(Position position);

    List<Position> getPositionsByElection(Election election);

    List<Position> getPositionsByElectionWithParent(Election
          election);

    boolean positionInUse(Position position);

    List<Position> getCandidatePositions(Election election);

}
```

```
import ph.edu.upm.agila.japiscos.model.Abstain;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Position;

@Service("positionService")
@Transactional
public class PositionServiceImpl implements PositionService {

    @Autowired
    private PositionDao positionDao;

    @Autowired
    private ElectionDao electionDao;
```

```
    @Autowired
    private CandidateDao candidateDao;

    @Autowired
    private AbstainDao abstainDao;

    private Position save(Position position) {
        return positionDao.save(position);
    }

    @Override
    public Position findById(long id) {
        return positionDao.findById(id);
    }

    @Override
    public void add(PositionBacker positionBacker) {
        Position position = new Position();
        position.setName(positionBacker.getName());
        position.setMaxElected(positionBacker.getMaxElected())
              ;
        position.setOrdinality(positionBacker.getOrdinality())
              ;
        position.setAbstain(positionBacker.isAbstain());
        Election election = electionDao.findById(
              positionBacker.getElectionId());
        position.setElection(election);
        Position positionAdded = save(position);
        if (positionBacker.isAbstain()) {
            Abstain abstain = new Abstain();
            abstain.setPosition(positionAdded);
            abstainDao.save(abstain);
        }
    }

    @Override
    public void edit(PositionBacker positionBacker) {
        Position positionToUpdate = findById(positionBacker.
              getId());
        Election election = electionDao.findById(
              positionBacker.getElectionId());
        positionToUpdate.setName(positionBacker.getName());
        positionToUpdate.setElection(election);
        positionToUpdate.setMaxElected(positionBacker.
              getMaxElected());
        positionToUpdate.setOrdinality(positionBacker.
              getOrdinality());
        positionToUpdate.setAbstain(positionBacker.isAbstain()
              );
        if (positionToUpdate.isAbstain() && !positionBacker.
              isAbstain()) {
            Abstain abstain = abstainDao.findByPosition(
                  positionToUpdate);
            abstainDao.delete(abstain);
        }
        Position positionUpdated = save(positionToUpdate);
        if (!positionToUpdate.isAbstain() && positionBacker.
              isAbstain()) {
            Abstain abstain = new Abstain();
            abstain.setPosition(positionUpdated);
            abstainDao.save(abstain);
        }
    }

    @Override
    public void delete(Position position) {
        if (position.isAbstain()) {
            abstainDao.delete(abstainDao.findByPosition(
                  position));
        }
        positionDao.delete(position);
    }

    @Override
    public List<Position> getPositionsByElection(Election
          election) {
        return positionDao.
              getPositionsByElectionOrderByOrdinalityAsc(
              election);
    }

    @Override
    public List<Position> getPositionsByElectionWithParent(
          Election election) {
        if (election.getParentElectionId() != 0) {
            Election parentElection = electionDao.findById(
                  election.getParentElectionId());
            List<Position> positions = getPositionsByElection(
                  parentElection);
            positions.addAll(getPositionsByElection(election))
                  ;
            return positions;
        } else {
            List<Position> positions = getPositionsByElection(
                  election);
            return positions;
        }
    }

    @Override
    public boolean positionInUse(Position position) {
        return (!candidateDao.getCandidateByPosition(position)
              .isEmpty());
    }

    @Override
    public List<Position> getCandidatePositions(Election
          election) {
        List<Candidate> candidates = candidateDao
              .getCandidatesByElectionOrderByLastNameAsc(
                    election);
        List<Position> positions = getPositionsByElection(
              election);
        List<Position> candidatePositions = new ArrayList<>();
        for (Candidate candidate : candidates) {
            candidatePositions.add(candidate.getPosition());
        }
        for (int i = 0; i < positions.size(); i++) {
            if (!candidatePositions.contains(positions.get(i))
                  ) {
                positions.remove(i);
            }
        }
        return positions;
    }

}
```

4.18. SendMailService.java

```
package ph.edu.upm.agila.japiscos.service;

public interface SendMailService {

    void sendMail(String from, String to, String subject,
          String msg);

}
```

4.19. SendMailServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailSender;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Service;

@Service("sendMailService")
public class SendMailServiceImpl implements SendMailService {

    @Autowired
    private MailSender mailSender;

    @Async
    @Override
    public void sendMail(String from, String to, String
          subject, String msg) {
        SimpleMailMessage simpleMailMessage = new
              SimpleMailMessage();
        simpleMailMessage.setFrom(from);
        simpleMailMessage.setTo(to);
        simpleMailMessage.setSubject(subject);
        simpleMailMessage.setText(msg);
        mailSender.send(simpleMailMessage);
    }

}
```

4.20. UserService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.List;

import ph.edu.upm.agila.japiscos.exception.
      EmailExistsException;
import ph.edu.upm.agila.japiscos.exception.
      UsernameExistsException;
import ph.edu.upm.agila.japiscos.formbacker.ChangePasswordForm
      ;
import ph.edu.upm.agila.japiscos.formbacker.UserCreateForm;
import ph.edu.upm.agila.japiscos.formbacker.UserEditForm;
import ph.edu.upm.agila.japiscos.model.Role;
import ph.edu.upm.agila.japiscos.model.User;

public interface UserService {

    User findByUsername(String username);

    User findById(long id);

    User add(UserCreateForm form) throws
          UsernameExistsException, EmailExistsException;

    User edit(UserEditForm form) throws
          UsernameExistsException, EmailExistsException;

    void delete(User user);

    List<User> getUsersByRole(Role votingOfficial);

    void changePassword(ChangePasswordForm changePasswordForm)
          ;

}
```

4.21. UserServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.
      BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.
      Transactional;

import ph.edu.upm.agila.japiscos.dao.UserDao;
import ph.edu.upm.agila.japiscos.exception.
      EmailExistsException;
import ph.edu.upm.agila.japiscos.exception.
      UsernameExistsException;
import ph.edu.upm.agila.japiscos.formbacker.ChangePasswordForm
      ;
import ph.edu.upm.agila.japiscos.formbacker.UserCreateForm;
import ph.edu.upm.agila.japiscos.formbacker.UserEditForm;
import ph.edu.upm.agila.japiscos.model.Role;
import ph.edu.upm.agila.japiscos.model.User;

@Service("userService")
@Transactional
public class UserServiceImpl implements UserService {

    @Autowired
    private UserDao userDao;

    @Autowired
    private VoterService voterService;

    @Autowired
    private SendMailService sendMailService;

    @Autowired
    private PasswordService passwordService;

    @Override
    public User findByUsername(String username) {
        return userDao.findByUsername(username);
    }

    @Override
    public User findById(long id) {
        return userDao.findById(id);
    }

    @Override
    public User add(UserCreateForm form) throws
          UsernameExistsException,
            EmailExistsException {
        if (userDao.findByUsername(form.getUsername()) != null
              ) {
            throw new UsernameExistsException(form.getUsername
                  ());
        }
        if (userDao.findByEmail(form.getEmail()) != null) {
            throw new EmailExistsException(form.getEmail());
        }
        User user = new User();
        user.setUsername(form.getUsername());
        String generatedPassword = passwordService.
              generatePassword(10);
        user.setPasswordHash(new BCryptPasswordEncoder().
              encode(generatedPassword));
        user.setRole(form.getRole());
        user.setEmail(form.getEmail());
        User savedUser = userDao.save(user);
        String mailMessage = "Please log in to iVote with the
              following credentials: \n\n"
            + "Username: " + form.getUsername()
            + "\nPassword: " + generatedPassword + "\n";
        sendMailService.sendMail("ivote.sharemind@gmail.com",
              form.getEmail(),
            "[iVote Sharemind] Login Credentials", mailMessage
                  );
        return savedUser;
    }

    @Override
    public User edit(UserEditForm form) throws
          UsernameExistsException,
            EmailExistsException {
        User checkUser = userDao.findByUsername(form.
              getUsername());
        User checkEmail = userDao.findByEmail(form.getEmail())
              ;
        if (checkUser != null) {
            if (checkUser.getId() != form.getId()) {
                throw new UsernameExistsException(form.
                      getUsername());
            }
        }
        if (checkEmail != null) {
            if (checkEmail.getId() != form.getId()) {
                throw new EmailExistsException(form.getEmail()
                      );
            }
        }
        User user = userDao.findById(form.getId());
        user.setUsername(form.getUsername());
        user.setRole(form.getRole());
        user.setEmail(form.getEmail());
        return userDao.save(user);
    }

    @Override
    public void delete(User user) {
        userDao.delete(user);
    }

    @Override
    public List<User> getUsersByRole(Role votingOfficial) {
        return userDao.getUsersByRole(votingOfficial);
    }

    @Override
    public void changePassword(ChangePasswordForm
          changePasswordForm) {
        User user = userDao.findById(changePasswordForm.getId
              ());
        user.setPasswordHash(new BCryptPasswordEncoder().
              encode(changePasswordForm.getNewPassword()));
        userDao.save(user);
        String mailMessage = "Your password has changed. \nNew
              login credentials: \n\n"
            + "Username: " + user.getUsername()
            + "\nPassword: " + changePasswordForm.
                  getNewPassword() + "\n";
        sendMailService.sendMail("ivote.sharemind@gmail.com",
              user.getEmail(),
            "[iVote Sharemind] Changed Password Details",
                  mailMessage);
    }

}
```

4.22. VoterService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.List;

import ph.edu.upm.agila.japiscos.exception.
      EmailExistsException;
import ph.edu.upm.agila.japiscos.exception.SharemindException;
import ph.edu.upm.agila.japiscos.exception.
      UsernameExistsException;
import ph.edu.upm.agila.japiscos.formbacker.VoterCreateForm;
import ph.edu.upm.agila.japiscos.formbacker.VoterEditForm;
import ph.edu.upm.agila.japiscos.model.User;
import ph.edu.upm.agila.japiscos.model.Voter;
import ph.edu.upm.agila.japiscos.model.VoterType;

public interface VoterService {

    Voter findById(long voterId);

    Voter findVoterByUser(User user);

    void add(VoterCreateForm voterForm) throws
          UsernameExistsException, EmailExistsException;

    void edit(VoterEditForm voterEditForm);

    void delete(Voter voter);

    List<Voter> getAllVoters();

    List<Voter> getVotersVoted();

    List<Voter> checkRecordedVoters(List<Voter> voters) throws
          SharemindException;

    Voter getVoterByAlias(String alias);

    List<Voter> getPublishedVoters();

    List<Voter> getAllActiveVoters();

    List<Voter> getVotersByVoterTypes(List<VoterType>
          voterTypes);

}
```

4.23. VoterServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.
      BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.
      Transactional;

import ph.edu.upm.agila.japiscos.dao.CandidateDao;
```

```
import ph.edu.upm.agila.japiscos.dao.ElectionDao;
import ph.edu.upm.agila.japiscos.dao.PositionDao;
import ph.edu.upm.agila.japiscos.dao.UserDao;
import ph.edu.upm.agila.japiscos.dao.VoterDao;
import ph.edu.upm.agila.japiscos.dao.VoterTypeDao;
import ph.edu.upm.agila.japiscos.exception.
      EmailExistsException;
import ph.edu.upm.agila.japiscos.exception.SharemindException;
import ph.edu.upm.agila.japiscos.exception.
      UsernameExistsException;
import ph.edu.upm.agila.japiscos.formbacker.VoterCreateForm;
import ph.edu.upm.agila.japiscos.formbacker.VoterEditForm;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Position;
import ph.edu.upm.agila.japiscos.model.Role;
import ph.edu.upm.agila.japiscos.model.User;
import ph.edu.upm.agila.japiscos.model.Voter;
import ph.edu.upm.agila.japiscos.model.VoterType;

@Service("voterService")
@Transactional
public class VoterServiceImpl implements VoterService {

    @Autowired
    private VoterDao voterDao;

    @Autowired
    private UserDao userDao;

    @Autowired
    private PositionDao positionDao;

    @Autowired
    private CandidateDao candidateDao;

    @Autowired
    private CandidateService candidateService;

    @Autowired
    private SendMailService sendMailService;

    @Autowired
    private ElectionDao electionDao;

    @Autowired
    private VoterTypeDao voterTypeDao;

    @Autowired
    private HashService hashService;

    @Autowired
    private PasswordService passwordService;

    @Override
    public Voter findById(long voterId) {
        return voterDao.findById(voterId);
    }

    @Override
    public Voter findVoterByUser(User user) {
        return voterDao.findVoterByUser(user);
    }

    @Override
    public void add(VoterCreateForm voterForm) throws
          UsernameExistsException,
            EmailExistsException {
        boolean voterExist = voterExists(voterForm);
        User user = null;
        if (userDao.findByUsername(voterForm.getUsername()) !=
               null) {
            user = userDao.findByUsername(voterForm.
                  getUsername());
            if (user.getRole() != Role.VOTER) {
                throw new UsernameExistsException(user.
                      getUsername());
            }
        } else if (userDao.findByEmail(voterForm.getEmail())
              != null) {
            user = userDao.findByEmail(voterForm.getEmail());
            if (user.getRole() != Role.VOTER) {
                throw new EmailExistsException(user.getEmail()
                      );
            }
        } else {
            user = new User();
        }
        user.setUsername(voterForm.getUsername());
        String generatedPassword = passwordService.
              generatePassword(10);
        user.setPasswordHash(new BCryptPasswordEncoder()
              .encode(generatedPassword));
        user.setEmail(voterForm.getEmail());
        user.setRole(Role.VOTER);
        User savedUser = userDao.save(user);
        String mailMessage = "Please log in to iVote with the
              following credentials: \n\n"
                + "Username: " + voterForm.getUsername()
                + "\nPassword: " + generatedPassword + "\n";
        sendMailService.sendMail("ivote.sharemind@gmail.com",
              voterForm.getEmail(),
            "[iVote Sharemind] Login Credentials", mailMessage
                  );
        Voter voter = null;
        if (!voterExist) {
            voter = new Voter();
        } else {
            voter = voterDao.findVoterByUser(savedUser);
            voter.setVoted(false);
            voter.setPublished(false);
            voter.setActive(true);
        }
        voter.setUser(savedUser);
        voter.setVoterType(voterForm.getVoterType());
        save(voter);
    }

    @Override
    public void edit(VoterEditForm voterEditForm) {
        Voter voter = findById(voterEditForm.getId());
        User user = voter.getUser();
        user.setUsername(voterEditForm.getUsername());
        user.setEmail(voterEditForm.getEmail());
        User updatedUser = userDao.save(user);
        voter.setUser(updatedUser);
        voter.setVoterType(voterEditForm.getVoterType());
        save(voter);
    }

    @Override
    public void delete(Voter voter) {
        voter.setActive(false);
        save(voter);
    }

    private Voter save(Voter voter) {
        String salt = "";
        try {
            salt = hashService.getSalt();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        voter.setSalt(salt);
        return voterDao.save(voter);
    }

    @Override
    public List<Voter> getAllVoters() {
        return (List<Voter>) voterDao.findAll();
    }

    @Override
    public List<Voter> getVotersVoted() {
        return voterDao.getVotersByVoted(true);
    }

    @Override
    public List<Voter> checkRecordedVoters(List<Voter> voters)
            throws SharemindException {
        List<Voter> recordedVoters = voterDao.
              getVotersByPublished(true);
        if (recordedVoters.isEmpty()) {
            recordedVoters = new ArrayList<>();
        }
        for (Voter voter : voters) {
            if (!voter.isPublished()) {
                String verificationCode = "";
                VoterType voterType = voter.getVoterType();
                List<Election> elections = voterType.
                      getElections();
                List<Position> electionPositions = new
                      ArrayList<>();
                for (Election election : elections) {
                    electionPositions.addAll(positionDao.
                          getPositionsByElection(election));
                }
                StringBuilder builder = new StringBuilder();
                for (Position position : electionPositions) {
                    ProcessBuilder pb = new ProcessBuilder("./
                          CheckVoter", String.valueOf(position
                          .getId()),
                        String.valueOf(voter.getId()));
                    pb.directory(new File("/home/sharemind/
                          DevTools/bin/"));
                    Process p = null;
                    try {
                        p = pb.start();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    BufferedReader reader = new BufferedReader
                          (new InputStreamReader(p.
                          getInputStream()));
                    String line = null;
                    int n = 0;
                    try {
                        while ((line = reader.readLine()) !=
                              null) {
                            if (line.contains("WARNING:")) {
                                throw new SharemindException()
                                      ;
                            }
                            if (n >= 3) {
                                if (line.contains("Candidates"
                                      )) {
                                    builder.append(line.
                                          substring(line.
                                          lastIndexOf("=") +
                                          1) + ":");
                                }
                            }
                            n++;
                        }
                    } catch (IOException e) {
```

```
                        e.printStackTrace();
                    }

                    try {
                        if (p.waitFor() == 0) {
                            continue;
                        }
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                verificationCode = hashService.getHash(builder
                      .toString(), voter.getSalt());
                if (voter.getVerificationCode().equals(
                      verificationCode)) {
                    voter.setPublished(true);
                    recordedVoters.add(voter);
                }
            }
        }
        return recordedVoters;
    }

    @Override
    public Voter getVoterByAlias(String alias) {
        return voterDao.getVoterByAlias(alias);
    }

    private boolean voterExists(VoterCreateForm voterForm) {
        if (userDao.findByUsername(voterForm.getUsername()) !=
              null
                || userDao.findByEmail(voterForm.getEmail()) !=
                      null) {
            return true;
        } else {
            return false;
        }
    }

    @Override
    public List<Voter> getPublishedVoters() {
        return voterDao.getVotersByPublished(true);
    }

    @Override
    public List<Voter> getAllActiveVoters() {
        return voterDao.getVotersByActive(true);
    }

    @Override
    public List<Voter> getVotersByVoterTypes(List<VoterType>
          voterTypes) {
        List<Voter> voters = new ArrayList<>();
        for (VoterType voterType : voterTypes) {
            voters.addAll(voterDao.getVotersByVoterType(
                  voterType));
        }
        return voters;
    }
}
```

4.24. VoterTypeService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.List;

import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.VoterType;

public interface VoterTypeService {

    VoterType findById(long voterTypeId);

    VoterType save(VoterType voterType);

    void delete(VoterType voterType);

    List<VoterType> getAllVoterTypes();

    List<VoterType> getTypesByElection(Election election);

}
```

4.25. VoterTypeServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.
      Transactional;

import ph.edu.upm.agila.japiscos.dao.VoterTypeDao;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.VoterType;

@Service("voterTypeService")
@Transactional
public class VoterTypeServiceImpl implements VoterTypeService
      {

    @Autowired
    private VoterTypeDao voterTypeDao;

    @Override
    public VoterType findById(long voterTypeId) {
        return voterTypeDao.findById(voterTypeId);
    }

    @Override
    public VoterType save(VoterType voterType) {
        return voterTypeDao.save(voterType);
    }

    @Override
    public void delete(VoterType voterType) {
        voterTypeDao.delete(voterType);
    }

    @Override
    public List<VoterType> getAllVoterTypes() {
        return (List<VoterType>) voterTypeDao.findAll();
    }

    @Override
    public List<VoterType> getTypesByElection(Election
          election) {
        List<Election> elections = new ArrayList<>();
        elections.add(election);
        return voterTypeDao.getTypesByElections(elections);
    }

}
```

