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

Voterify: Using The Homomorphic Encryption Project (THEP) to Provide Privacy and Verifiability in Internet Voting

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

Marvin Joseph R. Occeño

June 2016

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 "Voterify: Using The Homomorphic Encryption Project (THEP) to Provide Privacy and Verifiability in Internet Voting" prepared and submitted by Marvin Joseph R. Occeño 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. Marvin John C. Ignacio, M.Sc. (*cand.*)	_____	_____
5. Ma. Sheila A. Magboo, M.Sc.	_____	_____
6. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____

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

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

Internet voting (I-Voting) has already been used in several elections but its use is not yet widespread due to numerous security requirements. Among the top security requirements are privacy and verifiability. We created Voterify, an Internet voting application, that uses The Homomorphic Encryption Project (THEP) to look into the feasibility of using Paillier cryptosystem in providing privacy preserving and verifiable elections in Internet voting.

Keywords: Internet voting, homomorphic encryption, Paillier scheme, THEP

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Internet voting has the potential to replace traditional elections due to the emerging technologies in today's world. In fact, it is becoming a popular research topic and has been implemented in quite a number of elections over the last decade [1]. I-voting, compared to the conventional balloting system which is time-consuming in nature, speeds up the voting process, reduces the chance of human errors, and could increase voter turnout.

In spite of its benefits, I-voting still has a lot of challenges to face. These include satisfying certain security requirements such as privacy and verifiability in order to gain the trust of the voters [2] and for an election to be fair and successful. To achieve privacy, all votes must remain secret and should not be associated to the voters. Verifiability, on one hand, means that voters can check that their votes were really counted and that the final result of the election is accurate [3]. Other security properties may be considered as well, such as uniqueness, wherein no voter can cast a vote more than once; eligibility, in which only eligible voters are allowed to cast their ballots; and efficiency, which suggests that computations should be performed within a reasonable amount of time [4].

Most existing I-voting systems employ cryptographic techniques in order to satisfy the security requirements needed in the election process. Cryptography plays an important role in today's world even if many are not aware of that fact [5]. It can be used to protect sensitive information such as medical records and financial data. Such cryptographic primitive is homomorphic encryption, which makes it possible to perform certain computations on encrypted data without decrypting it. Thus, when it comes to e-voting, homomorphic encryption has the potential to calculate election results while maintaining the confidentiality of the votes.

B. Statement of the Problem

It is very important to ensure voters' privacy. There should be no way to determine how each voter voted [6]. With that, there is a need to encrypt the votes to maintain confidentiality. However, the server or other authorities might not be able to do some operations such as counting of votes if the data are encrypted. Tallying is obviously important and it's also a way to achieve verifiability. A trusted authority could decrypt these votes in order to do some computations on them but doing so greatly compromises privacy. Both privacy and verifiability must be satisfied by such I-voting system for it to be secure and reliable.

To address this problem, there is a need to find a way on how to perform computations directly on encrypted data without decrypting it. Such solution is to employ homomorphic encryption, which permits computing on encrypted data. Thus, election results can be tallied while votes remain encrypted.

C. Objectives of the Study

This study aims to create a simple, homomorphic encryption-based Internet voting system which has the following functionalities for the following user roles:

1. Allows a voter to
    (a) Cast a vote in an election
    (b) View his vote in hashed form
    (c) Download his hashed votes in text file
    (d) Raise a vote discrepancy ticket
    (e) Manage his account
2. Allows a voting official to

(a) Setup an event

i. Manage major and minor elections

ii. Manage positions

iii. Manage candidates

iv. Manage parties

v. Manage blocks

vi. Manage voters

vii. Start and stop an event

(b) Tally the results

(c) Receive vote discrepancy tickets

(d) Manage his account

3. Allows a public user to

(a) View election results

(b) Download election results in PDF

(c) View the bulletin board of votes

(d) Download the bulletin board of votes in PDF

4. Allows an administator to

(a) Manage list of voting officials

D. Significance of the Project

If votes are to be submitted electronically, election results will be calculated quickly and the chance of human errors will be eliminated. Also, electronic voting, particularly an Internet voting system, allows voters to vote at their own homes or at

any place they want to by using their personal computers, laptops, and even mobile phones. With this kind of convenience, there could be a significant increase in voter turnout, thus supporting democratic process [7].

Securing votes and verifying election results are the key features of this project. Privacy and verifiability are conflicting properties at times. To verify the accuracy of the election outcome without revealing individual votes is a challenging task. To make this possible, homomorphic encryption is then applied. This cryptographic primitive is responsible for the addition of encrypted votes and the decryption of the final result only. With this, privacy is guaranteed. Thus, this web-based voting system, aside from computerizing the voting process, has the potential to create fair and successful elections.

Generally, this project can significantly contribute in today's world where the demand for privacy of digital data (i.e, electronic votes, medical records, financial data) is very high [8].

E. Scope and Limitations

1. Voterify is ideal to environments where secure voting is needed.
2. However, it is not applicable to nationwide elections.
3. This system is designed for low-coercion elections.
4. It can be applied to student government elections but for the purpose of project demonstration only.
5. There is no voter registration in the system. The voting official provides the list of official or eligible voters.
6. The system doesn't take into account accessibility to people with disabilities.
7. Write-in votes are not accepted.

8. Failure of election is not considered.

9. Voters can only notify voting officials of the discrepancy of their votes during the period of verification.

10. Once a voting official starts an event, he/she can no longer make modifications in elections, positions, candidates, blocks, and voters. Once a voting official stops an event, voters for that event can no longer vote.

11. A voter can only vote in one event at a time.

12. All actions to address the complaints raised by voters are undertaken outside the voting system.

F. Assumptions

1. Email addresses of voters will be legitimate and secure.

2. Voting officials will be responsible enough to ensure the secrecy of private keys.

3. The administrator and voting officials will not collude to decrypt individual votes.

4. An abstain vote will mean not voting for any candidate in a certain position.

II. Review of Related Literature

Electronic voting has been employed in quite a number of elections over the last decade. Its potential to significantly speed up the voting process is one of the primary reasons for its development. However, the use of e-voting faces a lot of security challenges which include ensuring the privacy of the voters as well as the reliability of the voting system [1].

Several countries have utilized voting over the Internet. Estonia, as the first country to offer Internet voting nationally, introduced its online voting system in 2005 which basically works as follows: voters use their national ID cards to authenticate themselves to the server and to sign their encrypted ballots [9]. However, the voters cannot check that their votes are really counted and tabulated correctly; thus, the system lacks verifiability [1]. Springall et. al present a precise description plus a security analysis of the Estonian I-voting system in [9].

Norway used an Internet voting system based on ElGamal encryption of ballots and a mix-net before decryption [10]. Once they have voted, voters receive a code via SMS, which can be used to confirm that their votes are really inside the ballot box. This Norwegian voting protocol does not totally guarantee verifiability although it offers a so-called "proxy-verifiability", in which trusted supervisors can check the consistency of the ballot box and the proofs produced by the tallier when decrypting the ballots [1].

The above e-voting systems have problems or issues with verifiability. In order to achieve the said property, voters are usually provided with evidences (i.e., encrypted ballots and zero-knowledge proofs shown in a public bulletin board) which they can use to check that their votes were really counted and that the published result is correct [3]. In [11], Culnane et. al present vVote, a verifiable voting system used in a real state election in Victoria, Australia. Verification is done by giving all the voters and passive observers access to the public Web Bulletin Board (WBB), an

authenticated public broadcast channel with memory. A voter can use his receipt, which is given to him upon casting a vote, to check that the information is correctly recorded on the public WBB. Also shown on this board are mixing and decryption proofs that all votes are correctly mixed and decrypted. With respect to security, the vVote system assumes that the authorities who share the decryption key do not collude in order to keep the votes private and the devices such as the printer and the EBM are trusted not to leak the ballot information. A detailed verification procedure and more security assumptions, claims, and analysis are discussed by Culnane et. al. In summary, the vVote system provides cast-as-intended verification, where each voter gets evidence that her vote is cast as she intended; recorded-as-cast verification, where each voter gets the opportunity to check that her (encrypted) ballot appears in a public list of recorded votes; and an output list of decrypted recorded votes, with a universally verifiable proof of the decryption [11].

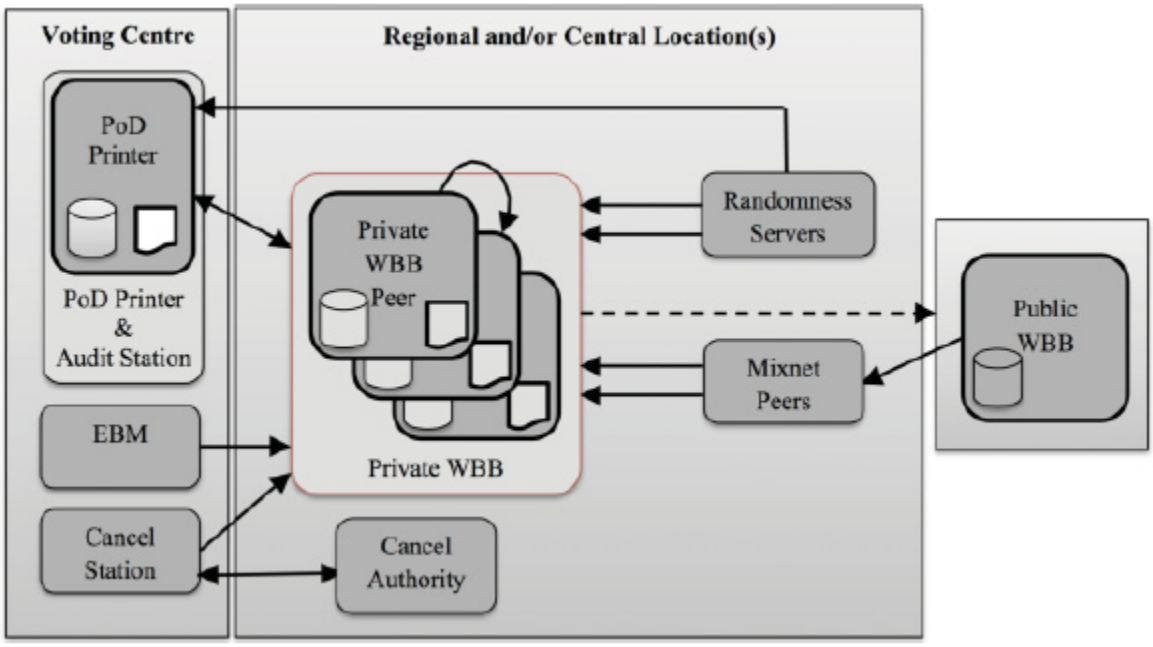


Figure 1: Architecture of the vVote system

Presented in [12] is Helios, a web-based open-audit voting system developed by Ben Adida. Helios was used by the Université catholique de Louvain (UCL) in Belgium

when it elected its President in March 2009. Voting for the University President of UCL is not mandatory. Thus, a registration phase was set up to ensure that only eligible voters got a credential to vote. A voter can be a faculty, staff, researcher or student. Also, voters were allowed to vote as many times they wanted, with the last vote being the only one counted and displayed on the web bulletin board [13]. According to Cortier, Helios offers both ballot privacy, individual, and universal verifiability.

Most existing e-voting systems rely on cryptographic techniques in order to overcome the security issues in the election process [14]. Secure voting schemes can be based on mix-nets, blind signatures, and homomorphic encryption.

The mix-net model. The basic idea of a mix-net is to shuffle a given input sequence of ciphertexts [6] or simply the encrypted votes in order to hide the relation between the voter and his vote. In a mix-net protocol, there are n mix-servers or mixes, each with its own public key and private key [15]. Each mix takes in an ordered list of votes, performs either re-encryption or decryption on them, secretly permutes their order, and forwards the result to the next mix. Every output of a mix-server is posted on a bulletin board which is publicly accessible. Thus, universal verifiability is achieved. However, due to many decryption or re-encryption operations, it may slow down the voting process. Also, each mix must provide a proof of correction to verify that all votes are forwarded correctly [6]. The Prêt á Voter voting system [8] is one such system that uses a mix-net [11]. More schemes based on mix-nets are discussed in [16].

The blind signature model. In a blind signature-based voting scheme, a ballot is blinded in order to achieve its confidentiality requirement. A voter is required to have his vote signed by an authority (or validator), whose task is to verify the voter's eligibility. To ensure privacy, a voter casts a ballot B, blinds B using a random number, and sends it to the validator. Let (n, e) be the validator's public key and

(n, d) be his private key. A voter generates a random number r such that $\gcd(r, n) = 1$ and sends the following to the validator:

$$B' = r^e B \ mod \ n$$

The random number r conceals the ballot from the validator. The validator then signs the blinded ballot. The signed value is as follows:

$$S' = (B')^d = rB^d \ mod \ n$$

After receiving the validated ballot, the voter unblinds the ballot to get the true signature S of the validator for the ballot by computing [14],

$$S = (S')r^{-1} \ mod \ n = B^d$$

Finally, S is the ballot B signed by the authority using his private key. Each voter then submit his own S through an anonymous channel to a voting bulletin board that will only accept ballots signed by the authority [6].

The homomorphic encryption model. The concept of homomorphic encryption was introduced by Rivest, Adleman and Dertouzous in 1978 [17] and was first instantiated for limited classes of functions like addition or multiplication [18]. Homomorphic encryption is said to be one of efficient security tools for such e-voting system due to homomorphic property, that is, certain operations such as addition and multiplication can be performed directly on ciphertexts such that upon decryption, the same answer is obtained if same operations are to be done on the original messages [19]. In a homomorphic encryption-based voting scheme, votes are either added or multiplied while encrypted, so no individual vote is revealed [6]. The election result can be easily obtained just by decrypting either the sum or product of the encrypted votes. Moreover, both individual and universal verifiability are guaranteed since there is a publicly available bulletin board. Displayed on this board are encrypted votes next to a voter name or voter identification number. However, the computational cost is high for large number of voters.

Paillier algorithm is one homomorphic cryptosystem which is widely used in voting systems. It is a probabilistic asymmetric algorithm for public key cryptography, invented by Pascal Paillier in 1999 [20]. In Paillier scheme, decrypting the product of two ciphertexts (encypted votes) yields the sum of their corresponding plaintexts (original votes). Having this property makes the Paillier scheme additive homomorphic.

In 1982, Shaff Goldwasser and Silvio Micali proposed a cryptosystem which was known as the "Goldwasser-Micali cryptosystem". It also possesses additive homomorphism and was proven to be a secure encryption scheme which reached a remarkable rating of safety. However, it is only capable of encrypting one bit at a time, which makes this scheme impractical at times [21].

In 2009, Craig Gentry in his PhD thesis introduced the first fully homomorphic encryption (FHE) scheme using ideal lattices. In an FHE scheme, there are no limitations on what computations can be performed. However, Gentry's approach to FHE is computationally expensive [21].

The said scheme was difficult and inefficient to implement until in 2013, when IBM presented an open source software library for homomorphic encryption named as HElib. This library implements Brakerski-Gentry-Vaikuntanathan (BGV) scheme, along with many optimizations to make homomorphic evaluation faster. HElib is written in C++ and uses the Number Theory Library (NTL) [5]. Another library which implements fully homomorphic encryption is Scarab, which is written in C. The Homomorphic Encryption Project (THEP) library, on one hand, uses Java to implement the Paillier scheme particularly.

The FHE scheme of HElib could be overwhelming if this library would be utilized to develop an Internet Voting system. Votes are to be added only in order to tally the results. Thus, THEP which focuses on Paillier's additive homomorphic property would be a more ideal tool in this study. HElib is far more complicated than THEP.

Looking at the former's documentation, it consists of classes and interfaces which are more difficult to comprehend compared to those provided by the latter. Moreover, generation of public and private keys and manipulation of encrypted data are easier with THEP.

III. Theoretical Framework

A. Electronic Voting

An electronic voting (e-voting) system is formally defined as a voting system in which the election data is recorded, stored and, processed primarily as digital information [7]. Basically, casting and counting of votes are held over electronic media.

Below is a taxonomy of e-voting as presented by Yumeng et. al [22].

1. Paper-Based E-Voting: Here, paper ballots could be marked manually, but counted electronically. Punch-card systems and optical scan systems fall under this category.

2. Direct Recording E-Voting (DRE) Machine: The DRE machine is a computer with a screen to display the ballot and an input device in the form of push buttons or a touch screen.

3. Internet Voting: Sometimes called as online voting. In general, it can be divided into two main types as follows:

   (a) Poll site. Voters are required to go to staffed polling sites and use computers to cast votes.

   (b) Remote voting system. Voters are allowed to cast from any computer or digital device connected to the Internet or to a private network, typically from home or workplace.

E-voting has been utilized due to its several benefits such as speeding up the counting of votes and reducing the chance of human errors. However, such e-voting system can be at risk of electoral fraud (i.e., vote modifying and ballot stuffling). Thus, voting systems, for them to be secure, reliable, and trustworthy, should satisfy the following security requirements which are described in [1, 2, 4, 20, 22]:

1. Privacy: A vote is kept secret and no one can determine for whom anyone else voted
2. Eligibility: Only authorized voters who satisfy predetermined criterion can vote.
3. Individual Verifiability: Each voter can check that his ballot really appears on the ballot box.
4. Universal Verifiability: The announced result is accurate and should correspond to the ballots on the box.
5. Uniqueness: No one can vote more than once.
6. Fairness: No partial result is available before the final result comes out.
7. Receipt-freeness: A voter does not gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way.
8. Uncoercibility: Any coercer should not be able to coerce a voter to cast his vote.
9. Robustness: No voter can disrupt the election, any invalid vote will be detected and not counted in the final tally.
10. Efficiency: Computations should be performed within a reasonable amount of time.

B. Homomorphic Encryption

Homomorphism is an algebraic property that allows certain mathematical operations to be carried out on encrypted data without decrypting it, thus improving privacy [5]. For example, given two encrypted numbers, a person can find the sum of their original values without decrypting their encrypted forms. In [23], homomorphic encryption is described as the following:

Let **PT** be the plaintext space and **CT** be the ciphertext space such that **PT** is a group under the operation $\oplus$ and **CT** is a group under the operation $\otimes$. Let $E_r(m)$ denote encryption of the message m using parameter r. An encryption scheme is homomorphic, if for given $c_1 = E_{r1}(m_1)$ and $c_2 = E_{r2}(m_2)$, there exists an r such that $c_1 \otimes c_2 = E_r(m_1 \oplus m_2)$.

This study utilizes the Paillier cryptosystem which possesses an additive homomorphic property. A homomorphic encryption is additive if:

$$Enc(x + y) = Enc(x) \cdot Enc(y)$$

On the other hand, a homomorphic encryption is multiplicative if:

$$Enc(x \cdot y) = Enc(x) \cdot Enc(y)$$

Below is the Paillier algorithm as shown in [5].

KeyGen(p, q)

Input: p and q should be prime

Compute: $n = pq$, $\lambda = lcm(p - 1, q - 1)$

Choose: $g \mid gcd(g, n^2) = 1$ and $gcd(L(g^\lambda \bmod n^2), n) = 1$ where $L(u) = (u-1)/n$

Compute: $\mu = L(g^\lambda \bmod n^2)^{-1}$

Output: (P_k, S_k)

Public Key: $P_k = (n, g)$

Secret Key: $S_k = (\lambda, \mu)$

Encrypt(m, P_k)

Input: $m \in \mathbb{Z}_n$

Choose: $r \mid gcd(r, n) = 1$

Compute: $c = g^m r^n \bmod n^2$

Output: c

Decrypt(c, S_k)

Input: c

Compute: $L(c^{\lambda} \bmod n^2) \cdot \mu \bmod n$

Output: $m \in \mathbb{Z}_n$

C. The Homomorphic Encryption Project (THEP)

THEP is a software library which implements the Paillier cryptosystem in Java, along with it's homomorphic operations and key generation. Saving/transporting keys and encrypted integers can be accomplished using methods inherited from *Serializable*, a Java interface which is responsible to represent an object as a sequence of bytes. This project aims to provide homomorphic encryption libraries to developers so they can in turn create privacy and confidentiality aware software [24].

D. Voting with Paillier Cryptosystem

Paillier's additive homomorphic property is utilized in order to add up the votes. Each vote is represented in numeric form before encrypting it with a public key generated by Paillier's algorithm. The entry 1 represents a vote for a candidate. Each encryption requires a secure random number, so that same votes will be encrypted to different ciphers.

Consider the voting scenario below where there are three voters and three candidates.

Voter	Alice	Bob	Charlie	Vote	Random Number	Encrypted Vote
V1	1	0	0	100	12943	394239429348
V2	0	0	1	001	4058	198923493489
V3	1	0	0	100	5172	702320132869
Total	2	0	1	201		1295483055706

Using their private keys, voting officials need to decrypt only the sum of the encrypted votes as individual votes are never decrypted, thereby providing voter

privacy. The resulting value is equal to the total of the plaintext votes or simply the election result. Shown on the bulletin board is a list of voters or their aliases with their respective encrypted votes. This in turn provides individual verifiability.

IV. Design and Implementation

A. Use Cases

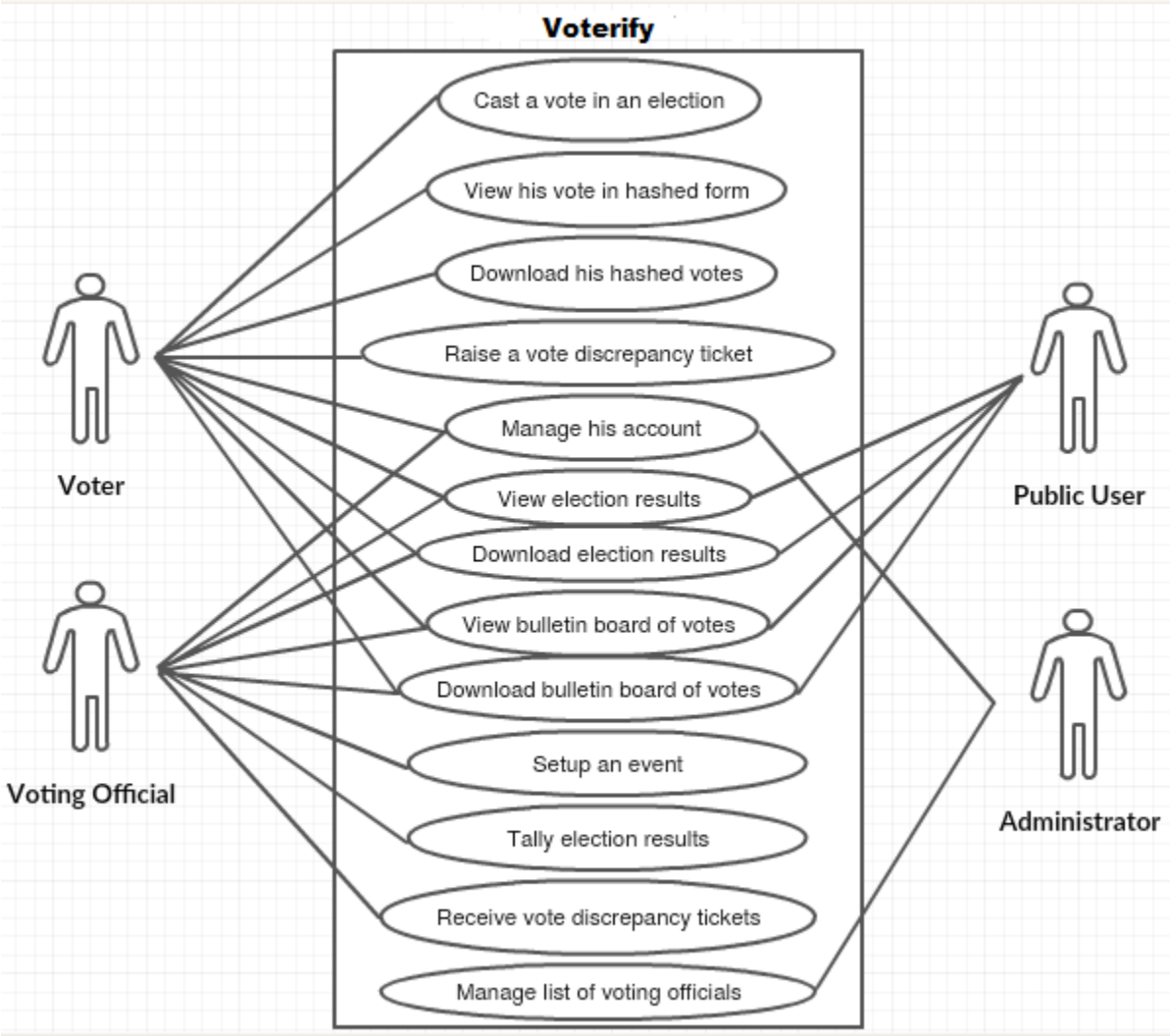


Figure 2: Top Level Use Case Diagram of Voterify

Voterify has four users: the voter, the voting official, the administrator, and a public user.

The voter can cast a vote in an election. Upon doing so, he can view the corresponding hashes of his votes and download a text file containing these hashes. This text file will serve as the voter's receipt. He can use this receipt to verify if his hashed votes appear on the bulletin board. Should there be such discrepancy in his votes

appearing on the said board, he can raise a ticket to notify the voting official who created the elections the voter participated in.

The voting official can create an event. Setting up an event requires managing elections, positions, parties, candidates, blocks, and voters. He can also start and stop an event. The voting official can add a voter or bulk upload voters via a .csv file which contains their desired usernames and email addresses. Each voter will receive an email containing these login credentials.

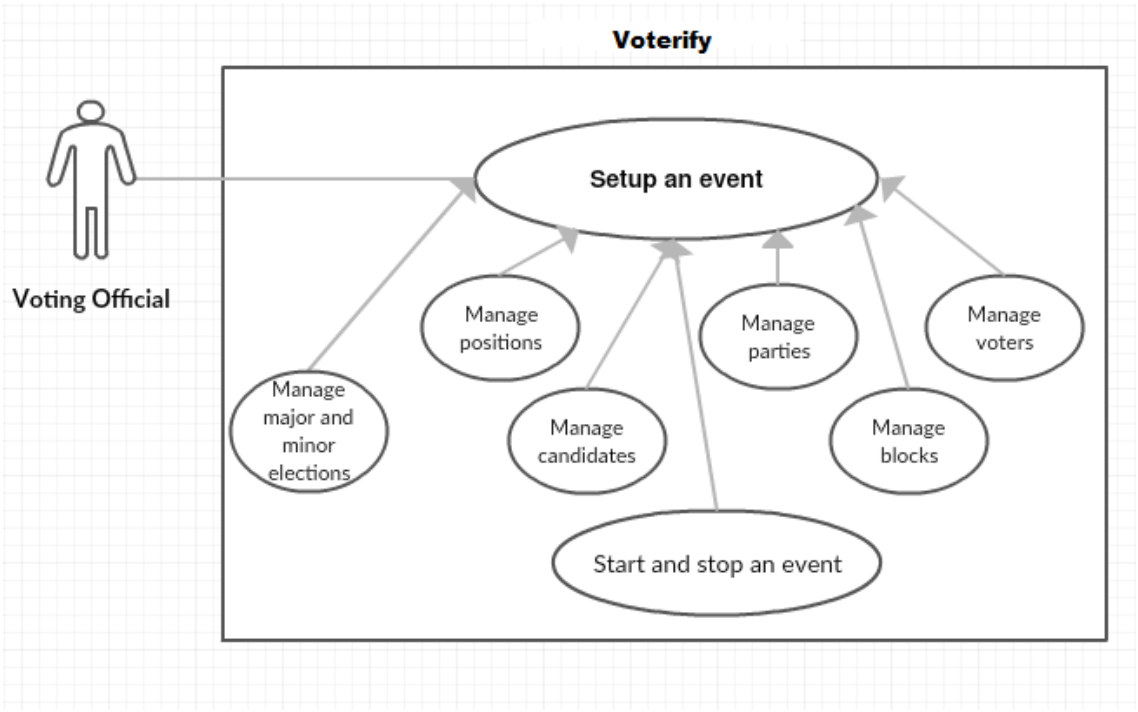


Figure 3: Setup an election use case diagram of Voterify

In addition, the voting official will have access to tally the vote. To tally the vote means to provide the election's private key in order to count the total number of votes while individual votes remain encrypted.

The voting official will be notified if such voter raises a vote discrepancy ticket. Arrangements to fix issues are done outside the system. Vote discrepancies should happen only when the voting system is compromised since attackers can manipulate votes. Otherwise, it means that the voter who raised a ticket is dishonest.

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

A public user can view and download the election results as well as the bulletin board of votes which displays hashed votes with the corresponding voters' aliases.

B. System Development

Voterify is implemented using the Apache Struts2, which is a Java Server Pages (JSP) framework, and Hibernate object-relational mapping framework. For the data storage, MySQL database management system (DBMS) is used.

The I-voting system utilizes The Homomorphic Encryption Project (THEP), a Java-based library which implements the Paillier algorithm, to perform the necessary homomorphic operations such as encrypting the votes and tallying election results. For every election created by a voting official, a 1024-bit public-and-private key pair is generated. The reason why a 1024-bit key pair is chosen is that key lengths shorter than 1024 bits are vulnerable to brute-force attacks nowadays. The public key consists of two parameters, n and g. Only n is stored in the database since in THEP, g is set to $n + 1$. On the other hand, the private key is split also into two values named as λ and μ. The former is stored in the database while the latter is given to a voting official as a *.key* file which he needs to upload in order to compute the tally. Shown below are sample values of n, λ, and μ. In THEP, they are all generated in decimal forms by default. The system converts them into hexadecimal values to conform with the cryptographic key standards which require keys to be in hex form.

Public Key (n)

```
96622ce886b9d1d63827fd7779ca5eda2654522267aae79b98d8e1fb59198f07d59
e1f0903715b7f0c9a1a45503de7fedf85541af9916604fa72b4923875e9eda68f08
0f43a01db4c3ab39e79331b9b7f1ba3fd3a0de761154199f8aa94c467ade4fe12b3
e77e57c78997f5290053e1d112098b5e29a1689d54d38537c71933d
```

Private Key (λ)

`96622ce886b9d1d63827fd7779ca5eda2654522267aae79b98d8e1fb59198f07d59`
`e1f0903715b7f0c9a1a45503de7fedf85541af9916604fa72b4923875e9ec164dae`
`8c639f77a57a888686153ec48b1225ef3b779c5beebd2a84e395f51af9a924633cb`
`639e4d00986aaff9f14a70d6a9122f810e4c4322396a43e3c08f1a0`

Private Key (μ)

`402b76dc13f47797f6f0b01c9760a558c4c079bc948129e76e5408664f99ab2f30c`
`c6618fe5c8eefff9651650148902060df116c1c737c602a9fb8370c827699d24e3`
`a2fd41a9f4f2b5217acbe92ef9a56cdc2ad6d18ce2d4e0c3b2fc536404456363523`
`41d593cfff37fd1c80aceebfcca5f50de9f9cb0d464338ff4d4bb1d`

The public key is mainly used to encrypt the votes. Vote encryption works as follows:

Suppose in an election E, there are n candidates C_1, C_2, ..., C_n running for a certain position P. A vote is initially represented in numeric or plaintext form before encrypting it. If a voter votes for C_i, his numeric vote is 10^{i-1}. For example, if he chooses C_3, his vote equals 100. If he is required to vote for more than one candidate, say he selects C_1 and C_3, his vote equals the sum of 1 (value obtained by choosing C_1) and 100 which is 101. The plaintext vote is then encrypted using the public key of E. The encryption function of THEP employs secure randomness so that same votes will result to different ciphers. An encrypted vote is in THEP's `EncryptedInteger` form whose cipher value has usual length of 630 characters. To achieve privacy, the cipher value, which is converted to string, is the one stored in the database, not the plaintext equivalent. Below are the key components of a vote saved in the database.

`voter_id`: ID of the voter

`position_id`: ID of the position the voter is voting for

`evote`: Encrypted vote which is a 630-character string

On the other hand, the private key is used to tally election results. Homomorphic tallying works as follows:

Vote counting is done position-wise. Each encrypted vote for a certain position P is fetched and then converted back to `EncryptedInteger`. Through THEP, all votes for P can now be added up to find the sum, which when decrypted already reveals the election results. The decryption process requires both λ, the one stored in the database, and μ, the one the voting official needs to upload.

To provide for individual verifiability, for every position listed in the ballot, a voter is shown the resulting hash value if he votes for a particular candidate. A hash is obtained by hashing via MD5 the respective long, encrypted vote. During ballot generation, the system already prepares the possible encrypted votes for all candidates. This is why a voter can already view the corresponding hash of an encrypted vote. Even if two voters voted for the same candidate, they will still have different hashes because their encrypted votes are different, which is made possible by THEP's encryption function. Once a voter submits his votes, he is redirected to a page where he can see again the hashes of the candidates he had voted. This provides evidence that his votes are cast as he intended. He can download these hashes in a text file which serves as his receipt. This receipt also contains the verification code which is simply the MD5 hash of the concatenation of the hashes.

Below is a sample receipt. The code and the hashes appearing on a receipt should be exactly the same as to what are displayed on the bulletin board of votes.

```
V100
President:  936aa2d51122f827004e568af835d1c6
Vice President:  be0608aeaa6a12e4b25c011ad358c0e8
Councilors:  5f0f716507f54b0569a3c833902afdbe
```

```
Verification Code:  7c2095341f9c2e22c4c259cd49fdae3d

Timestamp:  2016-05-25 00:28:48.716
```

C. Database Design

The voting system follows the below database design.

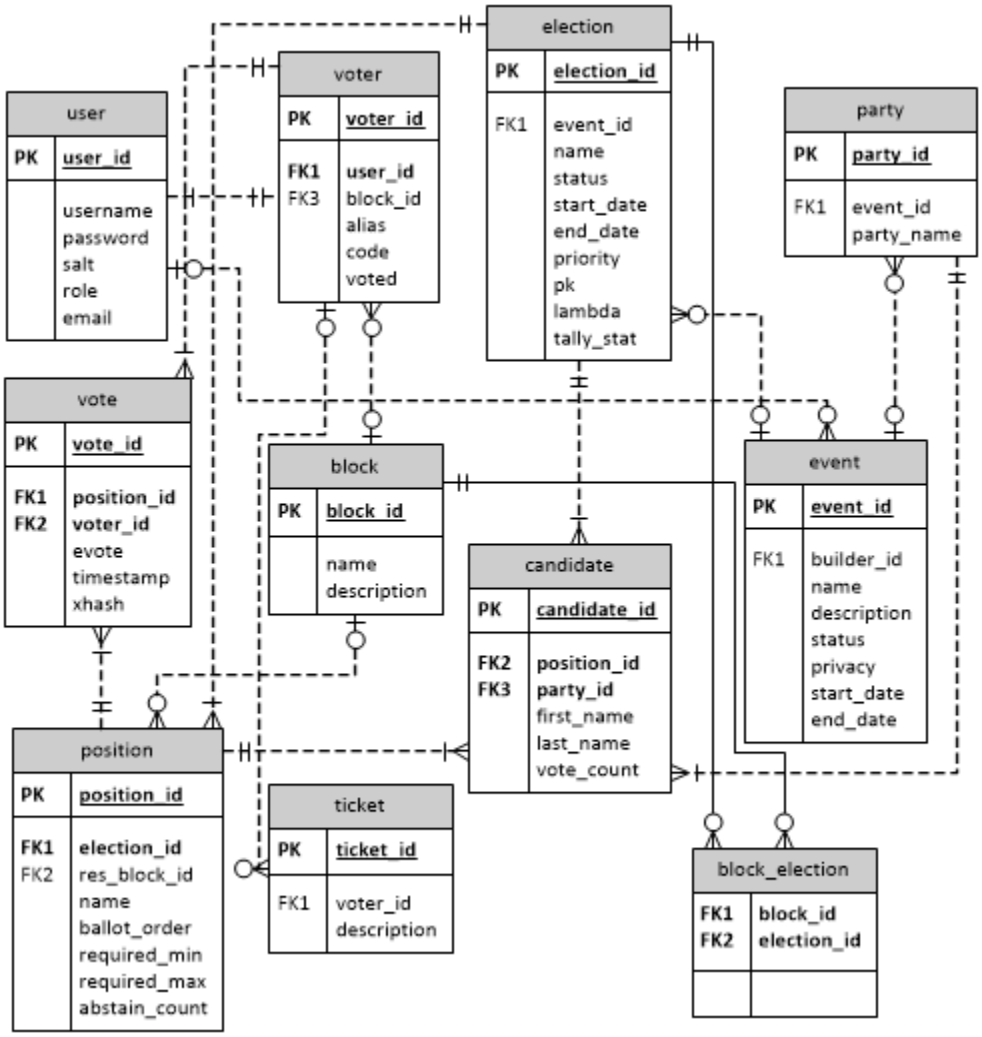


Figure 4: Entity Relationship Diagram of the Internet Voting System

D. Data Dictionary

Listed below are the different database tables used by the voting system.

Column	Type	Description
user_id	int(11)	ID of the user
username	varchar(20)	Username of the user
password	varchar(50)	Hashed password of the user
salt	varchar(50)	Salt of the password
email	varchar(50)	Email of the user
role	enum('VOTER', 'VOTING_OFFICIAL', 'ADMIN')	Role of the user

Table 1: user table

Column	Type	Description
block_id	int(11)	ID of the block
name	varchar(20)	Name of the block
description	text	Description of the block

Table 2: block table

Column	Type	Description
voter_id	int(11)	ID of the voter
user_id	int(11)	ID of the user
block_id	int(11)	Block ID of the voter
alias	varchar(10)	Alias of the voter
code	varchar(32)	Verification code of the voter
voted	bit(1)	Voting status of the voter

Table 3: voter table

Column	Type	Description
event_id	int(11)	ID of the event
builder_id	int(11)	User ID of event creator
name	varchar(50)	Name of the event
description	text	Description of the event
status	enum('IDLE', 'RUNNING', 'FINISHED', 'ARCHIVED')	Status of the event
privacy	bit(1)	Privacy of the event
start_date	datetime	Timestamp when the event is started
end_date	datetime	Timestamp when the event is ended

Table 4: event table

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

Table 5: party table

Column	Type	Description
election_id	int(11)	ID of the election
event_id	int(11)	ID of the event
name	varchar(20)	Name of the election
priority	bit(1)	Priority of the election
tally_stat	bit(1)	Tally status of the election
pk	varchar(310)	Public key of the election
lambda	varchar(650)	Lambda parameter of the private key of the election

Table 6: election table

Column	Type	Description
position_id	int(11)	ID of the position
election_id	int(11)	ID of the election the position belongs to
res_block_id	int(11)	ID of a block restricted to vote for this position
name	varchar(20)	Name of the position
required_min	int(11)	Minimum number of elected candidates for the position
required_max	int(11)	Maximum number of elected candidates for the position
ballot_order	int(11)	Order of the position in the ballot
abstain_count	int(11)	Number of abstains in the position

Table 7: position 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
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 8: candidate table

Column	Type	Description
block_id	int(11)	ID of the block
election_id	int(11)	ID of the election

Table 9: block_election table

Column	Type	Description
`ticket_id`	int(11)	ID of the ticket
`voter_id`	int(11)	ID of the voter who raised the ticket
`description`	text	Description of the ticket

Table 10: ticket table

Column	Type	Description
`voter_id`	int(11)	ID of the voter
`position_id`	int(11)	ID of the position
`evote`	varchar(650)	Encrypted vote
`timestamp`	timestamp	When the voter voted
`xhash`	varchar(32)	MD5 hash of a vote for a position which requires more than one candidate

Table 11: vote table

E. System Architecture

Figure 5 shows the system architecture of Voterify. The system has three layers — presentation, service, and database. The presentation layer involves the web interface to be coded in HTML, CSS, and JavaScript. The service layer is developed using Struts2 together with THEP that implements Paillier algorithm. Apache Tomcat is used as the system's web server, with MySQL as its database server. Hibernate handles the MySQL transactions.

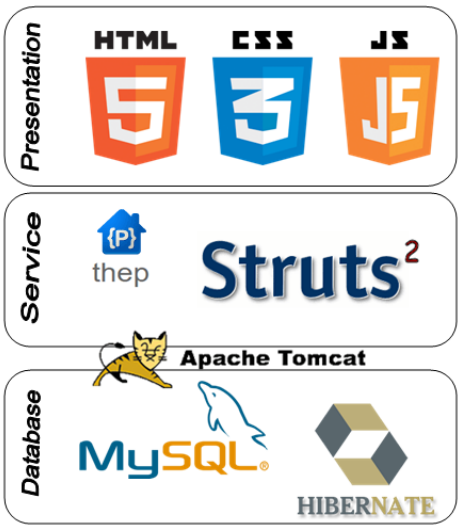


Figure 5: System Architecture

Memory leak and JDBC timeout exceptions were common problems encountered

during the software development phase. To address these issues, we use the following hibernate configurations (by changing the config file of hibernate, `hibernate.cfg.xml`).

```
hibernate.c3p0.privilegeSpawnedThreads = true
hibernate.c3p0.contextClassLoaderSource = library
```

These configurations are not present in most Struts2 and Hibernate tutorials found on the web. These workarounds are explained in [25].