4.26. VoteService.java

```
package ph.edu.upm.agila.japiscos.service;

import java.io.IOException;
import java.util.List;

import ph.edu.upm.agila.japiscos.exception.SharemindException;
import ph.edu.upm.agila.japiscos.formbacker.Vote;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Voter;

public interface VoteService {

    void addVotes(List<Vote> votes, Voter voter) throws
          IOException, SharemindException;

    void updateVotes(List<Vote> votes, Voter voter) throws
          IOException, SharemindException;

    void processResults(List<Candidate> candidates) throws
          IOException, InterruptedException;

}
```

4.27. VoteServiceImpl.java

```
package ph.edu.upm.agila.japiscos.service;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import ph.edu.upm.agila.japiscos.dao.AbstainDao;
import ph.edu.upm.agila.japiscos.dao.CandidateDao;
import ph.edu.upm.agila.japiscos.dao.PositionDao;
import ph.edu.upm.agila.japiscos.dao.VoteDao;
import ph.edu.upm.agila.japiscos.dao.VoterDao;
import ph.edu.upm.agila.japiscos.exception.SharemindException;
import ph.edu.upm.agila.japiscos.formbacker.PositionVote;
import ph.edu.upm.agila.japiscos.formbacker.Vote;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Position;
import ph.edu.upm.agila.japiscos.model.Voter;
import ph.edu.upm.agila.japiscos.model.VoterType;

@Service("voteService")
public class VoteServiceImpl implements VoteService {

    @Autowired
    private VoteDao voteDao;

    @Autowired
    private VoterDao voterDao;

    @Autowired
    private CandidateDao candidateDao;

    @Autowired
    private PositionDao positionDao;
```

```
    @Autowired
    private AbstainDao abstainDao;

    @Autowired
    private HashService hashService;

    @Override
    public void addVotes(List<Vote> votes, Voter voter) throws
            IOException,
             SharemindException {
        String candidateVotes = "";
        for (Vote vote : votes) {
            long[] candidateIds = vote.getCandidateIds();
            for (int i = 0; i < candidateIds.length; i++) {
                long id = candidateIds[i];
                Candidate candidate;
                if (id != 0) {
                    candidate = candidateDao.findById(id);
                } else {
                    candidate = new Candidate();
                    candidate.setId(0);
                    Position position = positionDao.findById(
                            vote.getPositionId());
                    candidate.setPosition(position);
                }
                candidateVotes += String.valueOf(candidate.
                        getId()) + ":";
                try {
                    Process p = voteDao.addVote(candidate,
                            voter);
                    BufferedReader reader = new BufferedReader
                            (new InputStreamReader(p.
                            getInputStream()));
                    String line = null;
                    while ((line = reader.readLine()) != null)
                             {
                        if (line.contains("WARNING:")) {
                            throw new SharemindException();
                        }
                    }
                    if (p.waitFor() == 0) {
                        continue;
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
        boolean voted = true;
        List<Voter> voters = voterDao.getVotersByVoted(voted);
        String verificationCode = hashService.getHash(
                candidateVotes,
                voter.getSalt());
        if (!voter.isVoted()) {
            int aliasNumber = voters.size() + 1;
            String alias = "V" + aliasNumber;
            Voter checkVoter = voterDao.getVoterByAlias(alias)
                    ;
            while (checkVoter != null) {
                aliasNumber++;
                alias = "V" + aliasNumber;
                checkVoter = voterDao.getVoterByAlias(alias);
            }
            voter.setAlias(alias);
            voter.setVoted(true);
        }
        voter.setVerificationCode(verificationCode);
        List<PositionVote> positionVotes = getVoteHashes(votes
                , voter);
        String trackerJson = getJson(positionVotes, voter);
        voter.setTracker(trackerJson);
        voterDao.save(voter);
    }

    @SuppressWarnings("unchecked")
    private String getJson(List<PositionVote> positionVotes,
            Voter voter) {
        JSONObject outerObject = new JSONObject();
        VoterType voterType = voter.getVoterType();
        List<Election> voterElections = voterType.getElections
                ();
        for (Election election : voterElections) {
            JSONArray outerArray = new JSONArray();
            for (PositionVote positionVote : positionVotes) {
                if (positionVote.getPosition().getElection().
                        getId() == election.getId()) {
                    JSONObject innerObject = new JSONObject();
                    List<String> voteHashes = positionVote.
                            getVoteHashes();
                    JSONArray innerArray = new JSONArray();
                    for (String voteHash : voteHashes) {
                        innerArray.add(voteHash);
                    }
                    innerObject.put(positionVote.getPosition()
                            .getName(), innerArray);
                    outerArray.add(innerObject);
                }
            }
            outerObject.put(election.getName(), outerArray);
        }
        return outerObject.toString();
    }

    private List<PositionVote> getVoteHashes(List<Vote> votes,
            Voter voter) {
        List<PositionVote> positionVotes = new ArrayList<>();
        for (Vote vote : votes) {
            List<String> hashes = new ArrayList<>();
            for (int i = 0; i < vote.getCandidateIds().length;
                    i++) {
                long id = vote.getCandidateIds()[i];
                String hash = hashService.getHash(String.
                        valueOf(id), voter.getSalt());
                hashes.add(hash);
            }
            Position position = positionDao.findById(vote.
                    getPositionId());
            PositionVote positionVote = new PositionVote();
            positionVote.setPosition(position);
            positionVote.setVoteHashes(hashes);
            positionVotes.add(positionVote);
        }
        return positionVotes;
    }

    @Override
    public void updateVotes(List<Vote> votes, Voter voter)
            throws IOException,
             SharemindException {
        for (Vote vote : votes) {
            long[] candidateIds = vote.getCandidateIds();
            for (int i = 0; i < vote.getCandidateIds().length;
                    i++) {
                long id = candidateIds[i];
                Candidate candidate;
                if (id != 0) {
                    candidate = candidateDao.findById(id);
                } else {
                    candidate = new Candidate();
                    candidate.setId(0);
                    Position position = positionDao.findById(
                            vote.getPositionId());
                    candidate.setPosition(position);
                }
                try {
                    Process p = voteDao.updateVote(candidate,
                            voter);
                    BufferedReader reader = new BufferedReader
                            (new InputStreamReader(p.
                            getInputStream()));
                    String line = null;
                    while ((line = reader.readLine()) != null)
                             {
                        if (line.contains("WARNING:")) {
                            throw new SharemindException();
                        }
                    }
                    if (p.waitFor() == 0) {
                        continue;
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
        addVotes(votes, voter);
    }

    @Override
    public void processResults(List<Candidate> candidates)
            throws IOException,
             InterruptedException {
        voteDao.processResults(candidates);
        List<Position> positions = new ArrayList<>();
        for (Candidate candidate : candidates) {
            if (!positions.contains(candidate.getPosition()))
                     {
                positions.add(candidate.getPosition());
            }
        }
        for (Position position : positions) {
            if (position.isAbstain()) {
                voteDao.processResults(abstainDao.
                        findByPosition(position));
            }
        }
    }

}
```

5. Data Access Objects (DAO)

5.1. AbstainDao.java

```
package ph.edu.upm.agila.japiscos.dao;

import org.springframework.data.repository.CrudRepository;

import ph.edu.upm.agila.japiscos.model.Abstain;
import ph.edu.upm.agila.japiscos.model.Position;

public interface AbstainDao extends CrudRepository<Abstain,
        Long> {

    Abstain findByPosition(Position position);

}
```

5.2. CandidateDao.java

```
package ph.edu.upm.agila.japiscos.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Party;
import ph.edu.upm.agila.japiscos.model.Position;

public interface CandidateDao extends CrudRepository<Candidate
      , Long> {
    Candidate findById(long id);

    List<Candidate> getCandidateByPosition(Position position);

    List<Candidate> getCandidatesByElection(Election election)
          ;

    List<Candidate> getCandidatesByElectionOrderByLastNameAsc(
          Election election);

    List<Candidate> getCandidatesByParty(Party party);
}
```

5.3. ElectionDao.java

```
package ph.edu.upm.agila.japiscos.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import ph.edu.upm.agila.japiscos.model.Election;

public interface ElectionDao extends CrudRepository<Election,
      Long> {
    Election findById(long id);

    List<Election> getElectionsByParentElectionId(long id);

    List<Election> getElectionsByStatus(boolean status);
}
```

5.4. PartyDao.java

```
package ph.edu.upm.agila.japiscos.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Party;

public interface PartyDao extends CrudRepository<Party, Long>
      {
    Party findById(long id);

    List<Party> getPartiesByElections(List<Election> elections
          );

    List<Party> getPartiesByElectionsOrderByNameAsc(List<
          Election> elections);
}
```

5.5. PositionDao.java

```
package ph.edu.upm.agila.japiscos.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.Position;

public interface PositionDao extends CrudRepository<Position,
      Long> {
    Position findById(long id);

    List<Position> getPositionsByElection(Election election);

    List<Position> getPositionsByElectionOrderByOrdinalityAsc(
          Election election);
}
```

5.6. UserDao.java

```
package ph.edu.upm.agila.japiscos.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import ph.edu.upm.agila.japiscos.model.Role;
import ph.edu.upm.agila.japiscos.model.User;

public interface UserDao extends CrudRepository<User, Long> {
    User findByUsername(String username);

    User findByEmail(String email);

    User findById(long id);

    List<User> getUsersByRole(Role votingOfficial);
}
```

5.7. VoteDao.java

```
package ph.edu.upm.agila.japiscos.dao;

import java.io.IOException;
import java.util.List;

import ph.edu.upm.agila.japiscos.model.Abstain;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Voter;

public interface VoteDao {

    Process addVote(Candidate candidate, Voter voter) throws
          IOException;

    Process updateVote(Candidate candidate, Voter voter)
          throws IOException;

    void processResults(List<Candidate> candidates) throws
          IOException, InterruptedException;

    void processResults(Abstain abstain) throws
          InterruptedException, IOException;
}
```

5.8. VoteDaoImpl.java

```
package ph.edu.upm.agila.japiscos.dao;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import ph.edu.upm.agila.japiscos.model.Abstain;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Voter;

@Repository("voteDao")
public class VoteDaoImpl implements VoteDao {

    @Autowired
    private CandidateDao candidateDao;

    @Autowired
    private AbstainDao abstainDao;

    @Override
    public Process addVote(Candidate candidate, Voter voter)
          throws IOException {
        String[] commands = new String[4];
        commands[0] = "/home/sharemind/DevTools/bin/./
              VoteDatabaseEntry";
        commands[1] = String.valueOf(candidate.getPosition().
              getId());
        commands[2] = String.valueOf(candidate.getId());
        commands[3] = String.valueOf(voter.getId());
        ProcessBuilder pb = new ProcessBuilder(commands);
        Process p = pb.start();
        return p;
    }

    @Override
    public Process updateVote(Candidate candidate, Voter voter
          )
            throws IOException {
        String[] commands = new String[3];
        commands[0] = "/home/sharemind/DevTools/bin/./
              UpdateVoteEntry";
        commands[1] = String.valueOf(candidate.getPosition().
              getId());
        commands[2] = String.valueOf(voter.getId());
        ProcessBuilder pb = new ProcessBuilder(commands);
        Process p = pb.start();
        return p;
    }
```

```
    @Override
    public void processResults(List<Candidate> candidates)
          throws IOException, InterruptedException {
        int i = 0;
        while (i < candidates.size()) {
            ProcessBuilder pb = new ProcessBuilder("./
                  ElectionResults",
                    String.valueOf(candidates.get(i).
                          getPosition().getId()),
                    String.valueOf(candidates.get(i).getId()))
                          ;
            pb.directory(new File("/home/sharemind/DevTools/
                  bin/"));
            Process p = pb.start();
            BufferedReader reader = new BufferedReader(new
                  InputStreamReader(p.getInputStream()));
            StringBuilder builder = new StringBuilder();
            String line = null;
            int n = 0;
            while ((line = reader.readLine()) != null) {
                if (n == 3) {
                    builder.append(line.substring(line.
                          lastIndexOf("=") + 1));
                    break;
                }
                n++;
            }
            String result = builder.toString();
            int count = Integer.parseInt(result);
            candidates.get(i).setVoteCount(count);
            candidateDao.save(candidates.get(i));
            if (p.waitFor() == 0) {
                i++;
            }
        }
    }

    @Override
    public void processResults(Abstain abstain) throws
          InterruptedException, IOException {
        ProcessBuilder pb = new ProcessBuilder("./
              ElectionResults",
                String.valueOf(abstain.getPosition().getId()),
                String.valueOf(0));
        pb.directory(new File("/home/sharemind/DevTools/bin/")
              );
        Process p = pb.start();
        BufferedReader reader = new BufferedReader(new
              InputStreamReader(p.getInputStream()));
        StringBuilder builder = new StringBuilder();
        String line = null;
        int n = 0;
        while ((line = reader.readLine()) != null) {
            if (n == 3) {
                builder.append(line.substring(line.lastIndexOf
                      ("=") + 1));
                break;
            }
            n++;
        }
        String result = builder.toString();
        int count = Integer.parseInt(result);
        abstain.setVoteCount(count);
        abstainDao.save(abstain);
        if (p.waitFor() == 0) {
            return;
        }
    }

}
```

5.9. VoterDao.java

```
package ph.edu.upm.agila.japiscos.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import ph.edu.upm.agila.japiscos.model.User;
import ph.edu.upm.agila.japiscos.model.Voter;
import ph.edu.upm.agila.japiscos.model.VoterType;

public interface VoterDao extends CrudRepository<Voter, Long>
      {

    Voter findById(long voterId);

    Voter findVoterByUser(User user);

    List<Voter> getVotersByVoted(boolean voted);

    Voter getVoterByAlias(String alias);

    List<Voter> getVotersByPublished(boolean published);

    List<Voter> getVotersByVoterType(VoterType voterType);

    List<Voter> getVotersByActive(boolean active);

}
```

5.10. VoterTypeDao.java

```
package ph.edu.upm.agila.japiscos.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import ph.edu.upm.agila.japiscos.model.Election;
import ph.edu.upm.agila.japiscos.model.VoterType;

public interface VoterTypeDao extends CrudRepository<VoterType
      , Long> {

    VoterType findById(long voterTypeId);

    List<VoterType> getTypesByElections(List<Election>
          elections);

}
```

6. Entity Models

6.1. Abstain.java

```
package ph.edu.upm.agila.japiscos.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "abstain")
public class Abstain {

    @Id
    @Column(name = "abstain_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @OneToOne
    @JoinColumn(name = "position_id")
    private Position position;

    @Column(name = "vote_count")
    private int voteCount;

    public Abstain() {
        this.voteCount = 0;
    }

    public Position getPosition() {
        return position;
    }

    public void setPosition(Position position) {
        this.position = position;
    }

    public int getVoteCount() {
        return voteCount;
    }

    public void setVoteCount(int voteCount) {
        this.voteCount = voteCount;
    }

}
```

6.2. Candidate.java

```
package ph.edu.upm.agila.japiscos.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.persistence.Transient;

@Entity
@Table(name = "candidate")
public class Candidate {

    @Id
    @Column(name = "candidate_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @Column(name = "last_name", length = 30)
    private String lastName;

    @Column(name = "first_name", length = 30)
    private String firstName;

    @ManyToOne
    @JoinColumn(name = "party_id")
    private Party party;
```

```
    @OneToOne
    @JoinColumn(name = "position_id")
    private Position position;

    @OneToOne
    @JoinColumn(name = "election_id")
    private Election election;

    @Column(name = "vote_count")
    private int voteCount;

    @Transient
    private String hash;

    public Candidate() {
        this.voteCount = 0;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public Party getParty() {
        return party;
    }

    public void setParty(Party party) {
        this.party = party;
    }

    public Position getPosition() {
        return position;
    }

    public void setPosition(Position position) {
        this.position = position;
    }

    public Election getElection() {
        return election;
    }

    public void setElection(Election election) {
        this.election = election;
    }

    public int getVoteCount() {
        return voteCount;
    }

    public void setVoteCount(int voteCount) {
        this.voteCount = voteCount;
    }

    public String getHash() {
        return hash;
    }

    public void setHash(String hash) {
        this.hash = hash;
    }
}
```

6.3. CurrentUser.java

```
package ph.edu.upm.agila.japiscos.model;

import org.springframework.security.core.authority.
       AuthorityUtils;

public class CurrentUser extends
        org.springframework.security.core.userdetails.User {

    private static final long serialVersionUID =
           4155877839666002405L;

    private User user;

    public CurrentUser(User user) {
        super(user.getUsername(), user.getPasswordHash(),
              AuthorityUtils
                .createAuthorityList(user.getRole().toString()
                    ));
        this.user = user;
    }

    public User getUser() {
        return user;
    }

    public Long getId() {
        return user.getId();
    }

    public Role getRole() {
        return user.getRole();
    }
}
```

6.4. Election.java

```
package ph.edu.upm.agila.japiscos.model;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

@Entity
@Table(name = "election")
public class Election {

    @Id
    @Column(name = "election_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @NotEmpty
    @Length(min = 5, max = 50)
    @Column(name = "name", length = 50)
    private String name;

    @Column(name = "parent_id")
    private long parentElectionId;

    @Column(name = "status")
    private boolean status;

    @Column(name = "start_date", columnDefinition = "DATETIME"
          )
    @Temporal(TemporalType.TIMESTAMP)
    private Date startDate;

    @Column(name = "end_date", columnDefinition = "DATETIME")
    @Temporal(TemporalType.TIMESTAMP)
    private Date endDate;

    public Election() {
        this.status = false;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public long getParentElectionId() {
        return parentElectionId;
    }

    public void setParentElectionId(long parentElectionId) {
        this.parentElectionId = parentElectionId;
    }

    public boolean isStatus() {
        return status;
    }

    public void setStatus(boolean status) {
        this.status = status;
    }

    public Date getStartDate() {
        return startDate;
    }

    public void setStartDate(Date startDate) {
        this.startDate = startDate;
    }
```

```
    public Date getEndDate() {
        return endDate;
    }

    public void setEndDate(Date endDate) {
        this.endDate = endDate;
    }

    @Override
    public boolean equals(Object object) {
        boolean sameSame = false;
        if (object != null && object instanceof Election) {
            sameSame = this.id == ((Election) object).id;
        }
        return sameSame;
    }

}
```

6.5. Party.java

```
package ph.edu.upm.agila.japiscos.model;

import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

@Entity
@Table(name = "party")
public class Party {

    @Id
    @Column(name = "party_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @NotEmpty
    @Length(min = 5, max = 20)
    @Column(name = "name", length = 20)
    private String name;

    @NotEmpty
    @ManyToMany
    @Column(name = "election_id")
    @JoinTable(name = "party_elections", joinColumns =
          @JoinColumn(name = "party_id"),
        inverseJoinColumns = @JoinColumn(name = "election_id")
              )
    private List<Election> elections;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<Election> getElections() {
        return elections;
    }

    public void setElections(List<Election> elections) {
        this.elections = elections;
    }

}
```

6.6. Position.java

```
package ph.edu.upm.agila.japiscos.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "position")
public class Position {

    @Id
    @Column(name = "position_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @Column(name = "name", length = 20)
    private String name;

    @Column(name = "max")
    private int maxElected;

    @Column(name = "ordinality")
    private int ordinality;

    @Column(name = "abstain")
    private boolean abstain;

    @OneToOne
    @JoinColumn(name = "election_id")
    private Election election;

    public Position() {

    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getMaxElected() {
        return maxElected;
    }

    public void setMaxElected(int maxElected) {
        this.maxElected = maxElected;
    }

    public int getOrdinality() {
        return ordinality;
    }

    public void setOrdinality(int ordinality) {
        this.ordinality = ordinality;
    }

    public boolean isAbstain() {
        return abstain;
    }

    public void setAbstain(boolean abstain) {
        this.abstain = abstain;
    }

    public Election getElection() {
        return election;
    }

    public void setElection(Election election) {
        this.election = election;
    }

}
```

6.7. Role.java

```
package ph.edu.upm.agila.japiscos.model;

public enum Role {
    VOTER("Voter"), ADMIN("Admin"), VOTING_OFFICIAL("Voting
          Official");

    private String value;

    private Role(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }

}
```

6.8. User.java

```
package ph.edu.upm.agila.japiscos.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "user")
public class User {

    @Id
    @Column(name = "user_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @Column(name = "username", unique = true, length = 20)
    private String username;

    @Column(name = "password_hash")
    private String passwordHash;

    @Column(name = "email", unique = true, length = 30)
    private String email;

    @Column(name = "role")
    @Enumerated(EnumType.STRING)
    private Role role;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPasswordHash() {
        return passwordHash;
    }

    public void setPasswordHash(String passwordHash) {
        this.passwordHash = passwordHash;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    }

}
```

6.9. Voter.java

```
package ph.edu.upm.agila.japiscos.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "voter")
public class Voter {

    @Id
    @Column(name = "voter_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @OneToOne
    @JoinColumn(name = "user_id")
    private User user;

    @Column(name = "voted")
    private boolean voted;

    @Column(name = "code")
    private String verificationCode;

    @OneToOne
    @JoinColumn(name = "voter_type_id")
    private VoterType voterType;

    @Column(name = "alias", length = 10)
    private String alias;

    @Column(name = "salt", length = 11)
    private String salt;

    @Column(name = "published")
    private boolean published;

    @Column(name = "tracker", length = 300)
    private String tracker;

    @Column(name = "active")
    private boolean active;

    public Voter() {
        this.voted = false;
        this.published = false;
        this.active = true;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public boolean isVoted() {
        return voted;
    }

    public void setVoted(boolean voted) {
        this.voted = voted;
    }

    public String getVerificationCode() {
        return verificationCode;
    }

    public void setVerificationCode(String verificationCode) {
        this.verificationCode = verificationCode;
    }

    public VoterType getVoterType() {
        return voterType;
    }

    public void setVoterType(VoterType voterType) {
        this.voterType = voterType;
    }

    public String getAlias() {
        return alias;
    }

    public void setAlias(String alias) {
        this.alias = alias;
    }

    public String getSalt() {
        return salt;
    }

    public void setSalt(String salt) {
        this.salt = salt;
    }

    public boolean isPublished() {
        return published;
    }

    public void setPublished(boolean published) {
        this.published = published;
    }

    public String getTracker() {
        return tracker;
    }

    public void setTracker(String tracker) {
        this.tracker = tracker;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

}
```

6.10. VoterType.java

```
package ph.edu.upm.agila.japiscos.model;

import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

@Entity
@Table(name = "voter_type")
public class VoterType {

    @Id
    @Column(name = "voter_type_id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @NotEmpty
    @Length(min = 5, max = 20)
    @Column(name = "name", length = 20)
    private String name;

    @NotEmpty
    @ManyToMany(fetch = FetchType.EAGER)
    @Column(name = "election_id")
    @JoinTable(name = "voter_type_elections", joinColumns =
           @JoinColumn(name = "voter_type_id"),
        inverseJoinColumns = @JoinColumn(name = "election_id")
                )
    private List<Election> elections;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public List<Election> getElections() {
        return elections;
    }

    public void setElections(List<Election> elections) {
        this.elections = elections;
    }

}
```

7. Form Backers

7.1. Ballot.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import java.util.ArrayList;
import java.util.List;

public class Ballot {

    private List<Vote> votes;
    private long electionId = -1;

    public Ballot() {
        votes = new ArrayList<>();
    }

    public long getElectionId() {
        return electionId;
    }

    public void setElectionId(long electionId) {
        this.electionId = electionId;
    }

    public List<Vote> getVotes() {
        return votes;
    }

    public void setVotes(List<Vote> votes) {
        this.votes = votes;
    }

}
```

7.2. CandidateBacker.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

public class CandidateBacker {

    private long id;

    @NotEmpty
    @Length(min = 2, max = 30)
    private String lastName;

    @NotEmpty
    @Length(min = 2, max = 30)
    private String firstName;

    private long electionId;

    private long partyId;

    private long positionId;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public long getPartyId() {
        return partyId;
    }

    public void setPartyId(long partyId) {
        this.partyId = partyId;
    }

    public long getPositionId() {
        return positionId;
    }

    public void setPositionId(long positionId) {
        this.positionId = positionId;
    }

    public long getElectionId() {
        return electionId;
    }

    public void setElectionId(long electionId) {
        this.electionId = electionId;
    }

}
```

7.3. ChangePasswordForm.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import org.hibernate.validator.constraints.NotEmpty;

public class ChangePasswordForm {

    private long id;

    @NotEmpty
    private String oldPassword;

    @NotEmpty
    private String newPassword;

    @NotEmpty
    private String newPasswordRepeated;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getOldPassword() {
```

```
        return oldPassword;
    }

    public void setOldPassword(String oldPassword) {
        this.oldPassword = oldPassword;
    }

    public String getNewPassword() {
        return newPassword;
    }

    public void setNewPassword(String newPassword) {
        this.newPassword = newPassword;
    }

    public String getNewPasswordRepeated() {
        return newPasswordRepeated;
    }

    public void setNewPasswordRepeated(String
          newPasswordRepeated) {
        this.newPasswordRepeated = newPasswordRepeated;
    }

}
```

7.4. ImportVoterForm.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import javax.validation.constraints.NotNull;

import org.springframework.web.multipart.MultipartFile;

import ph.edu.upm.agila.japiscos.model.VoterType;

public class ImportVoterForm {

    @NotNull
    private MultipartFile file;

    @NotNull
    private VoterType voterType;

    public MultipartFile getFile() {
        return file;
    }

    public void setFile(MultipartFile file) {
        this.file = file;
    }

    public VoterType getVoterType() {
        return voterType;
    }

    public void setVoterType(VoterType voterType) {
        this.voterType = voterType;
    }

}
```

7.5. PositionBacker.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

public class PositionBacker {

    private long id;

    @NotEmpty
    @Length(min = 5, max = 20)
    private String name;

    @NotNull
    @Min(1)
    private Integer maxElected;

    private long electionId;

    @NotNull
    @Min(1)
    private Integer ordinality;

    private boolean abstain;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getMaxElected() {
        return maxElected;
    }

    public void setMaxElected(Integer maxElected) {
        this.maxElected = maxElected;
    }

    public long getElectionId() {
        return electionId;
    }

    public void setElectionId(long electionId) {
        this.electionId = electionId;
    }

    public Integer getOrdinality() {
        return ordinality;
    }

    public void setOrdinality(Integer ordinality) {
        this.ordinality = ordinality;
    }

    public boolean isAbstain() {
        return abstain;
    }

    public void setAbstain(boolean abstain) {
        this.abstain = abstain;
    }

}
```

7.6. PositionVote.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import java.util.ArrayList;
import java.util.List;

import ph.edu.upm.agila.japiscos.model.Position;

public class PositionVote {

    private Position position;

    private List<String> voteHashes;

    public PositionVote() {
        voteHashes = new ArrayList<>();
    }

    public Position getPosition() {
        return position;
    }

    public void setPosition(Position position) {
        this.position = position;
    }

    public List<String> getVoteHashes() {
        return voteHashes;
    }

    public void setVoteHashes(List<String> voteHashes) {
        this.voteHashes = voteHashes;
    }

    @Override
    public String toString() {
        return "[position=" + position + ", " + "candidates="
              + voteHashes + "]";
    }

}
```

7.7. UserCreateForm.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

import ph.edu.upm.agila.japiscos.model.Role;
import ph.edu.upm.agila.japiscos.model.VoterType;

public class UserCreateForm {

    @NotEmpty
    @Length(min = 5, max = 20)
    private String username = "";

    @Email
    @NotEmpty
    @Length(min = 5, max = 30)
```

```
    private String email;

    @NotNull
    private Role role;

    private VoterType voterType;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    }

    public VoterType getVoterType() {
        return voterType;
    }

    public void setVoterType(VoterType voterType) {
        this.voterType = voterType;
    }

}
```

7.8. UserEditForm.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

import ph.edu.upm.agila.japiscos.model.Role;
import ph.edu.upm.agila.japiscos.model.VoterType;

public class UserEditForm {

    private long id;

    @NotEmpty
    @Length(min = 5, max = 20)
    private String username = "";

    @NotEmpty
    @Email
    @Length(min = 5, max = 30)
    private String email;

    @NotNull
    private Role role;

    private VoterType voterType;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Role getRole() {
        return role;
    }

    public void setRole(Role role) {
        this.role = role;
    }

    public VoterType getVoterType() {
        return voterType;
    }

    public void setVoterType(VoterType voterType) {
        this.voterType = voterType;
    }
}
```

7.9. Vote.java

```
package ph.edu.upm.agila.japiscos.formbacker;

public class Vote {

    private long[] candidateIds;
    private long positionId;

    public Vote() {
    }

    public long[] getCandidateIds() {
        return candidateIds;
    }

    public void setCandidateIds(long[] candidateIds) {
        this.candidateIds = candidateIds;
    }

    public long getPositionId() {
        return positionId;
    }

    public void setPositionId(long positionId) {
        this.positionId = positionId;
    }

}
```

7.10. VoterCreateForm.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

import ph.edu.upm.agila.japiscos.model.VoterType;

public class VoterCreateForm {

    @NotEmpty
    @Length(min = 5, max = 20)
    private String username;

    @NotEmpty
    @Email
    @Length(min = 5, max = 30)
    private String email;

    @NotNull
    private VoterType voterType;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public VoterType getVoterType() {
        return voterType;
    }

    public void setVoterType(VoterType voterType) {
        this.voterType = voterType;
    }

}
```

7.11. VoterEditForm.java

```
package ph.edu.upm.agila.japiscos.formbacker;

import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.Length;
import org.hibernate.validator.constraints.NotEmpty;

import ph.edu.upm.agila.japiscos.model.VoterType;

public class VoterEditForm {

    private long id;

    @NotEmpty
    @Length(min = 5, max = 20)
    private String username;

    @NotEmpty
    @Email
    @Length(min = 5, max = 30)
    private String email;

    @NotNull
    private VoterType voterType;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public VoterType getVoterType() {
        return voterType;
    }

    public void setVoterType(VoterType voterType) {
        this.voterType = voterType;
    }

}
```

7.12. VoterImport.java

```
package ph.edu.upm.agila.japiscos.formbacker;

public class VoterImport {

    private String username;

    private String email;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

}
```

7.13. VoterLoginCredentials.java

```
package ph.edu.upm.agila.japiscos.formbacker;

public class VoterLoginCredentials {

    private String username;

    private String generatedPassword;

    private String email;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getGeneratedPassword() {
        return generatedPassword;
    }

    public void setGeneratedPassword(String generatedPassword)
            {
        this.generatedPassword = generatedPassword;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

}
```

8. PDF Views

8.1. AbstractITextPdfView.java

```
package ph.edu.upm.agila.japiscos.pdf;

import java.io.ByteArrayOutputStream;
import java.io.OutputStream;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.view.AbstractView;

import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.PageSize;
import com.itextpdf.text.pdf.PdfWriter;

/**
 * This class is a work around for working with iText 5.x in
       Spring.
 *
 * @author www.codejava.net
 */
public abstract class AbstractITextPdfView extends
       AbstractView {

    public AbstractITextPdfView() {
        setContentType("application/pdf");
    }

    @Override
    protected boolean generatesDownloadContent() {
        return true;
    }

    @Override
    protected void renderMergedOutputModel(Map<String, Object>
           model, HttpServletRequest request,
        HttpServletResponse response) throws Exception {
        ByteArrayOutputStream baos =
              createTemporaryOutputStream();

        Document document = newDocument();
        PdfWriter writer = newWriter(document, baos);
        prepareWriter(model, writer, request);
        buildPdfMetadata(model, document, request);
        document.open();
        buildPdfDocument(model, document, writer, request,
              response);
        document.close();
        writeToResponse(response, baos);
    }

    protected Document newDocument() {
        return new Document(PageSize.A4);
    }

    protected PdfWriter newWriter(Document document,
          OutputStream os) throws DocumentException {
        return PdfWriter.getInstance(document, os);
    }

    protected void prepareWriter(Map<String, Object> model,
          PdfWriter writer, HttpServletRequest request)
        throws DocumentException {
        writer.setViewerPreferences(getViewerPreferences());
    }

    protected int getViewerPreferences() {
        return PdfWriter.ALLOW_PRINTING | PdfWriter.
              PageLayoutSinglePage;
    }

    protected void buildPdfMetadata(Map<String, Object> model,
        Document document, HttpServletRequest request) {
    }

    protected abstract void buildPdfDocument(Map<String,
          Object> model, Document document,
        PdfWriter writer, HttpServletRequest request,
              HttpServletResponse response) throws Exception;
}
```

8.2. BulletinBoardPdfBuilder.java

```
package ph.edu.upm.agila.japiscos.pdf;

import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import ph.edu.upm.agila.japiscos.model.Voter;

import com.itextpdf.text.BaseColor;
import com.itextpdf.text.Document;
import com.itextpdf.text.Font;
import com.itextpdf.text.FontFactory;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.pdf.PdfPCell;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;

public class BulletinBoardPdfBuilder extends
      AbstractITextPdfView {

    @SuppressWarnings("unchecked")
    @Override
    protected void buildPdfDocument(Map<String, Object> model,
            Document doc,
             PdfWriter writer, HttpServletRequest request,
             HttpServletResponse response) throws Exception {
        List<Voter> voters = (List<Voter>) model.get("
              publishedVoters");
        Font siteFont = FontFactory.getFont(FontFactory.
              HELVETICA);
        BaseColor customColor = new BaseColor(48, 165, 255);
        siteFont.setColor(customColor);
        siteFont.setSize(15f);
        siteFont.setStyle(Font.BOLD);
        doc.add(new Paragraph("iVote", siteFont));
        Font titleFont = FontFactory.getFont(FontFactory.
              HELVETICA);
        titleFont.setColor(BaseColor.BLACK);
        titleFont.setSize(14f);
        titleFont.setStyle(Font.BOLD);
        doc.add(new Paragraph("Bulletin Board", titleFont));
        PdfPTable table = new PdfPTable(3);
        table.setWidthPercentage(100.0f);
        table.setWidths(new float[] { 1.0f, 3.0f, 3.0f });
        table.setSpacingBefore(10);
        Font font = FontFactory.getFont(FontFactory.HELVETICA)
              ;
        font.setColor(BaseColor.WHITE);
        font.setStyle(Font.BOLD);
        PdfPCell cell = new PdfPCell();
        cell.setBackgroundColor(customColor);
        cell.setPadding(5);
        cell.setPhrase(new Phrase("Alias", font));
        table.addCell(cell);
        cell.setPhrase(new Phrase("Verification Code", font));
        table.addCell(cell);
        cell.setPhrase(new Phrase("Tracker", font));
        table.addCell(cell);
        Font tableFont = FontFactory.getFont(FontFactory.
              HELVETICA);
        tableFont.setSize(8f);
        for (Voter voter : voters) {
            PdfPCell newCell = new PdfPCell();
            newCell.setPhrase(new Phrase(voter.getAlias(),
                  tableFont));
            table.addCell(newCell);
            newCell.setPhrase(new Phrase(voter.
                  getVerificationCode(), tableFont));
            table.addCell(newCell);
            newCell.setPhrase(new Phrase(voter.getTracker(),
                  tableFont));
            table.addCell(newCell);
        }
        doc.add(table);
    }
}
```

8.3. ResultsPdfBuilder.java

```
package ph.edu.upm.agila.japiscos.pdf;

import java.math.RoundingMode;
import java.text.DecimalFormat;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import ph.edu.upm.agila.japiscos.model.Abstain;
import ph.edu.upm.agila.japiscos.model.Candidate;
import ph.edu.upm.agila.japiscos.model.Position;
import ph.edu.upm.agila.japiscos.model.Voter;

import com.itextpdf.text.BaseColor;
import com.itextpdf.text.Document;
import com.itextpdf.text.Font;
import com.itextpdf.text.FontFactory;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.pdf.PdfPCell;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;

public class ResultsPdfBuilder extends AbstractITextPdfView {

    @SuppressWarnings("unchecked")
    @Override
    protected void buildPdfDocument(Map<String, Object> model,
            Document doc,
             PdfWriter writer, HttpServletRequest request,
             HttpServletResponse response) throws Exception {
        List<Candidate> candidates = (List<Candidate>) model.
              get("candidates");
        List<Abstain> abstains = (List<Abstain>) model.get("
              abstains");
        List<Position> positions = (List<Position>) model.get(
              "positions");
        List<Voter> voters = (List<Voter>) model.get("voters")
              ;
        int voterCount = 0;
        if (!candidates.isEmpty()) {
            Font siteFont = FontFactory.getFont(FontFactory.
                  HELVETICA);
            BaseColor customColor = new BaseColor(48, 165,
                  255);
            siteFont.setColor(customColor);
            siteFont.setSize(15f);
            siteFont.setStyle(Font.BOLD);
            doc.add(new Paragraph("iVote", siteFont));
            Font titleFont = FontFactory.getFont(FontFactory.
                  HELVETICA);
            titleFont.setColor(BaseColor.BLACK);
            titleFont.setSize(14f);
            titleFont.setStyle(Font.BOLD);
            doc.add(new Paragraph(candidates.get(0).
                  getElection().getName() + " Election Results
                  ", titleFont));
            for (Position position : positions) {
                Font positionFont = FontFactory.getFont(
                      FontFactory.HELVETICA);
                positionFont.setColor(BaseColor.BLACK);
                positionFont.setSize(13f);
                positionFont.setStyle(Font.UNDERLINE);
                doc.add(new Paragraph(position.getName(),
                      positionFont));
                PdfPTable table = new PdfPTable(3);
                table.setWidthPercentage(100.0f);
                table.setWidths(new float[] { 3.0f, 2.0f, 2.0f
                      });
                table.setSpacingBefore(10);
                Font font = FontFactory.getFont(FontFactory.
                      HELVETICA);
                font.setColor(BaseColor.WHITE);
                font.setStyle(Font.BOLD);
                PdfPCell cell = new PdfPCell();
                cell.setBackgroundColor(customColor);
                cell.setPadding(5);
                cell.setPhrase(new Phrase("Candidate", font));
                table.addCell(cell);
                cell.setPhrase(new Phrase("Party", font));
                table.addCell(cell);
                cell.setPhrase(new Phrase("Votes", font));
                table.addCell(cell);
                for (Candidate candidate : candidates) {
                    if (candidate.getPosition().getId() ==
                          position.getId()) {
                        table.addCell(candidate.getLastName().
                              toUpperCase() + ", " + candidate
                              .getFirstName());
                        table.addCell(candidate.getParty().
                              getName());
                        table.addCell(String.valueOf(candidate
                              .getVoteCount()));
                    }
                }
                if (position.isAbstain()) {
                    for (Abstain abstain : abstains) {
                        if (abstain.getPosition().getId() ==
                              position.getId()) {
                            table.addCell("ABSTAIN");
                            table.addCell("-");
                            table.addCell(String.valueOf(
                                  abstain.getVoteCount()));
                        }
                    }
                }
                doc.add(table);
            }
            for (Voter voter : voters) {
                if (voter.isVoted()) {
                    voterCount++;
                }
            }
            doc.add(new Paragraph("Number of Voters Who Voted:
                  " + voterCount, titleFont));
            doc.add(new Paragraph("Total Number of Voters: " +
                  voters.size(), titleFont));
            DecimalFormat df = new DecimalFormat("#.###");
            df.setRoundingMode(RoundingMode.CEILING);
            doc.add(new Paragraph("Voter Turnout: " + df.
                  format((float) voterCount * 100/ voters.size
                  ())
                + "%", titleFont));
        }
    }
}
```

9. Validators

9.1. BallotValidator.java