F. Technical Architecture

The minimum requirements for the server machine include:

- Apache Tomcat 6
- MySQL 5.1.41
- 1GB RAM
- 2.4 MHz single core processor

The client side must have any of the following compatible web browsers plus some other minimum requirements:

- Mozilla Firefox 16.0.1
- Google Chrome 22.0.1229.94
- Safari 5.1.7
- Opera 12.02
- PDF Viewer
- Notepad

V. Results

Voters, voting officials, and the administrator can log in to the system by providing the login credentials.

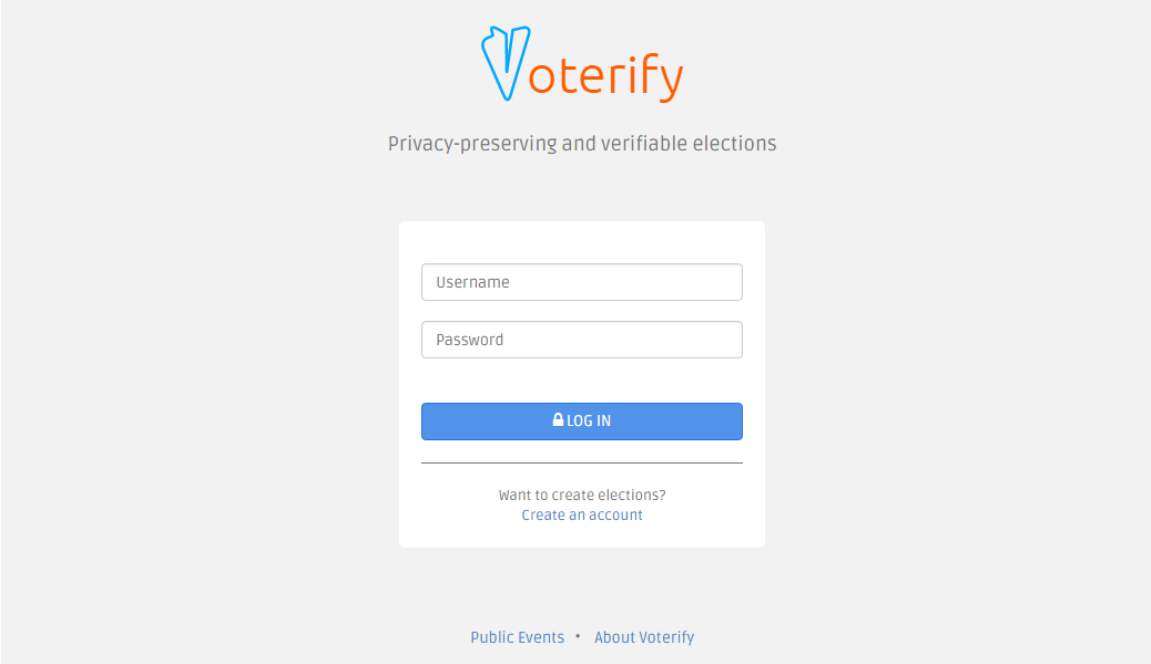


Figure 6: Login page

Error message appears whenever invalid credentials are provided.

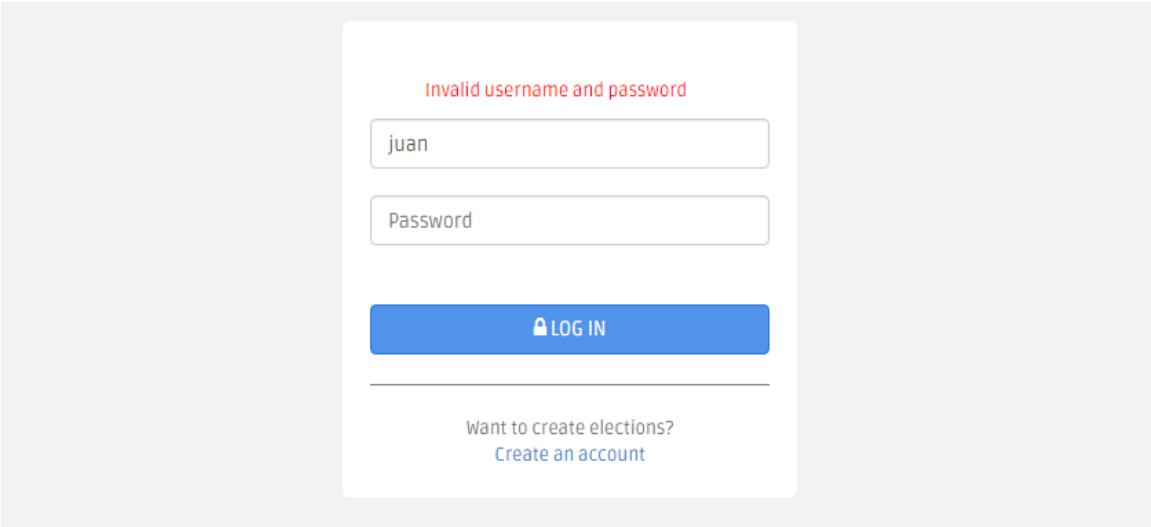


Figure 7: Login page - invalid credentials alert

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

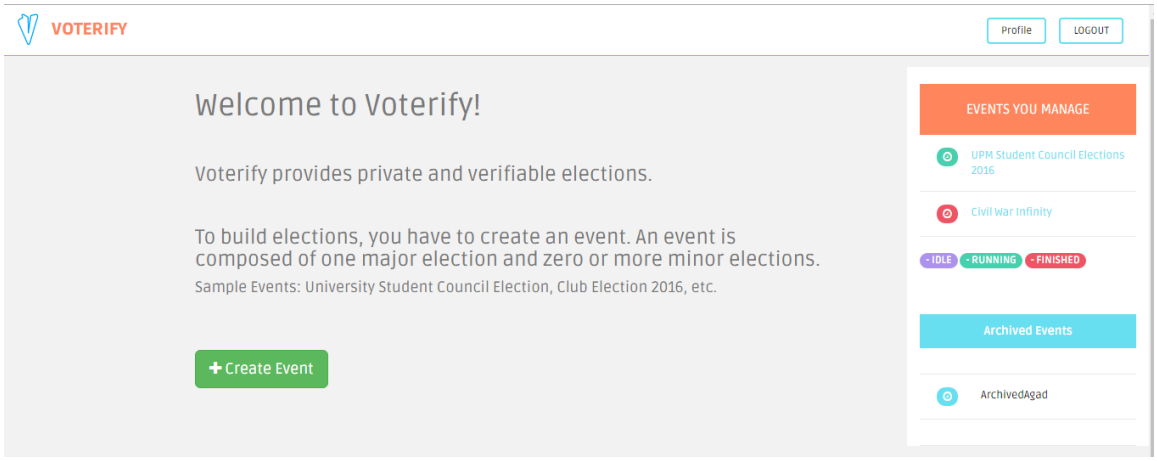


Figure 8: Home page for voting officials

A voting official can add or edit an event and later fill it up with elections, positions, parties, candidates, blocks, and voters. He can also start, stop, and archive an event.

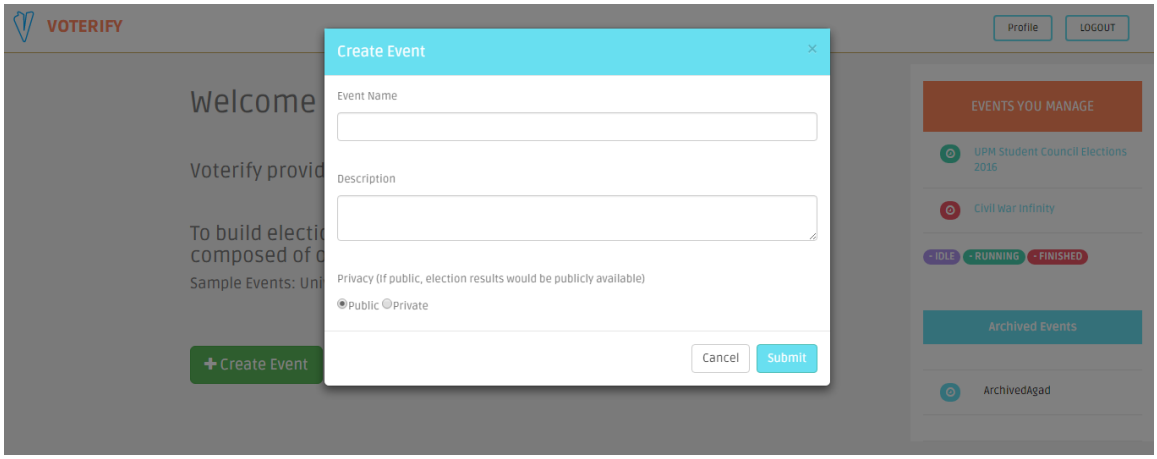


Figure 9: Create an event

Figure 10: Home page of an event

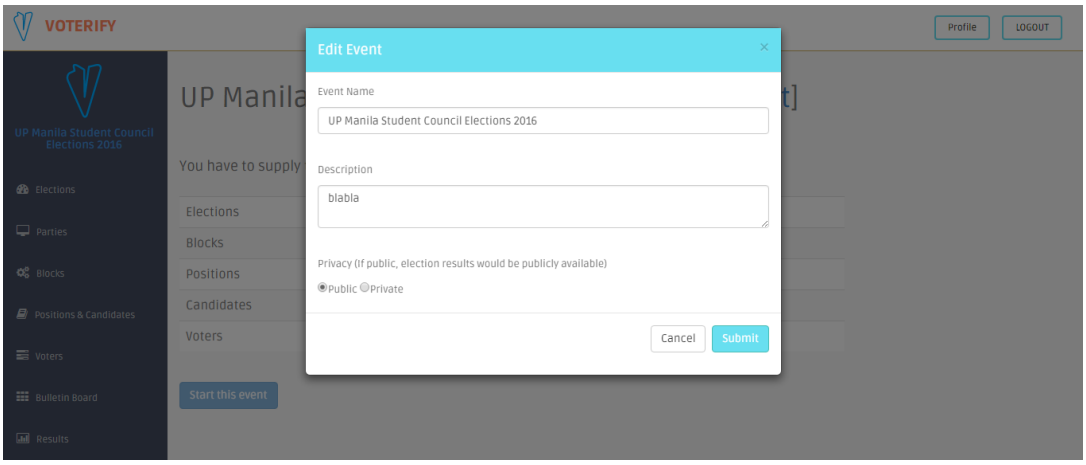


Figure 11: Edit an event

A voting official can add, edit, or delete an election and view the elections he created. He cannot delete elections in use by blocks and positions.

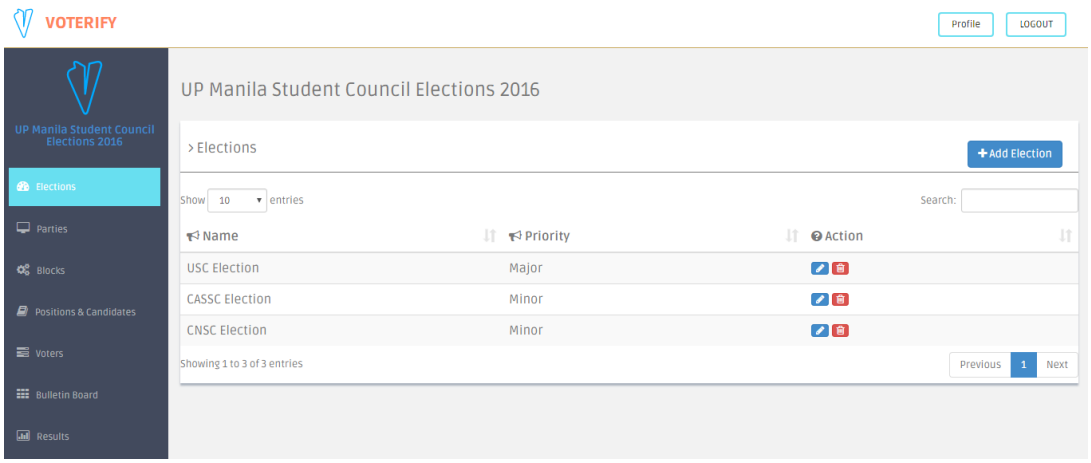


Figure 12: View elections

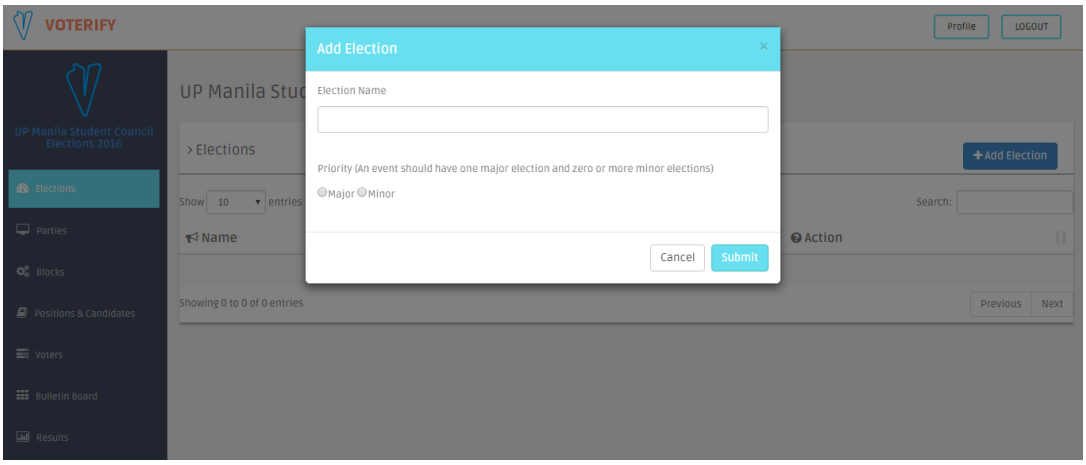


Figure 13: Add an election

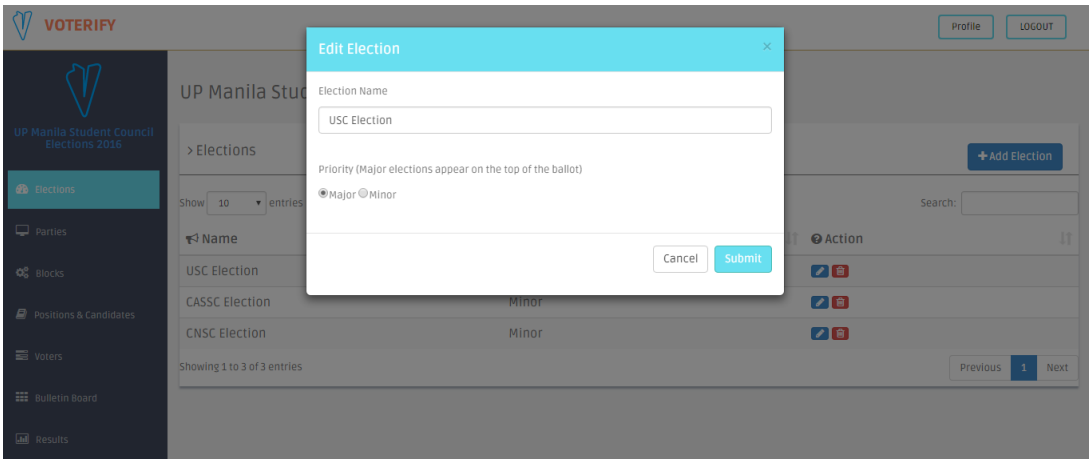


Figure 14: Edit an election

A voting official can add, edit, or delete a party and view the parties he created. He cannot delete parties in use by candidates.

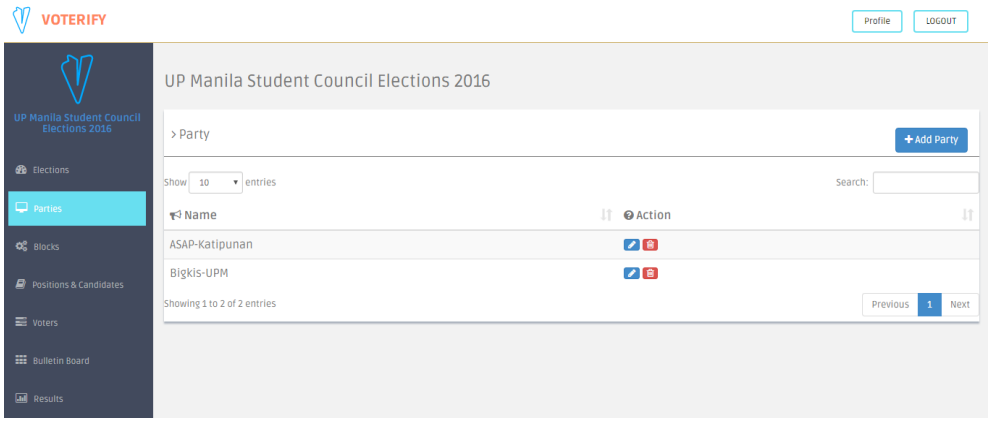


Figure 15: View parties

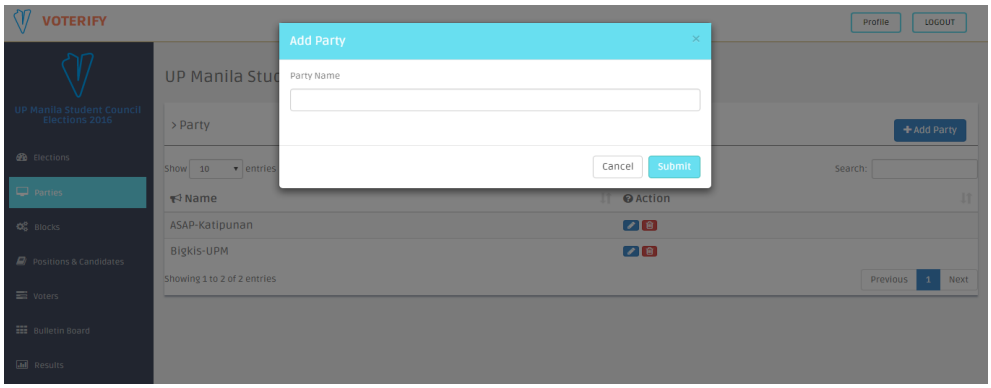


Figure 16: Add a party

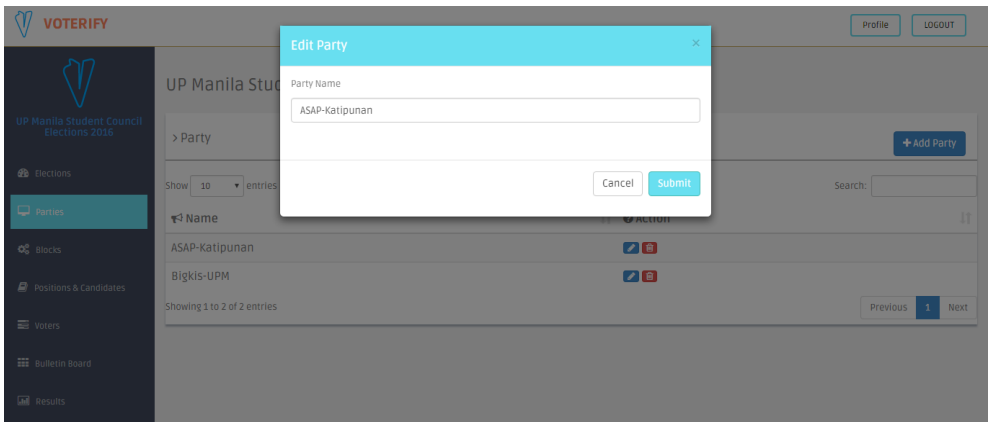


Figure 17: Edit a party

A voting official can add, edit, or delete a block (voter type) and view the blocks he created. He cannot delete blocks in use by positions and voters.

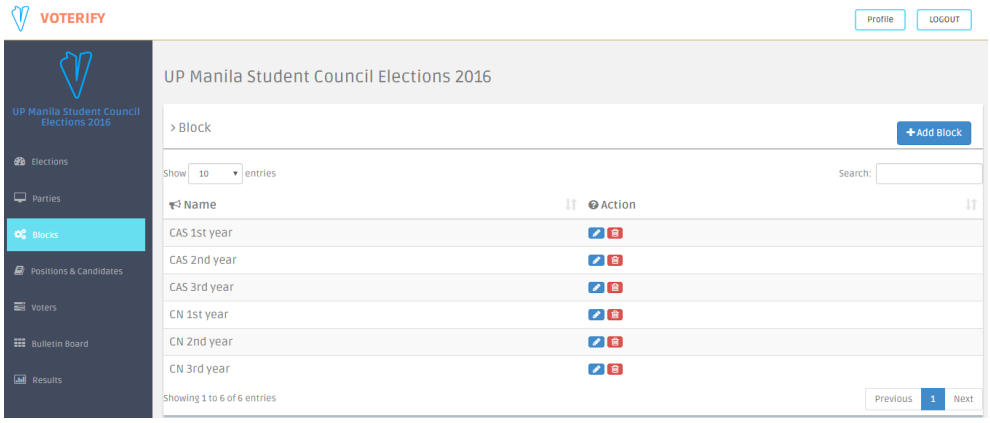


Figure 18: View blocks

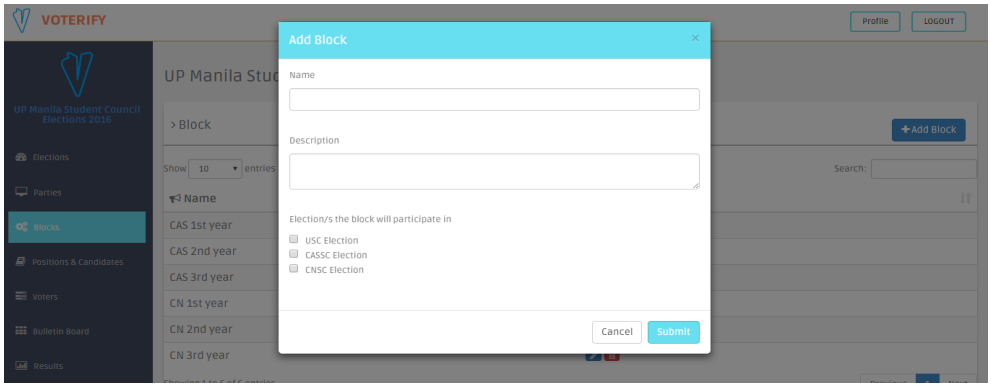


Figure 19: Add a block

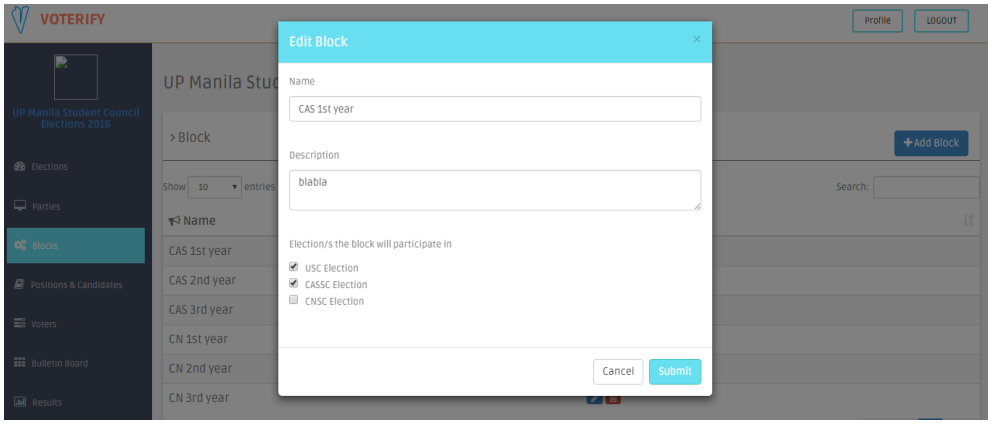


Figure 20: Edit a block

A voting official can add, edit, or delete a position and view the positions he created. If a position is deleted, the candidates for that position are also deleted.

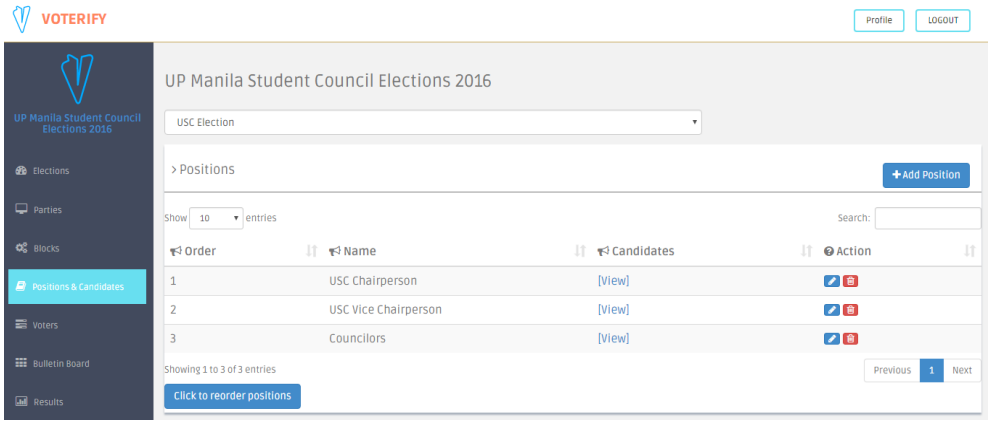


Figure 21: View positions

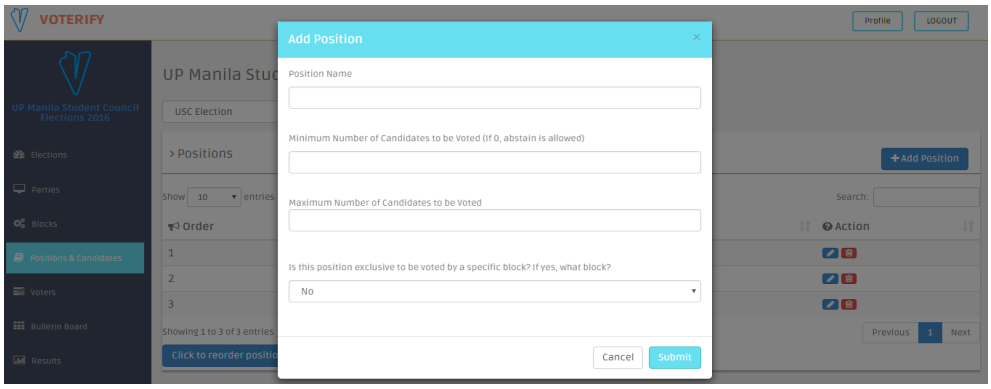


Figure 22: Add a position

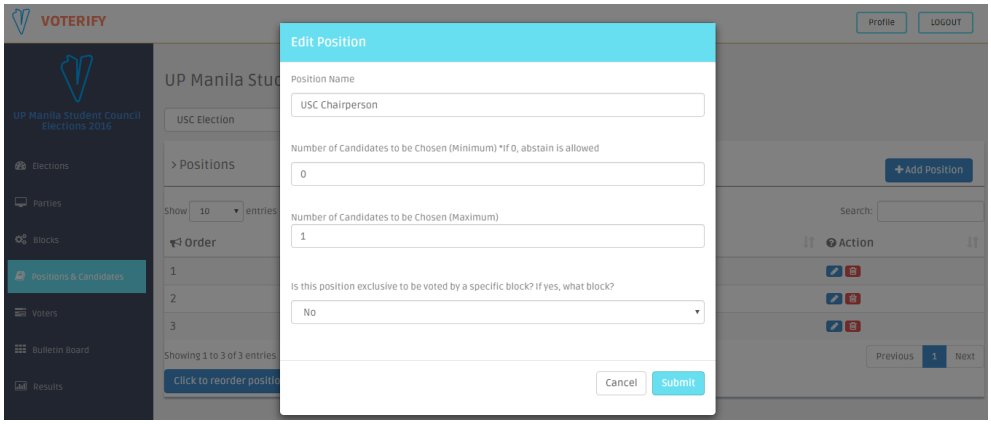


Figure 23: Edit a position

A voting official can reorder the positions through a drag and drop feature.

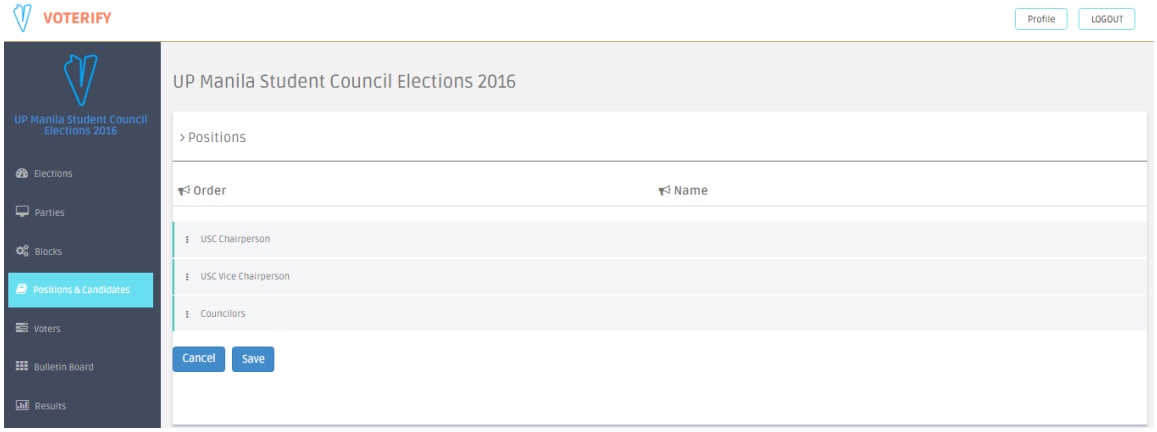


Figure 24: Reorder positions via drag and drop

A voting official can add, edit, or delete a candidate and view candidates for a certain position.

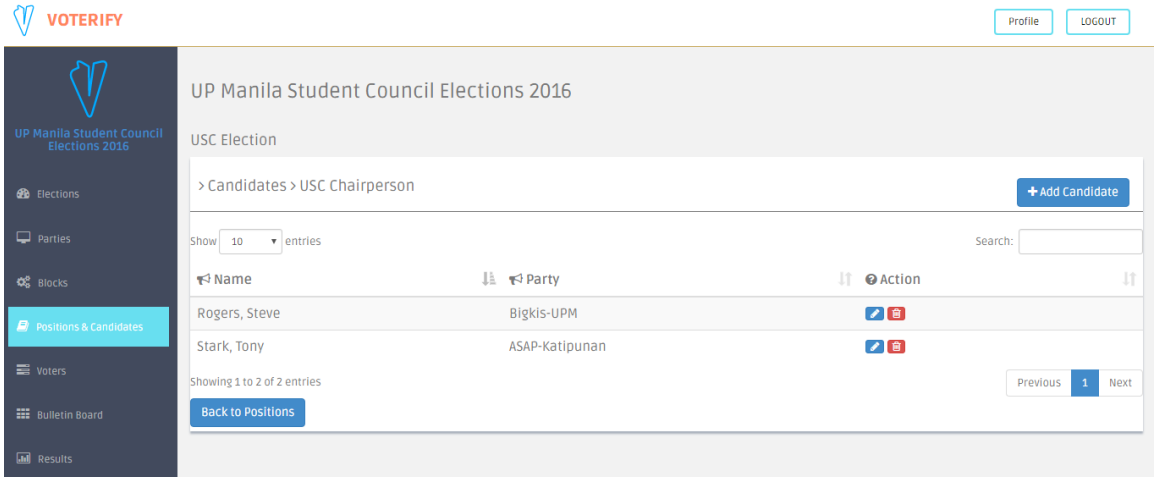


Figure 25: View candidates for a certain position

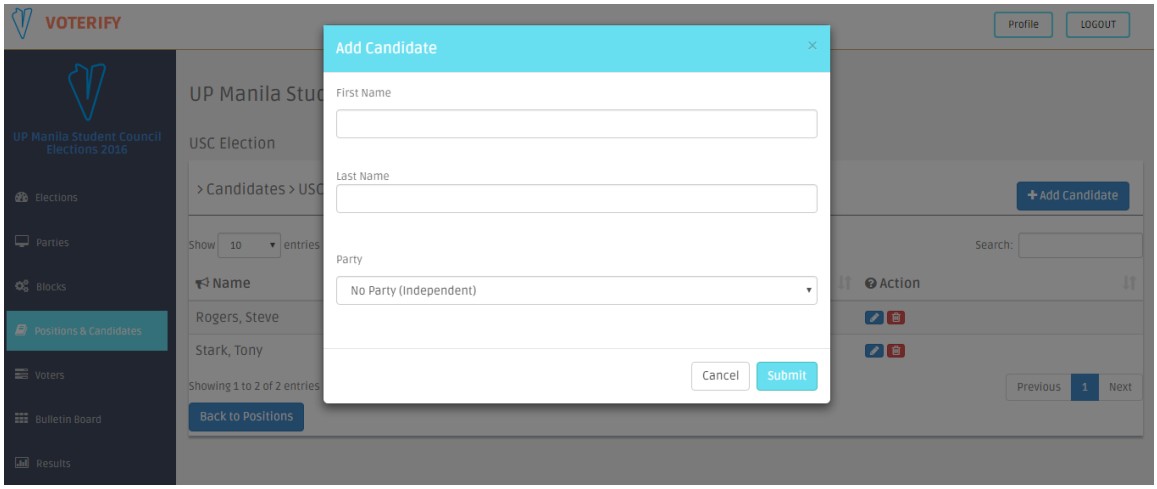


Figure 26: Add a candidate

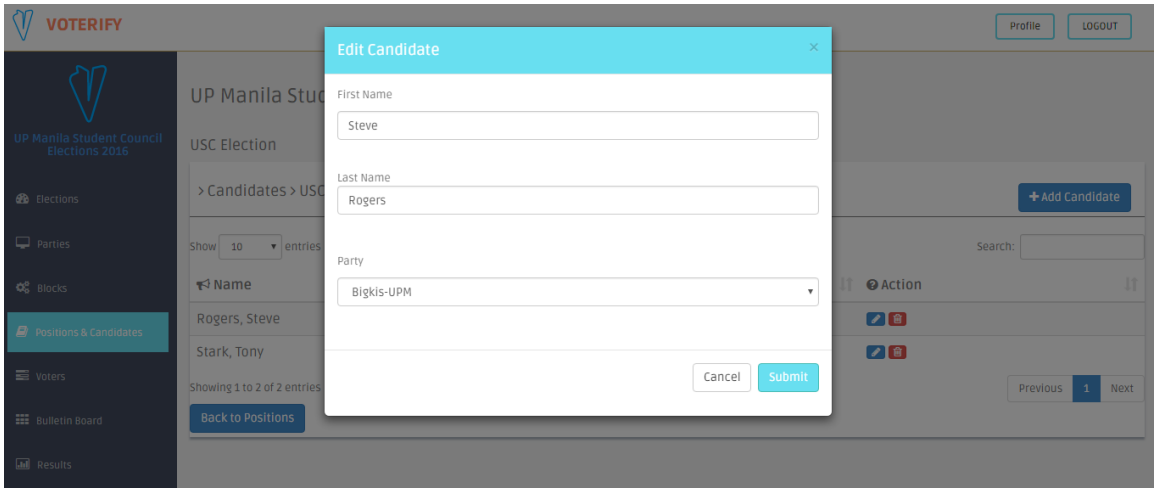


Figure 27: Edit a candidate

A voting official can add, edit, or delete a voter and view voters per block. Also, he can do bulk upload voters via .csv file which contains usernames and emails separated by a comma.

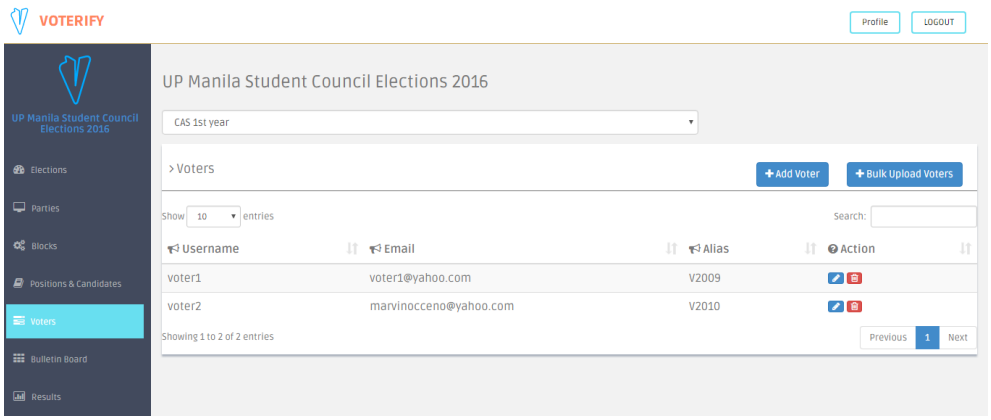


Figure 28: View voters per block

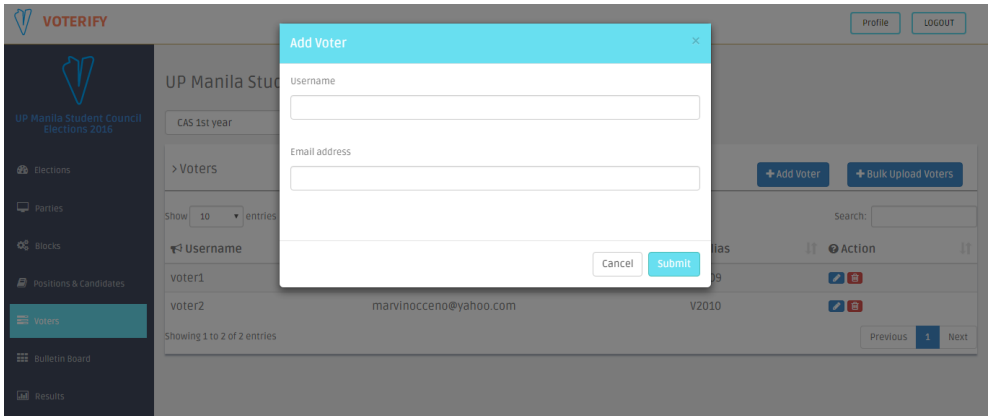


Figure 29: Add a voter

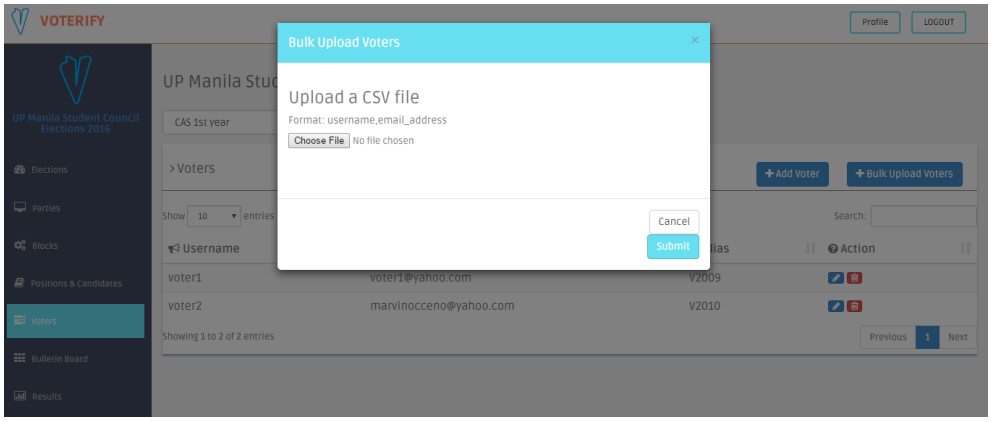


Figure 30: Bulk upload voters

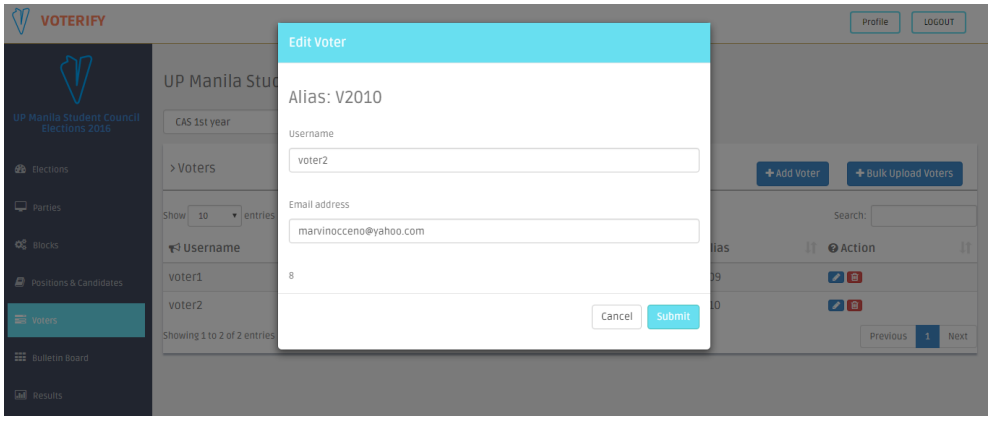


Figure 31: Edit a voter

A voting official can now start an event as long as the elections, blocks, voters, positions, and candidates are not empty. Private keys, which will be used to compute the tally, are generated once they start an event. These keys will be automatically downloaded.