```
package ph.edu.upm.agila.japiscos.validator;

import java.util.List;

import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import ph.edu.upm.agila.japiscos.formbacker.Ballot;
import ph.edu.upm.agila.japiscos.formbacker.Vote;

public class BallotValidator implements Validator {

    @Override
    public boolean supports(Class<?> clazz) {
        return Ballot.class.equals(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        Ballot ballot = (Ballot) target;
        List<Vote> votes = ballot.getVotes();
        for (Vote vote : votes) {
            if (vote.getCandidateIds() == null) {
                errors.rejectValue("votes", "required.votes");
                break;
            }
            if (vote.getCandidateIds() != null
                    && vote.getCandidateIds().length == 0) {
                errors.rejectValue("votes", "required.votes");
                break;
            }

        }
    }
}
```

9.2. CandidateFormValidator.java

```
package ph.edu.upm.agila.japiscos.validator;

import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import ph.edu.upm.agila.japiscos.formbacker.CandidateBacker;

public class CandidateFormValidator implements Validator {

    @Override
    public boolean supports(Class<?> clazz) {
        return CandidateBacker.class.equals(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        CandidateBacker candidateBacker = (CandidateBacker)
            target;
        if (candidateBacker.getPositionId() == 0) {
            errors.rejectValue("positionId", "empty.position")
                ;
        }
        if (candidateBacker.getPartyId() == 0) {
            errors.rejectValue("partyId", "empty.party");
        }
    }

}
```

9.3. FileValidator.java

```
package ph.edu.upm.agila.japiscos.validator;

import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import ph.edu.upm.agila.japiscos.formbacker.ImportVoterForm;

public class FileValidator implements Validator {

    @Override
    public boolean supports(Class<?> clazz) {
        return ImportVoterForm.class.equals(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        ImportVoterForm form = (ImportVoterForm) target;
        if (form.getFile().isEmpty()) {
            errors.rejectValue("file", "empty.file");
            return;
        }
        if (!form.getFile().getContentType().equalsIgnoreCase(
            "text/csv")) {
            errors.rejectValue("file", "type.file");
        }

    }

}
```

9.4. PasswordValidator.java

```
package ph.edu.upm.agila.japiscos.validator;

import org.springframework.security.crypto.bcrypt.BCrypt;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

import ph.edu.upm.agila.japiscos.formbacker.ChangePasswordForm
    ;
import ph.edu.upm.agila.japiscos.model.User;

public class PasswordValidator implements Validator {

    private User user;

    public PasswordValidator(User user) {
        this.user = user;
    }

    @Override
    public boolean supports(Class<?> clazz) {
        return ChangePasswordForm.class.equals(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        ChangePasswordForm form = (ChangePasswordForm) target;
        if (!BCrypt.checkpw(form.getOldPassword(), user.
            getPasswordHash())) {
            errors.rejectValue("oldPassword", "NotMatched.
                oldPassword");
        }
        if (!form.getNewPassword().equals(form.
            getNewPasswordRepeated())) {
            errors.rejectValue("newPassword", "NotMatched.
                newPassword");
            errors.rejectValue("newPasswordRepeated", "
                NotMatched.newPassword");
        }
    }
}
```

10. Exceptions

10.1. EmailExistsException.java

```
package ph.edu.upm.agila.japiscos.exception;

public class EmailExistsException extends Exception {

    private static final long serialVersionUID =
        5479039918234896459L;

    private String email;

    public EmailExistsException() {
        super();
    }

    public EmailExistsException(String email) {
        this.email = email;
    }

    public String getEmail() {
        return email;
    }

}
```

10.2. IncorrectCSVFormatException.java

```
package ph.edu.upm.agila.japiscos.exception;

public class IncorrectCSVFormatException extends Exception {

    private static final long serialVersionUID =
        -904138772380882959L;

    public IncorrectCSVFormatException() {
        super();
    }
}
```

10.3. InvalidEmailException.java

```
package ph.edu.upm.agila.japiscos.exception;

public class InvalidEmailException extends Exception {

    private static final long serialVersionUID =
          1138539645421657922L;

    private String username;

    private String email;

    public InvalidEmailException() {
        super();
    }

    public InvalidEmailException(String username, String email
          ) {
        this.username = username;
        this.email = email;
    }

    public String getUsername() {
        return username;
    }

    public String getEmail() {
        return email;
    }

}
```

10.4. SharemindException.java

```
package ph.edu.upm.agila.japiscos.exception;

public class SharemindException extends Exception {

    private static final long serialVersionUID =
          -584877235203143591 2L;

    public SharemindException() {
        super();
    }

}
```

10.5. UsernameExistsException.java

```
package ph.edu.upm.agila.japiscos.exception;

public class UsernameExistsException extends Exception {

    private static final long serialVersionUID =
          -3744897550762456672L;

    private String username;

    public UsernameExistsException() {
        super();
    }

    public UsernameExistsException(String username) {
        this.username = username;
    }

    public String getUsername() {
        return username;
    }
}
```

11. Properties

11.1. application.properties

```
# DataSource settings: set here configurations for the
      database connection
spring.datasource.url = jdbc:mysql://localhost:3306/ivote
spring.datasource.username = root
spring.datasource.password =
spring.datasource.driverClassName = com.mysql.jdbc.Driver

# Specify the DBMS
spring.jpa.database = MYSQL

# Show or not log for each sql query
spring.jpa.show-sql = true

# Hibernate settings are prefixed with spring.jpa.hibernate.*
spring.jpa.hibernate.ddl-auto = update
spring.jpa.hibernate.dialect = org.hibernate.dialect.
      MySQL5Dialect
spring.jpa.hibernate.naming_strategy = org.hibernate.cfg.
      ImprovedNamingStrategy
spring.jpa.hibernate.jdbc.batch_size = 20

# Email (MailProperties)
spring.mail.host = smtp.gmail.com
spring.mail.port = 587
spring.mail.username = ivote.sharemind@gmail.com
spring.mail.password = bogdanov
spring.mail.properties.mail.smtp.auth= true
spring.mail.properties.mail.smtp.starttls.enable = true
spring.mail.properties.mail.smtp.ssl.trust = smtp.gmail.com

# File Upload
multipart.maxFileSize = 128KB
multipart.maxRequestSize = 128KB
```

11.2. messages.properties

```
NotEmpty.election.name = Election name is required!
Length.election.name = Name must be between 5-50 characters.

NotEmpty.positionBacker.name = Position name is required!
Length.positionBacker.name = Name must be between 5-20
      characters.
NotNull.positionBacker.maxElected = Maximum number is required
      !
Min.positionBacker.maxElected = Maximum number should be at
      least 1.
typeMismatch.postionBacker.maxElected = Maximum number must be
       an integer value.
NotNull.positionBacker.ordinality = Ordinality is required!
Min.positionBacker.ordinality = Ordinality should be > 0.
typeMismatch.postionBacker.ordinality = Ordinality must be an
      integer value.

NotEmpty.party.name = Party name is required!
Length.party.name = Name must be between 5-20 characters.
NotEmpty.party.elections = Select an election!

NotEmpty.candidateBacker.firstName = First name is required!
Length.candidateBacker.firstName = First name must be between
      2-30 characters.
NotEmpty.candidateBacker.lastName = Last name is required!
Length.candidateBacker.lastName = Last name must be between
      2-30 characters.

NotEmpty.voterType.name = Voter type name is required!
Length.voterType.name = Name must be between 5-20 characters.
NotEmpty.voterType.elections = Select an election!

NotEmpty.userCreateForm.username = Username is required!
Length.userCreateForm.username = Username must be between 5-20
       characters.
NotEmpty.userCreateForm.email = Email address is required!
Length.userCreateForm.email = Email must be between 5-30
      characters.
Email.userCreateForm.email = Email address should be a valid
      email address.
NotNull.userCreateForm.role = Select a role!

NotEmpty.userEditForm.username = Username is required!
Length.userEditForm.username = Username must be between 5-20
      characters.
NotEmpty.userEditForm.email = Email address is required!
Length.userEditForm.email = Email must be between 5-30
      characters.
Email.userEditForm.email = Email address should be a valid
      email address.
NotNull.userEditForm.role = Select a role!

NotEmpty.voterCreateForm.username = Username is required!
Length.voterCreateForm.username = Username must be between
      5-20 characters.
NotEmpty.voterCreateForm.email = Email address is required!
Length.voterCreateForm.email = Email must be between 5-30
      characters.
Email.voterCreateForm.email = Email address should be a valid
      email address.
NotNull.voterCreateForm.voterType = Select a voter type!

NotEmpty.voterEditForm.username = Username is required!
Length.voterEditForm.username = Username must be between 5-20
      characters.
NotEmpty.voterEditForm.email = Email address is required!
Length.voterEditForm.email = Email must be between 5-30
      characters.
Email.voterEditForm.email = Email address should be a valid
      email address.
NotNull.voterEditForm.voterType = Select a voter type!

NotEmpty.changePasswordForm.oldPassword = Old Password is
      required!
NotEmpty.changePasswordForm.newPassword = New Password is
      required!

NotNull.importVoterForm.file = File is required!
NotNull.importVoterForm.voterType = Voter type is required!

required.votes = Vote is required!
NotMatched.oldPassword = Wrong password!
NotMatched.newPassword = Passwords do not match!
empty.position = Add a position first!
empty.party = Add a party first!

empty.file = File is empty!
type.file = File should be a CSV file!
size.file = Maximum upload size is 128KB;
```

11.3. views.properties

```
pdfView.(class)=ph.edu.upm.agila.japiscos.pdf.
      ResultsPdfBuilder
```

```
pdfBulletinBoardView.(class)=ph.edu.upm.agila.japiscos.pdf.
      BulletinBoardPdfBuilder
```

12. HTML Pages

12.1. header.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
</head>
<body>
<nav th:fragment="header" class="navbar navbar-inverse navbar-
      fixed-top" role="navigation">
<div class="container-fluid">
 <div class="navbar-header">
  <button type="button" class="navbar-toggle collapsed" data-
        toggle="collapse" data-target="#sidebar-collapse">
   <span class="sr-only">Toggle navigation</span>
   <span class="icon-bar"></span>
   <span class="icon-bar"></span>
   <span class="icon-bar"></span>
  </button>
  <a class="navbar-brand" href="/home">
   <img th:src="@{/img/logo.png}"/>
  </a>
  <ul class="user-menu" sec:authorize="isAuthenticated()">
   <li class="dropdown pull-right">
    <a href="#" class="dropdown-toggle" data-toggle="dropdown"
          ><span class="glyphicon glyphicon-user"></span>
     <span sec:authentication="principal.username"></span
          ><span class="caret"></span></a>
    <ul class="dropdown-menu" role="menu">
     <li><a th:href="@{|/user/${#authentication.principal.user
           .id}|}"><span class="glyphicon glyphicon-user"></
           span> Profile</a></li>
     <li><a th:href="@{|/user/edit/${#authentication.principal
           .user.id}|}"><span class="glyphicon glyphicon-cog">
           </span> Settings</a></li>
     <li role="presentation" class="divider"></li>
     <li>
      <form id="logout" th:action="@{/logout}" method="post">
       <input type="hidden" th:name="${_csrf.parameterName}"
             th:value="${_csrf.token}"/>
      </form>
      <a href="#" onclick="document.getElementById('logout').
            submit(); return false;"><span class="glyphicon
            glyphicon-log-out"></span> Log out</a>
     </li>
    </ul>
   </li>
  </ul>
 </div>
</div><!-- /.container-fluid -->
</nav>
</body>
</html>
```

12.2. navigation.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
</head>
<body>
<div th:fragment="navigation" id="sidebar-collapse" class="col
      -sm-3 col-lg-2 sidebar">
<ul class="nav menu">
 <li th:class="${activeSettingsElection}" sec:authorize="
       hasRole('VOTING_OFFICIAL')">
  <a th:href="@{/elections}"><span class="glyphicon glyphicon-
        tasks"></span> Elections</a>
 </li>
 <li th:class="${activeSettingsPosition}" sec:authorize="
       hasRole('VOTING_OFFICIAL')">
  <a th:href="@{/positions}"><span class="glyphicon glyphicon-
        th-list"></span> Positions</a>
 </li>
 <li th:class="${activeSettingsParty}" sec:authorize="hasRole
       ('VOTING_OFFICIAL')">
  <a th:href="@{/parties}"><span class="glyphicon glyphicon-
        bullhorn"></span> Parties</a>
 </li>
 <li th:class="${activeSettingsCandidate}" sec:authorize="
       hasRole('VOTING_OFFICIAL')">
  <a th:href="@{/candidates}"><span class="glyphicon glyphicon
        -list"></span> Candidates</a>
 </li>
 <li th:class="${activeSettingsVoterType}" sec:authorize="
       hasRole('VOTING_OFFICIAL')">
  <a th:href="@{/voter-types}"><span class="glyphicon
        glyphicon-th"></span> Voter Types</a>
 </li>
 <li th:class="${activeSettingsVotingOfficial}" sec:authorize=
       "hasRole('ADMIN')">
  <a th:href="@{/voting-officials}"><span class="glyphicon
        glyphicon-user"></span> Voting Officials</a>
 </li>
 <li th:class="${activeSettingsVoter}" sec:authorize="hasRole
       ('VOTING_OFFICIAL')">
  <a th:href="@{/voters}"><span class="glyphicon glyphicon-
        user"></span> Voters</a>
 </li>
 <li th:class="${activeSettingsVote}" sec:authorize="hasRole('
       VOTER')">
  <a th:href="@{/vote}"><span class="glyphicon glyphicon-edit"
        ></span> Vote</a>
 </li>
 <li th:class="${activeSettingsResults}">
  <a th:href="@{/results}"><span class="glyphicon glyphicon-
        search"></span> View Results</a>
 </li>
 <li th:class="${activeSettingsBulletinBoard}">
  <a th:href="@{/bulletin-board}"><span class="glyphicon
        glyphicon-list-alt"></span> Bulletin Board</a>
 </li>
</ul>
<div class="attribution">Template by <a href="http://www.
      medialoot.com/item/lumino-admin-bootstrap-template/">
      Medialoot</a></div>
</div><!--/.sidebar-->
</body>
</html>
```

12.3. add-candidate.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Candidate</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<script>
function getPositions() {
 var id = $('#electionId').val();
 var positions;
 $.ajax({
  async: false,
  url: '../positions/get-positions',
  data: {electionId : id},
  success: function(response){
    positions = response;
  }
 });
 $("#positionId").find('option').remove();
 $("#positionHelp").empty();
 if(positions.length != 0){
  $.each(positions, function (i, item) {
   $('#positionId').append(new Option(item.name, item.id));
  });
 } else {
   $('#positionId').append(new Option("No positions found", 0)
         );
   $('#positionHelp').append('Add a position <a href="
         positions/add">here</a>');
 }
}
function getParties() {
 var id = $('#electionId').val();
 var parties;
 $.ajax({
  async: false,
  url: '../parties/get-parties',
  data: {electionId : id},
  success: function(response){
    parties = response;
  }
 });
 $("#partyId").find('option').remove();
 $("#partyHelp").empty();
 if(parties.length != 0){
  $.each(parties, function (i, item) {
   $('#partyId').append(new Option(item.name, item.id));
  });
 } else {
  $('#partyId').append(new Option("No parties found", 0));
  $('#partyHelp').append('Add a party <a href="parties/add">
        here</a>');
 }
}
</script>
<body onload="getPositions(); getParties();">
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{/candidates}">Candidates</a></li>
  <li class="active">Add Candidate</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Add Candidate</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-plus"></span> Add
         Candidate
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form id="candidate-form" role="form" th:object="${
            candidateBacker}" th:action="@{/candidates/add}"
```

```
        method="post">
      <input type="hidden" th:name="${_csrf.parameterName}"
            th:value="${_csrf.token}" />
      <div class="form-group" th:classappend="${#fields.
            hasErrors('firstName')}? 'has-error'">
       <label>First Name*</label>
       <input class="form-control" th:field="*{firstName}"
             placeholder="Enter Candidate's First Name" />
       <p class="help block text-danger" th:if="${#fields.
             hasErrors('firstName')}"
        th:errors="*{firstName}">
        Error in first name
       </p>
      </div>
      <div class="form-group" th:classappend="${#fields.
            hasErrors('lastName')}? 'has-error'">
       <label>Last Name*</label>
       <input class="form-control" th:field="*{lastName}"
             placeholder="Enter Candidate's Last Name" />
       <p class="help block text-danger" th:if="${#fields.
             hasErrors('lastName')}"
        th:errors="*{lastName}">
        Error in last name
       </p>
      </div>
      <div class="form-group" >
       <label>Election*</label>
       <select class="form-control" th:field="*{electionId}"
             onchange="getPositions(); getParties();">
        <th:block th:each ="election : ${electionList}">
         <option th:value="${election.id}" th:text="${
               election.name}" >Election</option>
        </th:block>
       </select>
      </div>
      <div class="form-group" th:classappend="${#fields.
            hasErrors('positionId')}? 'has-error'">
       <label>Position*</label>
       <select class="form-control" th:field="*{positionId}">
       </select>
       <p class="help block text-danger" th:if="${#fields.
             hasErrors('positionId')}"
        th:errors="*{positionId}">
        Error in position
       </p>
       <p id="positionHelp" class="help block"></p>
      </div>
      <div class="form-group" th:classappend="${#fields.
            hasErrors('partyId')}? 'has-error'">
```

12.4. add-election.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Election</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{/elections}">Elections</a></li>
  <li class="active">Add Election</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Add Election</h1>
   <th:block th:if="${successMessage != null}">
    <div class="alert alert-success" th:text="${successMessage
          }">Success Message</div>
   </th:block>
   <th:block th:if="${errorMessage != null}">
    <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
   </th:block>
   <div class="panel panel-default">
    <div class="panel-heading">
     <span class="glyphicon glyphicon-plus"></span> Add
           Election
    </div>
    <div class="panel-body">
     <div class="row">
      <div class="col-lg-6">
       <form role="form" th:object="${election}" th:action="@
             {/elections/add}" method="post">
        <input type="hidden" th:name="${_csrf.parameterName}"
              th:value="${_csrf.token}" />
        <div class="form-group" th:classappend="${#fields.
              hasErrors('name')}? 'has-error'">
         <label>Election Name*</label>
         <input class="form-control" th:field="*{name}"
               placeholder="Enter Election Name" />
         <p class="help block text-danger" th:if="${#fields.
               hasErrors('name')}"
          th:errors="*{name}">
          Error in election name
         </p>
        </div>
        <div class="form-group">
         <label>Parent</label>
         <select class="form-control" th:field="*{
               parentElectionId}">
          <option th:value="0">No Parent</option>
          <th:block th:if="${electionList != null}" th:each ="
                electionItem : ${electionList}">
           <option th:value="${electionItem.id}" th:text="${
                 electionItem.name}">Election</option>
          </th:block>
         </select>
        </div>
        <input type="submit" class="btn btn-default" value="
              Submit"/>
        <input type="reset" class="btn btn-default" value="
              Reset" />
       </form>
      </div>
     </div>
     <!-- /.row (nested) -->
    </div>
    <!-- /.panel-body -->
   </div>
   <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.5. add-party.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Party</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{/elections}">Elections</a></li>
  <li class="active">Add Party</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Add Party</h1>
   <th:block th:if="${successMessage != null}">
    <div class="alert alert-success" th:text="${successMessage
          }">Success Message</div>
   </th:block>
   <th:block th:if="${errorMessage != null}">
    <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
   </th:block>
   <div class="panel panel-default">
    <div class="panel-heading">
     <span class="glyphicon glyphicon-plus"></span> Add
           Party
    </div>
    <div class="panel-body">
     <div class="row">
      <div class="col-lg-6">
       <form role="form" th:object="${party}" th:action="@
             {/parties/add}" method="post">
        <input type="hidden" th:name="${_csrf.parameterName}"
              th:value="${_csrf.token}" />
        <div class="form-group" th:classappend="${#fields.
              hasErrors('name')}? 'has-error'">
         <label>Party*</label>
         <select class="form-control" th:field="*{partyId}">
         </select>
         <p class="help block text-danger" th:if="${#fields.
               hasErrors('partyId')}"
          th:errors="*{partyId}">
          Error in party
         </p>
         <p id="partyHelp" class="help block"></p>
        </div>
        <input type="submit" class="btn btn-default" value="
              Submit"/>
        <input type="reset" class="btn btn-default" value="
              Reset" />
       </form>
      </div>
     </div>
     <!-- /.row (nested) -->
    </div>
    <!-- /.panel-body -->
   </div>
   <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
 //<![CDATA[
 !function ($) {
  $(document).on("click","ul.nav li.parent > a > span.icon",
        function(){
   $(this).find('em:first').toggleClass("glyphicon-minus");
  });
  $(".sidebar span.icon").find('em:first').addClass("glyphicon
        -plus");
 }(window.jQuery);
 $(window).on('resize', function () {
   if ($(window).width() > 768) $('#sidebar-collapse').
        collapse('show')
 })
 $(window).on('resize', function () {
   if ($(window).width() <= 767) $('#sidebar-collapse').
        collapse('hide')
 })
 //]]>
</script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Election</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<link th:href="@{/css/chosen.css}" rel="stylesheet" type="text
      /css" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{/parties}">Parties</a></li>
  <li class="active">Add Party</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Add Party</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-plus"></span> Add Party
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${party}" th:action="@{/
            parties/add}" method="post">
       <input type="hidden" th:name="${_csrf.parameterName}"
             th:value="${_csrf.token}" />
       <div class="form-group" th:classappend="${#fields.
             hasErrors('name')}? 'has-error'">
        <label>Party Name*</label>
        <input class="form-control" th:field="*{name}"
              placeholder="Enter Party Name" />
        <p class="help block text-danger" th:if="${#fields.
              hasErrors('name')}"
         th:errors="*{name}">
         Error in party name
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
             hasErrors('elections')}? 'has-error'">
        <label>Elections*</label><br/>
        <th:block th:if="${party.elections == null}">
         <select th:if="${party.elections == null}" class="
               chosen-select form-control" multiple="multiple"
                name="elections" data-placeholder="Choose an
               Election"
           tabindex="4">
          <option th:each="election : ${electionList}" th:
                value="${election.id}" th:text="${election.
                name}" />
         </select>
        </th:block>
        <th:block th:if="${party.elections != null}" >
         <select class="chosen-select form-control" multiple="
               multiple" name="elections" data-placeholder="
               Choose an Election"
           tabindex="4">
          <option th:each="election : ${electionList}" th:
                value="${election.id}" th:text="${election.
                name}"
                th:selected="${party.elections.contains(
                election)}"/>
         </select>
        </th:block>
        <p class="help block text-danger" th:if="${#fields.
              hasErrors('elections')}"
         th:errors="*{elections}">
         Error in elections
        </p>
       </div>
       <input type="submit" class="btn btn-default" value="
             Submit"/>
       <input type="reset" class="btn btn-default" value="
             Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script th:src="@{/js/chosen.jquery.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
<script type="text/javascript">
var config = {
  '.chosen-select'           : {},
  '.chosen-select-deselect'  : {allow_single_deselect:true},
  '.chosen-select-no-single' : {disable_search_threshold:10},
  '.chosen-select-no-results': {no_results_text:'Oops, nothing
       found!'},
  '.chosen-select-width'     : {width:"95%"}
}
for (var selector in config) {
  $(selector).chosen(config[selector]);
}
</script>
</body>
</html>
```

12.6. add-position.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Position</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{/positions}">Positions</a></li>
  <li class="active">Add Position</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Add Position</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-plus"></span> Add
         Position
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${positionBacker}" th:
            action="@{/positions/add}" method="post">
       <input type="hidden" th:name="${_csrf.parameterName}"
             th:value="${_csrf.token}" />
       <div class="form-group" th:classappend="${#fields.
             hasErrors('name')}? 'has-error'">
        <label>Position Name*</label>
        <input class="form-control" th:field="*{name}"
              placeholder="Enter Position Name" />
        <p class="help block text-danger" th:if="${#fields.
              hasErrors('name')}"
         th:errors="*{name}">
         Error in position name
        </p>
       </div>
       <div class="form-group" >
        <label>Election*</label>
        <select class="form-control" th:field="*{electionId}">
         <th:block th:each ="election : ${electionList}">
          <option th:value="${election.id}" th:text="${
               election.name}" >Election</option>
         </th:block>
        </select>
       </div>
       <div class="form-group" th:classappend="${#fields.
             hasErrors('maxElected')}? 'has-error'">
        <label>Maximum number of candidates to be elected for
              the position*</label>
        <input class="form-control" type="number" th:field="*{
              maxElected}" />
        <p class="help block text-danger" th:if="${#fields.
              hasErrors('maxElected')}"
         th:errors="*{maxElected}">
```

```
          Error in maxElected
         </p>
        </div>
        <div class="form-group" th:classappend="${#fields.
              hasErrors('ordinality')}? 'has-error'">
         <label>Ordinality (order in the ballot)*</label>
         <input class="form-control" type="number" th:field="*{
               ordinality}" />
         <p class="help block text-danger" th:if="${#fields.
               hasErrors('ordinality')}"
          th:errors="*{ordinality}">
          Error in ordinality
         </p>
        </div>
        <div class="checkbox">
         <label>
          <input type="checkbox" th:field="*{abstain}"/>
               Abstain
         </label>
        </div>
        <input type="submit" class="btn btn-default" value="
              Submit"/>
        <input type="reset" class="btn btn-default" value="
              Reset" />
       </form>
      </div>
     </div>
     <!-- /.row (nested) -->
    </div>
    <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.7. add-voter.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Voter</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li><a th:href="@{/voters}">Voters</a></li>
  <li class="active">Add Voter</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Add Voter</h1>
  <div class="alert bg-primary" role="alert">
   <span class="glyphicon glyphicon-info-sign"></span> Adding
        a voter with the same username/email as a previous
        voter record will update that record.
  </div>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
        ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-plus"></span> Add Voter
    <div class="pull-right">
     <a href="/voters/import" type="button" class="btn btn-
          primary" >
      <span class="glyphicon glyphicon-upload"></span> Import
           CSV file
     </a>
    </div>
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${voterCreateForm}" th:
           action="@{/voters/add}" method="post">
       <input type="hidden" th:name="${_csrf.parameterName}"
            th:value="${_csrf.token}" />
       <div class="form-group" th:classappend="${#fields.
            hasErrors('username')}? 'has-error'">
        <label>Username:</label>
        <input class="form-control" th:field="*{username}"
             placeholder="Enter Username" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('username')}"
         th:errors="*{username}">
         Error in username
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
            hasErrors('email')}? 'has-error'">
        <label>Email:</label>
        <input class="form-control" type="text" th:field="*{
             email}" placeholder="Enter Email" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('email')}"
         th:errors="*{email}">
         Error in email
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
            hasErrors('voterType')}? 'has-error'">
        <label>Voter Types:</label>
        <select class="form-control" th:field="*{voterType}" >
         <option th:each="voterType : ${voterTypeList}" th:
              value="${voterType.id}"
           th:text="${voterType.name}"></option>
        </select>
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('voterType')}"
         th:errors="*{voterType}">
         Error in voter type
        </p>
       </div>
       <input type="submit" class="btn btn-default" value="
            Submit"/>
       <input type="reset" class="btn btn-default" value="
            Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.8. add-voter-type.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Voter Type</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<link th:href="@{/css/chosen.css}" rel="stylesheet" type="text
      /css" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li><a th:href="@{/voter-types}">Voter Types</a></li>
  <li class="active">Add Voter Type</li>
```

```
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Add Voter Type</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
        ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-plus"></span> Add Voter
         Type
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${voterType}" th:action="@
           {/voter-types/add}" method="post">
       <input type="hidden" th:name="${_csrf.parameterName}"
            th:value="${_csrf.token}" />
       <div class="form-group" th:classappend="${#fields.
            hasErrors('name')}? 'has-error'">
        <label>Voter Type Name*</label>
        <input class="form-control" th:field="*{name}"
             placeholder="Enter Voter Type Name" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('name')}"
         th:errors="*{name}">
         Error in voter type name
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
            hasErrors('elections')}? 'has-error'">
        <label>Elections*</label><br/>
        <th:block th:if="${voterType.elections == null}">
         <select th:if="${voterType.elections == null}" class=
              "chosen-select form-control" multiple="multiple
              " name="elections" data-placeholder="Choose an
              Election"
           tabindex="4">
          <option th:each="election : ${electionList}" th:
               value="${election.id}" th:text="${election.
               name}" />
         </select>
        </th:block>
        <th:block th:if="${voterType.elections != null}">
         <select th:if="${voterType.elections != null}" class=
              "chosen-select form-control" multiple="multiple
              " name="elections" data-placeholder="Choose an
              Election"
           tabindex="4">
          <option th:each="election : ${electionList}" th:
               value="${election.id}" th:text="${election.
               name}"
                th:selected="${voterType.elections.contains(
                     election)}"/>
         </select>
        </th:block>
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('elections')}"
         th:errors="*{elections}">
         Error in elections
        </p>
       </div>
       <input type="submit" class="btn btn-default" value="
            Submit"/>
       <input type="reset" class="btn btn-default" value="
            Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script th:src="@{/js/chosen.jquery.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
      function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
      plus");
}(window.jQuery);

$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
<script type="text/javascript">
var config = {
  '.chosen-select'           : {},
  '.chosen-select-deselect'  : {allow_single_deselect:true},
  '.chosen-select-no-single' : {disable_search_threshold:10},
  '.chosen-select-no-results': {no_results_text:'Oops, nothing
       found!'},
  '.chosen-select-width'     : {width:"95%"}
}
for (var selector in config) {
  $(selector).chosen(config[selector]);
}
</script>
</body>
</html>
```

12.9. add-voting-official.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Voting Official</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
     main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li><a th:href="@{/voting-officials}">Voting Officials</a></
       li>
  <li class="active">Add Voting Official</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
 <h1 class="page-header">Add Voting Official</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
        ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-plus"></span> Add Voting
         Official
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${userCreateForm}" th:
           action="@{/voting-officials/add}" method="post">
       <input type="hidden" th:name="${_csrf.parameterName}"
            th:value="${_csrf.token}" />
       <input type="hidden" th:name="role" th:value="*{role}"
            />
       <div class="form-group" th:classappend="${#fields.
            hasErrors('username')}? 'has-error'">
        <label>Username:</label>
        <input class="form-control" th:field="*{username}"
             placeholder="Enter Username" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('username')}"
         th:errors="*{username}">
         Error in username
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
            hasErrors('email')}? 'has-error'">
        <label>Email:</label>
        <input class="form-control" type="text" th:field="*{
             email}" placeholder="Enter Email" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('email')}"
         th:errors="*{email}">
         Error in email
        </p>
       </div>
       <input type="submit" class="btn btn-default" value="
            Submit"/>
       <input type="reset" class="btn btn-default" value="
            Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
      function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
      plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
```

```
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
        collapse('hide')
})
```

12.10. bulletin-board.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Bulletin Board</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/bootstrap-table.css}" rel="stylesheet" /
      >
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li class="active">Bulletin Board</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Bulletin Board</h1>
  <th:block th:if="${errorMessage != null}">
     <div class="alert alert-danger" th:text="${errorMessage}"
          >Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-list-alt"></span> Votes
         Recorded
    <div class="pull-right" th:if="${!recordedVoters.isEmpty()
         }">
     <a class="btn btn-primary" th:href="@{/bulletin-board/
          export}"><span class="glyphicon glyphicon-download-
          alt"></span> Export Bulletin Board</a>
    </div>
    </div>
   <div class="panel-body">
    <table data-toggle="table" data-show-refresh="true" data-
         search="true" data-pagination="true" data-sort-name=
         "name" data-sort-order="asc" th:if="${recordedVoters
          != null}" >
     <thead>
      <tr>
       <th data-field="alias" data-sortable="true">Alias</th>
       <th data-field="verification-code" data-sortable="true"
            >Verification Code</th>
      </tr>
     </thead>
     <tbody>
      <th:block th:each="voter, iter : ${recordedVoters}">
       <tr>
       <td th:text="${voter.alias}"></td>
       <td>
        <span th:text="${voter.verificationCode}"></span>
        <span><a th:href="@{/bulletin-board/view-tracker/__${
             voter.alias}__}">[view]</a></span>
       </td>
       </tr>
      </th:block>
     </tbody>
    </table>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script th:src="@{/js/bootstrap-table.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
        ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
        collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.11. candidates.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Candidates</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<script>
function getCandidates(){
 var id = $('#electionId').val();
 $('#candidates-div').empty();

 var positions;
 $.ajax({
  async: false,
  url: 'positions/get-positions',
  data: {electionId : id},
  success: function(response){
    positions = response;
  }
 });

 var candidates;
 $.ajax({
  async: false,
  url: 'candidates/get-candidates',
  data: {electionId : id},
  success: function(response){
   candidates = response;
  }
 });

 if(positions.length == 0){
  $('#candidates-div').append('No positions found.<br/><br/>')
       ;
 } else {
  $.each(positions, function(h, item1){
   var candidateNumber = 0;
   $('#candidates-div').append("<h4>" + item1.name + " (" +
        item1.maxElected + ")" + "</h4>");
   var table = document.createElement('table');
   table.setAttribute('class', 'table table-striped');
   var thead  = document.createElement('thead');
   var tbody  = document.createElement('tbody');
   var th = document.createElement('th');
   th.innerHTML = "No.";
   thead.appendChild(th);
   th = document.createElement('th');
   th.innerHTML = "Candidate";
   thead.appendChild(th);
   th = document.createElement('th');
   th.innerHTML = "Party";
   thead.appendChild(th);
   th = document.createElement('th');
   th.innerHTML = "Actions";
   thead.appendChild(th);
   table.appendChild(thead);
   var candidateCount = 0;
   $.each(candidates, function (i, item) {
    if(item.election.id == item1.election.id){
     if(item.position.id == item1.id){
      candidateNumber++;
      candidateCount++;
      var editStr= 'candidates/edit/' + item.id;
      var deleteStr= 'candidates/delete/' + item.id;

      var editLink = document.createElement('a');
      editLink.setAttribute('href', editStr);
      editLink.setAttribute('class', 'btn btn-primary btn-
           circle');
      editLink.setAttribute('style', 'margin-right:5px;');
      editLink.innerHTML = '<span class="glyphicon glyphicon-
           pencil"></span>';

      var deleteLink = document.createElement('a');
      deleteLink.onclick = function(){
       return confirm('Are you sure you want to delete this
            candidate?');
      };
      deleteLink.setAttribute('href', deleteStr);
      deleteLink.setAttribute('class', 'btn btn-danger btn-
           circle');
      deleteLink.innerHTML = '<span class="glyphicon glyphicon
           -trash"></span>';

      var tr = document.createElement('tr');
      var td = document.createElement('td');
      td.innerHTML = candidateNumber;
      tr.appendChild(td);
      td = document.createElement('td');
      td.innerHTML = item.lastName.toUpperCase() + ", " + item
           .firstName;
      tr.appendChild(td);
      td = document.createElement('td');
      td.innerHTML = item.party.name;
      tr.appendChild(td);
      td = document.createElement('td');
      td.appendChild(editLink);
      td.appendChild(deleteLink);
      tr.appendChild(td);
      tbody.appendChild(tr);
     }
    }
```

```
    });
    if(candidateCount != 0){
     table.appendChild(tbody);
     $('#candidates-div').append(table);
    } else {
     $('#candidates-div').append('No candidates found.<br/><br/
          >');
    }
   });
  }
}
</script>
<body onload="getCandidates()">
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li class="active">Candidates</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Candidates</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-list"></span> Candidates
    <div class="pull-right">
     <a th:href="@{/candidates/add}" type="button" class="btn
          btn-primary btn-sm" >
      <span class="glyphicon glyphicon-plus"></span> Add
           Candidate
     </a>
    </div>
   </div>
   <div class="panel-body">
    <div>
     <span th:if="${electionList.isEmpty()}">No candidates. <a
          th:href="@{/elections/add}">Add an election</a>
          first.</span>
     <form class="form-inline" th:if="${!electionList.isEmpty
          ()}">
      <label>Select an election:</label>
      <select class="form-control input-sm" id="electionId"
            onchange="getCandidates()">
       <th:block th:each ="election : ${electionList}" >
        <option th:value="${election.id}" th:text="${election
             .name}" >Election</option>
       </th:block>
      </select>
     </form>
    </div>
    <br/>
    <div class="table-responsive" th:if="${!electionList.
         isEmpty()}">
     <div id="candidates-div">
     </div>
    </div>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) {
   $('#sidebar-collapse').collapse('show');
  }
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) {
   $('#sidebar-collapse').collapse('hide');
  }
})
//]]>
</script>
</body>
</html>
```