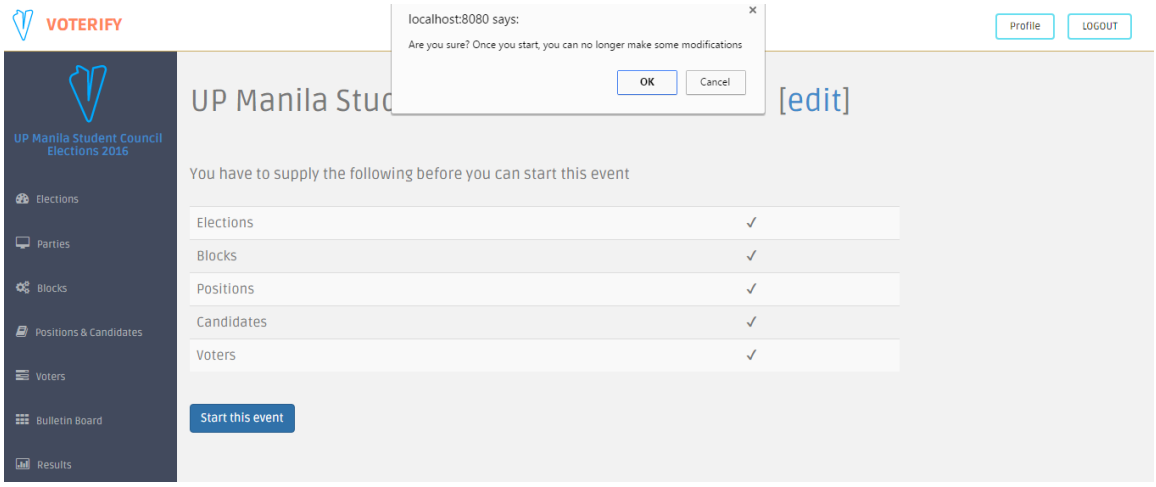


Figure 32: Start an event

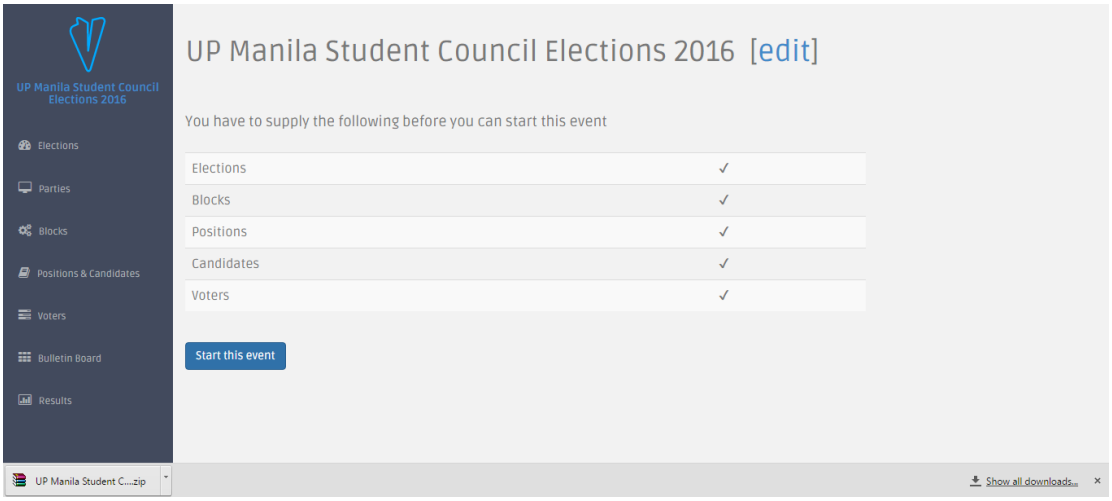


Figure 33: Download private keys

Once an event has been stopped, voting officials can now compute the tally by uploading the private keys. Results will immediately be available.

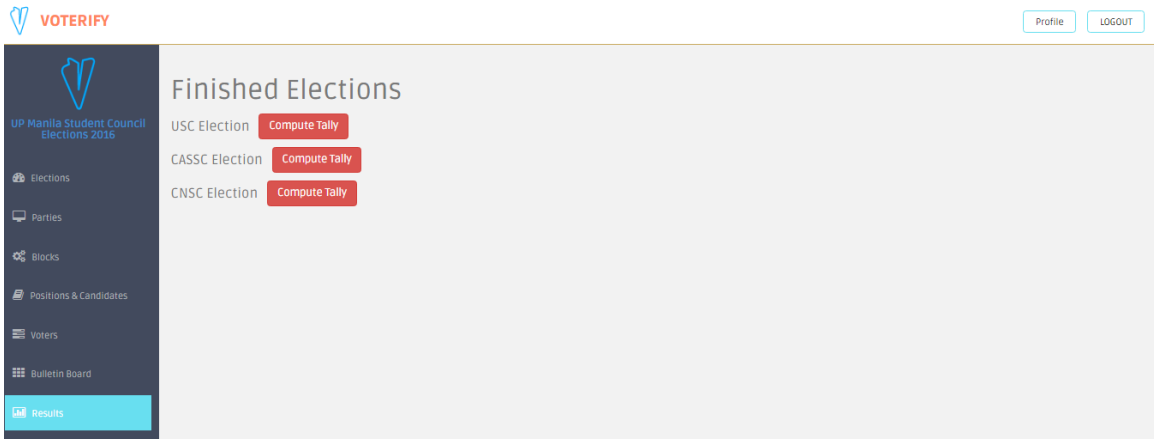


Figure 34: Tallying of results home page for voting officials

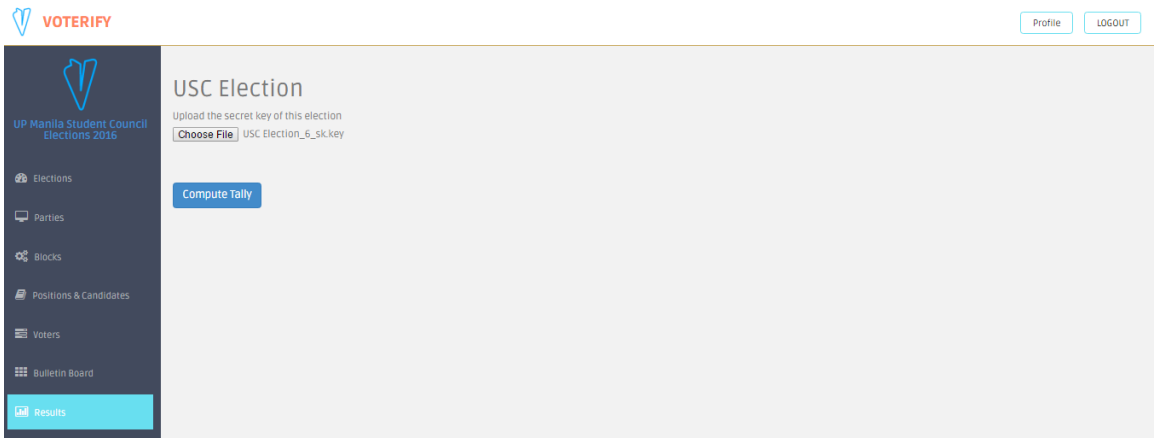


Figure 35: Private key upload

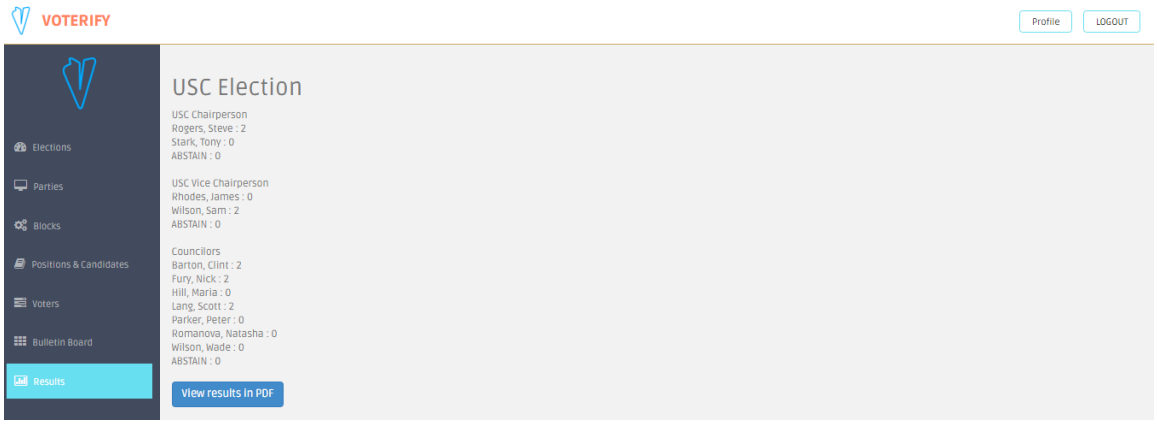


Figure 36: Election results

A voting official can receive vote discrepancy tickets.

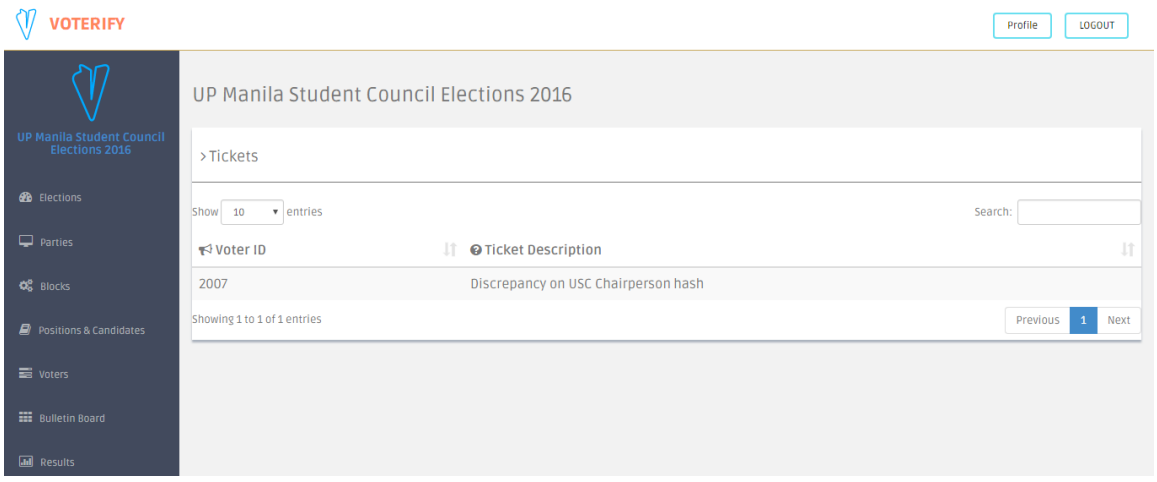


Figure 37: Tickets raised by voters

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

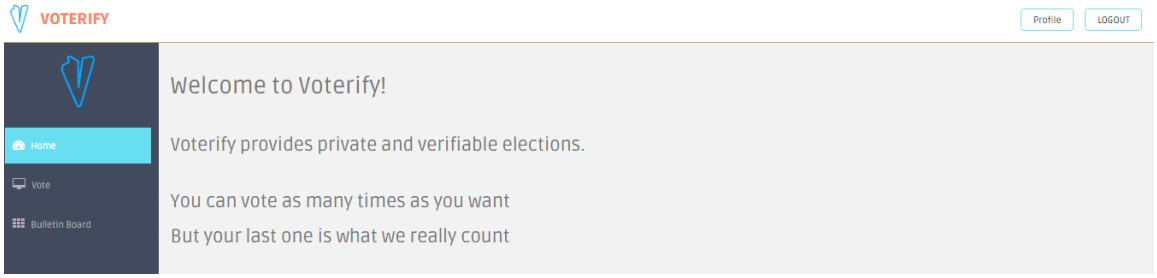


Figure 38: Home page for voters

Voters can cast a vote and see the hashes which correspond to their encrypted votes. Once the votes are submitted, they can download a receipt as a text file which contains these hashes.

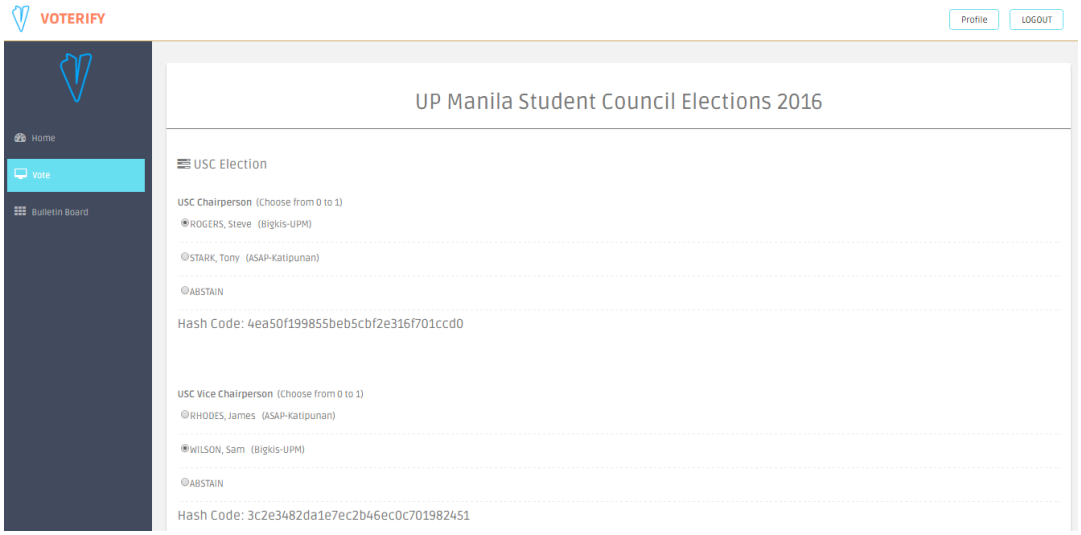


Figure 39: Cast a vote

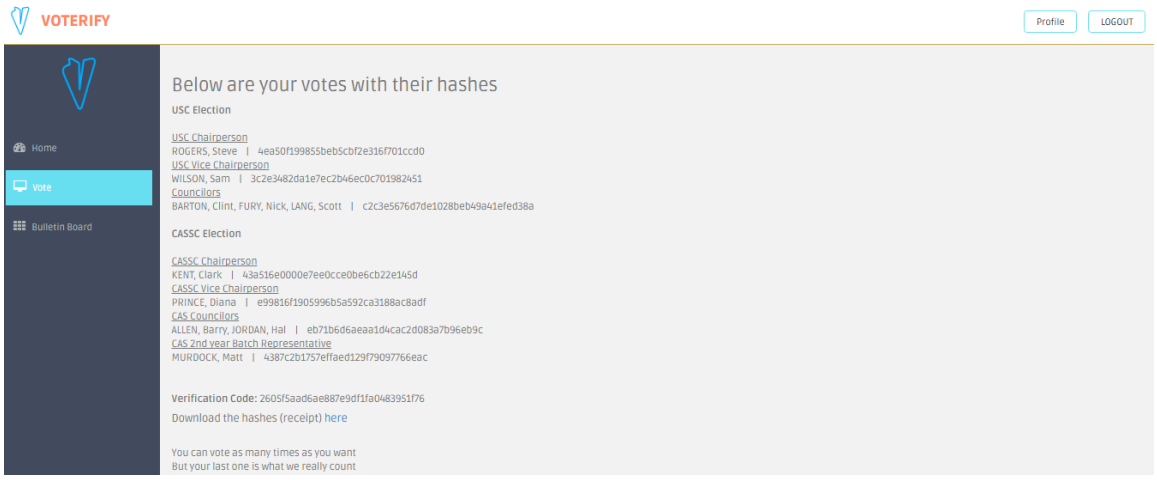


Figure 40: After casting a vote

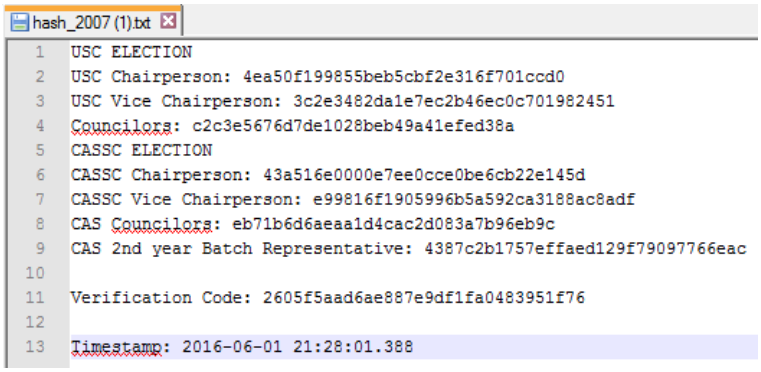


Figure 41: Voter's receipt

A voter can raise a ticket should there be a discrepancy in his votes appearing on the bulletin board.

Figure 42: Raise a ticket

Registered users can edit their profile.

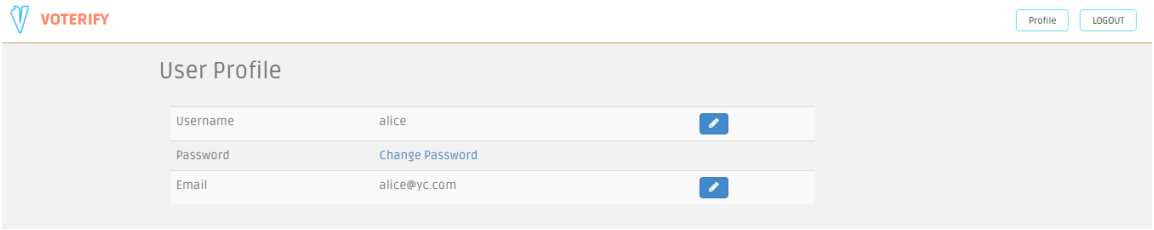


Figure 43: Manage account

The bulletin board of votes, which is publicly available, displays the encrypted votes of the voters.

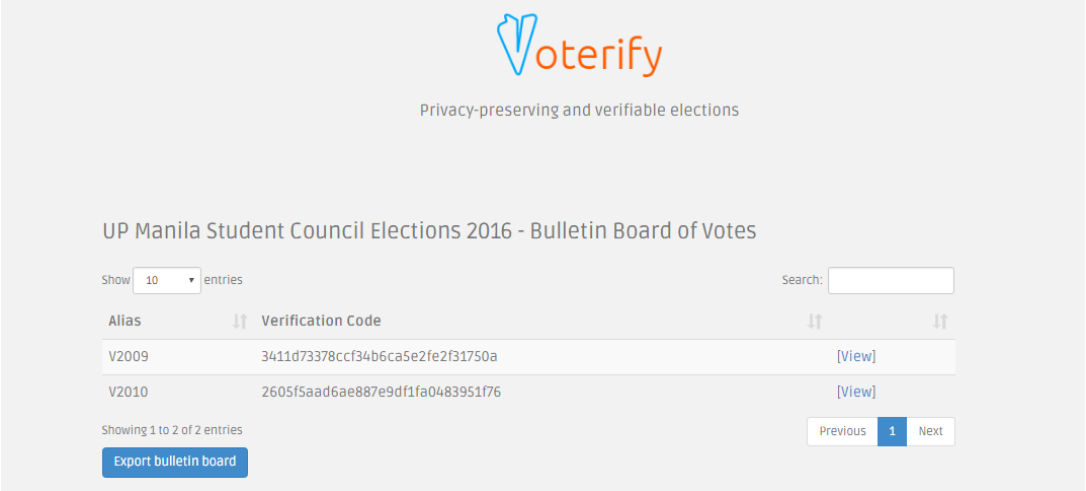
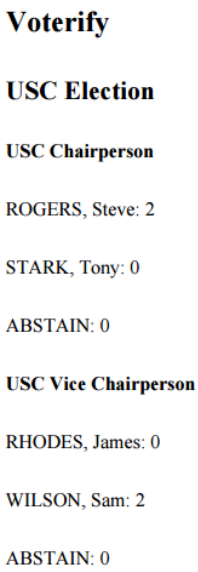


Figure 44: Bulletin Board Menu

Figure 45: Encrypted votes per voter

Public users can download the election results as well as the bulletin board of votes.



Voterify

USC Election

USC Chairperson

ROGERS, Steve: 2

STARK, Tony: 0

ABSTAIN: 0

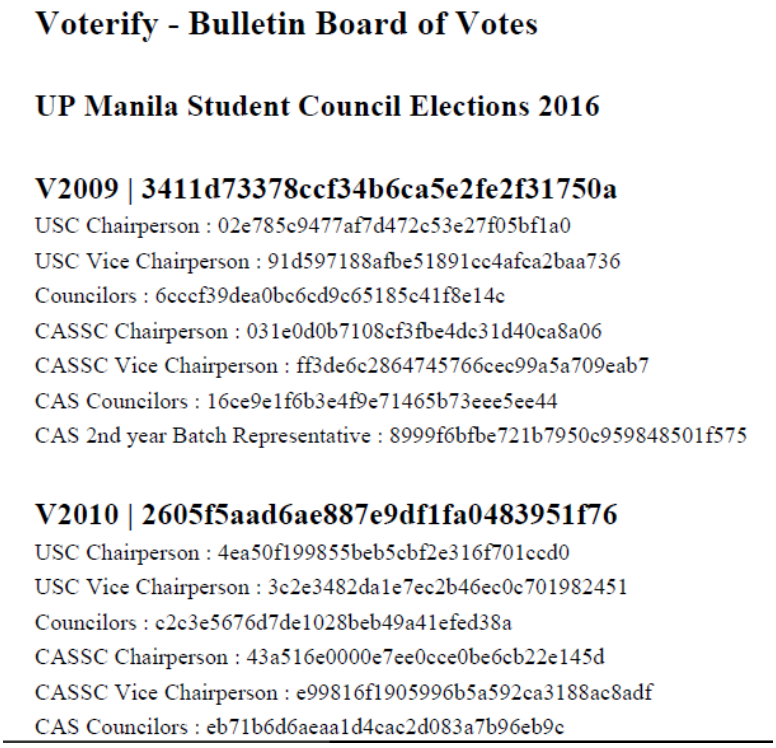
USC Vice Chairperson

RHODES, James: 0

WILSON, Sam: 2

ABSTAIN: 0

Figure 46: Election results



Voterify - Bulletin Board of Votes

UP Manila Student Council Elections 2016

V2009 | 3411d73378ccf34b6ca5e2fe2f31750a

USC Chairperson : 02e785c9477af7d472c53e27f05bf1a0
USC Vice Chairperson : 91d597188afbe51891cc4afca2baa736
Councilors : 6cccf39dea0bc6cd9c65185c41f8e14c
CASSC Chairperson : 031e0d0b7108cf3fbe4dc31d40ca8a06
CASSC Vice Chairperson : ff3de6c2864745766cec99a5a709eab7
CAS Councilors : 16ce9e1f6b3e4f9e71465b73eee5ee44
CAS 2nd year Batch Representative : 8999f6bfbe721b7950c959848501f575

V2010 | 2605f5aad6ae887e9df1fa0483951f76

USC Chairperson : 4ea50f199855beb5cbf2e316f701ccd0
USC Vice Chairperson : 3c2e3482da1e7ec2b46ec0c701982451
Councilors : c2c3e5676d7de1028beb49a41efed38a
CASSC Chairperson : 43a516e0000e7ee0cce0be6cb22e145d
CASSC Vice Chairperson : e99816f1905996b5a592ca3188ac8adf
CAS Councilors : eb71b6d6aeaa1d4cac2d083a7b96eb9c

Figure 47: Bulletin board of votes

The administrator can add, edit, or delete a voting official and view all voting officials.

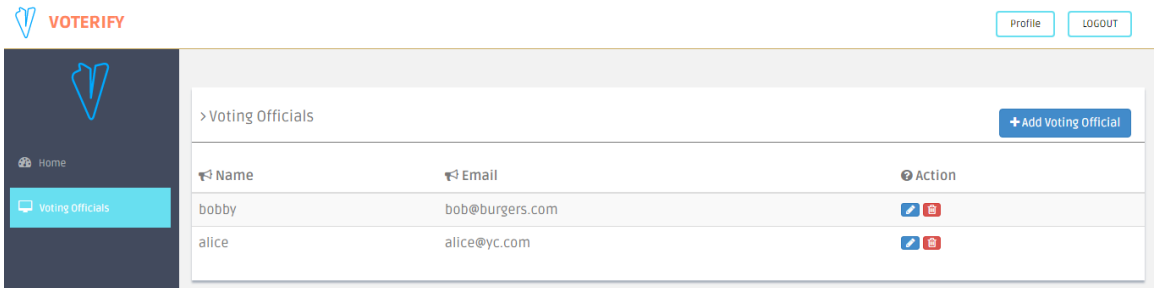


Figure 48: View all voting officials

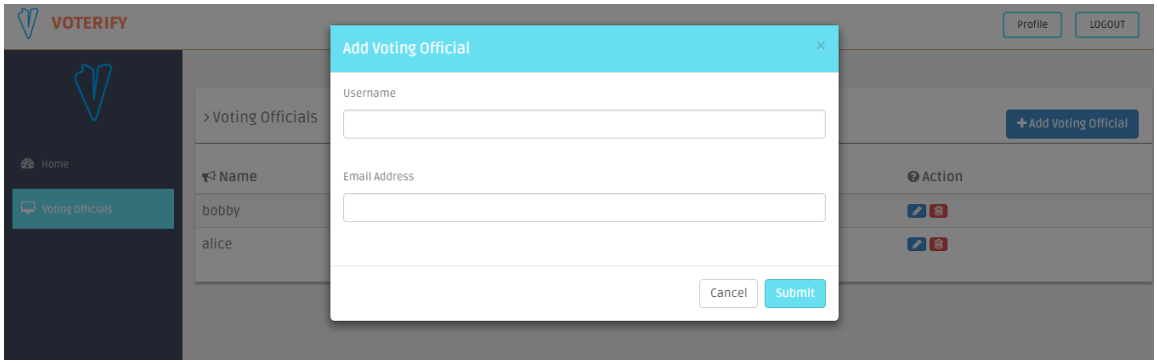


Figure 49: Add a voting official

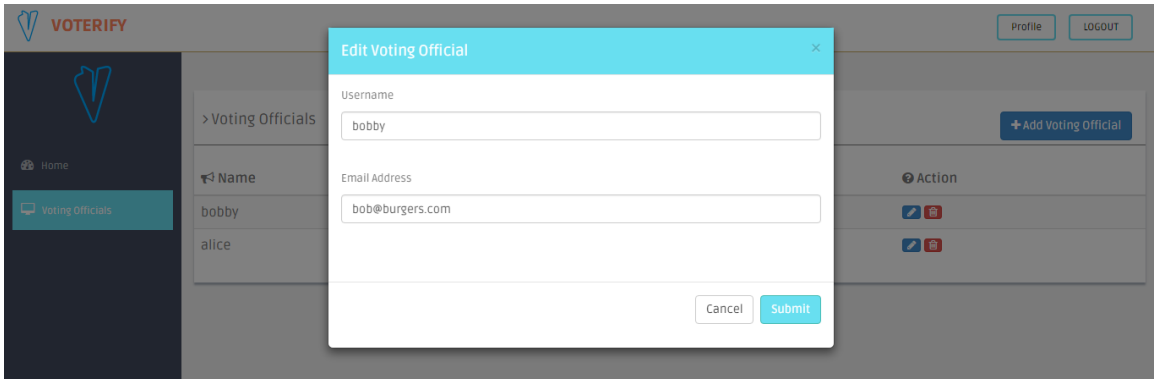


Figure 50: Edit a voting official

VI. Discussions

The Internet voting system named as Voterify provides privacy-preserving and verifiable elections. Homomorphic encryption, particularly the Paillier scheme, is the underlying cryptographic primitive which helps the system achieve these properties.

Voterify utilizes THEP to perform the following fundamental homomorphic operations – key generation, encryption, and decryption. Every election created has a 1024-bit public-and-private key pair. The private and public keys are generated with the following:

```
PrivateKey sk = new PrivateKey(1024);
PublicKey pk = sk.getPublicKey();
// Update election's public key; set it to n in the database
election.setPk_n(new BigInteger(pk.getN().toString()).toString(16));
// Update election's private key; set it to λ in the database
election.setSk_lambda(new BigInteger(sk.getLambda().toString())
.toString(16));
// Give μ to the voting official; do not save in the database
String mu = new BigInteger(sk.getMu().toString()).toString(16);
```

To achieve privacy, votes are encrypted and are never decrypted individually. Recall in Chapter IV that such vote has a plaintext form and the key components of a vote stored in the database are the `voter_id`, `position_id`, and `evote`. Suppose that the plaintext vote is 100. The mechanism on how it is converted to its lengthy encrypted form and how that vote is mapped to a certain position is as follows:

```
PublicKey pk = new PublicKey(1024, new BigInteger(election.getPk_n(),
16));
BigInteger bi = new BigInteger("100");
EncryptedInteger ei = new EncryptedInteger(bi, pk);
```

```
vote.setEvote(ei.getCipherVal().toString());
vote.setPosition_id(position_id);
vote.setVoter_id(voter_id);
```

The field `evote` is stored as a string rather than an object since it's easier to fetch strings than objects. Recall that tallying is position-wise. Once all the `evote`s for a certain position are fetched to begin the counting of votes, their sum is computed through the following process:

```
List<Vote> votes = getVotes(position_id);
EncryptedInteger[] evotes = new EncryptedInteger[votes.size()];
EncryptedInteger sum = new EncryptedInteger(pk);
for(int i=0; i<votes.size(); i++){
   evotes[i] = new EncryptedInteger(pk);
   evotes[i].setCipherVal(new BigInteger(votes.get(i).getEvote());
   sum = sum.add(evotes[i]);
}
```

In THEP, the method `setCipherVal` under the `EncryptedInteger` class is private by default. We changed it to public so that other Java classes can use it. This method is employed during the tallying process as shown above.

```
public void setCipherVal(BigInteger cipherval) {
   this.cipherval = cipherval;
}
```

The sum needs to be decrypted in order to reveal the election results. The steps needed in the decryption process are as follows:

```
PublicKey pk = new PublicKey(1024, new BigInteger(election.getPk_n(),16));
```

```
PrivateKey sk = new PrivateKey(1024);
sk.setLambda(new BigInteger(election.getSk_lambda(),16));
// secretKey comes from the voting official
sk.setMu(new BigInteger(secretKey, 16));
sum.decrypt(sk).toString();
```

We added the following methods to THEP under the `PrivateKey` class. These methods are called during the decryption process as shown above. The values of `pub`, `lambda`, and `mu` are set once a private key is generated.

```
public void setPublicKey(PublicKey pub) {
   this.pub = pub;
}
public void setLambda(BigInteger lambda) {
   this.lambda = lambda;
}
public void setMu(BigInteger mu) {
   this.mu = mu;
}
```

The idea of the decryption process is to set up a private key whose values are set to λ, the one fetched from the database, and μ, the one uploaded by the voting official via the *.key* file. To detect whether the key uploaded is valid, we created a key validation method which accepts two parameters - the election's public key and the private key uploaded by the official. The method encrypts an arbitrary `BigInteger` using the public key and decrypts it using the private key. If the result matches the arbitrary `BigInteger`, it means that the private key is valid. This method is useful whenever a voting official uploads an incorrect key.

There is a publicly available bulletin board per event. This board is automatically published once the event has been stopped by the voting official. The board lists down the aliases and verification codes of the voters. Once the verification code of a voter is clicked, the position-wise hashed votes of that voter are displayed. Recall that these hashes are the MD5 hashes of their encrypted votes. A voter can verify his/her votes by checking if the hashed votes appearing on his receipt is the same with what was posted on the bulletin board.

Voterify has the following set of users: voting officials, voters, and the admin. Voting officials can create an event which consists of one or more elections. They can also manage parties, positions, candidates, blocks, and voters. Also, they will be given private keys which they will use for tallying purposes. It is assumed that they are responsible enough to safeguard the keys. On one hand, the voters can securely cast a vote in an election, view their votes in an encrypted form, and notify the voting official should there be such discrepancy in their votes. The admin can manage the list of voting officials. A public user can view and download results and bulletin boards if the voting official considers an event public. All these signify that the voting system is able to fulfill all its objectives.

VII. Conclusions

Voterify is created in answer to the growing need of automated, private, and verifiable elections. Homomorphic encryption is considered to achieve these desirable properties a secure voting system should possess. Paillier algorithm, a special form of homomorphic encryption, is utilized to encrypt the votes and tally election results. The voting system makes use of The Homomorphic Encryption Project (THEP) to implement the said algorithm.

The web application is able to provide the several functionalities for different user roles. Voting officials can setup an event which consists of elections, positions, candidates, parties, blocks, and voters. They can also tally election results while keeping the votes private. Voters can securely cast their votes and verify if these appear correctly on the bulletin board.

It can be concluded that Voterify indeed provides privacy-preserving and verifiable elections. Thus, the system contributes in today's world where the demand for privacy of digital data (i.e., electronic votes, financial data, medical records) is very high.

VIII. Recommendations

Voterify currently uses 1024-bit Paillier keys. As computer processors become faster, brute-force attacks are made simpler. Thus, it is quite recommended to have longer key sizes (i.e., 2048-bit or 4096-bit) to improve security. However, system performance especially the vote counting part must not be compromised since one primary purpose of Internet voting in general is to speed up the counting of votes.

Votes, in this web application, are encrypted on the server side. It has always been an argument if server-side encryption is better than client side or the other way around. Both have pitfalls. Server-side encryption means that one sends a data in plaintext form, through a trusted third party, before encrypting it. The problem with this is that when the packets are intercepted, sensitive data (i.e., votes) can be revealed. On the other hand, client-side encryption ensures that data are encrypted first before sending them to the server. However, it also poses a big threat. Users can see and even manipulate the encryption algorithm. In the case of I-voting, election results might be greatly affected since tweaking the said algorithm can potentially create irregular votes. With this regard, future developers need to be critical enough to pick a side.

It is also suggested to consider more secure software libraries other than THEP which also implement Paillier algorithm. Take into consideration the ones which perform Threshold Paillier or other Paillier variants that have a better secret-key sharing scheme. Again, more security is needed especially in today's world where information security is in short supply.

In addition, one can try to create graphs and charts out of election results for analytical purposes. The aesthetics of the system as well as its user-friendliness can also be improved.

IX. Bibliography

[1] V. Cortier, "Formal verification of e-voting: solutions and challenges," *ACM SIGLOG News*, vol. 2, pp. 25–34, January 2015.

[2] J. A. L. Piscos, "Secure multiparty computation for internet voting," Undergraduate Thesis, University of the Philippines Manila, Manila, Philippines, 2015.

[3] R. Kusters, T. Truderung, and A. Vogt, "Clash attacks on the verifiability of e-voting systems," *IEEE Computer Society*, pp. 395–409, 2012.

[4] H. N. Oo and A. M. Aung, "Implementation and analysis of secure electronic voting system," *International Journal of Scientific and Technology Research*, vol. 2, pp. 158–161, 2013.

[5] D. Hrestak and S. Picek, "Homomorphic encryption in the cloud," *MIPRO*, pp. 1400–1404, May 2014.

[6] A. Kiayias, M. Korman, and D. Walluck, "An internet voting system supporting user privacy," *IEEE Computer Society*, 2006.

[7] D. Gritzalis, "Secure electronic voting," *Kluwer Academic Publishers*, 2002.

[8] J. Sen, "Homomorphic encryption: Theory and application," *Theory and Practice of Cryptography and Network Security Protocols and Technologies*, 2013.

[9] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J. A. Halderman, "Security analysis of the Estonian internet voting system," *Proc. 21st ACM Conference on Computer and Communications Security (CCS14)*, 2014.

[10] K. Gjosteen, "The Norwegian internet voting protocol," *Cryptology ePrint Archive*, August 9 2013.

[11] C. Culnane, P. Y. A. Ryan, S. Schneider, and V. Teague, "vVote: A verifiable voting system," *ACM Trans. Info. Syst.*, vol. 1, June 2015.

[12] B. Adida, "Helios: Web-based open-audit voting," *USENIX Association*, pp. 335–348, August 2008.

[13] B. Adida, O. Pereira, O. de Marneffe, and J.-J. Quisquater, "Electing a university president using open-audit voting: Analysis of real-world use of helios," *USENIX Association*, June 25 2009.

[14] S. Ibrahim, M. Kamat, M. Salleh, and S. R. A. Aziz, "Secure e-voting with blind signature," *NCTT 2003*, pp. 193–197, 2003.

[15] R. A. Al-Saidi, "A secure electronic voting scheme based on evox-ma and revs e voting blind signature protocols," Masters Thesis, Middle East University, July 2011.

[16] K. Sampigethaya and R. Poovendran, "A survey on mix networks and their secure applications," *Proceedings of the IEEE*, pp. 2142–2181, December 2006.

[17] R. L. Rivest, L. Adleman, and M. L. Dertouzous, "On data banks and privacy homomorphisms," *Foundations of Secure Computation, Academic Press*, pp. 169–179, 1978.

[18] "Lecture 17: Introduction to electronic voting." http://courses.csail.mit.edu/6.897/spring04/L17.pdf. Accessed: 2015-11-19.

[19] L. J. M. Aslett, P. M. Esperanca, and C. C. Holmes, "A review of homomorphic encryption and software tools for encrypted statistical machine learning," *Department of Statistics, University of Oxford*, August 26 2015.

[20] H. Hussien and H. Aboelnaga, "Design of a secured e-voting system," *Computer Applications Technology (ICCAT), 2013 International Conference on*, pp. 1–5, Jan 2013.

[21] N. A. Lopez, "Database encryption using CryptDB for DentISt: Dental Information System (DentISt) 4.0," Undergraduate Thesis, University of the Philippines Manila, Manila, Philippines, 2014.

[22] F. Yumeng, T. Liye, L. Fanbao, and G. Chong, "Electronic voting: A review and taxonomy," *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on*, pp. 912–917, Aug 2012.

[23] A. Huszti, "A homomorphic encryption-based secure electronic voting scheme," *Publ. Math. Debrecen*, pp. 479–496, 2011.

[24] "thep." https://code.google.com/p/thep/. Accessed: 2015-12-08.

[25] "c3p0 - jdbc3 connection and statement pooling." http://www.mchange.com/projects/c3p0-0.9.5-pre4/#configuring_to_avoid_memory_leaks_on_redeploy. Accessed: 2016-05-31.