12.12. change-password.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Change Password</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{|/user/edit/${#authentication.principal.
        user.id}|}">Settings</a></li>
  <li class="active">Change Password</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Change Password</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default" th:if="${errorMessage ==
       null}">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-user"></span>Change
         Password
   </div>
   <div class="panel-body">
    <form role="form" th:object="${changePasswordForm}" method
         ="post">
     <input type="hidden" th:name="${_csrf.parameterName}" th:
          value="${_csrf.token}" />
     <div class="table-responsive">
      <table class="table table-striped">
       <tbody>
        <tr>
         <th>Old Password:</th>
         <td>
          <div class="form-group" th:classappend="${#fields.
               hasErrors('oldPassword')}? 'has-error'">
           <input class="form-control" type="password" th:
                field="*{oldPassword}" placeholder="Enter Old
                Password" />
           <p class="help block text-danger" th:if="${#fields.
                hasErrors('oldPassword')}"
            th:errors="*{oldPassword}">
            Error in old password
           </p>
          </div>
         </td>
        </tr>
        <tr>
         <th>New Password:</th>
         <td>
          <div class="form-group" th:classappend="${#fields.
               hasErrors('newPassword')}? 'has-error'">
           <input class="form-control" type="password" th:
                field="*{newPassword}" placeholder="Enter New
                Password" />
           <p class="help block text-danger" th:if="${#fields.
                hasErrors('newPassword')}"
            th:errors="*{newPassword}">
            Error in new password
           </p>
          </div>
         </td>
        </tr>
        <tr>
         <th>Confirm New Password:</th>
         <td>
          <div class="form-group" th:classappend="${#fields.
               hasErrors('newPasswordRepeated')}? 'has-error'
               ">
           <input class="form-control" type="password" th:
                field="*{newPasswordRepeated}" placeholder="
                Confirm New Password" />
           <p class="help block text-danger" th:if="${#fields.
                hasErrors('newPasswordRepeated')}"
            th:errors="*{newPasswordRepeated}">
            Error in new password
           </p>
          </div>
         </td>
        </tr>
       </tbody>
      </table>
     </div>
     <input type="submit" class="btn btn-default" value="
          Submit"/>
     <input type="reset" class="btn btn-default" value="Reset"
          />
    </form>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
```

```
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
```

12.13. edit-candidate.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Edit Candidate</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<script>
function getPositionsAndParties(positionId, partyId){
 var id = $('#electionId').val();
 getPositions(id, positionId);
 getParties(id, partyId);
}
function getPositions(id, positionId) {
 var positions;
 $.ajax({
  async: false,
  url: '../positions/get-positions',
  data: {electionId : id},
  success: function(response){
    positions = response;
  }
 });
 $("#positionId").find('option').remove();
 $.each(positions, function (i, item) {
  if(item.id == positionId){
   $('#positionId').append($("<option/>", {
    value: item.id,
    text: item.name,
    selected: "selected"
   }));
  } else {
   $('#positionId').append($("<option/>", {
    value: item.id,
    text: item.name
   }));
  }
 });
}
function getParties(id, partyId) {
 var parties;
 $.ajax({
  async: false,
  url: '../parties/get-parties',
  data: {electionId : id},
  success: function(response){
    parties = response;
  }
 });
 $("#partyId").find('option').remove();
 $.each(parties, function (i, item) {
  if(item.id == partyId){
   $('#partyId').append($("<option/>", {
    value: item.id,
    text: item.name,
    selected: "selected"
   }));
  } else {
   $('#partyId').append($("<option/>", {
    value: item.id,
    text: item.name
   }));
  }
 });
}
</script>
<body th:onload="'getPositionsAndParties(\'' + ${
      candidateBacker.positionId} + '\', \'' + ${
      candidateBacker.partyId} + '\');'">
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li><a th:href="@{/candidates}">Candidates</a></li>
  <li class="active">Edit Candidate</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Edit Candidate</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
        ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-pencil"></span> Edit
         Candidate
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${candidateBacker}" method=
           "post">
       <input type="hidden" th:name="${_csrf.parameterName}"
            th:value="${_csrf.token}" />
       <input type="hidden" th:value="${candidateBacker.id}"
            th:name="id" />
       <div class="form-group" th:classappend="${#fields.
            hasErrors('firstName')}? 'has-error'">
        <label>First Name*</label>
        <input class="form-control" th:field="*{firstName}"
             placeholder="Enter Candidate's First Name" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('firstName')}"
         th:errors="*{firstName}">
         Error in first name
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
            hasErrors('lastName')}? 'has-error'">
        <label>Last Name*</label>
        <input class="form-control" th:field="*{lastName}"
             placeholder="Enter Candidate's Last Name" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('lastName')}"
         th:errors="*{lastName}">
         Error in last name
        </p>
       </div>
       <div class="form-group" >
        <label>Election*</label>
        <select class="form-control" th:field="*{electionId}"
             th:onchange=
         "'getPositionsAndParties(\'' + *{positionId} + '\',
              \'' + *{partyId} + '\');'">
         <th:block th:each ="election : ${electionList}">
          <option th:if="${election.id} == ${candidateBacker.
               electionId}" selected="selected" th:value="${
               election.id}" th:text="${election.name}" >
               Election</option>
          <option th:if="${election.id} != ${candidateBacker.
               electionId}" th:value="${election.id}" th:text
               ="${election.name}" >Election</option>
         </th:block>
        </select>
       </div>
       <div class="form-group">
        <label>Position*</label>
        <select class="form-control" th:field="*{positionId}">
        </select>
       </div>
       <div class="form-group" >
        <label>Party*</label>
         <select class="form-control" th:field="*{partyId}">
        </select>
       </div>
       <input type="submit" class="btn btn-default" value="
            Submit"/>
       <input type="reset" class="btn btn-default" value="
            Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);

$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.14. edit-election.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Edit Election</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li><a th:href="@{/elections}">Elections</a></li>
  <li class="active">Edit Election</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Edit Election</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
        ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-pencil"></span> Edit
         Election
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${election}" method="post">
       <input type="hidden" th:name="${_csrf.parameterName}"
            th:value="${_csrf.token}" />
       <input type="hidden" th:value="${election.id}" th:name=
            "id" />
       <div class="form-group" th:classappend="${#fields.
            hasErrors('name')}? 'has-error'">
        <label>Election Name:</label>
        <input class="form-control" th:field="*{name}"
             placeholder="Enter Election Name" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('name')}"
         th:errors="*{name}">
         Error in election name
        </p>
       </div>
       <div class="form-group">
        <label>Parent:</label>
        <select class="form-control" th:field="*{
             parentElectionId}">
         <option th:if="*{parentElectionId == 0}" selected="
              selected" th:value="0">No Parent</option>
         <option th:if="*{parentElectionId != 0}" th:value="0"
              >No Parent</option>
         <th:block th:if="${electionList != null}" th:each ="
              electionItem : ${electionList}">
          <option th:if="*{parentElectionId} == ${electionItem
               .id}" selected="selected"
           th:value="${electionItem.id}" th:text="${
                electionItem.name}">Election </option>
          <option th:if="*{parentElectionId} != ${electionItem
               .id}"
           th:value="${electionItem.id}" th:text="${
                electionItem.name}">Election</option>
         </th:block>
        </select>
       </div>
       <input type="submit" class="btn btn-default" value="
            Submit"/>
       <input type="reset" class="btn btn-default" value="
            Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);

$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.15. edit-party.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Party</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<link th:href="@{/css/chosen.css}" rel="stylesheet" type="text
      /css" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li><a th:href="@{/parties}">Parties</a></li>
  <li class="active">Edit Party</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Edit Party</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
        ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-pencil"></span> Edit
         Party
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${party}" method="post">
       <input type="hidden" th:name="${_csrf.parameterName}"
            th:value="${_csrf.token}" />
       <input type="hidden" th:value="*{id}" th:name="id" />
       <div class="form-group" th:classappend="${#fields.
            hasErrors('name')}? 'has-error'">
        <label>Party Name*</label>
        <input class="form-control" th:field="*{name}"
             placeholder="Enter Party Name" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('name')}"
         th:errors="*{name}">
         Error in party name
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
            hasErrors('elections')}? 'has-error'">
        <label>Elections*</label><br/>
        <select class="chosen-select form-control" multiple="
             multiple" name="elections" data-placeholder="
             Choose an Election"
          tabindex="4">
         <option th:each="election : ${electionList}" th:value
              ="${election.id}" th:text="${election.name}"
          th:selected="${party.elections.contains(election)}"/
              >
        </select>
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('elections')}"
         th:errors="*{elections}">
         Error in elections
        </p>
       </div>
       <input type="submit" class="btn btn-default" value="
            Submit"/>
       <input type="reset" class="btn btn-default" value="
            Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script th:src="@{/js/chosen.jquery.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
```

```
        function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
        plus");
}(window.jQuery);

$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
        ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
        collapse('hide')
})
//]]>
```

12.16. edit-position.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Add Position</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{/positions}">Positions</a></li>
  <li class="active">Edit Position</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Edit Position</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-pencil"></span> Edit
         Position
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${positionBacker}" method="
            post">
       <input type="hidden" th:name="${_csrf.parameterName}"
             th:value="${_csrf.token}" />
       <input type="hidden" th:value="${positionBacker.id}" th
             :name="id" />
       <div class="form-group"  th:classappend="${#fields.
             hasErrors('name')}? 'has-error'">
        <label>Position Name*</label>
        <input class="form-control" th:field="*{name}"
              placeholder="Enter Position Name" />
        <p class="help block text-danger" th:if="${#fields.
              hasErrors('name')}"
         th:errors="*{name}">
         Error in position name
        </p>
       </div>
       <div class="form-group">
        <label>Election*</label>
        <select class="form-control" th:field="*{electionId}">
         <th:block th:each ="election : ${electionList}">
          <option th:if="*{electionId} == ${election.id}"
                selected="selected" th:value="${election.id}"
                th:text="${election.name}" >Election</option>
          <option th:if="*{electionId} != ${election.id}" th:
                value="${election.id}" th:text="${election.
                name}" >Election</option>
```

```
         </th:block>
        </select>
       </div>
       <div class="form-group" th:classappend="${#fields.
             hasErrors('maxElected')}? 'has-error'">
        <label>Maximum number of candidates to be elected for
              the position:</label>
        <input class="form-control" type="number" th:field="*{
             maxElected}" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('maxElected')}"
         th:errors="*{maxElected}">
         Error in maxElected
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
             hasErrors('ordinality')}? 'has-error'">
        <label>Ordinality (order in the ballot)*</label>
        <input class="form-control" type="number" th:field="*{
             ordinality}" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('ordinality')}"
         th:errors="*{ordinality}">
         Error in ordinality
        </p>
       </div>
       <div class="checkbox">
        <label>
         <input type="checkbox" th:field="*{abstain}" th:
              checked="*{abstain}"/> Abstain
        </label>
       </div>
       <input type="submit" class="btn btn-default" value="
             Submit"/>
       <input type="reset" class="btn btn-default" value="
             Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);

$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.17. edit-voter.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Edit Voter</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{/voters}">Voters</a></li>
  <li class="active">Edit Voter</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Edit Voter</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-plus"></span> Edit Voter
   </div>
   <div class="panel-body">
```

```
<div class="row">
 <div class="col-lg-6">
  <form role="form" th:object="${voterEditForm}" method="
       post">
   <input type="hidden" th:name="${_csrf.parameterName}"
        th:value="${_csrf.token}" />
   <input type="hidden" th:value="*{id}" th:name="id" />
   <div class="form-group" th:classappend="${#fields.
        hasErrors('username')}? 'has-error'">
    <label>Username:</label>
    <input class="form-control" th:field="*{username}"
         placeholder="Enter Username" />
    <p class="help block text-danger" th:if="${#fields.
         hasErrors('username')}"
     th:errors="*{username}">
     Error in username
    </p>
   </div>
   <div class="form-group" th:classappend="${#fields.
        hasErrors('email')}? 'has-error'">
    <label>Email:</label>
    <input class="form-control" type="text" th:field="*{
         email}" placeholder="Enter Email" />
    <p class="help block text-danger" th:if="${#fields.
         hasErrors('email')}"
     th:errors="*{email}">
     Error in email
    </p>
   </div>
   <div class="form-group" th:classappend="${#fields.
        hasErrors('voterType')}? 'has-error'">
    <label>Voter Types:</label>
    <select class="form-control" th:field="*{voterType}" >
     <th:block th:each ="voterType : ${voterTypeList}">
      <option th:if="${voterType.id} == *{voterType.id}"
           selected="selected" th:value="${voterType.id}"
            th:text="${voterType.name}" >Voter Type</
           option>
      <option th:if="${voterType.id} != *{voterType.id}"
           th:value="${voterType.id}" th:text="${
           voterType.name}" >Voter Type</option>
     </th:block>
    </select>
    <p class="help block text-danger" th:if="${#fields.
         hasErrors('voterType')}"
     th:errors="*{voterType}">
     Error in voter type
    </p>
   </div>
   <input type="submit" class="btn btn-default" value="
        Submit"/>
   <input type="reset" class="btn btn-default" value="
        Reset" />
  </form>
 </div>
</div>
<!-- /.row (nested) -->
</div>
<!-- /.panel-body -->
</div>
<!-- /.panel -->
</div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
      function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
      plus");
}(window.jQuery);

$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.18. edit-voter-type.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
     scale=1" />
<title>iVote - Edit Voter Type</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<link th:href="@{/css/chosen.css}" rel="stylesheet" type="text
     /css" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li><a th:href="@{/voter-types}">Voter Types</a></li>
  <li class="active">Edit Voter Type</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Edit Voter Type</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
        ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-pencil"></span> Edit
         Voter Type
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${voterType}" method="post"
           >
       <input type="hidden" th:name="${_csrf.parameterName}"
            th:value="${_csrf.token}" />
       <input type="hidden" th:value="*{id}" th:name="id" />
       <div class="form-group" th:classappend="${#fields.
            hasErrors('name')}? 'has-error'">
        <label>Voter Type Name:</label>
        <input class="form-control" th:field="*{name}"
             placeholder="Enter Voter Type Name" />
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('name')}"
         th:errors="*{name}">
         Error in voter type name
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
            hasErrors('elections')}? 'has-error'">
        <label>Elections:</label><br/>
        <select class="chosen-select form-control" multiple="
             multiple" name="elections" data-placeholder="
             Choose an Election"
         tabindex="4">
         <option th:each="election : ${electionList}" th:value
              ="${election.id}" th:text="${election.name}"
          th:selected="${voterType.elections.contains(election
               )}"/>
        </select>
        <p class="help block text-danger" th:if="${#fields.
             hasErrors('elections')}"
         th:errors="*{elections}">
         Error in elections
        </p>
       </div>
       <input type="submit" class="btn btn-default" value="
            Submit"/>
       <input type="reset" class="btn btn-default" value="
            Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script th:src="@{/js/chosen.jquery.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
      function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
      plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
<script type="text/javascript">
var config = {
  '.chosen-select'           : {},
  '.chosen-select-deselect'  : {allow_single_deselect:true},
  '.chosen-select-no-single' : {disable_search_threshold:10},
  '.chosen-select-no-results': {no_results_text:'Oops, nothing
       found!'},
  '.chosen-select-width'     : {width:"95%"}
}
for (var selector in config) {
  $(selector).chosen(config[selector]);
}
</script>
</body>
</html>
```

12.19. edit-voting-official.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Edit Voting Official</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{/voting-officials}">Voting Officials</a></
        li>
  <li class="active">Edit Voting Official</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Edit Official</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-pencil"></span> Edit User
   </div>
   <div class="panel-body">
    <div class="row">
     <div class="col-lg-6">
      <form role="form" th:object="${userEditForm}" method="
            post">
       <input type="hidden" th:name="${_csrf.parameterName}"
             th:value="${_csrf.token}" />
       <input type="hidden" name="id" th:value="*{id}"/>
       <input type="hidden" th:name="role" th:value="*{role}"
             />
       <div class="form-group" th:classappend="${#fields.
             hasErrors('username')}? 'has-error'">
        <label>Username:</label>
        <input class="form-control" th:field="*{username}"
              placeholder="Enter Username" />
        <p class="help block text-danger" th:if="${#fields.
              hasErrors('username')}"
         th:errors="*{username}">
         Error in username
        </p>
       </div>
       <div class="form-group" th:classappend="${#fields.
             hasErrors('email')}? 'has-error'">
        <label>Email:</label>
        <input class="form-control" type="text" th:field="*{
              email}" placeholder="Enter Email" />
        <p class="help block text-danger" th:if="${#fields.
              hasErrors('email')}"
         th:errors="*{email}">
         Error in email
        </p>
       </div>
       <input type="submit" class="btn btn-default" value="
             Submit"/>
       <input type="reset" class="btn btn-default" value="
             Reset" />
      </form>
     </div>
    </div>
    <!-- /.row (nested) -->
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
 //<![CDATA[
 !function ($) {
  $(document).on("click","ul.nav li.parent > a > span.icon",
        function(){
   $(this).find('em:first').toggleClass("glyphicon-minus");
  });
  $(".sidebar span.icon").find('em:first').addClass("glyphicon
        -plus");
 }(window.jQuery);
 $(window).on('resize', function () {
   if ($(window).width() > 768) $('#sidebar-collapse').
         collapse('show')
 })
 $(window).on('resize', function () {
   if ($(window).width() <= 767) $('#sidebar-collapse').
         collapse('hide')
 })
 //]]>
</script>
</body>
</html>
```

12.20. elections.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Elections</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li class="active">Elections</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Elections</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-tasks"></span> Elections
    <div class="pull-right">
     <div class="btn-group">
      <a th:href="@{/elections/add}" type="button" class="btn
            btn-primary btn-sm" >
       <span class="glyphicon glyphicon-plus"></span> Add
             Election
      </a>
     </div>
    </div>
   </div>
   <div class="panel-body">
    <div th:if="${electionList.isEmpty()}">No elections found.
          <br/><br/></div>
    <div class="table-responsive" th:if="${!electionList.
          isEmpty()}">
     <table class="table table-striped">
      <thead>
       <tr>
        <th>No.</th>
        <th>Election</th>
        <th>Status</th>
        <th>Actions</th>
       </tr>
      </thead>
      <tbody>
       <th:block th:each ="election, iter : ${electionList}">
        <tr>
         <td th:text="${iter.count}"></td>
         <td th:if="${election.parentElectionId == 0}" th:text
               ="${election.name}"></td>
         <td th:if="${election.parentElectionId > 0}" th:utext
               ="'   ' + ${election.name}"></td
               >
         <td th:if="${not election.status} and ${election.
               endDate != null}">Finished Election at <span th
               :text="${#dates.format(election.endDate, 'MMM
               dd, yyyy (HH:mm:ss)')}"></span></td>
         <td th:if="${not election.status} and ${election.
               endDate == null}">Not Yet Running [<a th:href="
               @{/elections/change-status/__${election.id}__}"
               >start</a>]</td>
         <td th:if="${election.status}">Running since <span th
               :text="${#dates.format(election.startDate, 'MMM
                dd, yyyy (HH:mm:ss)')}"></span> [<a th:href="@
               {/elections/change-status(id=${election.id})}">
               stop</a>]</td>
         <td>
          <a th:href="@{/elections/edit/__${election.id}__}"
                type="button" class="btn btn-primary btn-
                circle"><span class="glyphicon glyphicon-
                pencil"></span></a>
          <a th:href="@{/elections/delete/__${election.id}__}"
                 onclick="return confirm('Are you sure you
                want to delete this election?');" type="button
                " class="btn btn-danger btn-circle"><span
                class="glyphicon glyphicon-trash"></span></a>
         </td>
        </tr>
       </th:block>
      </tbody>
     </table>
    </div>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
```