X. Appendix

A. Source Code

A..1 Actions

AccountAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.UserDao;
import com.sp.vote.method.BCrypt;
import com.sp.vote.model.User;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;
import org.apache.commons.validator.routines.
        EmailValidator;
import org.apache.struts2.interceptor.
        SessionAware;

import java.util.Map;


public class AccountAction extends ActionSupport
implements SessionAware, Preparable {

private static final long serialVersionUID = 1L;

private User user;
private UserDao udao = new UserDao();

private String oldpass, newpass, retpass;
private boolean hasError[] = new boolean[3];

private String mode;
private int uid;
private Map<String, Object> sessionMap;


@Override
public void prepare() throws Exception {
uid = (Integer)sessionMap.get("user_id");
setUser(udao.getUserById(uid));
}


@Override
public String execute() throws Exception{
String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap)){
if(mode != null){
if(mode.equals("epw")){
if(myValidate()){
changePassword();
ret = "editPassSuccess";
}
else{
ret = "input";
}
}
else if(mode.equals("eun")){
if(!udao.getAllUsernames().contains
(user.getUsername())){
udao.editUser(user);
ret = "editPassSuccess";
}
else{
hasError[0] = true;
this.addFieldError("unF",
        "Username already exists");
ret = "input";
}
}
else if(mode.equals("eea")){
if(!EmailValidator.getInstance().
        isValid(user.getEmail())){
hasError[2] = true;
this.addFieldError("emailF",
        "Invalid email");
ret = "input";
}
else{
udao.editUser(user);
ret = "editPassSuccess";
}
}
}
else
ret = "display";
}

return ret;
}

public boolean myValidate(){
String salt = udao.getUser(user.getUsername()).
getSalt();
String hashedPass = BCrypt.hashpw(oldpass, salt);

if(!hashedPass.equals(user.getPassword())){
this.addFieldError("oldpF",
"Incorrect Password");
hasError[1] = true;
return false;
}
if(!newpass.equals(retpass)){
this.addFieldError("retpF",
"Passwords do not match");
hasError[1] = true;
return false;
}

return true;
}

public void changePassword(){
String salt = udao.getUser(user.getUsername())
.getSalt();
user.setPassword(BCrypt.hashpw(newpass, salt));
udao.editUser2(user);
}

@Override
public void setSession(Map<String, Object> sessionMap){
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

public String getOldpass() {
return oldpass;
}
public void setOldpass(String oldpass) { //
this.oldpass = oldpass;
}

public String getNewpass() {
return newpass;
}
public void setNewpass(String newpass) {
this.newpass = newpass;
}

public String getRetpass() {
return retpass;
}
public void setRetpass(String retpass) {
this.retpass = retpass;
}

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

public boolean[] isHasError() {
return hasError;
}
```

```
public void setHasError(boolean[] hasError) {
this.hasError = hasError;
}
}
```

BallotFormAction.java

```
package com.sp.vote.action;

import com.sp.vote.method.Hasher;
import com.sp.vote.model.*;
import com.sp.vote.dao.*;

import com.opensymphony.xwork2.ActionSupport;
import org.apache.struts2.interceptor.Session
        Aware;

import thep.paillier.EncryptedInteger;
import thep.paillier.PublicKey;

import java.math.BigInteger;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.Map;


public class BallotFormAction extends
        ActionSupport implements SessionAware {

private static final long serialVersionUID = 1L;

private int blockID;
private Event event;
private List<Election> elections;
private List<List<Position>> positions;
private List<List<List<Candidate>>> candidates;
private List<List<List<String>>> hashes;
private List<List<List<String>>> evotes;

private String[][] singlePos;
private String[][][] multiplePos;

private int maxPos, maxCan;
private String verificationCode;
private Map<String, Object> sessionMap;

private PublicKey pk;
private BigInteger bi;
private EncryptedInteger ei;


@Override
public String execute() throws Exception {
EventDao evdao = new EventDao();
ElectionDao eldao = new ElectionDao();
PositionDao podao= new PositionDao();
PartyDao padao = new PartyDao();
VoterDao vrdao = new VoterDao();
CandidateDao cdao = new CandidateDao();

Candidate abs = new Candidate("", "ABSTAIN");
Hasher h = new Hasher();

blockID = vrdao.getVoter((Integer) sessionMap.
        get("user_id")).getBlock_id();
elections = new ArrayList<Election>();
elections = eldao.getVoterElections(blockID);
sort_elections();
event = evdao.getEventOfAnElection(
        elections.get(0));

positions = new ArrayList<List<Position>>();
candidates = new ArrayList<List<
        List<Candidate>>>();
evotes = new ArrayList<List<List<String>>>();
hashes = new ArrayList<List<List<String>>>();

char[] chars;
maxCan = maxPos = 0;

Party partyCan;
String lastName;

for(int i=0; i<elections.size(); i++){
pk = new PublicKey(1024, new BigInteger
        (elections.get(i).getPk_n(), 16));

positions.add(new ArrayList<Position>());
positions.set(i, podao.getVoterPositions
        (elections.get(i).getElection_id(),
                blockID));
Collections.sort(positions.get(i));

if(maxPos < positions.get(i).size())
maxPos = positions.get(i).size();

candidates.add(new ArrayList<List<Candidate>>
        (positions.get(i).size()));
evotes.add(new ArrayList<List<String>>
        (positions.get(i).size()));
hashes.add(new ArrayList<List<String>>
        (positions.get(i).size()));
for(int j=0; j<positions.get(i).size(); j++){
candidates.get(i).add(new ArrayList<Candidate>());
candidates.get(i).set(j, cdao.getCandidatesBy
        Position(positions.get(i).get(j).
                getPosition_id()));
Collections.sort(candidates.get(i).get(j),
        Candidate.candidateNameComp);
evotes.get(i).add(new ArrayList<String>());
hashes.get(i).add(new ArrayList<String>());

// Abstain is allowed if required_min = 0
if(positions.get(i).get(j).getRequired_min() == 0)
candidates.get(i).get(j).add(abs);

if(maxCan < candidates.get(i).get(j).size())
maxCan = candidates.get(i).get(j).size();

chars = new char[candidates.get(i).get(j).size()];
Arrays.fill(chars, '0');

for(int k=0; k<candidates.get(i).get(j).size();
        k++){
lastName = candidates.get(i).get(j).get(k).
        getLast_name();
if(!lastName.equals("ABSTAIN")){
partyCan = padao.getParty(candidates.get(i).
        get(j).get(k).getParty_id());
candidates.get(i).get(j).get(k).setTemp
        ("(" +partyCan.getName()+ ")");
candidates.get(i).get(j).get(k).setLast_name
        (lastName.toUpperCase() + ",");
}

chars[k] = '1';
evotes.get(i).get(j).add(String.valueOf(chars));
hashes.get(i).get(j).add("");
bi = new BigInteger(evotes.get(i).get(j).get(k));
ei = new EncryptedInteger(bi, pk);

evotes.get(i).get(j).set(k, ei.getCipherVal().
        toString());
hashes.get(i).get(j).set(k, h.computeHash(
        evotes.get(i).get(j).get(k), "MD5"));
chars[k] = '0';
}

}
}

filler();

sessionMap.put("officialElections", elections);
sessionMap.put("officialPositions", positions);
sessionMap.put("officialCandidates", candidates);
sessionMap.put("officialEvotes", evotes);
sessionMap.put("officialHashes", hashes);
sessionMap.put("officialBlockID", blockID);

return SUCCESS;
}

public void filler(){
singlePos = new String[elections.size()][maxPos];
multiplePos = new String[elections.size()][maxPos]
        [maxCan];
}

public void sort_elections(){
Collections.sort(elections);
}

@Override
public void setSession(Map<String, Object> sessionMap){
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public int getBlockID() {
return blockID;
}
```

```
public void setBlockID(int blockID) {
this.blockID = blockID;
}

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

public List<List<Position>> getPositions() {
return positions;
}
public void setPositions(List<List<Position>>
        positions) {
this.positions = positions;
}

public List<List<List<Candidate>>>
        getCandidates() {
return candidates;
}
public void setCandidates(List<List<List
        <Candidate>>> candidates) {
this.candidates = candidates;
}

public List<List<List<String>>> getEvotes() {
return evotes;
}
public void setEvotes(List<List<List<String>>>
        evotes) {
this.evotes = evotes;
}

public List<List<List<String>>> getHashes() {
return hashes;
}
public void setHashes(List<List<List<String>>>
        hashes) {
this.hashes = hashes;
}

public String[][] getSinglePos() {
return singlePos;
}
public void setSinglePos(String[][] singlePos){
this.singlePos = singlePos;


public String[][][] getMultiplePos() {
return multiplePos;
}
public void setMultiplePos(String[][][] multiplePos){
this.multiplePos = multiplePos;
}

public String getVerificationCode(){
return verificationCode;
}
public void setVerificationCode(String
        verificationCode){
this.verificationCode = verificationCode;
}

public Event getEvent() {
return event;
}
public void setEvent(Event event) {
this.event = event;
}

}
```

BlankAction.java

```
package com.sp.vote.action;

import com.opensymphony.xwork2.ActionSupport;

public class BlankAction extends ActionSupport{

}
```

BlockAction.java

```
package com.sp.vote.action;

import com.sp.vote.model.Block;
import com.sp.vote.model.Election;
import com.sp.vote.model.Event;
import com.sp.vote.dao.BlockDao;
import com.sp.vote.dao.ElectionDao;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;
import org.apache.struts2.interceptor.
SessionAware;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Map;


public class BlockAction extends ActionSupport
 implements SessionAware, Preparable{

private static final long serialVersionUID = 1L;

private BlockDao bdao = new BlockDao();

private List<Event> events;
private List<Election> elections;
private List<Integer> savedElections;
private List<Integer> myElections;
private List<Block> blocks;

private Block block;
private Election  election;
private Event event;

private String mode;
private boolean[] hasError = new boolean[2];
private Map<String, Object> sessionMap;

@Override
public void prepare() throws Exception {
viewBlocks();

ElectionDao eldao = new ElectionDao();
elections = eldao.getElectionsOfAnEvent(
event.getEvent_id());
}

@Override
public String execute() {
String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap) && sessionMap.
get("role").equals("VOTING_OFFICIAL")){
if(mode != null){
if(mode.equals("addPage"))
ret = "addPage";
else if(mode.equals("add")){
ret = addBlock();
}
else if(mode.equals("editPage")){
ret = editPage();
}
else if(mode.equals("edit")){
ret = editBlock();
}
else if(mode.equals("delete")){
deleteBlock();
ret = "deleteSuccess";
}

viewBlocks();
}
else
ret = "viewBlocks";
}

return ret;
}

public void viewBlocks() {
event = (Event) sessionMap.get("event");
setBlocks(bdao.getBlocksOfAnEvent(event.
getEvent_id()));
}

public String addBlock(){
if(validatedFirst(0)){
HashSet<Election> electionSet = new
HashSet<Election>();
Election e;

for(int i=0; i<myElections.size(); i++){
e = new Election(myElections.get(i));
electionSet.add(e);
}
```

```
block.setElections(electionSet);
bdao.addBlock(block);
block = null;

return "addSuccess";
}
return "input";
}

public boolean validatedFirst(int flag) {
hasError[flag] = false;

if(block.getName().length() == 0){
this.addFieldError("block.name",
"Field required");
hasError[flag] = true;
}
if(block.getDescription().length() == 0){
this.addFieldError("block.description",
"Field required");
hasError[flag] = true;
}
if(myElections != null){
if(myElections.size() == 0)
this.addFieldError("blockh",
"Need to choose elections");
}
else{
this.addFieldError("blockh",
"Field Required");
hasError[flag] = true;
}

return !hasError[flag];
}

public String editBlock() {
if(validatedFirst(1)){
HashSet<Election> electionSet =
new HashSet<Election>();
Election e;

for(int i=0; i<myElections.size(); i++){
e = new Election(myElections.get(i));
electionSet.add(e);
}

block.setElections(electionSet);
bdao.updateBlock(block);
block = null;

return "editSuccess";
}
return "input2";
}

public String editPage(){
ElectionDao eldao = new ElectionDao();
List<Election> tempEl = eldao.
getVoterElections(block.getBlock_id());
savedElections = new ArrayList<Integer>();
for(Election el : tempEl)
savedElections.add(el.getElection_id());

return "editPage";
}

public void deleteBlock(){
bdao.deleteBlock(block);
}

@Override
public void setSession(Map<String, Object>
sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

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

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

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

public List<Block> getBlocks() {
return blocks;
}
public void setBlocks(List<Block> blocks) {
this.blocks = blocks;
}

public Block getBlock() {
return block;
}
public void setBlock(Block block) {
this.block = block;
}

public List<Integer> getMyElections() {
return myElections;
}
public void setMyElections(List<Integer>
myElections) {
this.myElections = myElections;
}

public List<Event> getEvents() {
return events;
}

public void setEvents(List<Event> events) {
this.events = events;
}

public Event getEvent() {
return event;
}

public void setEvent(Event event) {
this.event = event;
}

public boolean[] getHasError() {
return hasError;
}

public void setHasError(boolean[] hasError) {
this.hasError = hasError;
}

public List<Integer> getSavedElections() {
return savedElections;
}

public void setSavedElections(List<Integer>
savedElections) {
this.savedElections = savedElections;
}


}
```

BulletinBoardAction.java

```
package com.sp.vote.action;

import com.sp.vote.method.Hasher;
import com.sp.vote.model.Event;
import com.sp.vote.model.Vote;
import com.sp.vote.model.Voter;
import com.sp.vote.dao.EventDao;
import com.sp.vote.dao.PositionDao;
import com.sp.vote.dao.VoteDao;
import com.sp.vote.dao.VoterDao;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;
import org.apache.struts2.interceptor.
        SessionAware;

import java.util.List;
import java.util.Map;


public class BulletinBoardAction extends
        ActionSupport implements SessionAware,
        Preparable {
```

```
private static final long serialVersionUID = 1L;

private Event event;
private Voter voter;
private List<Event> events;
private List<Voter> voters;
private List<Vote> votes;
private int uid;
private String mode;
private String role;
private Map<String, Object> sessionMap;

@Override
public void prepare(){
if(sessionMap.get("event") != null)
event = (Event) sessionMap.get("event");
if(sessionMap.get("role") != null)
role = (String) sessionMap.get("role");
if(sessionMap.get("user_id") != null)
setUid((Integer) sessionMap.get("user_id"));
}

public String execute(){
String ret = "sorry";

if(mode != null){
if(mode.equals("viewVote")){
viewVote();
ret = "viewVoteSuccess";
}
}
else{
ret = publishBoard();
}


return ret;
}

public String publishBoard() {
EventDao evdao = new EventDao();
Event e = evdao.getEvent(event.getEvent_id());
event.setName(e.getName());

if(e.getStatus().equals("FINISHED")){
VoterDao vrdao = new VoterDao();
voters = vrdao.getVotersOfAnEvent(
event.getEvent_id());
event.setTemp(e.getName() + "_" +
e.getEvent_id() + "_board.pdf");
return "publishBoardSuccess";
}
else
return "publishBoardFailure";
}

public void viewVote() {
EventDao evdao = new EventDao();
Event ev = evdao.getEvent(event.getEvent_id());
event.setName(ev.getName());

Hasher h = new Hasher();
PositionDao podao = new PositionDao();
VoteDao vodao = new VoteDao();
setVotes(vodao.getVotes(voter.getAlias()));
for(int i=0; i<votes.size(); i++){
try {
if(podao.getPosition(votes.get(i).
        getPosition_id()).getRequired_max()==1)
        votes.get(i).setEvote(h.
                computeHash(votes.get(i).
                getEvote(), "MD5"));
else
        votes.get(i).setEvote(votes.get(i).
                getXhash());

} catch (Exception e) {
e.printStackTrace();
}
votes.get(i).setTemp(podao.
getPosition(votes.get(i).getPosition_id()).
getName());
}
}

public Event getEvent() {
return event;
}
public void setEvent(Event event) {
this.event = event;
}
public List<Event> getEvents() {
return events;
}
public void setEvents(List<Event> events) {
this.events = events;
}

@Override
public void setSession(Map<String, Object>
        sessionMap) {
this.sessionMap = sessionMap;
}

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

public List<Voter> getVoters() {
return voters;
}
public void setVoters(List<Voter> voters) {
this.voters = voters;
}

public List<Vote> getVotes() {
return votes;
}
public void setVotes(List<Vote> votes) {
this.votes = votes;
}

public Voter getVoter() {
return voter;
}
public void setVoter(Voter voter) {
this.voter = voter;
}

public String getRole() {
return role;
}

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

public int getUid() {
return uid;
}

public void setUid(int uid) {
this.uid = uid;
}


}
```

CandidateAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.CandidateDao;
import com.sp.vote.dao.ElectionDao;
import com.sp.vote.dao.PartyDao;
import com.sp.vote.dao.PositionDao;
import com.sp.vote.model.Candidate;
import com.sp.vote.model.Election;
import com.sp.vote.model.Event;
import com.sp.vote.model.Party;
import com.sp.vote.model.Position;

import org.apache.struts2.interceptor.
        SessionAware;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;

import java.util.Collections;
import java.util.List;
import java.util.Map;


public class CandidateAction extends ActionSupport
        implements SessionAware, Preparable {

private static final long serialVersionUID = 1L;

private CandidateDao cdao = new CandidateDao();
```

```
private Candidate candidate;
private Event event;
private Election election;
private Position position;
private List<Candidate> candidates;
private List<Election> elections;
private List<Position> positions;
private List<Party> parties;

private String mode;
private boolean hasError[] = new boolean[2];
private Map<String, Object> sessionMap;

@Override
public void prepare() throws Exception {
setEvent((Event) sessionMap.get("event"));
PartyDao padao = new PartyDao();
setParties(padao.getPartiesOfAnEvent(event.
        getEvent_id()));
Collections.sort(parties, Party.partyNameComp);
}

@Override
public String execute() {
String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap) && sessionMap.
        get("role").equals("VOTING_OFFICIAL")){

if(mode != null){
if(mode.equals("addPage"))
ret = "addPage";
else if(mode.equals("add")){
ret = addCandidate();
viewCandidates();
}
else if(mode.equals("editPage"))
ret = "editPage";
else if(mode.equals("edit")){
ret = editCandidate();
viewCandidates();
}
else if(mode.equals("delete")){
deleteCandidate();
viewCandidates();
ret = "deleteSuccess";
}
else if(mode.equals("cand")){
viewCandidates();
ret = "viewCandidates";
}

}
else
ret = "viewCandidates";

}

return ret;
}


public void viewCandidates() {
PositionDao podao = new PositionDao();
position = podao.getPosition(position.
        getPosition_id());
ElectionDao eldao = new ElectionDao();
election = eldao.getElection(position.
        getElection_id());
PartyDao padao = new PartyDao();
setCandidates(cdao.getCandidatesByPosition(
        position.getPosition_id()));

for(int i=0; i<candidates.size(); i++)
candidates.get(i).setTemp(padao.getParty(
        candidates.get(i).getParty_id()).getName());

Collections.sort(candidates,
        Candidate.candidateNameComp);
}

public String addCandidate(){
if(validatedFirst(0)){
cdao.addCandidate(candidate);
candidate = null;

return "addSuccess";
}
return "input";
}

public boolean validatedFirst(int flag) {
hasError[flag] = false;

if(candidate.getFirst_name().length() == 0){
this.addFieldError("candidate.first_name",
        "Field required");
hasError[flag] = true;
}
if(candidate.getLast_name().length() == 0){
this.addFieldError("candidate.last_name",
        "Field required");
hasError[flag] = true;
}

return !hasError[flag];
}

public String editCandidate() {
if(validatedFirst(1)){
cdao.editCandidate(candidate);
candidate = null;

return "editSuccess";
}
return "input";
}

public void deleteCandidate() {
cdao.deleteCandidate(candidate);
}

@Override
public void setSession(Map<String,
        Object> sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

public Candidate getCandidate() {
return candidate;
}
public void setCandidate(Candidate candidate) {
this.candidate = candidate;
}

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

public List<Candidate> getCandidates() {
return candidates;
}
public void setCandidates(List<Candidate>
        candidates) {
this.candidates = candidates;
}

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

public List<Position> getPositions() {
return positions;
}
public void setPositions(List<Position>
        positions) {
this.positions = positions;
}

public List<Party> getParties() {
return parties;
}
public void setParties(List<Party> parties) {
this.parties = parties;
}

public Event getEvent() {
return event;
```

```
}
public void setEvent(Event event) {
this.event = event;
}

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

public boolean[] getHasError() {
return hasError;
}

public void setHasError(boolean hasError[]) {
this.hasError = hasError;
}
}
```

DownloadFileAction.java

```
package com.sp.vote.action;

import org.apache.struts2.ServletActionContext;
import com.opensymphony.xwork2.ActionSupport;

import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import javax.servlet.ServletContext;


public class DownloadFileAction extends
        ActionSupport {

private static final long serialVersionUID = 1L;

private InputStream fileInputStream;
private String filename;

public InputStream getFileInputStream() {
return fileInputStream;
}

public String execute() throws Exception {
ServletContext context = ServletActionContext.
        getServletContext();
String path = context.getRealPath("/WEB-INF/files/");

File tempfile = new File(path + "/" + filename);
filename = tempfile.getName();
fileInputStream = new FileInputStream(tempfile);
return SUCCESS;
}

public String getFilename() {
return filename;
}
public void setFilename(String filename) {
this.filename = filename;
}

}
```

ElectionAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.ElectionDao;
import com.sp.vote.dao.EventDao;
import com.sp.vote.model.Election;
import com.sp.vote.model.Event;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;
import org.apache.struts2.interceptor.
        SessionAware;

import java.util.Collections;
import java.util.List;
import java.util.Map;


public class ElectionAction extends ActionSupport
        implements SessionAware, Preparable {

private static final long serialVersionUID = 1L;

private ElectionDao eldao = new ElectionDao();

private Election election;
private List<Election> elections;
private Event event;
private List<Event> events;

private String mode;
private int uid;
private Map<String, Object> sessionMap;

private boolean errorDel = true;
private boolean hasError[] = new boolean[2];


@Override
public void prepare() throws Exception {
uid = (Integer)sessionMap.get("user_id");

EventDao evdao = new EventDao();
events = evdao.getEvents(uid);
}

@Override
public String execute() throws Exception{
event = (Event) sessionMap.get("event");

String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap) && sessionMap.
        get("role").equals("VOTING_OFFICIAL")){
if(mode != null){
if(mode.equals("addPage"))
ret = "addPage";
else if(mode.equals("add")){
ret = addElection();
}
else if(mode.equals("editPage")){
ret = "editPage";
}
else if(mode.equals("edit")){
ret = editElection();
}
else if(mode.equals("delete")){
deleteElection();
ret = "deleteSuccess";
}

}
else
ret = "viewElections";
viewElections();

}

return ret;
}

public void viewElections() {
setElections(eldao.getElectionsOfAnEvent(
        event.getEvent_id()));
Collections.sort(elections);
}

public String addElection() throws Exception{
if(validatedFirst(0)){
election.setEvent_id(event.getEvent_id());
election.setPriority(election.getPriority()-1);
election.setPk_n("");
election.setSk_lambda("");

eldao.addElection(election);
election = null;

return "addSuccess";
}
return "input";
}

public boolean validatedFirst(int flag) {
viewElections();

hasError[flag] = false;
boolean hasMajor = false;
if(election.getPriority() == 1){
for(int i=0; i<elections.size(); i++){
if(elections.get(i).getPriority() == 0){
if(flag == 1){
if(elections.get(i).getElection_id() ==
        election.getElection_id())
        break;
}
hasMajor = true;
break;
```

```
}
}
}

if(election.getName().length() == 0){
this.addFieldError("ename",
        "This field is required");
hasError[flag] = true;
}
if(election.getPriority() == 0){
this.addFieldError("epriority",
        "This field is required");
hasError[flag] = true;
}
if(hasMajor){
this.addFieldError("epriority",
        "There is already an existing
major election");
hasError[flag] = true;
}

return !hasError[flag];
}

public String editElection() {
if(validatedFirst(1)){
election.setPriority(election.getPriority()-1);
eldao.updateElection(election);
election = null;
return "editSuccess";
}
return "input";
}

public void deleteElection(){
setErrorDel(eldao.deleteElection(election));
}

@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

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

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

public List<Event> getEvents() {
return events;
}
public void setEvents(List<Event> events) {
this.events = events;
}

public Event getEvent() {
return event;
}
public void setEvent(Event event) {
this.event = event;
}

public boolean isErrorDel() {
return errorDel;
}
public void setErrorDel(boolean errorDel) {
this.errorDel = errorDel;
}

public boolean[] getHasError() {
return hasError;
}
public void setHasError(boolean hasError[]) {
this.hasError = hasError;
}

}
```

EventAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.BlockDao;
import com.sp.vote.dao.CandidateDao;
import com.sp.vote.dao.ElectionDao;
import com.sp.vote.dao.EventDao;
import com.sp.vote.dao.PositionDao;
import com.sp.vote.dao.VoterDao;
import com.sp.vote.model.Election;
import com.sp.vote.model.Event;

import com.opensymphony.xwork2.
        ActionSupport;
import org.apache.struts2.
        ServletActionContext;
import org.apache.struts2.interceptor.
SessionAware;

import thep.paillier.PrivateKey;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.math.BigInteger;
import java.util.List;
import java.util.Map;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;
import javax.servlet.ServletContext;


public class EventAction extends ActionSupport
implements SessionAware {

private static final long serialVersionUID = 1L;

private EventDao evdao = new EventDao();

private Event event, eventForm;
private List<Event> events;
private List<Election> elections;

private int[] checker = new int[6];
private String[] sk_filenames;

private InputStream fileInputStream;
private String filename;

private int uid;
private String mode;
private boolean addr = false;
private boolean hasError[] = new boolean[2];
private Map<String, Object> sessionMap;


@Override
public String execute() throws Exception{
String ret = "sorry";

uid = (Integer)sessionMap.get("user_id");
SessionAction sa = new SessionAction();

if(sa.sessionExists(sessionMap) && sessionMap.
get("role").equals("VOTING_OFFICIAL")){

if(mode != null){
if(mode.equals("addPage"))
ret = "addPage";
else if(mode.equals("add")){
ret = addEvent();
}
else if(mode.equals("gteh")){
ret = goToEvent();
}
else if(mode.equals("goToEvent")){
viewEvents();
ret = goToEvent();
}
else if(mode.equals("editPage"))
ret = "editPage";
else if(mode.equals("edit")){
```

```
ret = editEvent();
}
else if(mode.equals("delete")){
deleteEvent();
ret = "deleteSuccess";
}
else if(mode.equals("startE")){
startEvent();
ret = "startSuccess";
}
else if(mode.equals("stopE")){
stopEvent();
ret = "stopSuccess";
}
else if(mode.equals("archivE")){
archiveEvent();
ret = "archiveSuccess";
}

viewEvents();
}
else
ret = "viewEvents";
}

return ret;
}

public void archiveEvent() {
Event e = evdao.getEvent(event.getEvent_id());
evdao.archiveEvent(e);
}

public String goToEvent(){
int eid = event.getEvent_id();
Event e = evdao.getEvent(eid);

if(e.getBuilder_id() == uid){
sessionMap.put("event", e);
setEvent((Event)sessionMap.get("event"));

ElectionDao eldao = new ElectionDao();
BlockDao bdao = new BlockDao();
PositionDao pdao = new PositionDao();
CandidateDao cdao = new CandidateDao();
VoterDao vdao = new VoterDao();

if(eldao.getElectionsOfAnEvent(eid).size()
> 0) checker[0] = 1;
if(bdao.getBlocksOfAnEvent(eid).size()
> 0) checker[1] = 1;
if(pdao.getPositionsOfAnEvent(eid).size()
> 0) checker[2] = 1;
if(cdao.getCandidatesOfAnEvent(eid).size()
> 0) checker[3] = 1;
if(vdao.getVotersOfAnEvent(eid).size()
> 0) checker[4] = 1;

checker[5] = 0;
for(int i=0; i<5; i++)
checker[5] += checker[i];

return "goToEvent";
}
else
return "sorry";

}

public void viewEvents() {
setEvent((Event)sessionMap.get("event"));
setEvents(evdao.getEvents(uid));
}

public String addEvent(){
if(validatedFirst(0)){
EventDao evdao = new EventDao();
eventForm.setBuilder_id(uid);
eventForm.setStatus("IDLE");

evdao.addEvent(eventForm);
eventForm = null;
setAddr(true);

return "addSuccess";
}
return "input";
}

public String editEvent(){
if(validatedFirst(1)){
EventDao evdao = new EventDao();
evdao.updateEvent(event);

return goToEvent();
}
return "input2";

}

public void deleteEvent(){
EventDao evdao = new EventDao();
evdao.deleteEvent(event);
}

public void startEvent() throws Exception{
setEvent((Event)sessionMap.get("event"));
evdao.start_stopEvent(event, "RUNNING");
createSecretKeys();
}

public void stopEvent() throws Exception{
setEvent((Event)sessionMap.get("event"));
evdao.start_stopEvent(event, "FINISHED");

PdfGenerationAction2 pga = new
        PdfGenerationAction2();
String pdfname = event.getName()+"_"+
event.getEvent_id()+"_board.pdf";
pga.generatePdf(pdfname, event);

goToEvent();
}

public void createSecretKeys() throws
        Exception {
ElectionDao eldao = new ElectionDao();
elections = eldao.getElectionsOfAnEvent(
event.getEvent_id());

ServletContext context = ServletActionContext.
getServletContext();
String path = context.getRealPath(
"/WEB-INF/files/");

String secretKey = "";
PrivateKey sk;

sk_filenames = new String[elections.size()];
for(int i=0; i<elections.size(); i++){
sk_filenames[i] =  elections.get(i).
getName() + "_"  + elections.get(i).
getElection_id() + "_sk.key";

sk = new PrivateKey(1024);

elections.get(i).setPk_n(new BigInteger(
sk.getPublicKey().getN().toString()).
toString(16));
elections.get(i).setSk_lambda(new BigInteger(
sk.getLambda().toString()).toString(16));

secretKey = new BigInteger(sk.getMu().toString()).
toString(16);

eldao.updateElectionKeys(elections.get(i));

File file = new File(path + "/" + sk_filenames[i]);
file.createNewFile();
FileWriter fw = new FileWriter(file);
BufferedWriter bw = new BufferedWriter(fw);
bw.write(secretKey);
bw.close();
}
autoDownload(sk_filenames, event.getEvent_id(),
event.getName());
}

public String viewPublicEvents(){
events = evdao.getAllFinishedEvents();

return SUCCESS;
}

public String viewPublicElections(){
ElectionDao eldao = new ElectionDao();
elections = eldao.getElectionsOfAnEvent(event.
getEvent_id());

for(int i=0; i<elections.size(); i++)
elections.get(i).setTemp(elections.get(i).getName()
+ "_" + elections.get(i).getElection_id()
+ "_results.pdf");
```

```
return SUCCESS;
}

public boolean validatedFirst(int flag) {
hasError[flag] = false;

if(flag == 0){
if(eventForm.getName().length() == 0){
this.addFieldError("eventForm.name",
"This field is required");
hasError[flag] = true;
}
if(eventForm.getDescription().length() == 0){
this.addFieldError("eventForm.description",
"This field is required");
hasError[flag] = true;
}
}
else{
if(event.getName().length() == 0){
this.addFieldError("event.name",
"This field is required");
hasError[flag] = true;
}
if(event.getDescription().length() == 0){
this.addFieldError("event.description",
"This field is required");
hasError[flag] = true;
}
}

return !hasError[flag];
}

public void autoDownload(String[] fnames,
int eid, String ename) throws Exception {
ServletContext context = ServletActionContext.
getServletContext();
String path = context.getRealPath("/WEB-INF
        /files/");

byte[] buffer = new byte[1024];

try{

FileOutputStream fos = new FileOutputStream(
path + "/" +  ename + "-" + eid + ".zip");

ZipOutputStream zos = new ZipOutputStream(fos);

for(int i=0; i<fnames.length; i++){
ZipEntry ze= new ZipEntry(fnames[i]);
zos.putNextEntry(ze);
FileInputStream in = new FileInputStream(path +
"/" + fnames[i]);

int len;
while ((len = in.read(buffer)) > 0) {
zos.write(buffer, 0, len);
}

in.close();

}

zos.closeEntry();
zos.close();

File tempfile = new File(path + "/" + ename + "-"
+ eid + ".zip"); // ORDINARY
setFilename(tempfile.getName());
fileInputStream = new FileInputStream(tempfile);

}catch(IOException ex){
ex.printStackTrace();
}
}

@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public Event getEvent(){
return event;
}
public void setEvent(Event event) {
this.event = event;
}

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

public List<Event> getEvents() {
return events;
}
public void setEvents(List<Event> events) {
this.events = events;
}

public Event getEventForm() {
return eventForm;
}
public void setEventForm(Event eventForm) {
this.eventForm = eventForm;
}

public int[] getChecker() {
return checker;
}
public void setChecker(int[] checker) {
this.checker = checker;
}

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

public String[] getSk_filenames() {
return sk_filenames;
}
public void setSk_filenames(String[]
sk_filenames) {
this.sk_filenames = sk_filenames;
}

public boolean isAddr() {
return addr;
}

public void setAddr(boolean addr) {
this.addr = addr;
}

public boolean[] getHasError() {
return hasError;
}
public void setHasError(boolean hasError[]) {
this.hasError = hasError;
}

public String getFilename() {
return filename;
}
public void setFilename(String filename) {
this.filename = filename;
}

public InputStream getFileInputStream() {
return fileInputStream;
}


}
```

HomeAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.EventDao;
import com.sp.vote.model.Event;

import org.apache.struts2.interceptor.
        SessionAware;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;

import java.util.List;
import java.util.Map;


public class HomeAction extends ActionSupport
        implements SessionAware, Preparable {
```

```
private static final long serialVersionUID = 1L;

private EventDao evdao = new EventDao();
private List<Event> events;
private Event event;

private Map<String, Object> sessionMap;
private String role;
private int uid;

@Override
public void prepare(){
uid = (Integer)sessionMap.get("user_id");
role = (String)sessionMap.get("role");
if(role.equals("VOTING_OFFICIAL"))
viewEvents();

}

@Override
public String execute(){
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap)){
sessionMap.remove("event");
return SUCCESS;
}
else
return ERROR;
}

public void viewEvents() {
setEvents(evdao.getEvents(uid));
}

@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public String getRole(){
return role;
}
public void setRole(String role) {
this.role = role;
}

public List<Event> getEvents() {
return events;
}
public void setEvents(List<Event> events){
this.events = events;
}

public Event getEvent() {
return event;
}
public void setEvent(Event event) {
this.event = event;
}
}
```

LoginAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.ElectionDao;
import com.sp.vote.dao.EventDao;
import com.sp.vote.dao.UserDao;
import com.sp.vote.dao.VoterDao;
import com.sp.vote.model.Event;
import com.sp.vote.model.User;
import com.sp.vote.method.BCrypt;

import org.apache.struts2.interceptor.
        SessionAware;
import com.opensymphony.xwork2.ActionSupport;

import java.util.List;
import java.util.Map;


public class LoginAction extends ActionSupport
        implements SessionAware {

private static final long serialVersionUID = 1L;

private User user;

private EventDao evdao = new EventDao();
private List<Event> events;
private Event event;

private Map<String, Object> sessionMap;

@Override
public void validate() {
if (user.getUsername().length() == 0)
this.addFieldError("user.username",
        "Name is required");
if (user.getPassword().length() == 0)
this.addFieldError("user.password",
        "Password is required");
}

@Override
public String execute() throws Exception {

String ret = "input";
UserDao udao = new UserDao();

String userName = user.getUsername();

if(udao.getUser(userName) != null){
if(udao.getUser(userName).getUsername().
        equals(userName)){
String salt = udao.getUser(user.getUsername())
        .getSalt();
String hashedPass = BCrypt.hashpw(user.getPassword(),
        salt);

if(udao.findUser(user.getUsername(), hashedPass)){
String role = udao.getUser(user.getUsername()).
        getRole();
sessionMap.put("role", role);
sessionMap.put("username", user.getUsername());
sessionMap.put("user_id", udao.getUser(user.
        getUsername()).getUser_id());

if(role.equals("VOTING_OFFICIAL")){
setEvents(evdao.getEvents(udao.getUser(user.
getUsername()).getUser_id()));
ret = "official_user";
}
else if(role.equals("VOTER")){
VoterDao vrdao = new VoterDao();
int block_id = vrdao.getVoter((Integer)
        sessionMap.get("user_id")).getBlock_id();
ElectionDao eldao = new ElectionDao();
setEvent(evdao.getEventOfAnElection(eldao.
        getVoterElections(block_id).get(0)));
sessionMap.put("event", event);
ret = "voter_user";
}
else if(role.equals("ADMIN"))
ret = "admin_user";

}
}
}

if(ret.equals("input"))
this.addActionError("Invalid username
        and password");

return ret;
}

@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

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

public Event getEvent(){
return event;
}
public void setEvent(Event event) {
this.event = event;
}

public List<Event> getEvents() {
return events;
}
public void setEvents(List<Event> events) {
```

```
this.events = events;
}

}
```

LogoutAction.java

```
package com.sp.vote.action;

import org.apache.struts2.interceptor.
        SessionAware;
import com.opensymphony.xwork2.ActionSupport;

import java.util.Map;


public class LogoutAction extends ActionSupport
        implements SessionAware {

private static final long serialVersionUID = 1L;

private Map<String, Object> sessionMap;

@Override
public String execute(){
sessionMap.clear();
return "success";
}

public Map<String, Object> getSessionMap() {
return sessionMap;
}
@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

}
```

PartyAction.java

```
package com.sp.vote.action;

import com.sp.vote.model.Event;
import com.sp.vote.model.Party;
import com.sp.vote.dao.EventDao;
import com.sp.vote.dao.PartyDao
;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;

import org.apache.struts2.interceptor.
‘SessionAware;

import java.util.Collections;
import java.util.List;
import java.util.Map;


public class PartyAction extends ActionSupport
        implements SessionAware, Preparable{

private static final long serialVersionUID = 1L;

private PartyDao padao = new PartyDao();

private Party party;
private List<Party> parties;
private Event event;
private List<Event> events;

private int uid;
private String mode;
private boolean hasError[] = new boolean[2];
private Map<String, Object> sessionMap;

@Override
public void prepare() throws Exception {
uid = (Integer)sessionMap.get("user_id");

viewParties(uid);
EventDao evdao = new EventDao();
events = evdao.getEvents(uid);

}

@Override
public String execute() {
String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap) && sessionMap.
        get("role").equals("VOTING_OFFICIAL")){
if(mode != null){
if(mode.equals("addPage"))
ret = "addPage";
else if(mode.equals("add")){
addParty(uid);
ret = "addSuccess";
}
else if(mode.equals("editPage"))
ret = "editPage";
else if(mode.equals("edit")){
ret = editParty();
}
else if(mode.equals("delete")){
deleteParty();
ret = "deleteSuccess";
}

viewParties(uid);
}
else
ret = "viewParties";

}

return ret;
}

public void viewParties(int uid) {
event = (Event) sessionMap.get("event");
setParties(padao.getPartiesOfAnEvent(event.
        getEvent_id()));
Collections.sort(parties, Party.partyNameComp);
}

public String addParty(int uid) {
if(validatedFirst(0)){
party.setEvent_id(event.getEvent_id());
padao.addParty(party);
party = null;

return "addSuccess";
}
else
return "input";
}

public boolean validatedFirst(int flag) {
hasError[flag] = false;

if(party.getName().length() == 0){
this.addFieldError("party.name",
        "Field required");
hasError[flag] = true;
}

return !hasError[flag];
}

public String editParty() {
if(validatedFirst(1)){
padao.updateParty(party);
party = null;

return "editSuccess";
}
return "input";
}

public void deleteParty() {
padao.deleteParty(party);
}

@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

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

```
}

public Event getEvent() {
return event;
}
public void setEvent(Event event) {
this.event = event;
}

public List<Event> getEvents() {
return events;
}
public void setEvents(List<Event> events) {
this.events = events;
}

public List<Party> getParties() {
return parties;
}
public void setParties(List<Party> parties) {
this.parties = parties;
}

public boolean[] getHasError() {
return hasError;
}
public void setHasError(boolean hasError[]) {
this.hasError = hasError;
}

}
```

PdfGenerationAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.CandidateDao;
import com.sp.vote.dao.PositionDao;
import com.sp.vote.model.Candidate;
import com.sp.vote.model.Election;
import com.sp.vote.model.Position;

import org.apache.struts2.ServletAction
Context;

import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import javax.servlet.ServletContext;

import com.itextpdf.text.Document;
import com.itextpdf.text.DocumentException;
import com.itextpdf.text.Font;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.PdfWriter;


public class PdfGenerationAction {
private InputStream fileStream;
private String FILE;
private static Font catFont = new Font(Font.
FontFamily.TIMES_ROMAN, 18, Font.BOLD);
private static Font norFont = new Font(Font.
FontFamily.TIMES_ROMAN, 12, Font.NORMAL);
private static Font subFont = new Font(Font.
FontFamily.TIMES_ROMAN, 16, Font.BOLD);
private static Font smallBold = new Font(Font.
FontFamily.TIMES_ROMAN, 12, Font.BOLD);

public void setFileStream(InputStream arg){
fileStream = arg;
}

public InputStream getFileStream() {
return fileStream;
}

public String generatePdf(String fname,
Election election) throws Exception {

try {
ServletContext context = ServletAction
        Context.getServletContext();
String path = context.getRealPath(
"/WEB-INF/files/");

FILE = path + "/" + fname;

Document document = new Document();
PdfWriter.getInstance(document, new
FileOutputStream(FILE));
document.open();
addMetaData(document);
addTitlePage(document, election);

document.close();
} catch (Exception e) {
e.printStackTrace();
}

try{
fileStream = new DataInputStream(new
FileInputStream(FILE));
} catch (IOException ioEx) {
ioEx.printStackTrace();
}

return "test-stream";
}

private static void addMetaData(Document
document) {
document.addTitle("Voterify");
document.addSubject("Elections");
}

private static void addTitlePage(Document
document, Election election) throws
DocumentException {
Paragraph preface = new Paragraph();
addEmptyLine(preface, 1);
preface.add(new Paragraph("Voterify",
catFont));
addEmptyLine(preface, 1);
preface.add(new Paragraph(election.getName(),
subFont));
addEmptyLine(preface, 1);

PositionDao podao = new PositionDao();
List<Position> positions = podao.
getPositionsOfAnElection(election.
getElection_id());
Collections.sort(positions);
CandidateDao cdao = new CandidateDao();
List<List<Candidate>> candidates = new
ArrayList<List<Candidate>>();
int cancount = 0;
for(int i=0; i<positions.size(); i++){
candidates.add(new ArrayList<Candidate>());
candidates.set(i, cdao.getCandidatesBy
Position(positions.get(i).getPosition_id
()));
Collections.sort(candidates.get(i),
Candidate.candidateNameComp);

cancount = cdao.getCandidatesByPosition(
positions.get(i).getPosition_id()).size();
if(positions.get(i).getRequired_min()
== 0)
cancount++;

preface.add(new Paragraph(positions.get(i).
getName(), smallBold));
addEmptyLine(preface, 1);
for(int j=0; j<cancount; j++){
if(j == cancount - 1){
if(positions.get(i).getRequired_min() == 0){
preface.add(new Paragraph("ABSTAIN: "
+ positions.get(i).getAbstain_count(),
`norFont));
addEmptyLine(preface, 1);
break;
}
}
preface.add(new Paragraph(candidates.get(i).
get(j).getLast_name().toUpperCase()
  +", "+candidates.get(i).get(j).
  getFirst_name() +": "+candidates.get(i).
  get(j).getVote_count(), norFont));
addEmptyLine(preface, 1);
}
}

document.add(preface);

}

private static void addEmptyLine(
Paragraph paragraph, int number) {
for (int i = 0; i < number; i++) {
```

```
paragraph.add(new Paragraph(" "));
}
}
}
```

PdfGenerationAction2.java

```
package com.sp.vote.action;

import com.sp.vote.dao.PositionDao;
import com.sp.vote.dao.VoteDao;
import com.sp.vote.dao.VoterDao;
import com.sp.vote.method.Hasher;
import com.sp.vote.model.Event;
import com.sp.vote.model.Vote;
import com.sp.vote.model.Voter;

import org.apache.struts2.ServletAction
Context;

import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import javax.servlet.ServletContext;

import com.itextpdf.text.Document;
import com.itextpdf.text.Document
        Exception;
import com.itextpdf.text.Font;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.PdfWriter;


public class PdfGenerationAction2 {
private InputStream fileStream;
private String FILE;
private static Font catFont = new Font(Font.
FontFamily.TIMES_ROMAN, 18, Font.BOLD);
private static Font norFont = new Font(Font.
FontFamily.TIMES_ROMAN, 12, Font.NORMAL);
private static Font subFont = new Font(Font.
FontFamily.TIMES_ROMAN, 16, Font.BOLD);

public void setFileStream(InputStream arg) {
fileStream = arg;
}

public InputStream getFileStream() {
return fileStream;
}


public String generatePdf(String fname,
Event event) throws Exception {

try {
ServletContext context = ServletAction
Context.getServletContext();
String path = context.getRealPath(
"/WEB-INF/files/");

FILE = path + "/" + fname;
Document document = new Document();
PdfWriter.getInstance(document, new
FileOutputStream(FILE));
document.open();
addMetaData(document);
addTitlePage(document, event);

document.close();
} catch (Exception e) {
e.printStackTrace();
}

try{
fileStream = new DataInputStream(
new FileInputStream(FILE));
} catch (IOException ioEx) {
ioEx.printStackTrace();
}

return "test-stream";
}

private static void addMetaData(Document
document) {
document.addTitle("Voterify");
document.addSubject("Bulletin
Board of Votes");
}

private static void addTitlePage(Document
document,
Event event) throws DocumentException {
Paragraph preface = new Paragraph();

preface.add(new Paragraph("Voterify -
Bulletin Board of Votes", catFont));
addEmptyLine(preface, 1);

preface.add(new Paragraph(event.getName(),
subFont));
addEmptyLine(preface, 1);

VoterDao vodao = new VoterDao();
List<Voter> voters = vodao.getVotersOfAn
Event(event.getEvent_id());

Hasher h = new Hasher();
PositionDao podao = new PositionDao();
VoteDao vdao = new VoteDao();
List<Vote> votes;

String x = "", y = "";

for(int i=0; i<voters.size(); i++){
preface.add(new Paragraph(voters.get(i).
getAlias()+" | "+voters.get(i).getCode(),
subFont));

votes = vdao.getVotes(voters.get(i).
getAlias());
for(int j=0; j<votes.size(); j++){
try {
x = podao.getPosition(votes.get(j).
getPosition_id()).getName();
if(podao.getPosition(votes.get(j).
getPosition_id()).getRequired_max()==1)
y = h.computeHash(votes.get(j).
getEvote(), "MD5");
else
y = votes.get(j).getXhash();
preface.add(new Paragraph(x + " : " + y,
norFont));

} catch (Exception e) {
e.printStackTrace();
}
}
addEmptyLine(preface, 1);

}

addEmptyLine(preface, 1);

document.add(preface);
}

private static void addEmptyLine(Paragraph
paragraph, int number) {
for (int i = 0; i < number; i++) {
paragraph.add(new Paragraph(" "));
}
}

}
```

PositionAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.BlockDao;
import com.sp.vote.dao.ElectionDao;
import com.sp.vote.dao.PositionDao;
import com.sp.vote.model.Block;
import com.sp.vote.model.Election;
import com.sp.vote.model.Event;
import com.sp.vote.model.Position;

import org.apache.commons.lang3.math.
        NumberUtils;
import org.apache.struts2.interceptor.
        SessionAware;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;

import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.util.StringTokenizer;
```

```
public class PositionAction extends ActionSupport
        implements SessionAware, Preparable {

private static final long serialVersionUID = 1L;

private PositionDao podao = new PositionDao();

private Position position;
private Election election;
private Event event;
private Block block;
private List<Position> positions;
private List<Election> elections;
private List<Block> blocks;

private int eid;
private String str_min, str_max;
private String mode;
private String newPositions;
private boolean hasError[] = new boolean[2];
private Map<String, Object> sessionMap;

@Override
public void prepare() throws Exception {
event = (Event) sessionMap.get("event");

ElectionDao eldao = new ElectionDao();
elections = eldao.getElectionsOfAnEvent(event.
        getEvent_id());

BlockDao bdao = new BlockDao();
blocks = bdao.getBlocksOfAnEvent(event.
        getEvent_id());

}

@Override
public String execute(){
String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap) && sessionMap.
        get("role").equals("VOTING_OFFICIAL")){
if(mode != null){
if(mode.equals("add")){
        ret = addPosition();
}
else if(mode.equals("editPage")){
        ret = "editPage";
}
else if(mode.equals("edit")){
        ret = editPosition();
}
else if(mode.equals("delete")){
        deletePosition();
        ret = "deleteSuccess";
}
else if(mode.equals("vpe"))
        ret = "viewPositions";
else if(mode.equals("dad"))
        ret = "dragPositions";

viewPositions();
}
else
ret = "positionHome";


}

return ret;
}

public void viewPositions(){
setPositions(podao.getPositionsOfAnElection
        (election.getElection_id()));
Collections.sort(positions);
BlockDao bdao = new BlockDao();
blocks = bdao.getBlocksOfAnElection(election.
        getElection_id());
}

public String addPosition(){
if(validatedFirst(0)){
setPositions(podao.getPositionsOfAnElection(
election.getElection_id()));

position.setBallot_order(positions.size() + 1);
position.setRequired_min(Integer.parseInt(
        str_min));
position.setRequired_max(Integer.parseInt(
        str_max));
podao.addPosition(position);
position = null;
str_min = str_max = null;

return "addSuccess";
}
else
return "input";
}

public boolean validatedFirst(int flag) {
hasError[flag] = false;
setPositions(podao.getPositionsOfAnElection(
election.getElection_id()));

for(int i=0; i<positions.size(); i++){
if(positions.get(i).getBallot_order() ==
position.getBallot_order()){
if(flag == 1){
if(positions.get(i).getPosition_id()
  == position.getPosition_id())
break;
}
break;
}
}

if(!NumberUtils.isNumber(str_min)){
this.addFieldError("str_min",
        "Must be a number");
hasError[flag] = true;
}
else{
if(Integer.parseInt(str_min) < 0){
this.addFieldError("str_min",
"Must be greater than or equal to 0");
hasError[flag] = true;
}
}

if(!NumberUtils.isNumber(str_max)){
this.addFieldError("str_max", "Must be a
        number");
hasError[flag] = true;
}
else{
if(Integer.parseInt(str_max) <= 0){
this.addFieldError("str_max", "Must be greater
than  or equal to 0");
hasError[flag] = true;
}
}

if(position.getName().length() == 0){
this.addFieldError("position.name",
        "Field required");
hasError[flag] = true;
}

return !hasError[flag];
}

public String editPosition() {
if(validatedFirst(1)){
setPositions(podao.getPositionsOfAnElection
(election.getElection_id()));

position.setRequired_min(Integer.parseInt(
        str_min));
position.setRequired_max(Integer.parseInt(
        str_max));
podao.updatePosition(position);
str_min = str_max = null;
position = null;

return "editSuccess";
}
return "input";
}

public void deletePosition(){
podao.deletePosition(position);
}

public String reorderPos(){
String[] pArr = new String[10];
StringTokenizer st = new StringTokenizer(
newPositions, ",");
int c = 0;
while(st.hasMoreTokens()){
pArr[c] = st.nextToken();
c++;
```

```
}

List<Position> p = podao.getPositionsOfAn
Election(eid);
Collections.sort(p);
for(int i=0; i<p.size(); i++){
p.get(i).setBallot_order(Integer.parseInt(
        pArr[i]));
podao.updatePositionOrder(p.get(i));
}

return "reorderSuccess";
}

public List<Election> getElections() {
return elections;
}
public void setElections(List<Election>
        elections){
this.elections = elections;
}
public Election getElection() {
return election;
}
public void setElection(Election election) {
this.election = election;
}
public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}
@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

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

public List<Position> getPositions() {
return positions;
}
public void setPositions(List<Position>
positions){
this.positions = positions;
}

public List<Block> getBlocks() {
return blocks;
}
public void setBlocks(List<Block> blocks) {
this.blocks = blocks;
}

public Block getBlock() {
return block;
}
public void setBlock(Block block) {
this.block = block;
}

public Event getEvent() {
return event;
}
public void setEvent(Event event) {
this.event = event;
}

public boolean[] getHasError() {
return hasError;
}

public void setHasError(boolean hasError[]){
this.hasError = hasError;
}

public String getStr_min() {
return str_min;
}

public void setStr_min(String str_min) {
this.str_min = str_min;
}

public String getStr_max() {
return str_max;
}

public void setStr_max(String str_max) {
this.str_max = str_max;
}

public String getNewPositions() {
return newPositions;
}

public void setNewPositions(String
newPositions) {
this.newPositions = newPositions;
}

public int getEid() {
return eid;
}
public void setEid(int eid) {
this.eid = eid;
}
}
```

ResultsAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.ElectionDao;
import com.sp.vote.model.Election;
import com.sp.vote.model.Event;

import org.apache.struts2.interceptor.
        SessionAware;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;

import java.util.List;
import java.util.Map;


public class ResultsAction extends ActionSupport
        implements SessionAware, Preparable {

private static final long serialVersionUID = 1L;

private ElectionDao eldao = new ElectionDao();

private List<Election> elections;
private Election election;
private Event event;

private String secretKey;


private String mode;
private String errorMsg;
private Map<String, Object> sessionMap;

@Override
public void prepare() throws Exception {
viewElections();
}


public String execute() throws Exception{
String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap) && sessionMap.
        get("role").equals("VOTING_OFFICIAL")){

if(mode != null){
if(mode.equals("computePage")){
ret = "computePage";
}
}
else
ret = "display";

}

return ret;

}

public void viewElections() {
event = (Event) sessionMap.get("event");
setElections(eldao.getElectionsOfAnEvent(
event.getEvent_id()));
}
```

```
@Override
public void setSession(Map<String, Object>
sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public String getSessionUser(){
String test = (String) sessionMap.get("username");
return test;
}

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

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

public String getSecretKey() {
return secretKey;
}
public void setSecretKey(String secretKey){
this.secretKey = secretKey;
}

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

public String getErrorMsg() {
return errorMsg;
}
public void setErrorMsg(String errorMsg) {
this.errorMsg = errorMsg;
}

public Event getEvent() {
return event;
}
public void setEvent(Event event) {
this.event = event;
}
}
```

SessionAction.java

```
package com.sp.vote.action;

import java.util.Map;


public class SessionAction {

public boolean sessionExists(Map<String,
        Object> sessionMap){
boolean b = true;

String username = (String) sessionMap.
        get("username");
if(username == null || username == "")
b = false;

return b;
}

public static int getUID(Map<String,
        Object> sessionMap){
return (Integer) sessionMap.get("user_id");
}

}
```

SignupAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.UserDao;
import com.sp.vote.model.User;
import com.sp.vote.method.BCrypt;
import org.apache.commons.validator.routines.EmailValidator;
import com.opensymphony.xwork2.ActionSupport;

public class SignupAction extends ActionSupport {

private static final long serialVersionUID = 1L;

private User user;
private String password, password2;

@Override
public void validate() {
if (user.getUsername().length() == 0)
this.addFieldError("user.username",
        "Name is required");
if (password.length() == 0)
this.addFieldError("passwordE",
        "Field is required");
if (password2.length() == 0)
this.addFieldError("password2E",
        "Field is required");
if(!EmailValidator.getInstance().
        isValid(user.getEmail()) || user.getEmail().
                length() == 0)
this.addFieldError("user.email",
        "Valid email is required");

UserDao udao = new UserDao();
if(udao.getAllUsernames().contains(user.
        getUsername()))
this.addFieldError("user.username",
        "Username already exists");
if(!password.equals(password2))
this.addFieldError("password2E",
        "Passwords do not match");
}

public String execute() {
User u = new User();
u.setUsername(user.getUsername());
u.setSalt(BCrypt.gensalt(12));
u.setPassword(BCrypt.hashpw(password,
        u.getSalt()));
u.setEmail(user.getEmail());
u.setRole("VOTING_OFFICIAL");

UserDao udao = new UserDao();
udao.addUser(u);

user = null;
return SUCCESS;
}

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

public String getPassword() {
return password;
}
public void setPassword(String password) {
this.password = password;
}

public String getPassword2() {
return password2;
}
public void setPassword2(String password2) {
this.password2 = password2;
}

}
```

TicketAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.TicketDao;
import com.sp.vote.dao.VoterDao;
import com.sp.vote.model.Event;
import com.sp.vote.model.Ticket;

import org.apache.struts2.interceptor.
        SessionAware;
import com.opensymphony.xwork2.ActionSupport;

import java.util.List;
import java.util.Map;
public class TicketAction extends ActionSupport
```

```
        implements SessionAware {

private static final long serialVersionUID = 1L;
private Ticket ticket;
private Event event;

private List<Ticket> tickets;

private int uid;
private String mode;
private Map<String, Object> sessionMap;

public String execute(){
String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap)){
if(sessionMap.get("role").equals("VOTER")){
if(mode != null){
if(mode.equals("raise")){
raiseTicket();
ret = "raiseSuccess";
}
}
else{
ret = "NONE";
}
}
else if(sessionMap.get("role").equals
        ("VOTING_OFFICIAL")){
if(mode != null){
if(mode.equals("tvo")){
displayTickets();
ret = "displaySuccess";
}
}
}

}

return ret;
}

public void displayTickets() {
event = (Event) sessionMap.get("event");
TicketDao tdao = new TicketDao();
tickets = tdao.getTicketsOfAnEvent(event.
        getEvent_id());
}

public void raiseTicket() {
TicketDao tdao = new TicketDao();

VoterDao vrdao = new VoterDao();
ticket.setVoter_id(vrdao.getVoter(uid).
        getVoter_id());
tdao.addTicket(ticket);
}

@Override
public void setSession(Map<String, Object>
        sessionMap) {
this.sessionMap = sessionMap;
}

public Ticket getTicket() {
return ticket;
}
public void setTicket(Ticket ticket) {
this.ticket = ticket;
}

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

public int getUid() {
return uid;
}

public void setUid(int uid) {
this.uid = uid;
}

public Event getEvent() {
return event;
}

public void setEvent(Event event) {
this.event = event;
}

public List<Ticket> getTickets() {
return tickets;
}

public void setTickets(List<Ticket> tickets) {
this.tickets = tickets;
}

}
```

UploadKeyAction.java

```
package com.sp.vote.action;

import com.sp.vote.method.VoteCounter;
import com.sp.vote.model.Candidate;
import com.sp.vote.model.Election;
import com.sp.vote.model.Position;
import com.sp.vote.model.Vote;
import com.sp.vote.model.Voter;
import com.sp.vote.dao.*;

import com.opensymphony.xwork2.ActionSupport;
import org.apache.commons.io.FileUtils;
import org.apache.struts2.ServletActionContext;

import thep.paillier.EncryptedInteger;
import thep.paillier.PrivateKey;
import thep.paillier.PublicKey;
import thep.paillier.exceptions.PublicKeysNotEqual
        Exception;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import javax.servlet.ServletContext;


public class UploadKeyAction extends
        ActionSupport{

private static final long serialVersionUID = 1L;

private Election election;
private Voter voter;
private List<Vote> votes;
private List<Position> positions;
private List<List<Candidate>> candidates;

private File myFile;
private String myFileContentType;
private String myFileFileName;
private String destPath;

private String ret;
private String pdfname;

public String execute(){
ServletContext context = ServletActionContext.
        getServletContext();
destPath = context.getRealPath("/WEB-INF/
files/");

try{
File destFile  = new File(destPath,
// myFileFileName);
FileUtils.copyFile(myFile, destFile);

readCSVFile();

}catch(IOException e){
e.printStackTrace();
return ERROR;
} catch (Exception e) {
e.printStackTrace();
}

return SUCCESS;
}

public void readCSVFile() throws Exception {
String csvFile = destPath + "/" + myFileFileName;
BufferedReader br = null;
String line = "", line2 = "";
```

```
try {
br = new BufferedReader(new FileReader(csvFile));
while ((line = br.readLine()) != null) {
line2 = line;
break;
}
} catch (FileNotFoundException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
} catch (Exception e) {
e.printStackTrace();
} finally {
if (br != null) {
try {
br.close();
} catch (IOException e) {
e.printStackTrace();
}
}
}

computeTally(line2);
}

public void computeTally(String secretKey)
        throws Exception{
setRet("computeFailure");

ElectionDao eldao = new ElectionDao();
election = eldao.getElection(
        election.getElection_id());
int candidateCount = 0;

PositionDao podao = new PositionDao();
CandidateDao cdao = new CandidateDao();
VoteDao vodao = new VoteDao();

VoteCounter vot = new VoteCounter();
try {
PublicKey pk = new PublicKey(1024, new Big
        Integer(election.getPk_n(),16));
PrivateKey sk = new PrivateKey(1024);
sk.setLambda(new BigInteger(election.
        getSk_lambda(),16));
sk.setMu(new BigInteger(secretKey, 16));
sk.setPublicKey(pk);

if(testDecryption(sk, pk)){
int[] voteCountArr;
candidates = new ArrayList<List<Candidate>>();

positions = podao.getPositionsOfAnElection(
        election.getElection_id());
Collections.sort(positions);
for(int i=0; i<positions.size(); i++){
candidates.add(new ArrayList<Candidate>());
candidates.set(i, cdao.getCandidatesByPosition(
        positions.get(i).getPosition_id()));
Collections.sort(candidates.get(i),
        Candidate.candidateNameComp);

votes = vodao.getVotes(positions.get(i).
getPosition_id());
candidateCount = cdao.getCandidatesByPosition(
positions.get(i).getPosition_id()).size();
if(positions.get(i).getRequired_min() == 0)
candidateCount++;

voteCountArr = new int[candidateCount];
voteCountArr = vot.countVotes(votes,
candidateCount, sk);

for(int j=0; j<candidateCount; j++){
if(j == candidateCount - 1){
if(positions.get(i).getRequired_min() == 0){
positions.get(i).setAbstain_count(
voteCountArr[j]);
podao.updateAbstain_count(positions.get(i),
voteCountArr[j]);
break;
}
}
candidates.get(i).get(j).setVote_count(
voteCountArr[j]);
cdao.updateCandidateCount(candidates.get(i).
get(j));
}
Thread.sleep(5000);
}
eldao.updateElectionTallyStat(election);

PdfGenerationAction pga = new
        PdfGenerationAction();
pdfname = election.getName()+"_"+election.
getElection_id()+"_results.pdf";
pga.generatePdf(pdfname, election);

setRet("computeSuccess");
}
else
setRet("computeFailure");
} catch (PublicKeysNotEqualException e) {
e.printStackTrace();
setRet("computeFailure");
}

}

public boolean testDecryption(PrivateKey sk,
PublicKey pk) throws Exception{
// Test 1
EncryptedInteger ei1 = new EncryptedInteger(new
BigInteger("10000"), pk);
if(!ei1.decrypt(sk).toString().equals("10000"))
return false;

// Test 1
EncryptedInteger ei2 = new EncryptedInteger(new
BigInteger("1"), pk);
if(!ei2.decrypt(sk).toString().equals("1"))
return false;

return true;

}

public File getMyFile() {
return myFile;
}
public void setMyFile(File myFile) {
this.myFile = myFile;
}

public String getMyFileContentType() {
return myFileContentType;
}
public void setMyFileContentType(String
myFileContentType) {
this.myFileContentType = myFileContentType;
}

public String getMyFileFileName() {
return myFileFileName;
}
public void setMyFileFileName(String
myFileFileName) {
this.myFileFileName = myFileFileName;
}

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

public Voter getVoter() {
return voter;
}
public void setVoter(Voter voter) {
this.voter = voter;
}

public List<Position> getPositions() {
return positions;
}
public void setPositions(List<Position>
positions) {
this.positions = positions;
}

public List<List<Candidate>> getCandidates() {
return candidates;
}
public void setCandidates(List<List<Candidate>>
candidates) {
this.candidates = candidates;
}

public String getRet() {
return ret;
```

```
}

public void setRet(String ret) {
this.ret = ret;
}

public String getPdfname() {
return pdfname;
}

public void setPdfname(String pdfname) {
this.pdfname = pdfname;
}

}
```

UploadVotersAction.java

```
package com.sp.vote.action;

import com.sp.vote.method.BCrypt;
import com.sp.vote.method.EmailSender;
import com.sp.vote.method.Hasher;
import com.sp.vote.method.MailFileReader;
import com.sp.vote.model.Block;
import com.sp.vote.model.User;
import com.sp.vote.model.Voter;
import com.sp.vote.dao.*;

import org.apache.commons.io.FileUtils;
import org.apache.struts2.
        ServletActionContext;
import org.apache.struts2.interceptor.
        SessionAware;
import com.opensymphony.xwork2.ActionSupport;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletContext;


public class UploadVotersAction extends
   ActionSupport implements SessionAware{

private static final long serialVersionUID = 1L;

private File myFile;
private String myFileContentType;
private String myFileFileName;
private String destPath;

private Voter voter;
private Block block;

private List<Block> blocks;
private List<String> duplicateUnames;
private boolean hasBulkError[] = new boolean[2];


public String execute(){
ServletContext context = ServletActionContext.
        getServletContext();
destPath = context.getRealPath("/WEB-INF/files/");

block.setBlock_id(voter.getBlock_id());

hasBulkError[0] = hasBulkError[1] = false;
if(myFile != null){
try{
File destFile  = new File(destPath, myFileFileName);
FileUtils.copyFile(myFile, destFile);

String mailString = MailFileReader.readMailFile(
        destPath, "mail_v.txt");
readCSVFile(mailString);

}catch(IOException ex1){
ex1.printStackTrace();
return ERROR;
} catch (Exception ex2) {
// TODO Auto-generated catch block
ex2.printStackTrace();
hasBulkError[1] = true;
}
}
else
hasBulkError[0] = true;

if(hasBulkError[0] == false && hasBulkError[1]
        == false)
return SUCCESS;
else{
if(hasBulkError[0] == true)
this.addFieldError("bulkF", "Choose a file");
if(hasBulkError[1] == true)
this.addFieldError("bulkF", "Invalid file");

return "input";
}

}

public void readCSVFile(String mailString) {
String csvFile = destPath + "/" +
        myFileFileName;
BufferedReader br = null;
String line = "";

UserDao udao = new UserDao();
VoterDao vrdao = new VoterDao();
User u = new User();
Voter vr = new Voter();
Hasher h = new Hasher();
EmailSender es = new EmailSender();


List<String> usernames = udao.getAllUsernames();
duplicateUnames = new ArrayList<String>();

String s1 = "", s2 = "";

try {
br = new BufferedReader(new FileReader(csvFile));
while ((line = br.readLine()) != null) {
String[] token = line.split(",");

if(usernames.contains(token[0])){
duplicateUnames.add(token[0]+","+token[1]);
}
else{
u.setUsername(token[0]);
s1 = h.generateStr2();
s2 = BCrypt.gensalt(12);
u.setPassword(BCrypt.hashpw(s1,s2));
u.setSalt(s2);
es.sendEmail("Voterify", mailString, token[1]);
u.setRole("VOTER");
u.setEmail(token[1]);
udao.addUser(u);

vr.setUser_id(udao.getUser(u.getUsername()).
        getUser_id());
vr.setBlock_id(voter.getBlock_id());
vr.setAlias("V" +vr.getUser_id());
vr.setCode("");
vrdao.addVoter(vr);

usernames.add(token[0]);
}

}
} catch (FileNotFoundException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
} catch (Exception e) {
e.printStackTrace();
} finally {
if (br != null) {
try {
br.close();
} catch (IOException e) {
e.printStackTrace();
}
}
}
}

public File getMyFile() {
return myFile;
}
public void setMyFile(File myFile) {
this.myFile = myFile;
}

public String getMyFileContentType() {
return myFileContentType;
}
```

```
public void setMyFileContentType(String
        myFileContentType) {
this.myFileContentType = myFileContentType;
}

public String getMyFileFileName() {
return myFileFileName;
}
public void setMyFileFileName(String
        myFileFileName) {
this.myFileFileName = myFileFileName;
}

public Voter getVoter() {
return voter;
}
public void setVoter(Voter voter) {
this.voter = voter;
}

public List<String> getDuplicateUnames() {
return duplicateUnames;
}
public void setDuplicateUnames(List<String>
        duplicateUnames) {
this.duplicateUnames = duplicateUnames;
}

public boolean[] getHasBulkError() {
return hasBulkError;
}

public void setHasBulkError(boolean
        hasBulkError[]) {
this.hasBulkError = hasBulkError;
}

public Block getBlock() {
return block;
}

public void setBlock(Block block) {
this.block = block;
}

public List<Block> getBlocks() {
return blocks;
}

public void setBlocks(List<Block> blocks) {
this.blocks = blocks;
}

@Override
public void setSession(Map<String, Object> arg0){
// TODO Auto-generated method stub

}
}
```

VoteAction.java

```
package com.sp.vote.action;

import com.sp.vote.method.CurrentTimeStamp;
import com.sp.vote.dao.EventDao;
import com.sp.vote.dao.VoteDao;
import com.sp.vote.dao.VoterDao;
import com.sp.vote.method.Hasher;
import com.sp.vote.model.*;

import com.opensymphony.xwork2.
        ActionSupport;
import com.opensymphony.xwork2.Preparable;
import org.apache.struts2.
        ServletActionContext;
import org.apache.struts2.interceptor.
        SessionAware;

import thep.paillier.*;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import javax.servlet.ServletContext;

public class VoteAction extends ActionSupport
        implements SessionAware, Preparable{

private static final long serialVersionUID = 1L;

private List<Election> elections;
private List<List<Position>> positions;
private List<List<List<Candidate>>> candidates;
private List<List<List<String>>> evotes;
private List<List<List<String>>> hashes;
private String[][][] parties;
private int blockID;

private VoterDao vrdao = new VoterDao();

private String[][] singlePos = new String
        [10][20];
private String[][][] multiplePos = new String
        [10][20][30];
private String[][] evote;
private String[] love;
private String verificationCode;
private String filename;
private Map<String, Object> sessionMap;

private Event event;
private int uid;

private PublicKey pk;
private EncryptedInteger ei, temp;

@SuppressWarnings("unchecked")
@Override
public void prepare(){
elections = (List<Election>)
        sessionMap.get("officialElections");
positions = (List<List<Position>>)
        sessionMap.get("officialPositions");
candidates = (List<List<List<Candidate>>>)
        sessionMap.get("officialCandidates");
evotes = (List<List<List<String>>>)
        sessionMap.get("officialEvotes");
hashes = (List<List<List<String>>>)
        sessionMap.get("officialHashes");
blockID = (Integer)
        sessionMap.get("officialBlockID");

filler();
}


public boolean validateVotes(){
Position p = null;
boolean valid = true;

try{

for(int i=0; i<elections.size(); i++){
for(int j=0; j<positions.get(i).size(); j++){
p = positions.get(i).get(j);
if(p.getRes_block_id() == 1 || p.
        getRes_block_id() == blockID){
if(p.getRequired_max()==1){
if(singlePos[i][j] != null){
// hi!
}
else{
valid = false;
}
}
else{
if(multiplePos[i][j].length == 0){
valid = false;
}
else if(multiplePos[i][j].length > p.
        getRequired_max()){
valid = false;
}

}
}
}
}

if(valid == false)
this.addFieldError("checker",
        "Please follow instructions");

EventDao evdao = new EventDao();
setEvent(evdao.getEventOfAnElection
        (elections.get(0)));
```

```
}catch(Exception e){
e.printStackTrace();
}

return valid;

}

@Override
public String execute() throws Exception {

CurrentTimeStamp ts = new Curren
        tTimeStamp();
Hasher h = new Hasher();

VoteDao vdao = new VoteDao();
Vote v = new Vote();
uid = (Integer)sessionMap.get("user_id");
v.setVoter_id(vrdao.getVoter(uid).
        getVoter_id());

int cid;

if(validateVotes()){
for(int i=0; i<elections.size(); i++){
pk = new PublicKey(1024, new BigInteger(
        elections.get(i).getPk_n(),16));
ei = new EncryptedInteger(pk);

for(int j=0; j<positions.get(i).
        size(); j++){
temp = new EncryptedInteger(pk);
evote[i][j] = "";
int ktemp = 0;
for(int k=0; k<candidates.get(i).get(j).
        size(); k++){
cid = candidates.get(i).get(j).get(k).
        getCandidate_id();
if(positions.get(i).get(j).
        getRequired_max() == 1){
if(singlePos[i][j].equals(String.
        valueOf(cid))){
evote[i][j] = hashes.get(i).get(j).get(k);
verificationCode += evote[i][j];
singlePos[i][j] = candidates.get(i).get(j).
        get(k).getLast_name() + " " + candidates.
        get(i).get(j).get(k).getFirst_name();

v.setPosition_id(positions.get(i).get(j).
        getPosition_id());
v.setEvote(evotes.get(i).get(j).get(k));
v.setTimestamp(ts.getCurrentDateTime());
if(vrdao.hasVoted(uid))
        vdao.updateVote(v);
else
        vdao.addVote(v);

break;
}
}
else{
if(Arrays.asList(multiplePos[i][j]).contains
        (String.valueOf(cid))){
multiplePos[i][j][ktemp] = candidates.get(i).get(j).
get(k).getLast_name() + " " + candidates.get(i).
get(j).get(k).getFirst_name();
ei.setCipherVal(new BigInteger(evotes.get(i).
get(j).get(k)));
temp = temp.add(ei);
evote[i][j] += evotes.get(i).get(j).get(k);
ktemp++;
}
if(k == candidates.get(i).get(j).size() - 1){
evote[i][j] = h.computeHash(evote[i][j], "MD5");
verificationCode += evote[i][j];

v.setPosition_id(positions.get(i).get(j).
        getPosition_id());getPosition_id());
v.setEvote(temp.getCipherVal().toString());
v.setXhash(evote[i][j]);
v.setTimestamp(ts.getCurrentDateTime());
if(vrdao.hasVoted(uid))
        vdao.updateVote(v);
else
        vdao.addVote(v);
}
}
}

}
}
```

```
try {
verificationCode = h.computeHash(
        verificationCode, "MD5");
vrdao.markVoted(v.getVoter_id(),
        verificationCode);
} catch (Exception e1) {
// TODO Auto-generated catch block
e1.printStackTrace();
}

printReceipt();

return SUCCESS;
}
return "input";

}

public void printReceipt() throws IOException{
String fileContent = "";

for(int i=0; i<elections.size(); i++){
fileContent += elections.get(i).
        getName().toUpperCase() + "\n";
for(int j=0; j<positions.get(i).size();
        j++){
fileContent += positions.get(i).
        get(j).getName() +": ";
fileContent += evote[i][j] + "\n";
}
}
fileContent += "\nVerification Code: "
        +verificationCode;
CurrentTimeStamp ts = new CurrentTimeStamp();
fileContent += "\n\nTimestamp: "
        +ts.getCurrentDateTime();

ServletContext context = ServletActionContext.
        getServletContext();
String path = context.getRealPath("/WEB-INF/
        files/");

filename = "hash_" + vrdao.getVoter(uid).
        getVoter_id()+ ".txt";
File file = new File(path + "/" + filename);
file.createNewFile();
FileWriter fw = new FileWriter(file);
BufferedWriter bw = new BufferedWriter(fw);
bw.write(fileContent);
bw.close();
}

public String display() {
return SUCCESS;
}

public void filler(){
evote = new String[elections.size()][10];
}

@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public String[][] getSinglePos() {
return singlePos;
}
public void setSinglePos(String[][] singlePos) {
this.singlePos = singlePos;
}

public String[][][] getMultiplePos() {
return multiplePos;
}
public void setMultiplePos(String[][][]
        multiplePos) {
this.multiplePos = multiplePos;
}

public String[][] getEvote(){
return evote;
}
public void setEvote(String[][] evote){
this.evote = evote;
}

public String getVerificationCode(){
return verificationCode;
}
```

```
public void setVerificationCode(String
        verificationCode){
this.verificationCode = verificationCode;
}


public String[] getLove() {
return love;
}
public void setLove(String[] love) {
this.love = love;
}

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

public List<List<Position>> getPositions() {
return positions;
}
public void setPositions(List<List<Position>>
        positions) {
this.positions = positions;
}

public List<List<List<Candidate>>> getCandidates() {
return candidates;
}
public void setCandidates(List<List<
        List<Candidate>>> candidates) {
this.candidates = candidates;
}

public List<List<List<String>>> getEvotes() {
return evotes;
}
public void setEvotes(List<List<List<String>>>
        evotes) {
this.evotes = evotes;
}

public List<List<List<String>>> getHashes() {
return hashes;
}
public void setHashes(List<List<List<String>>>
        hashes) {
this.hashes = hashes;
}

public int getBlockID() {
return blockID;
}
public void setBlockID(int blockID) {
this.blockID = blockID;
}

public String[][][] getParties() {
return parties;
}
public void setParties(String[][][] parties) {
this.parties = parties;
}

public Event getEvent() {
return event;
}
public void setEvent(Event event) {
this.event = event;
}

public String getFilename() {
return filename;
}
public void setFilename(String filename) {
this.filename = filename;
}

}
```

VoterAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.BlockDao;
import com.sp.vote.dao.UserDao;
import com.sp.vote.dao.VoterDao;
import com.sp.vote.method.BCrypt;
import com.sp.vote.method.EmailSender;
import com.sp.vote.method.Hasher;
import com.sp.vote.method.MailFileReader;
import com.sp.vote.model.Block;
import com.sp.vote.model.Event;
import com.sp.vote.model.User;
import com.sp.vote.model.Voter;

import com.opensymphony.xwork2.
        ActionSupport;
import com.opensymphony.xwork2.Preparable;
import org.apache.commons.validator.routines.
        EmailValidator;
import org.apache.struts2.
        ServletActionContext;
import org.apache.struts2.interceptor.
        SessionAware;

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

import javax.servlet.ServletContext;


public class VoterAction extends ActionSupport
        implements SessionAware, Preparable{

private static final long serialVersionUID = 1L;
private VoterDao vrdao = new VoterDao();
private UserDao udao = new UserDao();

private List<Voter> voters;
private List<Block> blocks;
private List<User> voterUsers;

private Event event;
private Voter voter;
private Block block;
private User user;
private String election;

private String mode;
private boolean hasError[] = new boolean[2];
private Map<String, Object> sessionMap;

@Override
public void prepare() throws Exception {
setEvent((Event) sessionMap.get("event"));

BlockDao bdao = new BlockDao();
blocks = bdao.getBlocksOfAnEvent(event.
        getEvent_id());

}

@Override
public String execute() throws IOException {
String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap) &&
        sessionMap.get("role").equals("VOTING_OFFICIAL")){

if(mode != null){
if(mode.equals("importPage"))
ret = "importPage";
else if(mode.equals("addPage"))
ret = "addPage";
else if(mode.equals("add")){
ret = addVoter();
}
else if(mode.equals("editPage")){
user = udao.getUser_Voter(voter);
ret = "editPage";
}
else if(mode.equals("edit")){
ret = editVoter();
}
else if(mode.equals("delete")){
deleteVoter();
ret = "deleteSuccess";
}
else if(mode.equals("vpe")){
ret = "viewVoters";
}

viewVoters();
}
else
ret = "viewVotersHome";
}
```

```
return ret;
}

public void viewVoters() {
setVoters(vrdao.getVotersByBlock(
        block.getBlock_id()));
setVoterUsers(vrdao.getVoterUsers(
        block.getBlock_id()));
for(int i=0; i<voterUsers.size(); i++)
voterUsers.get(i).setTemp2(vrdao.
        getVoter(voterUsers.get(i).
        getUser_id()).getVoter_id());

}

public String addVoter() throws
IOException {
if(validatedFirst(0)){
Hasher h = new Hasher();
String s1 = h.generateStr2();
String s2 = BCrypt.gensalt(12);
user.setPassword(BCrypt.hashpw(s1, s2));
user.setSalt(s2);
user.setRole("VOTER");
udao.addUser(user);

voter.setUser_id(udao.getUser(user.
        getUsername()).getUser_id());
voter.setAlias("V" + voter.getUser_id());
voter.setCode("");
vrdao.addVoter(voter);

ServletContext context = ServletActionContext.
        getServletContext();
String destPath = context.getRealPath("/WEB-INF/
        files/");
String mailString = MailFileReader.
        readMailFile(destPath, "mail_v.txt");
EmailSender es = new EmailSender();
es.sendEmail("Voterify", mailString,
        user.getEmail());
user = null;
voter = null;

return "addSuccess";
}
return "input";
}

public boolean validatedFirst(int flag) {
hasError[flag] = false;

List<String> usernames = udao.
        getAllUsernames();
if(flag == 1){
if(udao.getUser(user.getUsername()) != null){
int temp_id = udao.getUser(user.getUsername()).
        getUser_id();
if(temp_id == user.getUser_id())
usernames.remove(user.getUsername());
}
}

if(usernames.contains(user.getUsername())){
this.addFieldError("user.username",
        "Username already taken");
hasError[flag] = true;
}

if(!EmailValidator.getInstance().
        isValid(user.getEmail())){
this.addFieldError("user.email",
"Invalid email");
hasError[flag] = true;
}


return !hasError[flag];
}

public String editVoter() {
if(validatedFirst(1)){
udao.editUser(user);

user = null;

return "editSuccess";
}
return "input";
}

public void deleteVoter() {
vrdao.deleteVoter(voter);
}

@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public String getElection() {
return election;
}
public void setElection(String election) {
this.election = election;
}

public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

public List<Block> getBlocks() {
return blocks;
}

public void setBlocks(List<Block> blocks) {
this.blocks = blocks;
}

public Voter getVoter() {
return voter;
}

public void setVoter(Voter voter) {
this.voter = voter;
}

public List<Voter> getVoters() {
return voters;
}
public void setVoters(List<Voter> voters) {
this.voters = voters;
}

public User getUser() {
return user;
}

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

public Event getEvent() {
return event;
}
public void setEvent(Event event) {
this.event = event;
}

public List<User> getVoterUsers() {
return voterUsers;
}
public void setVoterUsers(List<User>
        voterUsers) {
this.voterUsers = voterUsers;
}

public Block getBlock() {
return block;
}
public void setBlock(Block block) {
this.block = block;
}

public boolean[] getHasError() {
return hasError;
}
public void setHasError(boolean hasError[]) {
this.hasError = hasError;
}

}
```

VotingOfficialAction.java

```
package com.sp.vote.action;

import com.sp.vote.dao.UserDao;
import com.sp.vote.method.BCrypt;
```

```
import com.sp.vote.method.EmailSender;
import com.sp.vote.method.Hasher;
import com.sp.vote.method.MailFileReader;
import com.sp.vote.model.User;

import com.opensymphony.xwork2.
        ActionSupport;
import com.opensymphony.xwork2.Preparable;
import org.apache.struts2.
        ServletActionContext;
import org.apache.struts2.interceptor.
        SessionAware;
import org.apache.commons.validator.
        routines.EmailValidator;


import java.util.List;
import java.util.Map;

import javax.servlet.ServletContext;


public class VotingOfficialAction extends
        ActionSupport implements SessionAware,
        Preparable {

private static final long serialVersionUID = 1L;

private UserDao udao = new UserDao();
private List<User> voting_officials;
private User voting_official;

private String mode;
private boolean hasError[] = new boolean[2];
private Map<String, Object> sessionMap;