```
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.21. error.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Error</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<div class="row">
 <div class="col-xs-10 col-xs-offset-1 col-sm-8 col-sm-offset
       -2 col-md-4 col-md-offset-4">
  <div class="panel panel-danger">
   <div class="panel-heading" >
    <span th:utext="'Oops, ' + ${status} + '!'">Error</span>
    <div class="pull-right">
     <a th:href="@{/home}" type="button" class="btn btn-danger
           btn-sm" >
      <span class="glyphicon glyphicon-home"></span> Back to
           Home
     </a>
    </div>
   </div>
   <div class="panel-body">
    <h3 th:text="${error}"></h3>
    <h5 th:text="${message}"></h5>
   </div>
  </div>
 </div><!-- /.col-->
</div><!-- /.row -->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) {
   $('#sidebar-collapse').collapse('show');
  }
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) {
   $('#sidebar-collapse').collapse('hide');
  }
})
//]]>
</script>
</body>
</html>
```

12.22. home.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a href="#"><span class="glyphicon glyphicon-home"></
       span></a></li>
  <li class="active">Home</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Welcome to iVote!</h1>
  <div class="jumbotron">
    <h2>Welcome to iVote!</h2>
    <p>iVote provides private and verifiable elections through
          Secure Multiparty Computation.</p>
    <div sec:authorize="!isAuthenticated()">
   <p><a class="btn btn-primary btn-lg" th:href="@{/login}"
         role="button">Get Started</a></p>
    </div>
    <div sec:authorize="isAuthenticated()">
   <p sec:authorize="hasRole('VOTING_OFFICIAL')"><a class="btn
         btn-primary btn-lg" th:href="@{/elections}" role="
         button">Get Started</a></p>
   <p sec:authorize="hasRole('ADMIN')"><a class="btn btn-
         primary btn-lg" th:href="@{/voting-officials}" role="
         button">Get Started</a></p>
    </div>
  </div>
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.23. import-voters.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Import CSV File</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li><a th:href="@{/voters}">Voters</a></li>
  <li class="active">Import CSV File</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Import CSV File</h1>
  <div class="alert bg-primary" role="alert">
   <span class="glyphicon glyphicon-info-sign"></span> Adding
        a voter with the same username/email as a previous
        voter record will update that record.
  </div>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success"
    th:utext="${successMessage}">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger"
    th:text="${errorMessage}">Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-upload"></span>Import CSV
         File
   </div>
   <div class="panel-body">
    <form role="form" th:object="${importVoterForm}" method="
         post" enctype="multipart/form-data">
     <input type="hidden" th:name="${_csrf.parameterName}" th:
          value="${_csrf.token}" />
     <div class="form-group" th:classappend="${#fields.
          hasErrors('file')}? 'has-error'">
      <label>CSV File:* (File should have comma-delimited
           voter fields (username, email))</label>
```

```
      <input type="file" th:field="*{file}"/>
      <p class="help block text-danger" th:if="${#fields.
            hasErrors('file')}"
       th:errors="*{file}">
       Error in file
      </p>
     </div>
     <div class="form-group" th:classappend="${#fields.
           hasErrors('voterType')}? 'has-error'">
       <label>Voter Type:*</label>
       <select class="form-control" th:field="*{voterType}">
       <option th:each="voterType : ${voterTypeList}" th:
             value="${voterType.id}"
        th:text="${voterType.name}"></option>
       </select>
       <p class="help block text-danger" th:if="${#fields.
             hasErrors('voterType')}"
       th:errors="*{voterType}">
       Error in voter type
      </p>
     </div>
     <input type="submit" class="btn btn-default" value="
          Submit"/>
    </form>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.24. login.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Login</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<div class="row">
 <div class="col-xs-10 col-xs-offset-1 col-sm-8 col-sm-offset
       -2 col-md-4 col-md-offset-4">
  <div class="login-panel panel panel-default">
   <div class="panel-heading">Log in</div>
   <div class="panel-body">
    <form th:action="@{/login}" method="post" role="form">
     <input type="hidden" th:name="${_csrf.parameterName}" th:
           value="${_csrf.token}" />
     <div th:if="${errorMessage != null}" th:text="${
           errorMessage}" class="alert alert-danger">
       Error
     </div>
     <div th:if="${param.error}" class="alert alert-danger">
       Invalid username or password.
     </div>
     <fieldset>
      <div class="form-group">
       <input class="form-control" placeholder="Username" name
            ="username" type="text" autofocus="autofocus" />
      </div>
      <div class="form-group">
       <input class="form-control" placeholder="Password" name
            ="password" type="password" value="" />
      </div>
      <input type="submit" class="btn btn-primary" value="
           Login"/>
     </fieldset>
    </form>
   </div>
  </div>
 </div><!-- /.col-->
</div><!-- /.row -->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) {
   $('#sidebar-collapse').collapse('show');
  }
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) {
   $('#sidebar-collapse').collapse('hide');
  }
})
//]]>
</script>
</body>
</html>
```

12.25. parties.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Parties</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<script>
function getParties(){
 var id = $('#electionId').val();
 $('#parties-div').empty();
 $('#parties > thead:last').empty();
 $('#parties > tbody:last').empty();
 var parties;
 $.ajax({
  async: false,
  url: 'parties/get-parties',
  data: {electionId : id},
  success: function(response){
   parties = response;
  }
 });
 if(parties.length > 0){
  $('#parties > thead:last').append('<tr><th>No.</th><th>Party
        </th><th>Actions</th></tr>');
  $.each(parties, function (i, item) {
   var count = i+1;
   var editStr= 'parties/edit/' + item.id;
   var deleteStr= 'parties/delete/' + item.id;
   var editLink = document.createElement('a');
   editLink.setAttribute('href', editStr);
   editLink.setAttribute('class', 'btn btn-primary btn-circle
         ');
   editLink.setAttribute('style', 'margin-right:5px;');
   editLink.innerHTML = '<span class="glyphicon glyphicon-
         pencil"></span>';
   var deleteLink = document.createElement('a');
   deleteLink.onclick = function(){
    return confirm('Are you sure you want to delete this party
          ?');
   };
   deleteLink.setAttribute('href', deleteStr);
   deleteLink.setAttribute('class', 'btn btn-danger btn-circle
         ');
   deleteLink.innerHTML = '<span class="glyphicon glyphicon-
         trash"></span>';
   var tr = document.createElement('tr');
   var td = document.createElement('td');
   td.innerHTML = count;
   tr.appendChild(td);
   td = document.createElement('td');
   td.innerHTML = item.name;
   tr.appendChild(td);
   td = document.createElement('td');
   td.appendChild(editLink);
   td.appendChild(deleteLink);
   tr.appendChild(td);
   $('#parties > tbody:last').append(tr);
  });
 } else {
  $('#parties-div').html('No parties found.');
 }
}
</script>
<body onload="getParties()">
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li class="active">Parties</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Parties</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
```

```
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-bullhorn"></span> Parties
    <div class="pull-right">
     <a th:href="@{/parties/add}" type="button" class="btn btn
          -primary btn-sm" >
      <span class="glyphicon glyphicon-plus"></span> Add Party
      </a>
    </div>
   </div>
   <div class="panel-body">
    <div>
     <span th:if="${electionList.isEmpty()}">No parties. <a th
          :href="@{/elections/add}">Add an election</a> first
          .</span>
     <form class="form-inline" th:if="${!electionList.isEmpty
          ()}">
      <label>Select an election:</label>
      <select class="form-control input-sm" id="electionId"
           onchange="getParties()">
       <th:block th:each ="election : ${electionList}" >
         <option th:value="${election.id}" th:text="${election
              .name}" >Election</option>
       </th:block>
      </select>
     </form>
    </div>
    <br/>
    <div class="table-responsive" th:if="${!electionList.
         isEmpty()}">
     <div id="parties-div">
     </div>
     <table id="parties" class="table table-striped">
      <thead>
      </thead>
      <tbody>
      </tbody>
     </table>
    </div>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
 //<![CDATA[
 !function ($) {
  $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
   $(this).find('em:first').toggleClass("glyphicon-minus");
  });
  $(".sidebar span.icon").find('em:first').addClass("glyphicon
       -plus");
 }(window.jQuery);
 $(window).on('resize', function () {
   if ($(window).width() > 768) {
    $('#sidebar-collapse').collapse('show');
   }
 })
 $(window).on('resize', function () {
   if ($(window).width() <= 767) {
    $('#sidebar-collapse').collapse('hide');
   }
 })
 //]]>
</script>
</body>
</html>
```

12.26. positions.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Positions</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<script>
function getPositions(){
 var id = $('#electionId').val();
 $('#positions-div').empty();
 $('#positions > thead:last').empty();
 $('#positions > tbody:last').empty();
 var positions;
 $.ajax({
  async: false,
  url: 'positions/get-positions',
  data: {electionId : id},
  success: function(response){
    positions = response;
  }
 });
 if(positions.length > 0){
  $('#positions > thead:last').append('<tr><th>No.</th><th>
       Position</th><th>Actions</th></tr>');
  $.each(positions, function (i, item) {
   var count = i+1;
   var editStr= 'positions/edit/' + item.id;
   var deleteStr= 'positions/delete/' + item.id;
   var editLink = document.createElement('a');
   editLink.setAttribute('href', editStr);
   editLink.setAttribute('class', 'btn btn-primary btn-circle
        ');
   editLink.setAttribute('style', 'margin-right:5px;');
   editLink.innerHTML = '<span class="glyphicon glyphicon-
        pencil"></span>';
   var deleteLink = document.createElement('a');
   deleteLink.onclick = function(){
    return confirm('Are you sure you want to delete this
         position?');
   };
   deleteLink.setAttribute('href', deleteStr);
   deleteLink.setAttribute('class', 'btn btn-danger btn-circle
        ');
   deleteLink.innerHTML = '<span class="glyphicon glyphicon-
        trash"></span>';
   var tr = document.createElement('tr');
   var td = document.createElement('td');
   td.innerHTML = count;
   tr.appendChild(td);
   td = document.createElement('td');
   td.innerHTML = item.name;
   tr.appendChild(td);
   td = document.createElement('td');
   td.appendChild(editLink);
   td.appendChild(deleteLink);
   tr.appendChild(td);
   $('#positions > tbody:last').append(tr);
  });
 } else {
  $('#positions-div').html('No positions found.');
 }
}
</script>
<body onload="getPositions()">
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
 <div class="row">
  <ol class="breadcrumb">
   <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
   <li class="active">Positions</li>
  </ol>
 </div><!--/.row-->
 <div class="row">
  <div class="col-lg-12">
   <h1 class="page-header">Positions</h1>
   <th:block th:if="${successMessage != null}">
    <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
   </th:block>
   <th:block th:if="${errorMessage != null}">
    <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
   </th:block>
   <div class="panel panel-default">
    <div class="panel-heading">
     <span class="glyphicon glyphicon-th-list"></span>
          Positions
     <div class="pull-right">
      <a th:href="@{/positions/add}" type="button" class="btn
           btn-primary btn-sm" >
       <span class="glyphicon glyphicon-plus"></span> Add
            Position
      </a>
     </div>
    </div>
    <div class="panel-body">
     <div>
      <span th:if="${electionList.isEmpty()}">No positions. <a
           th:href="@{/elections/add}">Add an election</a>
           first.</span>
      <form class="form-inline" th:if="${!electionList.isEmpty
           ()}">
       <label>Select an election:</label>
       <select class="form-control input-sm" id="electionId"
            onchange="getPositions()">
        <th:block th:each ="election : ${electionList}" >
          <option th:value="${election.id}" th:text="${election
               .name}" >Election</option>
        </th:block>
       </select>
      </form>
     </div>
     <br/>
     <div class="table-responsive" th:if="${!electionList.
          isEmpty()}">
      <div id="positions-div">
      </div>
      <table id="positions" class="table table-striped">
       <thead>
       </thead>
       <tbody>
       </tbody>
      </table>
     </div>
    </div>
    <!-- /.panel-body -->
   </div>
   <!-- /.panel -->
  </div>
 </div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
      function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
      plus");
```

```
}(window.jQuery);

$(window).on('resize', function () {
  if ($(window).width() > 768) {
   $('#sidebar-collapse').collapse('show');
  }
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) {
   $('#sidebar-collapse').collapse('hide');
  }
})
//]]>
</script>
</body>
</html>
```

12.27. review-ballot.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Review Ballot</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li><a th:href="@{/vote}">Vote</a></li>
  <li class="active">Review Ballot</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Review Ballot</h1>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-edit"></span> Review
         Ballot
   </div>
   <div class="panel-body">
    <div class="alert bg-primary" role="alert">
     <span class="glyphicon glyphicon-info-sign"></span>
          Click 'Votes' on top to edit your votes.
     Remember the hash of your voted candidates or you can
          download it <a href="#" onclick="document.
          getElementById('download').submit(); return false;"
          >here</a>.
     <form id="download" th:action="@{/vote/ivote-candidate-
          hashes}" th:object="${ballot}" method="post">
      <input type="hidden" th:name="${_csrf.parameterName}" th
           :value="${_csrf.token}"/>
      <th:block th:each="vote, iter : *{votes}">
       <input type="hidden" th:name="votes[__${iter.index}__].
            positionId" th:value="${vote.positionId}"/>
       <th:block th:each="candidateId, iter2 : ${vote.
            candidateIds}">
        <input type="hidden" th:name="votes[__${iter.index}__
             ].candidateIds[__${iter2.index}__]" th:value="${
             candidateId}"/>
       </th:block>
      </th:block>
     </form>
    </div>
    <div th:each="position : ${positions}">
```

```
    <th:block th:if="${ballot.getElectionId() == -1} or ${
          ballot.getElectionId() != position.election.id}">
     <h2 th:text="${position.election.name}"></h2>
    </th:block>
    <h4 th:text="${position.name}"></h4>
    <div th:each="candidate : ${votedCandidates}" th:if="${
          position.id == candidate.position.id}">
     <span th:if="${candidate.id != 0}" th:text="${candidate.
           lastName + ', ' + candidate.firstName}"></span>
     <span th:if="${candidate.id == 0}">ABSTAIN</span>
      
     (Hash: <span th:text="${candidate.hash}"></span>)
    </div>
    <th:block th:text="${ballot.setElectionId(position.
          election.id)}"></th:block>
   </div>
   <br/>
   <form method="post" th:object="${ballot}">
    <input type="hidden" th:name="${_csrf.parameterName}" th:
          value="${_csrf.token}" />
    <th:block th:each="vote, iter : *{votes}">
     <input type="hidden" th:name="votes[__${iter.index}__].
           positionId" th:value="${vote.positionId}"/>
     <th:block th:each="candidateId, iter2 : ${vote.
           candidateIds}">
      <input type="hidden" th:name="votes[__${iter.index}__].
            candidateIds[__${iter2.index}__]" th:value="${
            candidateId}"/>
     </th:block>
    </th:block>
    <input type="submit" class="btn btn-default" value="
          Submit Votes" />
   </form>
  </div>
  <!-- /.panel-body -->
 </div>
 <!-- /.panel -->
</div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
      plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.28. settings.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Settings</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li class="active">Settings</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Settings</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success"
    th:text="${successMessage}">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger"
    th:text="${errorMessage}">Error Message</div>
  </th:block>
  <div class="panel panel-default" th:if="${errorMessage ==
        null}">
```

```
   <div class="panel-heading">
    <span class="glyphicon glyphicon-user"></span>Settings
   </div>
   <div class="panel-body">
    <form role="form" th:object="${userEditForm}" method="post
         ">
     <input type="hidden" th:name="${_csrf.parameterName}" th:
           value="${_csrf.token}" />
     <input type="hidden" th:name="role" th:value="*{role}" />
     <div class="table-responsive">
      <table class="table table-striped" th:if="${errorMessage
            == null}">
       <tbody>
        <tr>
         <th>Username:</th>
         <td>
          <div class="form-group" th:classappend="${#fields.
               hasErrors('username')}? 'has-error'">
           <input class="form-control" th:field="*{username}"
                placeholder="Enter Username" />
           <p class="help block text-danger" th:if="${#fields.
                hasErrors('username')}"
            th:errors="*{username}">
            Error in username
           </p>
          </div>
         </td>
        </tr>
        <tr sec:authorize="hasRole('VOTING_OFFICIAL') or
             hasRole('ADMIN')">
         <th>Password:</th>
         <td><a th:href="@{|/user/change-password/${#
              authentication.principal.user.id}|}">Change
              Password</a></td>
        </tr>
        <tr>
         <th>Email:</th>
         <td>
          <div class="form-group" th:classappend="${#fields.
               hasErrors('email')}? 'has-error'">
```