@Override
public void prepare(){
viewVotingOfficials();
}


public boolean myValidate(int flag){
hasError[flag] = false;

List<String> usernames = udao.getAllUsernames();
if(flag == 1){
if(udao.getUser(voting_official.getUsername())
        != null){
int temp_id = udao.getUser(voting_official.
        getUsername()).getUser_id();
if(temp_id == voting_official.getUser_id())
usernames.remove(voting_official.getUsername());
}
}

if(usernames.contains(voting_official.
        getUsername())){
this.addFieldError("userF",
        "Username already taken");
hasError[flag] = true;
}
if(voting_official.getUsername().length() == 0){
this.addFieldError("userF",
        "Field required");
hasError[flag] = true;
}
if(!EmailValidator.getInstance().
        isValid(voting_official.getEmail())){
this.addFieldError("emailF", "Invalid email");
hasError[flag] = true;
}

return !hasError[flag];
}

@Override
public String execute() throws Exception{
String ret = "sorry";
SessionAction sa = new SessionAction();
if(sa.sessionExists(sessionMap) &&
        sessionMap.get("role").equals("ADMIN")){

if(mode != null){
if(mode.equals("addPage"))
ret = "addPage";
else if(mode.equals("add")){
if(myValidate(0)){
addVotingOfficial();
ret = "addSuccess";
}
else{
ret = "input";
}
}
else if(mode.equals("editPage"))
ret = "editPage";
else if(mode.equals("edit")){
if(myValidate(1)){
editVotingOfficial();
ret = "editSuccess";
}
else
ret = "input";
}
else if(mode.equals("delete")){
deleteVotingOfficial();
ret = "deleteSuccess";
}

viewVotingOfficials();
}
else
ret = "viewVotingOfficials";
}

return ret;
}

//public String display(){}
public void addVotingOfficial()
        throws Exception {
Hasher h = new Hasher();
String s1 = h.generateStr2();
String s2 = BCrypt.gensalt(12);
voting_official.setPassword(BCrypt.hashpw(s1, s2));
voting_official.setSalt(s2);
voting_official.setRole("VOTING_OFFICIAL");
udao.addUser(voting_official);

ServletContext context = ServletActionContext.
        getServletContext();
String path = context.getRealPath(
        "/WEB-INF/files/");

EmailSender es = new EmailSender();
String body = MailFileReader.readMailFile(path,
        "mail_vo.txt");
body += "Username: " +voting_official.
        getUsername() + "\n";
body += "Password: " +s1+ "\n";
es.sendEmail("Voterify | User Credentials",
        body, voting_official.getEmail());

voting_official = null;
}

public void editVotingOfficial() {
udao.editUser(voting_official);
voting_official = null;
}

public void deleteVotingOfficial() {
udao.deleteUser(voting_official.getUser_id());
}

public void viewVotingOfficials() {
setVoting_officials(udao.getVotingOfficials());
}


@Override
public void setSession(Map<String, Object>
        sessionMap) {
// TODO Auto-generated method stub
this.sessionMap = sessionMap;
}

public List<User> getVoting_officials() {
return voting_officials;
}
public void setVoting_officials(List<User>
        voting_officials) {
this.voting_officials = voting_officials;
}

public User getVoting_official() {
return voting_official;
}
public void setVoting_official(User
        voting_official) {
this.voting_official = voting_official;
}
```

```
public String getMode() {
return mode;
}
public void setMode(String mode) {
this.mode = mode;
}

public boolean[] getHasError() {
return hasError;
}

public void setHasError(boolean hasError[]) {
this.hasError = hasError;
}

}
```

A..2 Data Access Objects

BlockDao.java

```
package com.sp.vote.dao;

import com.sp.vote.hibernate.
        HibernateUtil;
import com.sp.vote.model.*;

import org.hibernate.Hibernate
        Exception;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

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


public class BlockDao {

public void addBlock(Block block) {

Transaction trns = null;
Session session = HibernateUtil.
        getSessionFactory().open
        Session();
try {
trns = session.beginTransaction();
session.save(block);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}
}

@SuppressWarnings("unchecked")
public List<Block> getBlocksOfAn
        Event(int event_id) {
List<Block> blocks = new ArrayList
        <Block>();
Session session = HibernateUtil.
        getSessionFactory().open
        Session();
try {
String queryStr = "SELECT DISTINCT b
        from Block b join b.elections e ";
queryStr += "WHERE e.event_id = :eid";
Query query = session.createQuery(
        queryStr);
query.setInteger("eid", event_id);
blocks = (List<Block>) query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return blocks;
}

@SuppressWarnings("unchecked")
public List<Block> getBlocksOfAnElection
        (int election_id) {
List<Block> blocks = new ArrayList
        <Block>();
Session session = HibernateUtil.
        getSessionFactory().open
        Session();
try {
String queryStr = "SELECT DISTINCT b
from Block b join b.elections e ";
queryStr += "WHERE e.election_id = :eid";
Query query = session.createQuery
(queryStr);
query.setInteger("eid", election_id);
blocks = (List<Block>) query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return blocks;
}

public Block getBlock(int block_id) {
Block block = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from Block
where name = :bid";
Query query = session.createQuery(
queryString);
query.setInteger("bid", block_id);
block = (Block) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return block;
}

public Block getUserBlock(int user_id) {
Block block = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "Select b from
 Block b, Voter v WHERE b.block_id =
v.block_id AND v.user_id = :uid";
Query query = session.createQuery(
queryString);
query.setInteger("uid", user_id);
block = (Block) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return block;
}

public void updateBlock(Block block){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Block b = (Block)session.get(Block.class,
block.getBlock_id());
b.setName(block.getName());
b.setDescription(block.getDescription());
b.setElections(block.getElections());
session.update(b);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void deleteBlock(Block block){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Block b = (Block)session.get(
Block.class, block.getBlock_id());
session.delete(b);
tx.commit();
```

```
}catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
}finally {
session.close();
}
}
}
```

CandidateDao.java

```
package com.sp.vote.dao;

import com.sp.vote.hibernate.
        HibernateUtil;
import com.sp.vote.model.*;

import org.hibernate.Hibernate
        Exception;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

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

public class CandidateDao {

public void addCandidate(Candidate
        candidate) {

Transaction trns = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
trns = session.beginTransaction();
session.save(candidate);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}
}

public void editCandidate(Candidate
candidate){
Session session = HibernateUtil.get
SessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Candidate c = (Candidate)session.get(
Candidate.class, candidate.
getCandidate_id());
c.setFirst_name(candidate.get
First_name());
c.setLast_name(candidate.get
Last_name());
c.setParty_id(candidate.get
Party_id());

session.update(c);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void updateCandidateCount(
Candidate candidate){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Candidate c = (Candidate)session.get(
Candidate.class, candidate.
getCandidate_id());
c.setVote_count(candidate.
getVote_count());

session.update(c);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void deleteCandidate(Candidate
candidate){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Candidate c = (Candidate)session.get(
Candidate.class, candidate.
getCandidate_id());
session.delete(c);
tx.commit();
}catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
}finally {
session.close();
}
}

@SuppressWarnings("unchecked")
public List<Candidate> getCandidatesOf
AnEvent(int event_id) {
List<Candidate> candidates = new
ArrayList<Candidate>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "Select DISTINCT c
from Candidate c, Position p,
Election el, Event ev ";
queryStr += "WHERE c.position_id =
p.position_id AND p.election_id =
el.election_id AND el.event_id = :eid";
Query query = session.createQuery
(queryStr);
query.setInteger("eid", event_id);
candidates = (List<Candidate>)
query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return candidates;
}

@SuppressWarnings("unchecked")
public List<Candidate> get
CandidatesByPosition(int position_id){
List<Candidate> candidates =
 new ArrayList<Candidate>();
Session session = HibernateUtil.
getSessionFactory().openSession();

try {
String queryStr = "from Candidate
c WHERE c.position_id = :pid";
Query query = session.
createQuery(queryStr);
query.setInteger("pid", position_id);
candidates = (List<Candidate>)
query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return candidates;
}
}
```

ElectionDao.java

```
package com.sp.vote.dao;

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

import org.hibernate.Hibernate
        Exception;
import org.hibernate.Query;
import org.hibernate.Session;
```

```
import org.hibernate.Transaction;

import com.sp.vote.hibernate.
        HibernateUtil;
import com.sp.vote.model.*;


public class ElectionDao {

public void addElection(Election
election) {
Transaction trns = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
trns = session.beginTransaction();
session.save(election);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}
}

public Election getElection(int
election_id) {
Election election = null;
Session session = HibernateUtil.
sgetSessionFactory().openSession();
try {
String queryString = "from
Election e where e.election_id
= :eid";
Query query = session.
createQuery(queryString);
query.setInteger("eid", election_id);
election = (Election) query.
uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return election;
}

@SuppressWarnings("unchecked")
public List<Election> getElections
(int user_id) {
List<Election> elections = new
ArrayList<Election>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "Select el from
Election el, Event ev WHERE el.event_id =
ev.event_id AND ev.builder_id = :uid";
Query query = session.createQuery(
queryString);
query.setInteger("uid", user_id);
elections = (List<Election>)
query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return elections;
}

@SuppressWarnings("unchecked")
public List<Election> getElectionsOf
AnEvent(int event_id) {
List<Election> elections = new
ArrayList<Election>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "Select el from
Election el WHERE el.event_id = :eid";
Query query = session.createQuery(
queryString);
query.setInteger("eid", event_id);
elections = (List<Election>) query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return elections;
}

@SuppressWarnings("unchecked")
public List<Election> getVoterElections(
int block_id) {
List<Election> elections = new ArrayList
<Election>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "Select DISTINCT e
from Block b join b.elections e WHERE
b.block_id = :bid";
Query query = session.createQuery
(queryStr);
query.setInteger("bid", block_id);
elections = (List<Election>) query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return elections;
}

public void updateElection(Election
election){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Election e = (Election)session.
get(Election.class, election.
getElection_id());
e.setName(election.getName());
e.setPriority(election.getPriority());
session.update(e);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void updateElectionKeys(Election
election){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Election e = (Election)session.get(
Election.class, election.
getElection_id());
e.setPk_n(election.getPk_n());
e.setSk_lambda(election.getSk_lambda());

session.update(e);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void updateElectionTallyStat(
Election election){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Election e = (Election)session.get(
Election.class, election.getElection_id());
e.setTally_stat(1);
session.update(e);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
```

```
} finally {
session.close();
}
}

public boolean deleteElection(Election
 election){
boolean deleteSuccess = true;
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Election e = (Election)session.get(
Election.class, election.getElection_id());
session.delete(e);
tx.commit();
}catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
deleteSuccess = false;
}finally {
session.close();
}
return deleteSuccess;
}

}
```

EventDao.java

```
package com.sp.vote.dao;

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

import org.hibernate.
        HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import com.sp.vote.hibernate.
        HibernateUtil;
import com.sp.vote.method.
        CurrentTimeStamp;
import com.sp.vote.model.*;


public class EventDao {

public void addEvent(Event event) {
Transaction trns = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
trns = session.beginTransaction();
session.save(event);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}
}

public Event getEvent(int event_id) {
Event event = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from Event
where event_id = :eid";
Query query = session.
createQuery(queryString);
query.setInteger("eid", event_id);
event = (Event) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}

return event;
}

@SuppressWarnings("unchecked")
public List<Event> getEvents(int user_id) {
List<Event> events = new ArrayList<Event>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from Event
 where builder_id = :id";
Query query = session.createQuery(
queryString);
query.setInteger("id", user_id);
events = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return events;
}

@SuppressWarnings("unchecked")
public List<Event> getFinishedEvents
(int user_id) {
List<Event> events = new ArrayList<Event>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from Event where
builder_id = :id AND status = :stat";
Query query = session.createQuery(queryString);
query.setInteger("id", user_id);
query.setString("stat", "FINISHED");
events = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return events;
}

@SuppressWarnings("unchecked")
public List<Event> getAllFinishedEvents() {
List<Event> events = new ArrayList<Event>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from Event where
privacy = :p AND status = :s";
Query query = session.
createQuery(queryString);
query.setInteger("p", 0);
query.setString("s", "FINISHED");
events = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return events;
}

public Event getEventOfAnElection
(Election election) {
Event event = new Event();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "from Event
WHERE event_id = :eid";
Query query = session.createQuery(
queryStr);
query.setInteger("eid", election.
getEvent_id());
event = (Event) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return event;
}

public void updateEvent(Event event){
Session session = HibernateUtil.g
etSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
```

```
Event e = (Event)session.get(Event.class,
event.getEvent_id());
e.setName(event.getName());
e.setDescription(event.getDescription());
e.setPrivacy(event.getPrivacy());
session.update(e);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void deleteEvent(Event event){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Event e = (Event)session.get(Event.class,
event.getEvent_id());
session.delete(e);
tx.commit();
}catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
}finally {
session.close();
}
}

public void start_stopEvent(Event event,
String flag){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
CurrentTimeStamp ts = new
CurrentTimeStamp();
try{
tx = session.beginTransaction();
Event e = (Event)session.get(Event.class,
event.getEvent_id());
e.setStatus(flag);
if(flag.equals("RUNNING"))
e.setStart_date(ts.getCurrentDateTime());
else
e.setEnd_date(ts.getCurrentDateTime());
session.update(e);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void archiveEvent(Event event){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Event e = (Event)session.get(Event.class,
event.getEvent_id());
e.setStatus("ARCHIVED");

session.update(e);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

@SuppressWarnings("unchecked")
public List<Election> getVoterElections
(int block_id) {
List<Election> elections = new
ArrayList<Election>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "Select DISTINCT e
from Block b join b.elections e WHERE
b.block_id = :bid";
Query query = session.createQuery(
queryStr);
query.setInteger("bid", block_id);
elections = (List<Election>)
query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return elections;
}

}
```

PartyDao.java

```
package com.sp.vote.dao;

import com.sp.vote.hibernate.
        HibernateUtil;
import com.sp.vote.model.*;

import org.hibernate.
        HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

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


public class PartyDao {

public void addParty(Party party) {

Transaction trns = null;
Session session = HibernateUtil.
        getSessionFactory().
        openSession();
try {
trns = session.beginTransaction();
session.save(party);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}
}

public Party getParty(String
        party_name) {
Party party = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from Party
where name = :pname";
Query query = session.createQuery(
queryString);
query.setString("pname", party_name);
//query.setInteger("eid", election_id);
party = (Party) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}

return party;
}

public Party getParty(int party_id) {
Party party = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from Party
where party_id = :pid";
Query query = session.createQuery(
queryString);
query.setInteger("pid", party_id);
//query.setInteger("eid", election_id);
party = (Party) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
```

```
} finally {
session.flush();
session.close();
}

return party;
}

@SuppressWarnings("unchecked")
public List<Party> getPartiesOfAnEvent(
int event_id) {
List<Party> parties = new
ArrayList<Party>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "from Party p
WHERE p.event_id = :eid";
Query query = session.createQuery(
queryStr);
query.setInteger("eid", event_id);
parties = (List<Party>) query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return parties;
}

public void updateParty(Party party){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Party p = (Party)session.get(Party.class,
party.getParty_id());
p.setName(party.getName());
session.update(p);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void deleteParty(Party party){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Party p = (Party)session.get(Party.class,
party.getParty_id());
session.delete(p);
tx.commit();
}catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
}finally {
session.close();
}
}
}
```

PositionDao.java

```
package com.sp.vote.dao;

import com.sp.vote.hibernate.
        HibernateUtil;
import com.sp.vote.model.Position;

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

import org.hibernate.
        HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;


public class PositionDao {

public void addPosition(Position
position) {
Transaction trns = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
trns = session.beginTransaction();
session.save(position);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}
}

public Position getPosition(int
position_id) {
Position position = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from Position
where position_id = :pid";
Query query = session.createQuery(
queryString);
query.setInteger("pid", position_id);
position = (Position) query.
uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}

return position;
}

public void updatePosition(Position
position){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Position p = (Position)session.get(
Position.class,
position.getPosition_id());
p.setName(position.getName());
p.setRes_block_id(position.
getRes_block_id());
p.setRequired_min(position.
getRequired_min());
p.setRequired_max(position.
getRequired_max());
session.update(p);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void updatePositionOrder(
Position
position){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Position p = (Position)session.get(
Position.class, position.get
Position_id());
p.setBallot_order(position.
getBallot_order());

session.update(p);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}
```

```
public void updateAbstain_count(
Position position, int count){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Position p = (Position)session.get(
Position.class, position.getPosition_id());
p.setAbstain_count(count);
session.update(p);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void deletePosition(
Position position){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Position p = (Position)session.get(
Position.class, position.getPosition_id());
session.delete(p);
tx.commit();
}catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
}finally {
session.close();
}
}

@SuppressWarnings("unchecked")
public List<Position>
getPositionsOfAnEvent(int event_id) {
List<Position> positions =
new ArrayList<Position>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "SELECT p
FROM Position p, Election el ";
queryString += "WHERE p.election_id =
el.election_id AND el.event_id = :eid";
Query query = session.createQuery(
queryString);
query.setInteger("eid", event_id);
positions = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return positions;
}

@SuppressWarnings("unchecked")
public List<Position> getPositions
OfAnElection(int election_id) {
List<Position> positions = new
ArrayList<Position>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "SELECT p FROM
Position p ";
queryString += "WHERE p.election_id = :eid";
Query query = session.createQuery(queryString);
query.setInteger("eid", election_id);
positions = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return positions;
}

@SuppressWarnings("unchecked")
public List<Position> getVoterPositions
(int election_id, int block_id) {
List<Position> positions = new
ArrayList<Position>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "Select p from
Position p, Election e where p.
election_id = e.election_id AND
e.election_id = :eid"+ " AND (p.
res_block_id = 1 OR p.res_block_id=:bid)";
Query query = session.createQuery(
queryString);
query.setInteger("eid", election_id);
query.setInteger("bid", block_id);
positions = (List<Position>) query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return positions;
}
}
```

TicketDao.java

```
package com.sp.vote.dao;

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

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import com.sp.vote.hibernate.
        HibernateUtil;
import com.sp.vote.model.Ticket;

public class TicketDao {
public void addTicket(Ticket ticket){
Transaction trns = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
trns = session.beginTransaction();
session.save(ticket);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}

}

@SuppressWarnings("unchecked")
public List<Ticket> getTicketsOf
AnEvent(int event_id) {
List<Ticket> tickets = new
ArrayList<Ticket>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "Select DISTINCT t
FROM Ticket t, Voter v, Event ev,
 Block b join b.elections el WHERE
t.voter_id = v.voter_id AND v.block_id =
 b.block_id AND el.event_id = :eid";
Query query = session.createQuery(
queryStr);
query.setInteger("eid", event_id);
tickets = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return tickets;
}
}
```

UserDao.java

```
package com.sp.vote.dao;

import com.sp.vote.model.User;
import com.sp.vote.model.Voter;
```

```
import com.sp.vote.hibernate.
        HibernateUtil;

import org.hibernate.
        HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

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


public class UserDao {

public void addUser(User user) {
Transaction trns = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
trns = session.beginTransaction();
session.save(user);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}

}

public void deleteUser(int user_id) {
Transaction trns = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
trns = session.beginTransaction();
User user = (User) session.load(
User.class, new Integer(user_id));
session.delete(user);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}
}

public void editUser(User user){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
User u = (User)session.get(User.class,
user.getUser_id());
u.setUsername(user.getUsername());
u.setEmail(user.getEmail());
session.update(u);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void editUser2(User user){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
User u = (User)session.get(User.class,
user.getUser_id());
u.setPassword(user.getPassword());
session.update(u);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public User getUser(String username) {
User user = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from User
where username = :uname";
Query query = session.createQuery(
queryString);
query.setString("uname", username);
user = (User) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return user;
}

public User getUserById(int user_id) {
User user = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from User where
user_id = :uid";
Query query = session.createQuery(
queryString);
query.setInteger("uid", user_id);
user = (User) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return user;
}

public User getUser_Voter(Voter v) {
User user = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "Select u from
 User u WHERE u.user_id = :vid";
Query query = session.createQuery(
queryString);
query.setInteger("vid", v.getUser_id());
user = (User) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return user;
}

@SuppressWarnings("unchecked")
public boolean findUser(String username,
String password) {
List<User> users =
new ArrayList<User>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from User where
username = :uname and password = :pwd";
Query query = session.createQuery(
queryString);
query.setString("uname", username);
query.setString("pwd", password);
users = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
if(users.size() == 1)
return true;
else
return false;
}

@SuppressWarnings("unchecked")
public List<User> getVotingOfficials() {
```

```
List<User> votingOfficials =
new ArrayList<User>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from User
where role = :r";
Query query = session.createQuery(
queryString);
query.setString("r", "VOTING_OFFICIAL");
votingOfficials = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return votingOfficials;
}

@SuppressWarnings("unchecked")
public List<String> getAllUsernames() {
List<String> users =
new ArrayList<String>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "Select
username from User";
Query query = session.createQuery(
queryString);
users = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return users;
}
}
```

VoteDao.java

```
package com.sp.vote.dao;

import com.sp.vote.model.*;
import com.sp.vote.hibernate.
        HibernateUtil;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

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

public class VoteDao {

public void addVote(Vote vote) {
Transaction trns = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
trns = session.beginTransaction();
session.save(vote);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}
}


public void updateVote(Vote v){
Transaction tx=null;
Session session = HibernateUtil.
getSessionFactory().openSession();
Query query = session.createQuery(
"from Vote v where v.voter_id = ?
and v.position_id = ?");
query.setParameter(0, v.getVoter_id());
query.setParameter(1, v.getPosition_id());

Vote vote = (Vote) query.uniqueResult();
vote.setEvote(v.getEvote());
vote.setTimestamp(v.getTimestamp());
vote.setXhash(v.getXhash());

session.saveOrUpdate(vote);
tx = session.beginTransaction();
tx.commit();
session.close();
}

public int getLastVoteID() {
Session session = HibernateUtil.
getSessionFactory().getCurrentSession();
session.beginTransaction();
return ((Long)session.createQuery("
select count(*) from Vote").
uniqueResult()).intValue();
}

@SuppressWarnings("unchecked")
public List<Vote> getVotes(String alias){
Session session = HibernateUtil.
getSessionFactory().
getCurrentSession();
session.beginTransaction();
List<Vote> votes = new
ArrayList<Vote>();
String sql = "Select v from Vote v,
Voter vr
where vr.alias =:alias AND
v.voter_id=vr.voter_id";
Query query = session.createQuery(sql);
query.setParameter("alias", alias);
votes = query.list();
session.close();
return votes;
}

@SuppressWarnings("unchecked")
public List<Vote> getVotes(int
position_id){
Session session = HibernateUtil.
getSessionFactory().getCurrentSession();
session.beginTransaction();
List<Vote> votes = new ArrayList<Vote>();
String sql = "from Vote v
where v.position_id =:pid";
Query query = session.createQuery(sql);
query.setInteger("pid", position_id);
votes = query.list();
session.close();
return votes;
}

}
```

VoterDao.java

```
package com.sp.vote.dao;

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

import org.hibernate.
        HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import com.sp.vote.hibernate.
        HibernateUtil;
import com.sp.vote.model.User;
import com.sp.vote.model.Voter;

public class VoterDao {

public void addVoter(Voter voter) {
Transaction trns = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
trns = session.beginTransaction();
session.save(voter);
session.getTransaction().commit();
} catch (RuntimeException e) {
if (trns != null) {
trns.rollback();
}
e.printStackTrace();
} finally {
session.flush();
session.close();
}
}
```

```
public void editVoter(Voter voter){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Voter v = (Voter)session.get(
sVoter.class, voter.getVoter_id());
v.setBlock_id(voter.getBlock_id());
session.update(v);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}

public void deleteVoter(Voter voter){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
tx = session.beginTransaction();
Voter v = (Voter)session.get(
Voter.class, voter.getVoter_id());
session.delete(v);
tx.commit();
}catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
}finally {
session.close();
}
}

public Voter getVoter(int user_id) {
Voter voter = null;
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryString = "from Voter
where user_id = :uid";
Query query = session.createQuery(
queryString);
query.setInteger("uid", user_id);
voter = (Voter) query.uniqueResult();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return voter;
}

@SuppressWarnings("unchecked")
public List<Voter> getVotersOfAnEvent(
int event_id) {
List<Voter> voters = new ArrayList
<Voter>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "Select DISTINCT v
FROM Voter v, Event ev, Block b join
b.elections el WHERE v.block_id =
b.block_id  AND el.event_id = :eid";
Query query = session.createQuery(
queryStr);
query.setInteger("eid", event_id);
voters = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return voters;
}

@SuppressWarnings("unchecked")
public List<Voter> getVotersByBlock
(int block_id) {
List<Voter> voters = new
ArrayList<Voter>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "FROM Voter v WHERE
v.block_id = :bid";
Query query = session.createQuery(
queryStr);
query.setInteger("bid", block_id);
voters = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return voters;
}

@SuppressWarnings("unchecked")
public List<Voter> getVotersOfAnEvent(
int uid, int event_id) {
List<Voter> voters = new
ArrayList<Voter>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "Select DISTINCT v
FROM Voter v, Event ev, Block b join
b.elections el WHERE v.block_id=b.block_id
AND el.event_id = :eid AND
ev.builder_id = :uid";
Query query = session.createQuery(
queryStr);
query.setInteger("eid", event_id);
query.setInteger("uid", uid);
voters = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return voters;
}

@SuppressWarnings("unchecked")
public List<User> getVoterUsers(
sint block_id) {
List<User> user_voters = new ArrayList
<User>();
Session session = HibernateUtil.
getSessionFactory().openSession();
try {
String queryStr = "Select DISTINCT
u FROM User u, Voter v, Event ev, Block
b join b.elections el WHERE u.user_id
= v.user_id AND v.block_id = :bid";
Query query = session.createQuery(
queryStr);
query.setInteger("bid", block_id);
user_voters = query.list();
} catch (RuntimeException e) {
e.printStackTrace();
} finally {
session.flush();
session.close();
}
return user_voters;
}

public boolean hasVoted(int user_id){
Session session = HibernateUtil.
getSessionFactory().openSession();
session.beginTransaction();
String sql = " from Voter v
where v.user_id=:user_id and v.voted=:x";
Query query = session.createQuery(sql);
query.setParameter("user_id", user_id);
query.setParameter("x", 1);

@SuppressWarnings("unchecked")
List<Voter> list = query.list();
if (list.size() > 0) {
session.close();
return true;    // has already voted
}
session.close();
return false;
}

public void markVoted(int voter_id,
String code){
Session session = HibernateUtil.
getSessionFactory().openSession();
Transaction tx = null;
try{
```

```
tx = session.beginTransaction();
Voter voter = (Voter)session.get(
Voter.class, voter_id);
voter.setCode(code);
voter.setVoted(1);
session.update(voter);
tx.commit();
} catch (HibernateException e) {
if (tx!=null) tx.rollback();
e.printStackTrace();
} finally {
session.close();
}
}
}
```

A..3 Hibernate Utility

HibernateUtil.java

```
package com.sp.vote.hibernate;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.
  StandardServiceRegistryBuilder;
import org.hibernate.cfg.
  Configuration;
import org.hibernate.service.
  ServiceRegistry;

public class HibernateUtil
{
private static SessionFactory
sessionFactory = buildSessionFactory();

private static SessionFactory
buildSessionFactory()
{
try
{
if (sessionFactory == null)
{
Configuration configuration =
new Configuration().configure(
HibernateUtil.class.getResource(
"/hibernate.cfg.xml"));
StandardServiceRegistryBuilder
serviceRegistryBuilder = new
StandardServiceRegistryBuilder();
serviceRegistryBuilder.applySettings(
configuration.getProperties());
ServiceRegistry serviceRegistry =
serviceRegistryBuilder.build();
sessionFactory = configuration.
buildSessionFactory(serviceRegistry);
}
return sessionFactory;
} catch (Throwable ex)
{
System.err.println("Initial
SessionFactory creation failed." + ex);
throw new ExceptionInInitializerError(ex);
}
}

public static SessionFactory
getSessionFactory(){
return sessionFactory;
}

public static void shutdown(){
getSessionFactory().close();
}
}
```

A..4 Methods

CurrentTimeStamp.java

```
package com.sp.vote.method;

import java.sql.Timestamp;
import java.util.Date;

public class CurrentTimeStamp {

public Timestamp getCurrentDateTime(){
Date date = new Date();
return (new Timestamp(date.getTime()));
}
}
```

EmailSender.java

```
package com.sp.vote.method;

import javax.mail.*;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import java.util.Properties;

/**
* Created by anirudh on 28/10/14.
* http://examples.javacodegeeks.com/core-java/
send-email-with-gmail-in-java-example/
*/
public class EmailSender {

public void sendEmail(String subject,
String body, String toEmail) {

final String username = "voterify.he@gmail.com";
final String password = "voterify6969haha";

Properties props = new Properties();
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.starttls.enable", "true");
props.put("mail.smtp.host", "smtp.gmail.com");
props.put("mail.smtp.port", "587");

Session session = Session.getInstance(props,
new javax.mail.Authenticator() {
protected PasswordAuthentication
getPasswordAuthentication() {
return new PasswordAuthentication(username,
password);
}
});

try {

Message message = new MimeMessage(session);
message.setFrom(new InternetAddress(username));
message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(toEmail));
message.setSubject(subject);
message.setText(body);

Transport.send(message);

System.out.println("Mail sent succesfully!");

} catch (MessagingException e) {
throw new RuntimeException(e);
}
}

public void sendEmail2(String subject, String body,
String toEmail) {

final String username = "voterify.he@gmail.com";
final String password = "voterify6969haha";

Properties props = new Properties();
props.put("mail.smtp.host", "smtp.gmail.com");
props.put("mail.smtp.socketFactory.port", "465");
props.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
props.put("mail.smtp.auth", "true");
props.put("mail.smtp.port", "465");

Session session = Session.getDefaultInstance(props,
new javax.mail.Authenticator() {
protected PasswordAuthentication getPassword
Authentication() {
return new PasswordAuthentication(username,
password);}});

try {

Message message = new MimeMessage(session);
message.setFrom(new InternetAddress(username));
message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(toEmail));
message.setSubject(subject);
message.setText("body");

Transport.send(message);

System.out.println("Mail sent succesfully!");
```

```
} catch (MessagingException e) {
throw new RuntimeException(e);
}}
}
```

Hasher.java

```
package com.sp.vote.method;

import java.security.MessageDigest;
import java.util.Random;

public class Hasher {

public String computeHash(String plaintext,
String algorithm) throws Exception {

MessageDigest md = MessageDigest.
getInstance(algorithm);
md.update(plaintext.getBytes());

byte byteData[] = md.digest();

//convert the byte to hex format method 1
StringBuffer sb = new StringBuffer();
for (int i = 0; i < byteData.length; i++)
sb.append(Integer.toString((byteData[i]
& 0xff) + 0x100, 16).substring(1));

return sb.toString();
}

public String generateStr2(){
Random r = new Random();

String alphabet = "abcdefghijklm01234nopqrst
uvwxyz56789ABCDEFGHIJKLM56789NOPQRSTUVWXYZ1234";
String str = "";
for(int i=0; i<8; i++)  //(4)+3
str += alphabet.charAt(r.nextInt(alphabet.length()));

return str;
}


}
```

MailFileReader.java

```
package com.sp.vote.method;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class MailFileReader {

public static String readMailFile(String
destPath, String role) throws IOException {
String file = destPath + "/mail/" + role;
BufferedReader br = new BufferedReader(
new FileReader(file));
try {
StringBuilder sb = new StringBuilder();
String line = br.readLine();

while (line != null) {
sb.append(line);
sb.append("\n");
line = br.readLine();
}
return sb.toString();
} finally {
br.close();
}
}
}
```

VoteCounter.java

```
package com.sp.vote.method;

import com.sp.vote.model.Vote;

import thep.paillier.EncryptedInteger;
import thep.paillier.PrivateKey;
import thep.paillier.exceptions.
        BigIntegerClassNotValid;
import thep.paillier.exceptions.
        PublicKeysNotEqualException;

import java.math.BigInteger;
import java.util.List;

public class VoteCounter {

public int[] countVotes(List<Vote> votes,
        int candidateCount, PrivateKey sk)
        throws BigIntegerClassNotValid,
        PublicKeysNotEqualException{
int numberOfClusters = votes.size()/9;
if(votes.size() % 9 != 0)
numberOfClusters++;

BigInteger[] voteArr = new BigInteger
        [votes.size()];
EncryptedInteger[] result = new Encrypted
        Integer[numberOfClusters];
EncryptedInteger[] evote = new Encrypted
        Integer[votes.size()];
String[] officialResult = new String[
        numberOfClusters];
int[] votecount = new int[candidateCount];

for(int i=0; i<numberOfClusters; i++){
result[i] = new EncryptedInteger(sk.get
        PublicKey());
officialResult[i] = "";
}

int cluster=0;
String x;
for(int i=0; i<votes.size(); i++){
x = votes.get(i).getEvote();
voteArr[i] = new BigInteger(x);
evote[i] = new EncryptedInteger(sk.
getPublicKey());
evote[i].setCipherVal(voteArr[i]);

result[cluster] = result[cluster].
add(evote[i]);

if((i!=0 && (i+1)%9==0) ||
(i == votes.size()-1)){
officialResult[cluster] =
normalizeStr(result[cluster].decrypt(sk).
toString(), candidateCount);
cluster++;
}
}
votecount = tally(officialResult,
candidateCount);

return votecount;
}

private String normalizeStr(String string,
int candidateCount) {
String newStr="";
for(int i=0; i<candidateCount-
string.length(); i++)
newStr+="0";

return newStr+string;
}

public int[] tally(String[] result,
int candidateCount){
int[] tally = new int[candidateCount];
// no. of candidates
int sum;

for(int i=0; i<tally.length; i++){
sum=0;
for(int j=0; j<result.length; j++)
sum += Character.getNumericValue(
result[j].charAt(i));

tally[i] = sum;
}
return tally;
}
}
```

A..5 Models

Block.java

```
package com.sp.vote.model;

import java.util.Set;

public class Block {
```

```
private int block_id;
private String name;
private String description;
private Set<Election> elections;

public Block(){}

public int getBlock_id() {
return block_id;
}
public void setBlock_id(int block_id) {
this.block_id = block_id;
}

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

public String getDescription() {
return description;
}
public void setDescription(String
        description) {
this.description = description;
}

public Set<Election> getElections() {
return elections;
}
public void setElections(Set<Election>
        elections) {
this.elections = elections;
}


}
```

Candidate.java

```
package com.sp.vote.model;

import java.util.Comparator;

public class Candidate i
        mplements Comparable<Candidate>{

private int candidate_id;
private int position_id;
private int party_id;
private String first_name;
private String last_name;
private int vote_count;
private String temp;

public Candidate(){}

public Candidate(String first_name,
        String last_name){
this.first_name = first_name;
this.last_name = last_name;
}

public int getCandidate_id() {
return candidate_id;
}
public void setCandidate_id(int candidate_id){
this.candidate_id = candidate_id;
}

public int getPosition_id() {
return position_id;
}
public void setPosition_id(int position_id){
this.position_id = position_id;
}
public int getParty_id() {
return party_id;
}
public void setParty_id(int party_id) {
this.party_id = party_id;
}
public String getFirst_name() {
return first_name;
}
public void setFirst_name(String first_name) {
this.first_name = first_name;
}
public String getLast_name() {
return last_name;
}
public void setLast_name(String last_name) {
this.last_name = last_name;
}
public int getVote_count() {
return vote_count;
}
public void setVote_count(int vote_count) {
this.vote_count = vote_count;
}

public String getTemp() {
return temp;
}
public void setTemp(String temp) {
this.temp = temp;
}

public static Comparator<Candidate>
        candidateNameComp = new
        Comparator<Candidate>() {
public int compare(Candidate c1,
        Candidate c2) {
String can1 = c1.getLast_name();
String can2 = c2.getLast_name();

return can1.compareTo(can2);
}

};

@Override
public int compareTo(Candidate o) {
// TODO Auto-generated method stub
return 0;
}

}
```

Election.java

```
package com.sp.vote.model;

public class Election implements
        Comparable<Election> {

private int election_id;
private int event_id;
private String name;
private String pk_n;
private String sk_lambda;
private int priority;
private int tally_stat;
private String temp;

public Election() {}

public Election(int election_id) {
this.election_id = election_id;
}

public int getElection_id() {
return election_id;
}
public void setElection_id(int election_id){
this.election_id = election_id;
}

public int getEvent_id() {
return event_id;
}
public void setEvent_id(int event_id) {
this.event_id = event_id;
}

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

public String getPk_n() {
return pk_n;
}
public void setPk_n(String pk_n) {
this.pk_n = pk_n;
}

public String getSk_lambda() {
return sk_lambda;
```

```
}
public void setSk_lambda(String sk_lambda){
this.sk_lambda = sk_lambda;
}

public int getPriority() {
return priority;
}
public void setPriority(int priority) {
this.priority = priority;
}

@Override
public int compareTo(Election e) {
return(priority - e.priority);
}

public int getTally_stat() {
return tally_stat;
}
public void setTally_stat(int tally_stat){
this.tally_stat = tally_stat;
}

public String getTemp() {
return temp;
}
public void setTemp(String temp) {
this.temp = temp;
}
}
```

Event.java

```
package com.sp.vote.model;

import java.sql.Timestamp;

public class Event {

private int event_id;
private int builder_id;
private String name;
private String description;
private String status;
private int privacy;
private Timestamp start_date;
private Timestamp end_date;
private String temp;

public Event(){}

public int getEvent_id() {
return event_id;
}
public void setEvent_id(int event_id) {
this.event_id = event_id;
}

public int getBuilder_id() {
return builder_id;
}
public void setBuilder_id(int builder_id){
this.builder_id = builder_id;
}

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

public String getDescription() {
return description;
}
public void setDescription(String
        description){
this.description = description;
}

public String getStatus() {
return status;
}
public void setStatus(String status) {
this.status = status;
}

public int getPrivacy() {
return privacy;
}
public void setPrivacy(int privacy) {
this.privacy = privacy;
}

public Timestamp getStart_date() {
return start_date;
}
public void setStart_date(Timestamp
        start_date) {
this.start_date = start_date;
}

public Timestamp getEnd_date() {
return end_date;
}
public void setEnd_date(Timestamp
        end_date) {
this.end_date = end_date;
}

public String getTemp() {
return temp;
}
public void setTemp(String temp) {
this.temp = temp;
}

}
```

Party.java

```
package com.sp.vote.model;

import java.util.Comparator;

public class Party
        implements Comparable<Party>{

private int party_id;
private int event_id;
private String name;

public Party(){}

public int getParty_id() {
return party_id;
}
public void setParty_id(int party_id) {
this.party_id = party_id;
}

public int getEvent_id() {
return event_id;
}
public void setEvent_id(int event_id) {
this.event_id = event_id;
}

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

public static Comparator<Party>
        partyNameComp = new
        Comparator<Party>() {
public int compare(Party p1,
Party p2) {
String par1 = p1.getName();
String par2 = p2.getName();

return par1.compareTo(par2);
}

};

@Override
public int compareTo(Party o) {
// TODO Auto-generated method stub
return 0;
}

}
```

Position.java

```
package com.sp.vote.model;

public class Position implements
        Comparable<Position> {

private int position_id;
```

```
private int election_id;
private int res_block_id;
private String name;
private int ballot_order;
private int required_min;
private int required_max;
private int abstain_count;
private String temp;

public Position(){}

public int getPosition_id() {
return position_id;
}
public void setPosition_id(int position_id) {
this.position_id = position_id;
}

public int getElection_id() {
return election_id;
}
public void setElection_id(int election_id) {
this.election_id = election_id;
}

public int getRes_block_id() {
return res_block_id;
}
public void setRes_block_id(int res_block_id) {
this.res_block_id = res_block_id;
}

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

public int getBallot_order() {
return ballot_order;
}
public void setBallot_order(int ballot_order) {
this.ballot_order = ballot_order;
}

public int getRequired_min() {
return required_min;
}
public void setRequired_min(int required_min) {
this.required_min = required_min;
}

public int getRequired_max() {
return required_max;
}
public void setRequired_max(int required_max) {
this.required_max = required_max;
}

public int getAbstain_count() {
return abstain_count;
}
public void setAbstain_count(int abstain_count) {
this.abstain_count = abstain_count;
}

public String getTemp() {
return temp;
}
public void setTemp(String temp) {
this.temp = temp;
}

@Override
public int compareTo(Position p) {
// TODO Auto-generated method stub
return (ballot_order - p.ballot_order);
}

}
```

Ticket.java

```
package com.sp.vote.model;

public class Ticket {

private int ticket_id;
private int voter_id;
private String description;

public Ticket() {}

public int getTicket_id() {
return ticket_id;
}
public void setTicket_id(int ticket_id) {
this.ticket_id = ticket_id;
}

public int getVoter_id() {
return voter_id;
}
public void setVoter_id(int voter_id) {
this.voter_id = voter_id;
}

public String getDescription() {
return description;
}
public void setDescription(String
        description) {
this.description = description;
}


}
```

User.java

```
package com.sp.vote.model;

public class User implements {

private int user_id;
private String username;
private String password;
private String salt;
private String email;
private String role;
private int temp;
private int temp2;

public User() {}

public User(String username){
this.username = username;
}

public int getUser_id() {
return user_id;
}
public void setUser_id(int user_id) {
this.user_id = user_id;
}

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

public String getPassword() {
return password;
}
public void setPassword(String password) {
this.password = password;
}

public String getSalt() {
return salt;
}
public void setSalt(String salt) {
this.salt = salt;
}

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

public String getRole() {
return role;
}
public void setRole(String role) {
this.role = role;
}

public int getTemp() {
return temp;
```

```
}
public void setTemp(int temp) {
this.temp = temp;
}

public int getTemp2() {
return temp2;
}

public void setTemp2(int temp2) {
this.temp2 = temp2;
}


}
```

Vote.java

```
package com.sp.vote.model;

import java.sql.Timestamp;

public class Vote {

private int voter_id;
private int position_id;
private String evote;
private Timestamp timestamp;
private String xhash;
private String temp;

public Vote(){}

public Vote(int voter_id, int position_id,
String evote, Timestamp timestamp,
String xhash,
String temp) {
this.voter_id = voter_id;
this.position_id = position_id;
this.evote = evote;
this.setTimestamp(timestamp);
this.setXhash(xhash);
this.temp = temp;
}

public int getVoter_id() {
return voter_id;
}
public void setVoter_id(int voter_id) {
this.voter_id = voter_id;
}

public int getPosition_id() {
return position_id;
}
public void setPosition_id(int
position_id) {
this.position_id = position_id;
}

public String getEvote() {
return evote;
}
public void setEvote(String evote) {
this.evote = evote;
}

public Timestamp getTimestamp() {
return timestamp;
}
public void setTimestamp(Timestamp
timestamp) {
this.timestamp = timestamp;
}

public String getXhash() {
return xhash;
}
public void setXhash(String xhash) {
this.xhash = xhash;
}

public String getTemp() {
return temp;
}
public void setTemp(String temp) {
this.temp = temp;
}
}
```

Voter.java

```
package com.sp.vote.model;

public class Voter {

private int voter_id;
private int user_id;
private int block_id;
private String alias;
private String code;
private int voted;

public Voter(){}

public int getVoter_id() {
return voter_id;
}
public void setVoter_id(int voter_id) {
this.voter_id = voter_id;
}

public int getUser_id() {
return user_id;
}
public void setUser_id(int user_id) {
this.user_id = user_id;
}

public int getBlock_id() {
return block_id;
}
public void setBlock_id(int block_id) {
this.block_id = block_id;
}

public String getAlias() {
return alias;
}
public void setAlias(String alias) {
this.alias = alias;
}

public String getCode() {
return code;
}
public void setCode(String code) {
this.code = code;
}

public int getVoted() {
return voted;
}
public void setVoted(int voted) {
this.voted = voted;
}


}
```

A..6 Resources

Block.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.Block"
table="block">
<id name="block_id" type="int" column="block_id">
<generator class="increment" />
</id>
<property name="name"><column name="name" />
</property>
<property name="description">
<column name="description" />
</property>
<set name="elections" table="block_election">
<key column="block_id"/>
<many-to-many column="election_id"
class="com.sp.vote.model.Election"/>
</set>
</class>
</hibernate-mapping>
```

Candidate.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
```

```
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.Candidate"
table="candidate">
<id name="candidate_id" type="int"
column="candidate_id">
<generator class="increment" />
</id>
<property name="position_id">
<column name="position_id" />
</property>
<property name="party_id">
<column name="party_id" />
</property>
<property name="last_name">
<column name="last_name" />
</property>
<property name="first_name">
<column name="first_name" />
</property>
<property name="vote_count">
<column name="vote_count" />
</property>
</class>
</hibernate-mapping>
```

Election.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.Election"
table="election">
<id name="election_id" type="int"
column="election_id">
<generator class="increment" />
</id>
<property name="event_id">
<column name="event_id" />
</property>
<property name="name">
<column name="name" />
</property>
<property name="pk_n">
<column name="pk_n" />
</property>
<property name="sk_lambda">
<column name="sk_lambda" />
</property>
<property name="priority">
<column name="priority" />
</property>
<property name="tally_stat">
<column name="tally_stat" />
</property>
</class>
</hibernate-mapping>
```

Event.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.Event"
table="event">
<id name="event_id" type="int"
column="event_id">
<generator class="increment" />
</id>
<property name="builder_id">
<column name="builder_id" />
</property>
<property name="name">
<column name="name" />
</property>
<property name="description">
<column name="description" />
</property>
<property name="status">
<column name="status" />
</property>
<property name="privacy">
<column name="privacy" />
</property>
<property name="start_date">
<column name="start_date" />
</property>
<property name="end_date">
<column name="end_date" />
</property>

</class>
</hibernate-mapping>
```

Party.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.Party"
table="party">
<id name="party_id" type="int"
column="party_id">
<generator class="increment" />
</id>
<property name="name">
<column name="name" />
</property>
<property name="event_id">
<column name="event_id" />
</property>
</class>
</hibernate-mapping>
```

Position.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.Position"
table="position">
<id name="position_id" type="int"
column="position_id">
<generator class="increment" />
</id>
<property name="election_id">
<column name="election_id" />
</property>
<property name="res_block_id">
<column name="res_block_id" />
</property>
<property name="name">
<column name="name" />
</property>
<property name="ballot_order">
<column name="ballot_order" />
</property>
<property name="required_min">
<column name="required_min" />
</property>
<property name="required_max">
<column name="required_max" />
</property>
<property name="abstain_count">
<column name="abstain_count" />
</property>


</class>
</hibernate-mapping>
```

Ticket.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.Ticket"
table="ticket">
<id name="ticket_id" type="int"
column="ticket_id">
<generator class="increment" />
</id>
<property name="voter_id">
<column name="voter_id" />
</property>
<property name="description">
<column name="description" />
</property>
</class>
</hibernate-mapping>
```

User.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.User"
table="user">
<id name="user_id" type="int"
column="user_id">
<generator class="increment" />
</id>
<property name="username">
<column name="username" />
</property>
<property name="password">
<column name="password" />
</property>
<property name="salt">
<column name="salt" />
</property>
<property name="email">
<column name="email" />
</property>
<property name="role">
<column name="role" />
</property>
</class>
</hibernate-mapping>
```

Vote.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.Vote"
table="vote">
<composite-id>
<key-property name="voter_id"
column="voter_id" />
<key-property name="position_id"
column="position_id" />
</composite-id>
<property name="evote">
<column name="evote" />
</property>
<property name="timestamp">
<column name="timestamp" />
</property>
<property name="xhash">
<column name="xhash" />
</property>
</class>
</hibernate-mapping>
```

Voter.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/
hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.sp.vote.model.Voter"
table="voter">
<id name="voter_id" type="int"
column="voter_id">
<generator class="increment" />
</id>
<property name="user_id">
<column name="user_id" />
</property>
<property name="block_id">
<column name="block_id" />
</property>
<property name="alias">
<column name="alias" />
</property>
<property name="code">
<column name="code" />
</property>
<property name="voted">
<column name="voted" />
</property>
</class>
</hibernate-mapping>
```

hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//
Hibernate/Hibernate Configuration DTD 3.0//EN" "
http://www.hibernate.org/dtd/hibernate-
configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property name="hibernate.dialect">
org.hibernate.dialect.MySQLDialect</property>
<property name="hibernate.connection.driver_class">
com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">
${db.url}</property>
<property name="hibernate.connection.username">
${db.username}</property>
<property name="hibernate.connection.password">
${db.password}</property>
<property name="hibernate.connection.pool_size">
10</property>
<property name="hibernate.current_session_context
_class">thread</property>
<property name="hibernate.cache.provider_class">
org.hibernate.cache.internal.NoCacheProvider
</property>
<property name="hibernate.show_sql">true</property>

<property name="hibernate.c3p0.min_size">
5</property>
<property name="hibernate.c3p0.max_size">
20</property>
<property name="hibernate.c3p0.timeout">
300</property>
<property name="hibernate.c3p0.max_statements">
50</property>
<property name="hibernate.c3p0.idle_test_period">
3000</property>
<property name="hibernate.c3p0.privilegeSpawned
Threads">true</property>
<property name="hibernate.c3p0.contextClassLoader
Source">library</property>

<mapping resource="User.hbm.xml"/>
<mapping resource="Event.hbm.xml"/>
<mapping resource="Election.hbm.xml"/>
<mapping resource="Party.hbm.xml"/>
<mapping resource="Block.hbm.xml"/>
<mapping resource="Position.hbm.xml"/>
<mapping resource="Candidate.hbm.xml"/>
<mapping resource="Voter.hbm.xml"/>
<mapping resource="Vote.hbm.xml"/>
<mapping resource="Ticket.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

build.properties

```
db.url=jdbc:mysql://localhost:3306/Voterify
db.username=root
db.password=
```

A..7 Web Content

about.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>
<br><br>
<div align="center">
<img src="<s:url value='/dashgum/assets/img/
        Drawing.png'/>"><br><br>
<h4>Privacy-preserving and verifiable elections
</h4>
</div>
<br><br>
<div align="center">
<ul>
<li><h4>Voterify is a homomorphic encryption-based
Internet voting system. Through homomorphic
encryption, votes are encrypted via 1024-bit keys.
</h4></li>
<li><h4>Voterify provides bulletin boards of votes
which display voters' hashed encrypted votes.</h4>
</li>
<li><h4>Voterify utilizes The Homomorphic Encryption
Project (THEP) to perform the necessary homomorphic
operations.</h4></li>
<li><h4>Voterify makes use of the DashGum bootstrap
template.</h4></li>
</ul>
</div>
```

```
</body>
</html>
```

account.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<section id="main-content">
<section class="wrapper">

<div class="row">
<h2> User Profile </h2>
<div class="col-lg-8 main-chart">
<table class="table table-striped table-advance
table-hover"
 id="datazexample">
<tbody>
<tr>
<td>Username</td>
<td><s:property value="user.getUsername()" /></td>
<td>
<button class="btn btn-primary" data-toggle="modal"
data-target=
"#editUNmodal"><i class="fa fa-pencil"></i></button>
</td>
</tr>
<tr>
<td>Password</td>
<td>
<a data-toggle="modal" href="#editPWmodal">Change
Password</a>
</td>
<td> </td>
</tr>
<tr>
<td>Email</td>
<td><s:property value="user.getEmail()" /></td>
<td>
<button class="btn btn-primary" data-toggle="modal"
data-target="#
editEAmodal"><i class="fa fa-pencil"></i> </button>
</td>
</tr>
</tbody>
</table>

</div>

<!-- username -->
<s:form action="account" theme="simple">
<!-- Modal -->
<div aria-hidden="true" aria-labelledby="
myModalLabel"
role="dialog" tabindex="-1" id="editUNmodal"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<s:a href="account" cssClass="close">&times;</s:a>
<h4 class="modal-title">Edit Username</h4>
</div>
<div class="modal-body">
<p>Username</p>
<s:textfield name="user.username" value="%{user.
getUsername()}" cssClass="form-control"
required="required" /><br>
<s:fielderror fieldName="unF" /><br>

<s:hidden name="mode" value="eun" />
</div>
<div class="modal-footer">
<s:a href="account" cssClass="btn btn-default">
Cancel</s:a>
<s:submit value="Submit" cssClass="
btn btn-theme"/>
</div>
</div>
</div>
</div>
<!-- modal -->
</s:form>

<!-- email -->
<s:form action="account" theme="simple">
<!-- Modal -->
<div aria-hidden="true" aria-labelledby="
myModalLabel"
role="dialog" tabindex="-1" id="editEAmodal"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<s:a href="account" cssClass="close">&times;
</s:a>
<h4 class="modal-title">Edit Email</h4>
</div>
<div class="modal-body">
<p>Email</p>
<s:textfield name="user.email" value="%{
user.getEmail()}"
cssClass="form-control" required="required" />
<br>
<s:fielderror fieldName="emailF" /><br>

<s:hidden name="mode" value="eea" />
</div>
<div class="modal-footer">
<s:a href="account" cssClass="btn btn-default">
Cancel</s:a>
<s:submit value="Submit" cssClass="
btn btn-theme"/>
</div>
</div>
</div>
</div>
<!-- modal -->
</s:form>


<!-- password -->
<s:form action="account" theme="simple">
<!-- Modal -->
<div aria-hidden="true" aria-labelledby="m
yModalLabel"
role="dialog" tabindex="-1" id="editPWmodal"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<s:a href="account" cssClass="close">&times;
</s:a>
<h4 class="modal-title">Edit Password</h4>
</div>
<div class="modal-body">
<p>Old Password</p>
<s:password name="oldpass" cssClass="
form-control"
required="required" /><br>
<div class="error"><s:fielderror
fieldName="oldpF" />
</div> <br>

<p>New Password</p>
<s:password name="newpass" cssClass="
form-control" required="required" /><br>
<div class="error"><s:fielderror
fieldName="newpF" /></div> <br>

<p>Confirm New Password</p>
<s:password name="retpass" cssClass="
form-control" required="required" /><br>
<div class="error"><s:fielderror
fieldName="retpF" /></div> <br>

<s:hidden name="mode" value="epw" />
</div>
<div class="modal-footer">
<s:a href="account" cssClass="
btn btn-default">
Cancel</s:a>
<s:submit value="Submit" cssClass="
btn btn-theme"/></div>
</div>
</div>
</div>
<!-- modal -->
</s:form>

</div><! --/row -->
</section>
</section>

<!--main content end-->

</section>

<s:include value="/pages/includes/scripts.jsp"/>

</body>
</html>
```

blocks.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>

<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start -->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:url
value='/dashgum/assets/img/logo.png'/>"></a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="%{#e.event_id}" >
</s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}">${sessionScope.event.name}
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a class="active" href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="#" onclick="opennewtab('board')" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<!--main content start-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3>${sessionScope.event.name}</h3>

<div class="row mt">
<div class="col-md-12">
<div class="content-panel">
<table id="datazexample" class="table
table-striped table-advance table-hover">
<h4><i class="fa fa-angle-right"></i> Block
<s:if test='%{event.getStatus().equals("IDLE")}'>
<div class="pull-right">
<button class="btn btn-primary" data-toggle="modal"
data-target="#myModal"><i class="fa fa-plus"></i>
Add Block</button>
   
</div>
</s:if>
</h4>
<hr>
<thead>
<tr>
<th><i class="fa fa-bullhorn"></i> Name</th>
<th class="hidden-phone"><i class="fa
fa-question-circle">
</i> Action</th>
</tr>
</thead>
<tbody>
<s:iterator value="blocks" var="b">
<tr>
<td><s:property value="#b.name" /></td>
<td>
<s:if test='%{event.getStatus().
equals("IDLE")}'>
<s:url action="blocks" var="editURL"
escapeAmp="false">
<s:param name="block.block_id" value="
%{#b.block_id}">
</s:param>
<s:param name="block.name" value="%{#b.name}">
</s:param>
<s:param name="block.description" value="
%{#b.description}">
</s:param>
<s:param name="event.event_id" value="%{event.
getEvent_id()}">
</s:param>
<s:param name="mode" value="'editPage'"></s:param>
</s:url>
<s:a href="%{editURL}" cssClass="btn btn-
primary btn-xs">
<i class="fa fa-pencil"></i></s:a>
<s:url action="blocks" var="deleteURL" >
<s:param name="block.block_id" value="
%{#b.block_id}">
</s:param>
<s:param name="mode" value="'delete'"></s:param>
</s:url>
<s:a href="%{deleteURL}" cssClass="btn
btn-danger btn-xs"
onclick="return
confirm('Are you sure you want to delete this
block?')">
<i class="fa fa-trash-o"></i></s:a>
</s:if>
<s:else> </s:else>
</td>
</tr>
</s:iterator>

</tbody>
</table>
</div><!-- /content-panel -->
</div><!-- /col-md-12 -->
</div><!-- /row -->


<s:form action="blocks" theme="simple">
<!-- Modal -->
<div aria-hidden="true" aria-labelledby="
myModalLabel"
role="dialog" tabindex="-1" id="myModal"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<s:url action="blocks" var="lolURL" />
<s:a href="%{lolURL}" cssClass="close">&times;
</s:a>
<h4 class="modal-title">Add Block</h4>
</div>
<div class="modal-body">
<p>Name</p>
<s:textfield name="block.name" cssClass=
"form-control" required="required" /><br>
<s:fielderror fieldName="block.name"
cssClass="myerror" /><br>

<p>Description</p>
<s:textarea name="block.description"
cssClass="form-control" required="required" />
<br>
```

```
<s:fielderror fieldName="block.description"
cssClass="myerror" /><br>
<p>Election/s the block will participate in
</p>
<s:iterator value="elections" var="e">
<input type="checkbox" name="myElections"
value="<s:property value='%{#e.election_id}' />">
  <s:property value="%{#e.name}"/><br>

</s:iterator>
<s:fielderror fieldName="blockh"
cssClass="myerror" /><br>
<br>

<s:hidden name="mode" value="add" />
</div>
<div class="modal-footer">
<s:url action="blocks" var="lolURL" />
<s:a href="%{lolURL}"
cssClass="btn btn-default">Cancel</s:a>
<s:submit value="Submit" cssClass="
btn btn-theme"/>
</div>
</div>
</div>
</div>
<!-- modal -->
</s:form>

<s:if test="errorDel == false">
<script>
alert("You cannot delete an election in
use by positions, candidates, ...");
</script>
</s:if>
</div><!-- /col-lg-9 END SECTION MIDDLE -->
</div><! --/row -->
</section>
</section>

<!--main content end-->

</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})

function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>

<s:if test="hasError[0] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myModal').modal('show');
});
</script>
</s:if>

</body>
</html>
```

board-error.jsp

```
<s:include value="/pages/includes/head.jsp"/>
<body>
<br><br>
<div align="center">
<img src="<s:url value='/dashgum/assets/img/
Drawing.png'/>"><br><br>
<h4>Privacy-preserving and verifiable elections
</h4>
</div>

<section id="container" >

<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3><s:property value="event.getName()" />
</h3><br>
<h4>Bulletin Board of Votes</h4>

<div class="row mt">
<div class="col-md-12">
<b>Not yet available</b><br>

The bulletin board will only be available
once this event has been stopped by the voting
official.
</div><!-- /col-md-12 -->
</div><!-- /row -->

</div><! --/row -->
</section>
</section>


</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
</script>
</body>
</html>
```

board-ind.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>

<body>

<br><br>
<div align="center">
<img src="<s:url value='/dashgum/assets/img/
Drawing.png'/>"><br><br>
<h4>Privacy-preserving and verifiable elections
</h4>
</div>

<section id="container" >

<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h2><s:property value="%{event.getName()}" /></h2>
<br>
<h3>Bulletin Board of Votes</h3>

<div class="row mt">
<div class="col-md-12">
<h4><s:property value="%{voter.getAlias()}" />
</h4>
<s:iterator value="votes" var="v">
<b><s:property value="%{#v.temp}"/></b>:
<s:property value="%{#v.evote}"/><br>
</s:iterator>
<br>

<s:url action="board" var="backBoard">
<s:param name="event.event_id" value="
%{event.getEvent_id()}" />
</s:url>
<s:a href="%{backBoard}" cssClass=
"btn btn-primary">Back</s:a>

</div><!-- /col-md-12 -->
</div><!-- /row -->

</div><!-- /col-lg-9 END SECTION MIDDLE -->
</div><! --/row -->
</section>
</section>
</section>
<s:include value="/pages/includes/scripts.jsp"/>
<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
```

```
</script>

</body>
</html>
```

board-main.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>
<br><br>
<div align="center">
<img src="<s:url value='/dashgum/assets/img/
Drawing.png'/>"><br><br>
<h4>Privacy-preserving and verifiable elections
</h4>
</div>

<section id="container" >
<section id="main-content">
<section class="wrapper">
<div class="row">
<div class="col-lg-12 main-chart">

<h3><s:property value="event.getName()" />
- Bulletin Board of Votes</h3>

<div class="row mt">


<div class="col-md-9">

<table class="table table-striped table-advance
table-hover"
id="datazexample">
<thead>
<tr>
<th><i class="icon_profile"></i> Alias</th>
<th><i class="icon_profile"></i> Verification
Code</th>
<th> </th>
</tr>
</thead>
<tbody>
<s:iterator value="voters" var="v">
<tr>
<td><s:property value="#v.alias" /></td>
<td><s:property value="#v.code" /></td>
<td>
<s:if test="%{#v.code != ''}">
<s:url action="board" var="urlTag" >
<s:param name="voter.alias" value="%{#v.alias}">
</s:param>
<s:param name="event.event_id" value="%{event.
getEvent_id()}">
</s:param>
<s:param name="mode" value="'viewVote'"></s:param>
</s:url>
[<s:a href="%{urlTag}">View</s:a>]
</s:if>
<s:else> </s:else>
</td>
</tr>
</s:iterator>
</tbody>
</table>
</div><!-- /col-md-12 -->
</div><!-- /row -->

<s:url var="pdfDownload" action="key_download" >
<s:param name="filename" value="%{event.getTemp()}">
</s:param>
</s:url>
<s:a href="%{pdfDownload}" cssClass="btn
btn-primary">
Export bulletin board</s:a>
<s:if test='%{role.equals("VOTER")}'>
<s:url action="ticket" var="tickU">
<s:param name="uid" value="%{getUid()}" />
</s:url>
<s:a href="%{tickU}" cssClass="btn btn-primary">
Raise a Ticket</s:a>
</s:if>
</div><!-- /col-lg-9 END SECTION MIDDLE -->
</div><! --/row -->
</section>
</section>
</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
</script>

</body>
</html>
```

candidates.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="
<s:url value='/dashgum/assets/img/logo.png'/>">
</a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="
%{#e.event_id}"></s:param>
<s:param name="mode" value="'goToEvent'">
</s:param>
</s:url>
<s:a href="%{viewURL}">${sessionScope.event.name}
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a class="active" href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="#" onclick="opennewtab('board')">
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<!--main content start-->
<section id="main-content">
<section class="wrapper">
```

```
<div class="row">
<div class="col-lg-12 main-chart">

<h3>${sessionScope.event.name}</h3>

<div class="row mt">
<div class="col-md-12">

<h4><s:property value="%{election.getName()}" />
</h4>
<div class="content-panel">
<table id="datazexample" class="table
table-striped table-advance table-hover">
<h4><i class="fa fa-angle-right"></i>
Candidates <i class="fa fa-angle-right"></i>
<s:property value="%{position.getName()}" />
<s:if test='%{event.getStatus().equals("IDLE")}'>
<div class="pull-right">
<button class="btn btn-primary" data-toggle="modal"
data-target="#myModal"><i class="fa fa-plus"></i>
Add Candidate</button>
   
</s:if>
</div>
</h4>
<hr>
<thead>
<tr>
<th><i class="fa fa-bullhorn"></i> Name</th>
<th><i class="fa fa-bullhorn"></i> Party</th>
<th class="hidden-phone"><i class="
fa fa-question-circle"></i> Action</th>
</tr>
</thead>
<tbody>
<s:iterator value="candidates" var="c">
<tr>
<td>
<s:property value="%{#c.last_name}" />,
<s:property value="%{#c.first_name}" />
</td>
<td>
<s:property value="%{#c.temp}" />
</td>
<td>
<s:if test='%{event.getStatus().equals("IDLE")}'>
<s:url action="candidates" var="editURL"
escapeAmp="false">
<s:param name="candidate.candidate_id"
value="%{#c.candidate_id}"></s:param>
<s:param name="candidate.first_name"
value="%{#c.first_name}"></s:param>
<s:param name="candidate.last_name"
value="%{#c.last_name}"></s:param>
<s:param name="candidate.party_id"
value="%{#c.party_id}"></s:param>
<s:param name="candidate.position_id"
value="%{#c.position_id}"></s:param>
<s:param name="position.position_id"
value="%{#c.position_id}"></s:param>
<s:param name="position.election_id"
value="%{election.getElection_id()}"></s:param>
<s:param name="mode" value="'editPage'"></s:param>
</s:url>
<s:a href="%{editURL}" data-toggle="modal"
data-target="#myeditdialogc" data-backdrop="static"
data-keyboard="false" cssClass="btn
btn-primary btn-xs"><i class="fa fa-pencil"></i>
</s:a>

<s:url action="candidates" var="deleteURL" >
<s:param name="candidate.candidate_id"
value="%{#c.candidate_id}"></s:param>
<s:param name="position.position_id"
value="%{#c.position_id}"></s:param>
<s:param name="position.election_id"
value="%{election.getElection_id()}"></s:param>
<s:param name="mode" value="'delete'"></s:param>
</s:url>
<s:a href="%{deleteURL}" cssClass="btn btn-danger
btn-xs"
onclick="return confirm('Are you sure you want to
delete this candidate?')">
<i class="fa fa-trash-o"></i></s:a>
</s:if>
<s:else> </s:else>
</td>
</tr>
</s:iterator>
</tbody>
</table>
<s:url action="positions" var="lolURL">
<s:param name="mode" value="'vpe'" />
<s:param name="election.election_id"
value="%{election.getElection_id()}" />
</s:url>
<s:a href="%{lolURL}" cssClass="
btn btn-primary">Back to Positions</s:a>
</div><!-- /content-panel -->
</div><!-- /col-md-12 -->
</div><!-- /row -->
<s:form action="candidates" theme="simple">
<!-- Modal -->
<div aria-hidden="true" aria-labelledby="
myModalLabel" role="dialog" tabindex="-1"
id="myModal" class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<button type="button" class="close"
data-dismiss="modal"
aria-hidden="true">&times;</button>
<h4 class="modal-title">Add Candidate</h4>
</div>
<div class="modal-body">

<p>First Name</p>
<s:textfield name="candidate.first_name"
cssClass="form-control" /><br>
<s:fielderror fieldName="candidate.first_name"
cssClass="myerror" /><br>
<p>Last Name</lp>
<s:textfield name="candidate.last_name"
cssClass="form-control" /><br>
<s:fielderror fieldName="candidate.last_name"
cssClass="myerror" /><br>

<p>Party</p>
<s:select headerKey="1" headerValue="No Party
(Independent)"
list="parties" listKey="party_id" listValue="name"
name="candidate.party_id" cssClass="form-control
m-bot15"
required="required"/><br><br>

<s:hidden name="position.position_id"
value="%{position.getPosition_id()}" />
<s:hidden name="candidate.position_id"
value="%{position.getPosition_id()}" />

<s:hidden name="mode" value="add" />
</div>
<div class="modal-footer">
<button data-dismiss="modal" class="
btn btn-default" type="button">Cancel</button>
<s:submit value="Submit" cssClass="
btn btn-theme"/>
</div>
</div>
</div>
</div>
</s:form>
<div aria-hidden="true" aria-labelledby="
myModalLabel" role="dialog" tabindex="-1"
id="myeditdialogc" class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
</div>
</div>
</div>
<div aria-hidden="true" aria-labelledby="
myModalLabel" role="dialog" tabindex="-1"
 id="myeditdialog2" class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<s:form action="candidates" method="post"
theme="simple">
<div class="modal-header">
<s:url action="candidates" var="lolURL1" >
<s:param name="position.position_id" value="
%{position.getPosition_id()}"></s:param>
<s:param name="mode" value="'cand'"></s:param>
</s:url>
<s:a href="%{lolURL1}" cssClass="close">&times;
</s:a>
<h4 class="modal-title">Edit Candidate</h4>
</div>
<div class="modal-body">
<p>First Name</p>
<s:textfield name="candidate.first_name" value="
%{candidate.getFirst_name()}" cssClass="
form-control" /><br>
<s:fielderror fieldName="candidate.first_name"
cssClass="myerror" /><br>
```

```
<p>Last Name</lp>
<s:textfield name="candidate.last_name" value="
%{candidate.getLast_name()}" cssClass="form-control"
/><br>
<s:fielderror fieldName="candidate.last_name"
cssClass="myerror" /><br>
<p>Party</p>
<s:select headerKey="1" headerValue="No Party
(Independent)"
list="parties" listKey="party_id" listValue="name"
name="candidate.party_id" cssClass="form-control
m-bot15"  required="required"/><br><br>
<s:hidden name="candidate.candidate_id"
value="%{candidate.getCandidate_id()}" />
<s:hidden name="candidate.position_id"
value="%{position.getPosition_id()}" />
<s:hidden name="position.position_id"
value="%{candidate.getPosition_id()}" />
<s:hidden name="mode" value="edit" />
</div>
<div class="modal-footer">
<s:url action="candidates" var="lolURL" >
<s:param name="position.position_id"
value="%{position.getPosition_id()}"></s:param>
<s:param name="mode" value="'cand'"></s:param>
</s:url>
<s:a href="%{lolURL}" cssClass="btn btn-default">
Cancel</s:a>
<s:submit value="Submit" cssClass="btn btn-theme"/>
</div>


</s:form>

</div>
</div>
</div>
<!-- modal -->


<s:if test="errorDel == false">
<script>
alert("You cannot delete an election in use by positions,
candidates, ...");
</script>
</s:if>


</div><!-- /col-lg-9 END SECTION MIDDLE -->


</div><! --/row -->
</section>
</section>

<!--main content end-->
<!--footer start-->

<!--footer end-->
</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({

});
})

function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>

<s:if test="hasError[0] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myModal').modal('show');
});
</script>
</s:if>
<s:if test="hasError[1] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myeditdialog2').modal('show');
});
</script>
</s:if>


</body>
</html>
```

elections.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>

<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>

<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<
s:url value='/dashgum/assets/img/logo.png'/>">
</a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" escapeAmp=
"false">
<s:param name="event.event_id"></s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}">${sessionScope.event.name}
</s:a>
</h5>

<li class="mt">
<a class="active" href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>
<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="#" onclick="opennewtab('board')" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->

<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3>${sessionScope.event.name}</h3>

<div class="row mt">
<div class="col-md-12">
<div class="content-panel">
<table id="datazexample" class="table t
able-striped table-advance table-hover">
```

```
<h4><i class="fa fa-angle-right"></i> Elections
<s:if test='%{event.getStatus().equals("IDLE")}'>
<div class="pull-right">
<button class="btn btn-primary" data-toggle="modal"
 data-target="#myModal"><i class="fa fa-plus"></i>
 Election</button>
    

</div>
</s:if>
</h4>
<hr>
<thead>
<tr>
<th><i class="fa fa-bullhorn"></i> Name</th>
<th><i class="fa fa-bullhorn"></i> Priority</th>
<th class="hidden-phone"><i class="fa fa-questio
n-circle"></i> Action</th>
</tr>
</thead>
<tbody>
<s:iterator value="elections" var="e">
<tr>
<td><s:property value="#e.name" /></td>
<td>
<s:if test="%{#e.priority == 0}">
Major
</s:if>
<s:else>
Minor
</s:else>
</td>
<td>
<s:if test='%{event.getStatus().equals("IDLE")}'>
<s:url action="elections" var="editURL">
<s:param name="election.election_id" value="%{#e.e
lection_id}"></s:param>
<s:param name="election.name" value="%{#e.name}">
</s:param>
<s:param name="election.priority" value="%{#e.prio
rity}"></s:param>
<s:param name="mode" value="'editPage'"></s:param>
</s:url>
<s:a href="%{editURL}" data-toggle="modal" dat
a-target="#myeditdialog" data-backdrop="static" d
ata-keyboard="false" cssClass="btn btn-primary bt
n-xs"><i class="fa fa-pencil"></i></s:a>


<s:url action="elections" var="deleteURL" >
<s:param name="election.election_id" value="%{#e.el
ction_id}"></s:param>
<s:param name="mode" value="'delete'"></s:param>
</s:url>
<s:a href="%{deleteURL}" cssClass="btn btn-danger bt
n-xs" onclick="return confirm('Are you sure you want
 to
delete this election?')">
<i class="fa fa-trash-o"></i></s:a>

</s:if>
<s:else> </s:else>
</td>
</tr>
</s:iterator>

</tbody>
</table>
</div><!-- /content-panel -->
</div><!-- /col-md-12 -->
</div><!-- /row -->


<s:form action="elections" theme="simple">
<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel" role=
"dialog" tabindex="-1" id="myModal" class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<s:url action="elections" var="lolURL" />
<s:a href="%{lolURL}" cssClass="close">&times;</s:a>
<h4 class="modal-title">Add Election</h4>
</div>
<div class="modal-body">
<p>Election Name</p>
<s:textfield name="election.name" cssClass="form-c
ontrol" /><br>
<s:fielderror fieldName="ename" cssClass="myerror" /><br>

<p>Priority (An event should have one major electio
and zero or more minor elections)</p>
<s:radio name="election.priority" list="#{'1':'Major',
 '2':'Minor'}" /><br>
<s:fielderror fieldName="epriority" cssClass="myerror"/>
<br>
<s:hidden name="mode" value="add" />
</div>
<div class="modal-footer">
<s:a href="elections" cssClass="btn btn-default">C
ancel</s:a>
<s:submit value="Submit" cssClass="btn btn-theme"/>
</div>
</div>
</div>
</div>
<!-- modal -->
</s:form>


<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
 role="dialog" tabindex="-1" id="myeditdialog"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">

</div>
</div>
</div>
<!-- modal -->

<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabe
l" role="dialog" tabindex="-1" id="myeditdialog2"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<s:form action="elections" method="post" theme="simple">

<div class="modal-header">
<s:url action="elections" var="lolURL" />
<s:a href="%{lolURL}" cssClass="close">&times;</s:a>
<h4 class="modal-title">Edit Election</h4>
</div>
<div class="modal-body">
<p>Election Name</p>
<s:textfield name="election.name" value="%{election.
getName()}"
 cssClass="form-control" required="required" /><br>
<s:fielderror fieldName="ename" cssClass="myerror" />
<br>

<p>Priority (Major elections appear on the top of the
ballot)</p>
<s:radio name="election.priority" list="#{'1':'Major','
2':'Minor'}" value="%{election.getPriority()}" /><br>
<s:fielderror fieldName="epriority" cssClass="
myerror" /><br>

<s:hidden name="election.election_id" value="
%{election.ge
tElection_id()}" />
<s:hidden name="mode" value="edit" />
</div>
<div class="modal-footer">
<s:url action="elections" var="lolURL2" />
<s:a href="%{lolURL2}" cssClass="btn btn-default">
Cancel</s:a>
<s:submit value="Submit" cssClass="btn btn-theme"/>
</div>
</s:form>
</div>
</div>
</div>
<!-- modal -->

<s:if test="errorDel == false">
<script>
alert("You cannot delete an election in use
by blocks,
 positions, and candidates.");
</script>
</s:if>

</div><!-- /col-lg-9 END SECTION MIDDLE -->
</div><! --/row -->
</section>
</section>
```

```
<!--main content end-->

</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})

function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>

<s:if test="hasError[0] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myModal').modal('show');
});
</script>
</s:if>
<s:if test="hasError[1] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myeditdialog2').modal('show');
});
</script>
</s:if>

</body>
</html>
```

event-home.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>

<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>

<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:url
value='/dashgum/assets/img/logo.png'/>"></a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}">${sessionScope.event.name}
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="#" onclick="opennewtab('board')">
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<!--main content start-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-9 main-chart">

<h1><s:property value="event.getName()" /> 
<s:if test="%{event.getStatus().equals('IDLE')}">
<s:url action="events" var="editEvent">
<s:param name="event.event_id" value="event.
getEvent_id()" />
<s:param name="event.name" value="event.getName()"/>
<s:param name="event.description" value="event.
getDescription()" />
<s:param name="event.privacy" value="event.getPrivacy()"/>
<s:param name="mode" value="'editPage'" />
</s:url>
[<s:a href="%{editEvent}" data-toggle="modal"
data-target="#myeditdialog" data-backdrop="static"
data-keyboard="false">edit</s:a>]
</s:if>
</h1>
<br><br>
<h4>You have to supply the following before you can
start this event</h4><br>
<table class="table table-striped table-advance
table-hover" id="datazexample">
<tbody>
<tr>
<td>Elections</td>
<td>
<s:if test="checker[0] == 1">
&#10004;
</s:if>
<s:else>
 
</s:else>
</td>
</tr>
<tr>
<td>Blocks</td>
<td>
<s:if test="checker[1] == 1">
&#10004;
</s:if>
<s:else>
 
</s:else>
</td>
</tr>
<tr>
<td>Positions</td>
<td>
<s:if test="checker[2] == 1">
&#10004;
</s:if>
<s:else>
 
</s:else>
</td>
</tr>
<tr>
<td>Candidates</td>
<td>
<s:if test="checker[3] == 1">
&#10004;
</s:if>
<s:else>
 
</s:else>
</td>
```

```
</tr>
<tr>
<td>Voters</td>
<td>
<s:if test="checker[4] == 1">
&#10004;
</s:if>
<s:else>
 
</s:else>
</td>
</tr>
</tbody>
</table>

<br>
<s:if test="%{event.getStatus() == 'IDLE'}">
<s:if test="checker[5] == 5">
<s:url action="events" var="urlTag" >
<s:param name="event.event_id" value="
%{#e.event_id}"></s:param>
<s:param name="mode" value="'startE'">
</s:param>
</s:url>
<s:a cssClass="btn btn-primary" href="%{urlTag}"
onclick="return confirm('Are you sure? Once you start,
you can no longer make some modifications')"
title="Start this event">
Start this event
</s:a>
<!--<s:a href="elections" cssClass="btn btn-danger">
View Sample Ballot</s:a>-->
</s:if>
<s:else>
<a class="btn btn-primary disabled" href=""
title="You cannot start the event yet">
Start this event
</a>
</s:else>
</s:if>
<s:elseif test="%{event.getStatus() == 'RUNNING'}">
<s:url action="events" var="urlTag" >
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'stopE'"></s:param>
</s:url>
<s:a cssClass="btn btn-primary" href="%{urlTag}"
onclick="return confirm('Are you sure?')"
title="Stop this event">
Stop this event
</s:a>
</s:elseif>
<s:elseif test="%{event.getStatus() == 'FINISHED'}">
<s:url action="ticket" var="tickTag" >
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'tvo'"></s:param>
</s:url>
<s:a cssClass="btn btn-primary pull-right"
href="%{tickTag}" title="View Tickets">
View Tickets
</s:a>
<h3>This event is over</h3><br>
<s:url action="events" var="urlTag2" >
<s:param name="event.event_id" value="
%{event.getEvent_id()}"></s:param>
<s:param name="mode" value="'archivE'">
</s:param>
</s:url>
<s:a cssClass="btn btn-primary" href="%{urlTag2}"
onclick="return confirm('Are you sure?')"
title="Archive this event">
Archive this event
</s:a>
</s:elseif>
<s:else>
<h3>This event has been archived</h3><br>
</s:else>

</div><!-- /col-lg-9 END SECTION MIDDLE -->


<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myeditdialog"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">

</div>
</div>
</div>

<!-- modal -->

<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myeditdialog2"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<s:form action="events" method="post" theme="simple">

<div class="modal-header">
<button type="button" class="close" data-dismiss="modal"
aria-hidden="true">&times;</button>
<h4 class="modal-title">Edit Event</h4>
</div>
<div class="modal-body">
<p>Event Name</p>
<s:textfield name="event.name" value="%{event.getName()}"
cssClass="form-control" required="required"/><br>
<s:fielderror fieldName="event.name" cssClass="myerror" />
<br>

<p>Description</p>
<s:textarea name="event.description" value="%{event.
getDescription()}" cssClass="form-control"
required="required" /><br>
<s:fielderror fieldName="event.name"
cssClass="myerror" /><br>

<p>Privacy (If public, election results would be
publicly available)</p>
<s:radio name="event.privacy" list="#{'0':'Public',
'1':'Private'}" value="%{event.getPrivacy()}" r
equired="required" /><br>

<s:hidden name="event.event_id" value="%{event.
getEvent_id()}" />
<s:hidden name="mode" value="edit" />

</div>
<div class="modal-footer">
<button data-dismiss="modal" class="btn btn-default"
type="button">Cancel</button>
<s:submit value="Submit" cssClass="btn btn-theme"/>
</div>
</s:form>
</div>
</div>
</div>
<!-- modal -->

</div><! --/row -->
</section>
</section>

<!--main content end-->
<!--footer start-->

<!--footer end-->
</section>

<s:include value="/pages/includes/scripts.jsp"/>

<s:if test="hasError[1] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myeditdialog2').modal('show');
});
</script>
</s:if>