```
          <input class="form-control" th:field="*{email}"
                 placeholder="Enter Email" />
         </div>
         <p class="help block text-danger" th:if="${#fields.
              hasErrors('email')}"
          th:errors="*{email}">
          Error in email
         </p>
        </td>
       </tr>
      </tbody>
     </table>
    </div>
    <input type="submit" class="btn btn-default" value="
          Submit"/>
    <input type="reset" class="btn btn-default" value="Reset"
          />
   </form>
  </div>
  <!-- /.panel-body -->
 </div>
 <!-- /.panel -->
</div>
</div><!--/.row-->
</div> <!--/.main-->
```

12.29. user-profile.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Profile</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li class="active">User Profile</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Profile</h1>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default" th:if="${errorMessage ==
       null}">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-user"></span>Profile
   </div>
   <div class="panel-body">
    <div class="table-responsive">
    <table class="table table-striped" th:if="${errorMessage
         == null}">
     <tbody>
      <tr>
       <th>Username:</th>
       <td th:text="${user.username}"></td>
      </tr>
      <tr>
       <th>Email:</th>
       <td th:text="${user.email}"></td>
      </tr>
      <tr>
       <th>Role</th>
       <td th:text="${user.role.value}"></td>
      </tr>
     </tbody>
    </table>
    </div>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.30. view-results.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - View Results</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<script>
function getCandidates(id) {
 $('#ballot').empty();
 var positions;
 $.ajax({
  async: false,
  url: 'results/get-positions',
  data: {electionId : id},
  success: function(response){
    positions = response;
  }
 });
 var candidates;
 $.ajax({
  async: false,
  url: 'results/get-candidates-results',
  data: {electionId : id},
  success: function(response){
    candidates = response;
  }
 });
 var electionId = -1;
 $.each(positions, function (i, item) {
  if (electionId == -1 || electionId != item.election.id){
   $('#ballot').append("<h2>" + item.election.name + "</h2>");
  }
  $('#ballot').append("<h4>" + item.name + " (" + item.
       maxElected + ")" + "</h4>");
  $.each(candidates, function (j, item2) {
   if(item2.election.id == item.election.id){
    if (item2.position.id == item.id){
     $('#ballot').append(item2.lastName.toUpperCase() + ", " +
          item2.firstName + " (" + item2.party.name + "), "
         + item2.voteCount + " vote/s <br/>");
    }
   }
  });
  if(item.abstain){
   $.ajax({
    async: false,
    url: 'results/get-abstain',
    data: {positionId : item.id},
    success: function(response){
      voteCount = response;
    }
   });
   $('#ballot').append("ABSTAIN, " + voteCount + " vote/s<br/>
        ");
  }
  electionId = item.election.id;
 });
 $.ajax({
  async: false,
  url: 'results/get-voters',
  data: {electionId : id},
  success: function(response){
    voters = response;
  }
 });
 var voterCount = 0;
 $.each(voters, function (i, item) {
  if(item.voted){
   voterCount++;
  }
 });
 $('#ballot').append("<br/>");
 $('#ballot').append("<b>Number of Voters Who Voted:</b> " +
       voterCount + "<br/>");
 $('#ballot').append("<b>Total Number of Voters:</b> " +
```

```
        voters.length + "<br/>");
 $('#ballot').append("<b>Voter Turnout:</b> " + ((voterCount/
        voters.length)*100).toFixed(3) + "% <br/>");
 var linkStr= 'results/download-results/' + id;
 $('#ballot').append("<br/>");
 var editLink = document.createElement('a');
 editLink.setAttribute('href', linkStr);
 editLink.setAttribute('class', 'btn btn-primary btn-lg');
 editLink.innerHTML = '<span class="glyphicon glyphicon-
        download-alt"></span> Download Results';
 $('#ballot').append(editLink);
}
</script>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
        main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li class="active">View Results</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">View Results</h1>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-search"></span> View
          Results
    <div class="pull-right" th:if="${electionList != null}">
     <div class="btn-group">
      <button type="button" class="btn btn-default btn-sm
            dropdown-toggle" data-toggle="dropdown">
       Elections
       <span class="caret"></span>
      </button>
      <ul class="dropdown-menu pull-right" role="menu" >
       <li th:each="election : ${electionList}">
       <a href="#" th:onclick="'javascript:getCandidates(\'' +
            ${election.id} + '\');'"
          th:text="${election.name}"></a>
       </li>
      </ul>
     </div>
    </div>
   </div>
   <div class="panel-body">
    <div id="ballot" th:if="${electionList != null}">
    <h5>Select an election</h5>
     </div>
     <div class="alert alert-danger" th:if="${electionList ==
           null}">You cannot view the results at this time.</
           div>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.31. view-tracker.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Bulletin Board</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/bootstrap-table.css}" rel="stylesheet" /
      >
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li><a th:href="@{/bulletin-board}">Bulletin Board</a></li>
  <li class="active">View verification code tracker</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Bulletin Board</h1>
   <div class="panel panel-default">
    <div class="panel-heading">
     <span class="glyphicon glyphicon-search"></span> View
          verification code tracker
    </div>
    <div class="panel-body">
     <p>Verification Code: <span th:text="${verificationCode}"
          ></span></p>
     <p>Details: </p>
     <div class="form-group">
      <textarea class="form-control" rows="12" th:text="${json
           }"></textarea>
     </div>
    </div>
    <!-- /.panel-body -->
   </div>
   <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script th:src="@{/js/bootstrap-table.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.32. vote.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Vote</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<script>
function abstain(el, iterIndex){
 $("input[id^=votes" + iterIndex + "]").each( function () {
  if($(this).val() != 0) {
   if($(el).is(':checked')){
    $(this).attr("checked", false);
    $(this).attr("disabled", true);
   } else {
    $(this).removeAttr("checked");
    $(this).removeAttr("disabled");
   }
  }
 });
}
function multipleCandidates(el, iterIndex, max){
 var numOfCheckedBoxes = $("input[id^=votes" + iterIndex + "]:
       checked").length;
 if (numOfCheckedBoxes > max){
  alert("A maximum of " + max + " candidates can be voted for
       this position!");
  $(el).attr("checked", false);
 }
}
</script>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
       main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li class="active">Vote</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Vote</h1>
```

```
<th:block th:if="${successMessage != null}">
 <div class="alert alert-success" th:utext="${successMessage
       }">Success Message</div>
</th:block>
<div class="alert alert-warning" th:if="${voted != null}">
  You have already voted! <br/>
  You can vote again below (overwrites your previous votes).
</div>
<div class="panel panel-default">
 <div class="panel-heading">
  <span class="glyphicon glyphicon-edit"></span> Vote
 </div>
 <div class="panel-body">
  <p th:if="${positions == null} and ${candidates == null}
        and ${session.voted == null}">You cannot vote at
        this time.</p>
  <form role="form" th:object="${ballot}" action="vote/
        confirm" method="post" th:if="${positions != null}
        and ${candidates != null} and ${session.voted ==
        null}">
    <div class="alert alert-danger" th:if="${#fields.
          hasErrors('votes')}"
   th:errors="*{votes}">Error Message</div>
    <div class="form-group" th:each="position, iter : ${
          positions}"  th:if="${positions != null} and ${
          candidates != null} and ${session.voted == null}">
   <input type="hidden" th:name="${_csrf.parameterName}" th:
         value="${_csrf.token}" />
   <th:block th:if="${ballot.getElectionId() == -1} or ${
         ballot.getElectionId() != position.election.id}">
    <h2 th:text="${position.election.name}"></h2>
   </th:block>
   <label th:utext="${position.name} + ' (' + ${position.
         maxElected} + ')'">
   </label>
   <th:block th:if="${position.maxElected == 1}">
    <th:block th:each="candidate, iterStat : ${candidates}"
          th:if="${candidate.position.maxElected == 1}">
     <div class="radio" th:if="${position.election.id ==
           candidate.election.id}
       and ${position.id == candidate.position.id}">
      <label>
       <input type="hidden" th:name="votes[__${iter.index}
             __].positionId" th:value="${position.id}"/>
       <input type="radio" th:field="*{votes[__${iter.
             index}__].candidateIds}" th:value="${
             candidate.id}" /> <span th:text="${candidate.
             lastName + ', ' + candidate.firstName}"></
             span>
       (<span th:text="${candidate.party.name}"></span>)
      </label>
     </div>
    </th:block>
    <input th:if="${position.abstain}" type="radio" th:field
          ="*{votes[__${iter.index}__].candidateIds}" th:
          value="0" /> <span th:if="${position.abstain}">
          ABSTAIN</span>
    <br/><br/>
     </th:block>
     <th:block th:if="${position.maxElected > 1}">
    <th:block th:each="candidate, iterStat : ${candidates}"
          th:if="${candidate.position.maxElected > 1}">
     <div class="checkbox" th:if="${position.election.id ==
           candidate.election.id}
       and ${position.id == candidate.position.id}">
      <label>
       <input type="hidden" th:name="votes[__${iter.index}
             __].positionId" th:value="${position.id}"/>
       <input type="checkbox" th:field="*{votes[__${iter.
             index}__].candidateIds}"
      th:value="${candidate.id}" th:onchange="'javascript:
            multipleCandidates(this, \'' + ${iter.index} +
             '\', \'' + ${position.maxElected} + '\');'" /
            ><span th:text="${candidate.lastName + ', ' +
            candidate.firstName}"></span>
       (<span th:text="${candidate.party.name}"></span>)
      </label>
     </div>
    </th:block>
    <input th:if="${position.abstain}" type="checkbox" th:
          onchange="'javascript:abstain(this, \'' + ${iter.
          index} + '\');'" th:field="*{votes[__${iter.index}
          __].candidateIds}" th:value="0" /> <span th:if="${
          position.abstain}">ABSTAIN</span>
    <br/><br/>
   </th:block>
   <th:block th:text="${ballot.setElectionId(position.
         election.id)}"></th:block>
    </div>
    <input th:if="${positions != null and candidates != null
          }" type="submit" class="btn btn-default" value="
          Submit"/>
    <input th:if="${positions != null and candidates != null
          }" type="reset" class="btn btn-default" value="
          Reset" />
  </form>
 </div>
 <!-- /.panel-body -->
</div>
<!-- /.panel -->
</div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) $('#sidebar-collapse').collapse
       ('show')
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) $('#sidebar-collapse').
       collapse('hide')
})
//]]>
</script>
</body>
</html>
```

12.33. voters.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Voters</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/bootstrap-table.css}" rel="stylesheet" /
      >
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
        home"></span></a></li>
  <li class="active">Users</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Voters</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
         ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
         Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-user"></span> Voters
    <div class="pull-right">
     <a th:href="@{/voters/add}" type="button" class="btn btn-
           primary btn-sm" >
      <span class="glyphicon glyphicon-plus"></span> Add Voter
     </a>
    </div>
   </div>
   <div class="panel-body">
    <table data-toggle="table" data-show-refresh="true" data
          -search="true" data-pagination="true" data-sort-
          name="name" data-sort-order="asc" th:if="${
          voterList != null}">
     <thead>
      <tr>
     <th data-field="no." data-sortable="true">No.</th>
     <th data-field="username" data-sortable="true">Username<
          /th>
     <th data-field="email" data-sortable="true">Email</th>
     <th data-field="voter-type" data-sortable="true">Voter
          Type</th>
     <th>Actions</th>
      </tr>
     </thead>
     <tbody>
    <th:block th:each ="voter, iter : ${voterList}">
     <tr>
      <td th:text="${iter.count}"></td>
      <td th:text="${voter.user.username}"></td>
      <td th:text="${voter.user.email}"></td>
      <td th:text="${voter.voterType.name}"></td>
      <td>
       <a th:href="@{/voters/edit/__${voter.id}__}" type="
            button" class="btn btn-primary btn-circle"><span
             class="glyphicon glyphicon-pencil"></span></a>
       <a th:href="@{/voters/delete/__${voter.id}__}" onclick
            ="return confirm('Are you sure you want to
            delete this voter?');" type="button" class="btn
            btn-danger btn-circle"><span class="glyphicon
            glyphicon-trash"></span></a>
      </td>
     </tr>
      </th:block>
     </tbody>
    </table>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script th:src="@{/js/bootstrap-table.js}"></script>
<script>
//<![CDATA[
!function ($) {
```

```
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) {
   $('#sidebar-collapse').collapse('show');
  }
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) {
   $('#sidebar-collapse').collapse('hide');
  }
})
//]]>
</script>
</body>
</html>
```

12.34. voter-types.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Voter Types</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<script>
function getTypes(){
 var id = $('#electionId').val();
 $('#types-div').empty();
 $('#types > thead:last').empty();
 $('#types > tbody:last').empty();
 var types;
 $.ajax({
  async: false,
  url: 'voter-types/get-voter-types',
  data: {electionId : id},
  success: function(response){
   types = response;
  }
 });
 if(types.length > 0){
  $('#types > thead:last').append('<tr><th>No.</th><th>Type</
        th><th>Actions</th></tr>');
  $.each(types, function (i, item) {
   var count = i+1;
   var editStr= 'voter-types/edit/' + item.id;
   var deleteStr= 'voter-types/delete/' + item.id;
   var editLink = document.createElement('a');
   editLink.setAttribute('href', editStr);
   editLink.setAttribute('class', 'btn btn-primary btn-circle
         ');
   editLink.setAttribute('style', 'margin-right:5px;');
   editLink.innerHTML = '<span class="glyphicon glyphicon-
        pencil"></span>';
   var deleteLink = document.createElement('a');
   deleteLink.onclick = function(){
    return confirm('Are you sure you want to delete this voter
          type?');
   };
   deleteLink.setAttribute('href', deleteStr);
   deleteLink.setAttribute('class', 'btn btn-danger btn-circle
         ');
   deleteLink.innerHTML = '<span class="glyphicon glyphicon-
        trash"></span>';
   var tr = document.createElement('tr');
   var td = document.createElement('td');
   td.innerHTML = count;
   tr.appendChild(td);
   td = document.createElement('td');
   td.innerHTML = item.name;
   tr.appendChild(td);
   td = document.createElement('td');
   td.appendChild(editLink);
   td.appendChild(deleteLink);
   tr.appendChild(td);
   $('#types > tbody:last').append(tr);
  });
 } else {
  $('#types-div').html('No voter types found.');
 }
}
</script>
<body onload="getTypes()">
<nav th:replace="fragments/header :: header"></nav>
<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li class="active">Voter Types</li>
 </ol>
</div><!--/.row-->

<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Voter Types</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
        ">Success Message</div>
  </th:block>
  <th:block th:if="${errorMessage != null}">
   <div class="alert alert-danger" th:text="${errorMessage}">
        Error Message</div>
  </th:block>
  <div class="panel panel-default">
   <div class="panel-heading">
    <span class="glyphicon glyphicon-th"></span> Voter Types
    <div class="pull-right">
     <a th:href="@{/voter-types/add}" type="button" class="btn
          btn-primary btn-sm" >
      <span class="glyphicon glyphicon-plus"></span> Add Voter
           Type
     </a>
    </div>
   </div>
   <div class="panel-body">
    <div>
     <span th:if="${electionList.isEmpty()}">No voter types. <
          a th:href="@{/elections/add}">Add an election</a>
          first.</span>
     <form class="form-inline" th:if="${!electionList.isEmpty
          ()}">
      <label>Select an election:</label>
      <select class="form-control input-sm" id="electionId"
           onchange="getTypes()">
       <th:block th:each ="election : ${electionList}" >
        <option th:value="${election.id}" th:text="${election
             .name}" >Election</option>
       </th:block>
      </select>
     </form>
    </div>
    <br/>
    <div class="table-responsive" th:if="${!electionList.
         isEmpty()}">
     <div id="types-div">
     </div>
     <table id="types" class="table table-striped">
      <thead>
      </thead>
      <tbody>
      </tbody>
     </table>
    </div>
   </div>
   <!-- /.panel-body -->
  </div>
  <!-- /.panel -->
 </div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
       function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
       plus");
}(window.jQuery);
$(window).on('resize', function () {
  if ($(window).width() > 768) {
   $('#sidebar-collapse').collapse('show');
  }
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) {
   $('#sidebar-collapse').collapse('hide');
  }
})
//]]>
</script>
</body>
</html>
```

12.35. voting-officials.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-
      scale=1" />
<title>iVote - Voting Officials</title>
<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet" />
<link th:href="@{/css/datepicker3.css}" rel="stylesheet" />
<link th:href="@{/css/styles.css}" rel="stylesheet" />
<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->
</head>
<body>
<nav th:replace="fragments/header :: header"></nav>

<div th:replace="fragments/navigation :: navigation"></div>
<div class="col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
      main">
<div class="row">
 <ol class="breadcrumb">
  <li><a th:href="@{/home}"><span class="glyphicon glyphicon-
       home"></span></a></li>
  <li class="active">Voting Officials</li>
 </ol>
</div><!--/.row-->
<div class="row">
 <div class="col-lg-12">
  <h1 class="page-header">Voting Officials</h1>
  <th:block th:if="${successMessage != null}">
   <div class="alert alert-success" th:text="${successMessage}
        ">Success Message</div>
  </th:block>
```

```
<th:block th:if="${errorMessage != null}">
 <div class="alert alert-danger" th:text="${errorMessage}">
       Error Message</div>
</th:block>
<div class="panel panel-default">
 <div class="panel-heading">
  <span class="glyphicon glyphicon-user"></span> Voting
       Officials
  <div class="pull-right">
   <a th:href="@{/voting-officials/add}" type="button" class
        ="btn btn-primary btn-sm" >
    <span class="glyphicon glyphicon-plus"></span> Add
         Voting Official
   </a>
  </div>
 </div>
 <div class="panel-body">
  <div th:if="${votingOfficialList.isEmpty()}">No voting
       officials found.</div>
  <div class="table-responsive" th:if="${!votingOfficialList
       .isEmpty()}">
   <table class="table table-striped">
    <thead>
     <tr>
      <th>No.</th>
      <th>Username</th>
      <th>Email</th>
      <th>Actions</th>
     </tr>
    </thead>
    <tbody>
     <th:block th:each ="votingOfficial, iter : ${
          votingOfficialList}">
      <tr>
       <td th:text="${iter.count}"></td>
       <td th:text="${votingOfficial.username}"></td>
       <td th:text="${votingOfficial.email}"></td>
       <td>
        <a th:href="@{/voting-officials/edit/__${
             votingOfficial.id}__}" type="button" class="
             btn btn-primary btn-circle"><span class="
             glyphicon glyphicon-pencil"></span></a>
        <a th:href="@{/voting-officials/delete/__${
             votingOfficial.id}__}" onclick="return confirm
             ('Are you sure you want to delete this voting
             official?');" type="button" class="btn btn-
             danger btn-circle"><span class="glyphicon
             glyphicon-trash"></span></a>
       </td>
      </tr>
     </th:block>
    </tbody>
   </table>
  </div>
 </div>
 <!-- /.panel-body -->
</div>
<!-- /.panel -->
</div>
</div><!--/.row-->
</div> <!--/.main-->
<script th:src="@{/js/jquery-1.11.1.min.js}"></script>
<script th:src="@{/js/bootstrap.min.js}"></script>
<script>
//<![CDATA[
!function ($) {
 $(document).on("click","ul.nav li.parent > a > span.icon",
      function(){
  $(this).find('em:first').toggleClass("glyphicon-minus");
 });
 $(".sidebar span.icon").find('em:first').addClass("glyphicon-
      plus");
}(window.jQuery);

$(window).on('resize', function () {
  if ($(window).width() > 768) {
   $('#sidebar-collapse').collapse('show');
  }
})
$(window).on('resize', function () {
  if ($(window).width() <= 767) {
   $('#sidebar-collapse').collapse('hide');
  }
})
//]]>
</script>
</body>
</html>
```

XI. Acknowledgement

Four years in the university taught me a lot of important life lessons—to aim higher and not settle for less, to express our opinions freely without any hesitations, to grab every opportunity, and to balance work and play. Studying in UP is indeed a learning experience, and this experience is about to end. Thank you, UP, for making me experience being an *Iskolar ng Bayan* and for introducing me to the people who helped made all of this possible.

I would like to express my sincere gratitude to my adviser, Mr. Richard Bryann Chua, for entrusting me with this topic and guiding me all the way until the end. Thank you for taking the time out of your busy schedule to continuously provide necessary comments and suggestions to my work. I would also like to thank Mr. Dan Bogdanov, Sharemind's information security project manager, for answering all of my queries and granting us the license to Sharemind 3. Thank you for your quick responses even if we are more than a thousand miles apart. I would also like to acknowledge Orange and Bronze Software Labs for teaching me the technologies I used in this study. I am extremely grateful to them for providing us interns with the knowledge we need in this growing IT industry.

To my blockmates, thank you for all the words of encouragement and for being there when I needed help. Thank you Mich, Jaimee, Larisse, Francis, Dado, and Ron, for being the true friends whom I can rely and trust. Thank you for all the fun moments despite the struggles we faced in the university. Thank you also to Jhai for helping me out in finishing this SP. To Marx, thank you for being a responsible blockhead who constantly updates and cares for us.

To my family, thank you for all the unwavering support. Thank you to my mom who never failed to provide me with all of the things I needed since day one. Thank you to the person up there for being my inspiration. This is for you, dad.

Above all, I would like to thank God for all the blessings He bestowed upon me

and my family. Thank You for giving me the courage, strength, and wisdom to finish my SP. Thank You for always being there to guide me in times of my troubles and successes. Thank You, Lord!