<script>
function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>

</body>
</html>
```

event-keys.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>

<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>

<aside>
```

```
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:url
value='/dashgum/assets/img/logo.png'/>"></a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}">${sessionScope.event.name}
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="board" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<!--main content start-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3><s:property value="event.getName()" />
</h3>

<div class="row mt">
<div class="col-md-12">
<div class="content-panel">
<h1> The event has started</h1>
<%-- <s:iterator value="elections" var="e"
status="c">
<h3><s:property value="%{#e.name}" /></h3>


<s:url id="keyDownload" action="key_download" >
<s:param name="filename" value="
%{sk_filenames[#c.index]}">
</s:param>
</s:url>
<h5>Download secret key file <s:a href="
%{keyDownload}">
<s:property value="%{sk_filenames[#c.index]}" />
</s:a></h5>

<br><br>
</s:iterator> --%>
</div><!-- /content-panel -->
</div><!-- /col-md-12 -->
</div><!-- /row -->
</div><!-- /col-lg-9
END RIGHT SIDEBAR CONTENT
</div><! --/row -->
</section>
</section>

<!--main content end-->
<!--footer start-->

<!--footer end-->
</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
</script>

</body>
</html>
```

head.jsp

```
<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<%
response.setHeader("Pragma","no-cache");
response.setHeader("Cache-Control","no-store");
response.setHeader("Expires", "0");
response.setDateHeader("Expires", -1);
%
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<meta name="description" content="">
<meta name="author" content="Dashboard">
<meta name="keyword" content="Dashboard,
Bootstrap, Admin, Template, Theme, Responsive,
Fluid, Retina">

<title>Voterify</title>

<!-- Bootstrap core CSS -->
<link href="<s:url value='/dashgum/assets/css/
bootstrap.css'/>" rel="stylesheet">
<!--external css-->
<link href="<s:url value='/dashgum/assets/
font-awesome/css/font-awesome.css'/>"
rel="stylesheet">


<!-- Custom styles for this template -->
<link href="<s:url value='/dashgum/assets/
css/style.css'/>" rel="stylesheet">
<link href="<s:url value='/dashgum/assets/
css/style-responsive.css'/>" rel="stylesheet">

<link href="https://cdn.datatables.net/1.10.11/
css/dataTables.bootstrap.min.css" rel="stylesheet">

<link rel="icon" type="image/gif" href="
<s:url value='/dashgum/assets/img/logo-s.png'/>" />

<!-- HTML5 shim and Respond.js IE8 support of HTML5 e
lements and media queries -->
<!--[if lt IE 9]>
<script src="https://oss.maxcdn.com/libs/html5shiv/
3.7.0/html5shiv.js"></script>
<script src="https://oss.maxcdn.com/libs/respond.js/
1.4.2/respond.min.js"></script>
<![endif]-->
</head>
```

header.jsp

```
<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
```

```
pageEncoding="ISO-8859-1"%>
<%@taglib uri="/struts-tags" prefix="s"%>

<header class="header black-bg">
<div class="sidebar-toggle-box">
<a href="home"><img src="<s:url value='/
dashgum/assets/img/logo-s.png'/>"></a>
</div>
<!--logo start-->
<a href="home" class="logo"><b>VOTERIFY</b>
</a>
<!--logo end-->

<div class="top-menu">
<ul class="nav pull-right top-menu">
<li><a class="logout" href="account">
Profile</a></li>
<li><a class="logout" href="login">
LOGOUT</a></li>
</ul>
</div>
</header>
```

hello.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<%
response.setHeader("Pragma","no-cache");
response.setHeader("Cache-Control","no-store");
response.setHeader("Expires", "0");
response.setDateHeader("Expires", -1);
%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/
loose.dtd">
<html>
<head>
<title>Voterify</title>
</head>
<body>
<s:if test="role=='VOTING_OFFICIAL'">
<s:include value="official/home_vo.jsp"/>
</s:if>
<s:elseif test="role=='VOTER'">
<s:include value="voter/home_v.jsp"/>
</s:elseif>
<s:elseif test="role=='ADMIN'">
<s:include value="admin/home_a.jsp"/>
</s:elseif>
<s:else>
<s:include value="login.jsp"/>
</s:else>


</body>
</html>
```

home-a.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>
<section id="container">
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar" class="nav-collapse">
<ul class="sidebar-menu" id="nav-accordion">
<p class="centered"><a href=""><img src="
<s:url value='/dashgum/assets/img/logo.png'/>">
</a></p>
<li class="mt">
<a class="active" href="home">
<i class="fa fa-dashboard"></i>
<span>Home</span>
</a>
</li>
<li class="sub-menu">
<a href="voting_officials">
<i class="fa fa-desktop"></i>
<span>Voting Officials</span>
</a>
</li>
</ul>
</div>
</aside>
<section id="main-content">
<section class="wrapper">
<div class="row">
<div class="col-lg-12 main-chart">
<h2>Welcome to Voterify!</h2>
<h3>Good day, administrator!</h3>
</div>
</div><! --/row -->
</section>
</section>
</section>
<s:include value="/pages/includes/scripts.jsp"/>
</body>
</html>
```

home-vo.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>

<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>

<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-9 main-chart">


<b><h1>Welcome to Voterify!</h1></b><br>

<h3>Voterify provides private and verifiable
elections.</h3><br>
<h3>To build elections, you have to create
an event.
An event is <br> composed of one major
election and zero
or more minor elections.</h3>
<h4>Sample Events: University Student
Council Election,
Club Election 2016, etc. </h4>
<br><br><br>
<button class="btn btn-success btn-lg"
data-toggle="modal"
data-target="#myModal"><i class="fa fa-plus"></i>
Create Event</button>
<br><br><br><br><br><br><br><br>
</div>

<s:form action="events" theme="simple">
<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog"
tabindex="-1" id="myModal" class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<s:url action="events" var="lolURL">
<s:param name="mode" value="'homeE'" />
</s:url>
<s:a href="home" cssClass="close">&times;</s:a>
<h4 class="modal-title">Create Event</h4>
</div>
<div class="modal-body">
<p>Event Name</p>
<s:textfield name="eventForm.name" cssClass="
form-control"
required="required" /><br>
<s:fielderror fieldName="eventForm.name"
cssClass="myerror" /><br>
<p>Description</p>
<s:textarea name="eventForm.description"
cssClass="form-control"
required="required" /><br>
<s:fielderror fieldName="eventForm.description"
cssClass="myerror" /><br>
<p>Privacy (If public, election results would be
publicly available)</p>
<s:radio name="eventForm.privacy" list="#{'0':'Public',
'1':'Private'}" value="0" /><br>

<s:hidden name="mode" value="add" />
</div>
<div class="modal-footer">
<button data-dismiss="modal" class="btn btn-default"
type="button">Cancel</button>
<s:submit value="Submit"
cssClass="btn btn-theme"/>
</div>
</div>
</div>
</div>
<!-- modal -->
</s:form>
```

```
<div class="col-lg-3 ds">
<!--COMPLETED ACTIONS DONUTS CHART-->
<h3>EVENTS YOU MANAGE</h3>

<s:iterator value="events" var="e">
<s:if test='%{!#e.status.equals("ARCHIVED")}'>
<div class="desc">
<div class="thumb">
<s:if test='%{#e.status.equals("IDLE")}'>
<span class="badge bg-theme02">
</s:if>
<s:elseif test='%{#e.status.equals("RUNNING")}'>
<span class="badge bg-theme03">
</s:elseif>
<s:elseif test='%{#e.status.equals("FINISHED")}'>
<span class="badge bg-theme04">
</s:elseif>
<i class="fa fa-clock-o"></i>
</span>
</div>

<s:url action="events" var="viewURL" escapeAmp="false">
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'gteh'"></s:param>
</s:url>
<s:a href="%{viewURL}">
<s:property value="%{#e.name}" />
</s:a>

</div>
</s:if>
</s:iterator>
<br><br>
<span class="badge bg-theme02"> - IDLE </span>
<span class="badge bg-theme03"> - RUNNING </span>
<span class="badge bg-theme04"> - FINISHED </span>
<br><br><br><br>

<h4>Archived Events</h4>
<s:iterator value="events" var="e">
<div class="desc">
<s:if test='%{#e.status.equals("ARCHIVED")}'>
<div class="thumb">
<span class="badge bg-theme">
<i class="fa fa-clock-o"></i>
</span>
</div>
<font color="black">    
<s:property value="%{#e.name}" /></font>
</s:if>
</div>
</s:iterator>
</div>


</div><
<br><br><br><br><br>

</div><! --/row -->
</section>
</section>

<!--main content end-->
<!--footer start-->
<!--footer end-->
</section>

<!-- JS section -->
<s:include value="/pages/includes/scripts.jsp"/>

<s:if test="addr == true">
<script class="include" type="text/javascript"
src="<s:url value='dashgum/assets/js/
jquery.dcjqaccordion.2.7.js' />"></script>
<script src="<s:url value='dashgum/assets/
js/gritter/js/jquery.gritter.js' />"></script>
<script src="<s:url value='dashgum/assets/
js/gritter-conf.js' />"></script>

<script type="text/javascript">
$(document).ready(function () {
var unique_id = $.gritter.add({
title: 'EVENT ADDED!',
text: 'Click the event to fill it with election/s'
image: 'dashgum/assets/img/logo.png',
sticky: true,
time: '',
class_name: 'my-sticky-class'
});

return false;
});
</script>
</s:if>

<s:if test="hasError[0] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myModal').modal('show');
});
</script>
</s:if>

</body>
</html>
```

input-sk.jsp

```
<html lang="en">

<s:include value="/pages/includes/head.jsp"/>
<body>
<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:url
value='/dashgum/assets/img/logo.png'/>"></a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}"><s:property value="
event.getName()" /></s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="board" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a class="active" href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>

<section id="main-content">
```

```
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">


<h2><s:property value="%{election.getName()}" /></h2>

<form action="upload_key" method="post"
enctype="multipart/form-data">
<label for="myFile">Upload the secret key
of this election</label>
<input type="file" name="myFile" /><br><br><br>

<s:hidden name="election.election_id"
value="%{election.getElection_id()}" />
<button type="submit" id="buttonSelector"
class="btn btn-primary"
data-loading-text="<span class='glyphicon-left
 glyphicon glyphicon-refresh spinning'></span>...
 Tallying ...">
Compute Tally
</button>
</form>

</div><!-- /col-md-12 -->
</div><!-- /row -->
<s:if test="errorDel == false">
<script>
alert("You cannot delete an election in use by possibly
candidates, ...");
</script>
</s:if>
</div><!-- /col-lg-9 END SECTION MIDDLE -->

</div><! --/row -->
</section>
</section>

<br><br><br><br><br><br><br><br><br><br><br><br><br>
<!--main content end-->
<!--footer start-->

<!--footer end-->
</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
</script>

<script>
$("#buttonSelector").click(function ()
{
$(this).button('loading');
// Long waiting operation here
$(this).button('reset');
}
);
</script>


</body>
</html>
```

login.jsp

```
<!DOCTYPE html>
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>
<br><br>
<div align="center">
<img src="<s:url value='/dashgum/assets/img/
Drawing.png'/>"><br><br>
<h4>Privacy-preserving and verifiable elections
</h4>
</div>
<div id="login-page">
<div class="container">
<s:form cssClass="form-login" action="rdr"
theme="simple">
<div class="login-wrap">
<div class="myerror">  <s:actionerror/>
</div>
<s:textfield name="user.username"
cssClass="form-control"
placeholder="Username" />
<s:fielderror fieldName="user.username"
cssClass="myerror" />
<br>
<s:password name="user.password"
cssClass="form-control"
placeholder="Password" />
<s:fielderror fieldName="user.password"
cssClass="myerror" />
<label class="checkbox">
<span class="pull-right">
</span>
</label>
<button class="btn btn-facebook btn-block"
type="submit">
<i class="fa fa-lock"></i> LOG IN</button>
<hr>
<div class="registration">
Want to create elections?<br/>
<s:a href="signup_page"> Create an account</s:a>
</div>

</div>
</s:form>
<br><br><br>
<div class="login-social-link centered">
<h5>
<a href="public_events">Public Events</a>
   &bull;   
<a href="about_page">About Voterify</a>
</h5>
</div>

<!-- Modal: Forget password -->
<div aria-hidden="true"
aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myModal"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<button type="button" class="close"
data-dismiss="modal"
 aria-hidden="true">&times;</button>
<h4 class="modal-title">Forgot Password ?</h4>
</div>
<div class="modal-body">
<h1>Under Construction</h1>
</div>
</div>
</div>
</div>
<!-- modal -->

</div>
</div>

<s:include value="/pages/includes/scripts.jsp"/>
</body>
</html>
```

parties.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>

<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img
src="<s:url value='/dashgum/assets/img/logo.png'/>">
</a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}">${sessionScope.event.name}
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
```

```
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a class="active" href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="#" onclick="opennewtab('board')" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3>${sessionScope.event.name}</h3>

<div class="row mt">
<div class="col-md-12">
<div class="content-panel">
<table id="datazexample" class="table
table-striped table-advance table-hover">
<h4><i class="fa fa-angle-right"></i> Party
<s:if test='%{event.getStatus().equals("IDLE")}'>
<div class="pull-right">
<button class="btn btn-primary" data-toggle="modal"
data-target="#myModal"><i class="fa fa-plus"></i>
Add Party</button>
   
</div>
</s:if>
</h4>
<hr>
<thead>
<tr>
<th><i class="fa fa-bullhorn"></i> Name</th>
<th class="hidden-phone"><i class="fa fa-question-circle">
</i> Action</th>
</tr>
</thead>
<tbody>
<s:iterator value="parties" var="p">
<tr>
<td><s:property value="#p.name" /></td>
<td>
<s:if test='%{event.getStatus().equals("IDLE")}'>
<s:url action="parties" var="editURL" escapeAmp="false">
<s:param name="party.party_id" value="%{#p.party_id}">
</s:param>
<s:param name="party.name" value="%{#p.name}">
</s:param>
<s:param name="mode" value="'editPage'">
</s:param>
</s:url>
<s:a href="%{editURL}" data-toggle="modal"
data-target="#myeditdialog" data-backdrop="static"
data-keyboard="false"
cssClass="btn btn-primary btn-xs"><i
class="fa fa-pencil"></i></s:a>

<s:url action="parties" var="deleteURL" >
<s:param name="party.party_id" value="%{#p.party_id}"></s:param>
<s:param name="mode" value="'delete'"></s:param>
</s:url>
<s:a href="%{deleteURL}" cssClass="btn
btn-danger btn-xs"
onclick="return confirm('Are you sure
you want to delete this party?')">
<i class="fa fa-trash-o"></i></s:a>
</s:if>
<s:else> </s:else>
</td>
</tr>
</s:iterator>

</tbody>
</table>
</div><!-- /content-panel -->
</div><!-- /col-md-12 -->
</div><!-- /row -->


<s:form action="parties" theme="simple">
<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myModal" class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<s:url action="elections" var="lolURL" />
<s:a href="%{lolURL}" cssClass="close">&times;</s:a>
<h4 class="modal-title">Add Party</h4>
</div>
<div class="modal-body">
<p>Party Name</p>
<s:textfield name="party.name" requiredLabel="true"
cssClass="form-control" /><br>
<s:fielderror fieldName="party.name" cssClass="myerror" /><br>
<s:hidden name="mode" value="add" />
</div>
<div class="modal-footer">
<s:url action="parties" var="lolURL" />
<s:a href="%{lolURL}" cssClass="btn btn-default">Cancel</s:a>
<s:submit value="Submit" cssClass="btn btn-theme"/>
</div>
</div>
</div>
</div>
<!-- modal -->
</s:form>


<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myeditdialog"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
</div>
</div>
</div>
<!-- modal -->

<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myeditdialog2"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<s:form action="parties" method="post" theme="simple">

<div class="modal-header">
<s:url action="elections" var="lolURL" />
<s:a href="%{lolURL}" cssClass="close">&times;</s:a>
<h4 class="modal-title">Edit Party</h4>
</div>
<div class="modal-body">
<p>Party Name</p>
<s:textfield name="party.name" value="%{party.getName()}"
cssClass="form-control" required="required" /><br>
<s:fielderror fieldName="party.name"
cssClass="myerror" /><br>
<s:hidden name="party.party_id"
value="%{party.getParty_id()}" />
<s:hidden name="mode" value="edit" />
```

```
</div>
<div class="modal-footer">
<s:url action="parties" var="lolURL" />
<s:a href="%{lolURL}"
cssClass="btn btn-default">Cancel</s:a>
<s:submit value="Submit"
 cssClass="btn btn-theme"/>
</div>

</s:form>
</div>
</div>
</div>
<!-- modal -->


<s:if test="errorDel == false">
<script>
alert("You cannot delete an election in
use by positions, candidates, ...");
</script>
</s:if>

<script>
function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>


</div><!-- /col-lg-9 END SECTION MIDDLE -->


</div><! --/row -->
</section>
</section>

<!--main content end-->

</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
</script>

<s:if test="hasError[0] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myModal').modal('show');
});
</script>
</s:if>
<s:if test="hasError[1] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myeditdialog2').modal('show');
});
</script>
</s:if>


</body>
</html>
```

positions.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:url
value='/dashgum/assets/img/logo.png'/>"></a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="%{#e.
event_id}"></s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}"><s:property value="event.
getName()" /></s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a class="active" href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="#" onclick="opennewtab('board')" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3>${sessionScope.event.name}</h3>

<div class="row mt">
<div class="col-md-8">


<s:form action="positions" method="post"
theme="simple">

<%-- <s:select headerKey="-1" headerValue="
Choose an election" name="election.election_id"
list="elections" listKey="election_id"
listValue="name" onchange="handleChange(this.value)"
cssClass="form-control m-bot8" /> --%>
<s:select headerKey="-1" headerValue="Choose an election"
name="election.election_id"
list="elections" listKey="election_id" listValue="name"
value="%{election.getElection_id()}"
cssClass="form-control" onchange="form.submit()" />


<s:hidden name="mode" value="vpe" />
</s:form>
</div>

<s:if test="%{election.getElection_id() != -1}">
<div class="col-md-12">

<div class="content-panel">

<table id="datazexample" class="table table-striped
table-advance table-hover">
```

```
<h4><i class="fa fa-angle-right"></i> Positions
<s:if test='%{event.getStatus().equals("IDLE")}'>
<div class="pull-right">
<button class="btn btn-primary" data-toggle="modal"
data-target="#myModal"><i class="fa fa-plus"></i>
 Add Position</button>
   
</div>
</s:if>
</h4>
<hr>
<thead>
<tr>
<th><i class="fa fa-bullhorn"></i> Order</th>
<th><i class="fa fa-bullhorn"></i> Name</th>
<th><i class="fa fa-bullhorn"></i> Candidates</th>
<th class="hidden-phone"><i class="fa
fa-question-circle"></i> Action</th>
</tr>
</thead>
<tbody>
<s:iterator value="positions" var="p">
<tr>
<td><s:property value="%{#p.ballot_order}" /></td>
<td><s:property value="%{#p.name}" /></td>
<td>
<s:url var="candURL" action="candidates"
escapeAmp="false">
<s:param name="position.position_id"
value="%{#p.position_id}"></s:param>
<s:param name="position.election_id"
value="%{election.getElection_id()}"></s:param>
<s:param name="mode" value="'cand'"></s:param>
</s:url>
<s:a href="%{candURL}">[View]</s:a>
</td>
<td>
<s:if test='%{event.getStatus().equals("IDLE")}'>
<s:url action="positions" var="editURL"
escapeAmp="false">
<s:param name="position.position_id"
value="%{#p.position_id}"></s:param>
<s:param name="position.name"
value="%{#p.name}"></s:param>
<s:param name="election.election_id"
value="%{#p.election_id}"></s:param>
<s:param name="position.res_block_id"
value="%{#p.res_block_id}"></s:param>
<s:param name="position.ballot_order"
value="%{#p.ballot_order}"></s:param>
<s:param name="position.required_min"
value="%{#p.required_min}"></s:param>
<s:param name="position.required_max"
value="%{#p.required_max}"></s:param>
<s:param name="mode" value="'editPage'"></s:param>
</s:url>
<s:a href="%{editURL}" data-toggle="modal" data-target=
"#myeditdialog" data-backdrop="static" data-keyboard="false"
cssClass="btn btn-primary btn-xs"><i class="fa fa-pencil">
</i></s:a>

<s:url action="positions" var="deleteURL">
<s:param name="position.position_id" value="%{#p.position_id}">
</s:param>
<s:param name="election.election_id" value="%{#p.election_id}">
</s:param>
<s:param name="mode" value="'delete'"></s:param>
</s:url>
<s:a href="%{deleteURL}" cssClass="btn btn-danger btn-xs"
onclick="return confirm('Are you sure you want to delete
 this position?')">
<i class="fa fa-trash-o"></i></s:a>

</s:if>
<s:else> </s:else>
</td>
</tr>
</s:iterator>
</tbody>
</table>
<s:url action="positions" var="dadURL">
<s:param name="mode" value="'dad'" />
<s:param name="election.election_id" value="%{election.
getElection_id()}" />
</s:url>
<s:if test="%{positions.size() > 0}">
<s:a href="%{dadURL}" cssClass="btn btn-primary">Click
to reorder positions</s:a>
</s:if>
</div><!-- /content-panel -->
</div><!-- /col-md-12 -->
</s:if>

</div><!-- /row -->
<s:form action="positions" theme="simple">
<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myModal" class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<s:url action="positions" var="lolURL">
<s:param name="mode" value="'vpe'" />
<s:param name="election.election_id"
value="%{election.getElection_id()}" />
</s:url>
<s:a href="%{lolURL}" cssClass="close">&times;</s:a>
<h4 class="modal-title">Add Position</h4>
</div>
<div class="modal-body">
<p>Position Name</p>
<s:textfield name="position.name" cssClass=
"form-control" required="required" /><br>
<s:fielderror fieldName="position.name"
cssClass="myerror" /><br>

<p>Minimum Number of Candidates to be Voted (If 0,
abstain is allowed)</p>
<s:textfield name="str_min" cssClass="form-control" /><br>
<s:fielderror fieldName="str_min" cssClass="myerror" /><br>
<p>Maximum Number of Candidates to be Voted</lp>
<s:textfield name="str_max" cssClass="form-control" /><br>
<s:fielderror fieldName="str_max" cssClass="myerror" /><br>

<p>Is this position exclusive to be voted by a
specific block? If yes, what block?</p>
<s:select headerKey="1" headerValue="No" list="blocks"
listKey="block_id" listValue="name"
name="position.res_block_id" cssClass="form-control
m-bot15" required="required"/><br><br>

<s:hidden name="election.election_id"
value="%{election.getElection_id()}" />
<s:hidden name="position.election_id"
value="%{election.getElection_id()}" />
<s:hidden name="mode" value="add" />
</div>
<div class="modal-footer">
<s:url action="positions" var="lolURL">
<s:param name="mode" value="'vpe'" />
<s:param name="election.election_id"
value="%{election.getElection_id()}" />
</s:url>
<s:a href="%{lolURL}"
cssClass="btn btn-default">Cancel</s:a>
<s:submit value="Submit"
cssClass="btn btn-theme"/>
</div>
</div>
</div>
<!-- modal -->
</s:form>

<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myeditdialog"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">

</div>
</div>
</div>
<!-- modal -->

<!-- Modal -->
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myeditdialog2"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<s:form action="positions" method="post" theme="simple">
<div class="modal-header">
<h4 class="modal-title">Edit Position</h4>
</div>
<div class="modal-body">
<p>Position Name</p>
<s:textfield name="position.name"
value="%{position.getName()}" cssClass="form-control"
```

```
required="required" /><br>
<s:fielderror fieldName="position.name"
cssClass="myerror" /><br>

<p>Number of Candidates to be Chosen (Minimum)
*If 0, abstain is allowed</p>
<s:textfield name="str_min" value="%{position.
getRequired_min()}" cssClass="form-control" /><br>
<s:fielderror fieldName="str_min" cssClass="myerror" /><br>
<p>Number of Candidates to be Chosen (Maximum)</lp>
<s:textfield name="str_max" value="%{position.
getRequired_max()}" cssClass="form-control" /><br>
<s:fielderror fieldName="str_max" cssClass="myerror" /><br>

<p>Is this position exclusive to be voted by a
specific block? If yes, what block?</p>
<s:select headerKey="1" headerValue="No" list="blocks"
listKey="block_id" listValue="name"
name="position.res_block_id" cssClass="form-control mt-
bot15" required="required"/><br><br>

<s:hidden name="election.election_id" value="%{election.getElection_id()}" />
<s:hidden name="position.election_id" value="%{election.getElection_id()}" />
<s:hidden name="position.position_id" value="%{position.getPosition_id()}" />
<s:hidden name="mode" value="edit" />
</div>
<div class="modal-footer">
<s:url action="positions" var="lolURL">
<s:param name="mode" value="'vpe'" />
<s:param name="election.election_id" value="%{election.
getElection_id()}" />
</s:url>
<s:a href="%{lolURL}" cssClass="btn btn-default">Cancel</s:a>
<s:submit value="Submit" cssClass="btn btn-theme"/>
</div>

</s:form>
</div>
</div>
</div>
<!-- modal -->


<s:if test="errorDel == false">
<script>
alert("You cannot delete an election in use by blocks,
positions, and candidates.");
</script>
</s:if>


</div><!-- /col-lg-9 END SECTION MIDDLE -->


</div><! --/row -->
</section>
</section>

<!--main content end-->
<!--footer start-->

</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})

function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>

<s:if test="hasError[0] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myModal').modal('show');
});
</script>
</s:if>
<s:if test="hasError[1] == true">
<script type="text/javascript">
$(window).load(function(){
$('#myeditdialog2').modal('show');
});
</script>
</s:if>
```

positions2.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="
<s:url value='/dashgum/assets/img/logo.png'/>"></a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}"><s:property value="event.getName()" />
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a class="active" href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="#" onclick="opennewtab('board')" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3>${sessionScope.event.name}</h3>

<div class="row mt">
<div class="col-md-8">


</div>
```

```
<s:if test="%{election.getElection_id() != -1}">
<div class="col-md-12">

<div class="content-panel">

<table id="datazexample" class="table
table-striped table-advance table-hover">
<h4><i class="fa fa-angle-right"></i> Positions
<s:if test='%{event.getStatus().equals("IDLE")}'>

</s:if>
</h4>
<hr>
<thead>
<tr>
<th><i class="fa fa-bullhorn"></i> Order</th>
<th><i class="fa fa-bullhorn"></i> Name</th>
</tr>
</thead>
</table>
<div class="task-content">
<ul id="sortable" class="task-list">
<s:iterator value="positions" var="p" status="ctr">
<li class="list-primary" id='<s:property value="
%{#ctr.count}" />'>
<i class=" fa fa-ellipsis-v"></i>
<div class="task-title">
  <span class="task-title-sp"><s:property value="
  %{#p.name}" /></span>
</div>
</li>
</s:iterator>
</ul>
</div>


<s:url action="positions" var="lolURL">
<s:param name="mode" value="'vpe'" />
<s:param name="election.election_id" value="%{
election.getElection_id()}" />
</s:url>
<s:if test="%{positions.size() > 0}">
<s:a href="%{lolURL}" cssClass="btn btn-primary">
Cancel</s:a>  
<input id="save-dad" type="button" class="btn btn-primary"
value="Save" />
<p id="dragger" style="visibility: hidden;">
<s:property value="%{election.getElection_id()}" /></p>
</s:if>
<br><br>
</div><!-- /content-panel -->
</div><!-- /col-md-12 -->
</s:if>
</div><!-- /row -->

</div><!-- /col-lg-9 END SECTION MIDDLE -->

</div><! --/row -->
</section>
</section>

<!--main content end-->
<!--footer start-->

</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script class="include" type="text/javascript"
src="<s:url value='dashgum/assets/js/jquery.
dcjqaccordion.2.7.js' />"></script>

<script>
function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>

<script src="http://code.jquery.com/ui/1.10.3/j
query-ui.js"></script>
<script src="<s:url value='dashgum/assets/js/
tasks.js' />"></script>

<script>
jQuery(document).ready(function() {
TaskList.initTaskWidget();
});

$('#save-dad').click(function() {
var tempData = $('#sortable').sortable('toArray');
var myData = tempData.toString();
var eid = document.getElementById("dragger").innerHTML;
//alert(myData);

$.ajax({
type : "POST",
url : "./reorder_positions",
data: { newPositions: myData,
eid: eid
},

success : function(msg) {
window.location.href = "positions.action?
election.election_id="+eid+"&mode=vpe";
},
error: function (thrownError) {
alert("error");
}

});
});


$(function() {
$( "#sortable" ).sortable();
$( "#sortable" ).disableSelection();
});

</script>


</body>
</html>
```

publicevents.jsp

```
<body>

<br><br>
<div id="login-page">
<div class="container">
<h2>Public Events</h2><br>

<s:iterator value="events" var="e">
<h3><s:property value="%{#e.name}" />
</h3>
<s:url var="pbe"
action="public_elections" >
<s:param name="event.event_id"
value="%{#e.event_id}">
</s:param>
</s:url>
<s:a href="%{pbe}"
cssClass="btn btn-primary">
View results in PDF</s:a>
  
<s:url action="board" var="viewbURL">
<s:param name="event.event_id"
value="%{#e.event_id}" />
</s:url>
<s:a href="%{viewbURL}"
cssClass="btn btn-primary">
View Bulletin Board</s:a><br>
<br>
</s:iterator>
</div>
</div>
<script src="<s:url value='
dashgum/assets/
js/jquery.js' />"></script>
<script src="<s:url value='
dashgum/assets/
js/bootstrap.min.js' />"></script>

</body>
</html>
```

publicevents2.jsp

```
<body>
<br><br>
<div id="login-page">
<div class="container">
<h2>Public Events</h2><br>

<s:iterator value="elections" var="e">
<h3><s:property value="%{#e.name}" />
</h3>
<s:if test="%{#e.tally_stat == 1}">
<s:url var="pdfDownload"
action="key_download" >
<s:param name="filename"
value="%{#e.getTemp()}">
</s:param>
```

```
</s:url>
<s:a href="%{pdfDownload}"
cssClass="btn btn-primary">
Download results</s:a>
</s:if>
</s:iterator>
</div>
</div>

<script src="<s:url value='
dashgum/assets/
js/jquery.js' />"></script>
<script src="<s:url value='
dashgum/assets/
js/bootstrap.min.js' />">
</script>

</body>
</html>
```

result-aftervote.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:url
value='/dashgum/assets/img/logo.png'/>"></a></p>

<li class="mt">
<a href="home">
<i class="fa fa-dashboard"></i>
<span>Home</span>
</a>
</li>

<li class="sub-menu">
<a class="active" href="vote">
<i class="fa fa-desktop"></i>
<span>Vote</span>
</a>
</li>

<li class="sub-menu">
<a href="#" onclick="opennewtab('board')">
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3>Below are your votes with their hashes</h3>
<s:iterator value="elections" var="e" status="counter">
<b><s:property value="%{#e.name}" /></b>
<br>
<s:iterator value="positions.get(#counter.index)"
var="p" status="counter2">
<br>
<u><s:property value="%{#p.name}" /></u>
<s:if test="%{#p.required_max == 1}">
<br><s:property value="%{singlePos[#counter.index]
[#counter2.index]}" />
</s:if>
<s:else>
<br><s:property value="%{multiplePos[#counter.index]
[#counter2.index]}" />
</s:else>
   |   
<s:property value="%{evote[#counter.index
counter2.index]}" />
</s:iterator><br><br>
</s:iterator>

<br>
<b>Verification Code:</b> <s:property
value="%{verificationCode}" />
<br>
<s:url id="hashDownload" action="key_download" >
<s:param name="filename" value="%{filename}"></s:param>
</s:url>
<h5>Download the hashes (receipt) <s:a href="
%{hashDownload}">here</s:a></h5>
<br>
You can vote as many times as you want<br>
But your last one is what we really count

</div><!-- /col-lg-9 END SECTION MIDDLE -->

</div><! --/row -->
</section>
</section>

<!--main content end-->

</section>

<s:include value="/pages/includes/scripts.jsp"/>
<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
</script>
<script>
function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>

</body>
</html>
```

results-home.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>

<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:url
value='/dashgum/assets/img/logo.png'/>"></a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}"><s:property value="event.getName()" />
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
```

```
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="board" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a class="active" href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">
<h1>Finished Elections</h1>

<s:if test='%{event.getStatus().
equals("FINISHED")}'>
<s:iterator value="elections" var="e">
<h4>
<s:property value="%{#e.name}" />  
<s:url action="results" var="resultsURL" >
<s:param name="election.election_id"
value="%{#e.election_id}"></s:param>
<s:param name="election.name"
value="%{#e.name}"></s:param>
<s:param name="mode" value="'
computePage'"></s:param>
</s:url>
<s:if test="%{#e.tally_stat == 0}">
<s:a href="%{resultsURL}" cssClass="btn btn-danger"
onclick="return confirm('Are you sure?')">
<i class="icon_plus_alt2"></i>  Compute Tally</s:a>
</s:if>
<s:else>
<s:a href="" cssClass="btn btn-danger">
<i class="icon_plus_alt2"></i>  View Results</s:a>
</s:else>
</h4>
</s:iterator>
</s:if>
<s:else>
<h3>Cannot compute tally yet since the event
is not yet finished</h3>
</s:else>

</div><!-- /col-md-12 -->
</div><!-- /row -->
<s:if test="errorDel == false">
<script>
alert("You cannot delete an election in use by
positions, candidates, ...");
</script>
</s:if>


</div><!-- /col-lg-9 END SECTION MIDDLE -->

</div><! --/row -->
</section>
</section>

<br><br><br><br><br><br><br><br><br><br><br><br>
<!--main content end-->
<!--footer start-->

<!--footer end-->
</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
</script>

</body>
</html>
```

scripts.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@taglib uri="/struts-tags" prefix="s"%>

<!--common script for all pages-->
<script src="<s:url value='/dashgum/assets/
js/common-scripts.js' />"></script>

<script src="<s:url value='/dashgum/assets/
js/jquery.js' />"></script>
<script src="<s:url value='/dashgum/assets/
js/bootstrap.min.js' />"></script>
<script src="<s:url value='/dashgum/assets/
js/jquery.nicescroll.js' />"></script>
<script src="<s:url value='/dashgum/assets/
js/jquery.scrollTo.min.js' />"></script>

<script src="https://cdn.datatables.net/
1.10.11/js/jquery.dataTables.min.js"></script>
<script src="https://cdn.datatables.net/
1.10.11/js/dataTables.bootstrap.min.js"></script>
```

ticket.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>

<!--main content start-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">


<h1>Raise a Ticket</h1>
<h3>If you raise a ticket, it means that there is a
discrepancy in your votes appearing on the bulletin board
</h3>

<s:form action="ticket" method="post" theme="simple">
<div class="form-group">
<label for="exampleInputPassword1"><h4>Description</h4>
</label><br>
<s:textarea name="ticket.description" required="required"
rows="20" cols="150"/>
</div>

<s:hidden name="uid" value="%{getUid()}" />
<s:hidden name="mode" value="raise" />
<s:submit value="Submit" cssClass="btn btn-primary"/>
</s:form>

</div><!-- /col-lg-9 END SECTION MIDDLE -->

</div><! --/row -->
</section>
</section>
</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
</script>
</body>
</html>
```

ticket-vo.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
```

```
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:url
value='/dashgum/assets/img/logo.png'/>"></a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}">${sessionScope.event.name}
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="#" onclick="opennewtab('board')" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3>${sessionScope.event.name}</h3>

<div class="row mt">
<div class="col-md-12">
<div class="content-panel">
<table id="datazexample" class="table table-
striped table-advance table-hover">
<h4><i class="fa fa-angle-right"></i> Tickets
</h4>
<hr>
<thead>
<tr>
<th><i class="fa fa-bullhorn"></i> Voter ID</th>
<th class="hidden-phone"><i class="fa fa-
question-circle"></i> Ticket Description</th>
</tr>
</thead>
<tbody>
<s:iterator value="tickets" var="t">
<tr>
<td><s:property value="#t.voter_id" /></td>
<td><s:property value="#t.description" /></td>
</tr>
</s:iterator>

</tbody>
</table>
</div><!-- /content-panel -->
</div><!-- /col-md-12 -->
</div><!-- /row -->


<s:include value="/pages/includes/scripts.jsp"/>
<script>

$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})

function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>
</div>
</div><! --/row -->
</section>
</section>


</section>

</body>
</html>
```

uploadkeysuccess.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:
url value='/dashgum/assets/img/logo.png'/>">
</a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="%{#e.event_id}">
</s:param>
<s:param name="mode" value="'goToEvent'"></s:param>
</s:url>
<s:a href="%{viewURL}"><s:property value="event.
getName()" />
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
```

```
</li>
<li class="sub-menu">
<a href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="board" >
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a class="active" href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">


<h2><s:property value="%{election.getName()}" />
</h2>
<s:if test='%{ret.equals("computeSuccess")}'>
<s:iterator value="positions" var="p" status="c1">
<s:property value="%{#p.name}" /><br>
<s:iterator value="%{candidates.get(#c1.index)}"
var="c" status="c2">
<s:property value="%{#c.last_name}" />,
<s:property value="%{#c.first_name}" /> :
<s:property value="%{#c.vote_count}" /><br>
</s:iterator>
<s:if test="%{#p.required_min == 0}">
ABSTAIN : <s:property value="%{#p.abstain_count}" />
</s:if>
<br><br>
</s:iterator>

<s:url var="pdfDownload" action="key_download" >
<s:param name="filename" value="%{pdfname}">
</s:param>
</s:url>
<s:a href="%{pdfDownload}" cssClass="
btn btn-primary">View results in PDF</s:a>
</s:if>
<s:else>
<h2>Tallying Failed</h2><br>
<h4>That private key seems to be invalid</h4>

</s:else>
</div><!-- /col-md-12 -->
</div><!-- /row -->
<s:if test="errorDel == false">
<script>
alert("You cannot delete an election in
use by positions, candidates, ...");
</script>
</s:if>
</div><!-- /col-lg-9 END SECTION MIDDLE -->
</div><! --/row -->
</section>
</section>
</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
$(document).ready( function() {
$('#datazexample').dataTable({
/* Disable initial sort */
"aaSorting": []
});
})
</script>
</body>
</html>
```

uploadvotersuccess.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img
src="<s:url value='/dashgum/assets/img/ui-
sam.jpg'/>"
class="img-circle" width="60"></a></p>
<h5 class="centered">
<s:url action="events" var="viewURL" >
<s:param name="event.event_id" value="
%{#e.event_id}">
</s:param>
<s:param name="mode" value="'goToEvent'">
</s:param>
</s:url>
<s:a href="%{viewURL}">${sessionScope.event.name}
</s:a>
</h5>

<li class="mt">
<a href="elections">
<i class="fa fa-dashboard"></i>
<span>Elections</span>
</a>
</li>

<li class="sub-menu">
<a href="parties" >
<i class="fa fa-desktop"></i>
<span>Parties</span>
</a>
</li>

<li class="sub-menu">
<a href="blocks" >
<i class="fa fa-cogs"></i>
<span>Blocks</span>
</a>
</li>
<li class="sub-menu">
<a href="positions" >
<i class="fa fa-book"></i>
<span>Positions & Candidates</span>
</a>
</li>
<li class="sub-menu">
<a class="active" href="voters" >
<i class="fa fa-tasks"></i>
<span>Voters</span>
</a>
</li>
<li class="sub-menu">
<a href="#" onclick="opennewtab('board')">
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>
<li class="sub-menu">
<a href="results" >
<i class=" fa fa-bar-chart-o"></i>
<span>Results</span>
</a>
</li>

</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->
<section id="main-content">
<section class="wrapper">

<div class="row">
<div class="col-lg-12 main-chart">

<h3>${sessionScope.event.name}</h3>

<div class="row mt">
<div class="col-md-12">
<div class="content-panel">

<s:if test="%{duplicateUnames.size() != 0}">
<h3>The following was/were not added
since this/these username/s already exist/s.<h3>
<s:iterator value="duplicateUnames">
<h4><s:property /></h4>
```

```
</s:iterator>
</s:if>
<s:else>
<h3>Upload Success! All voters were added!</h3>
</s:else>
</div><!-- /content-panel -->
</div><!-- /col-md-12 -->
</div><!-- /row -->

</div><!-- /col-lg-9 END SECTION MIDDLE -->


</div><! --/row -->
</section>
</section>

<!--main content end-->
<!--footer start-->

<!--footer end-->
</section>

<s:include value="/pages/includes/scripts.jsp"/>

<script>
function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>

</body>
</html>
```

vote.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<link rel="stylesheet" href="<s:url value='/
dashgum/assets/css/to-do.css'/>">
<body>

<section id="container" >
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar"  class="nav-collapse ">
<!-- sidebar menu start-->
<ul class="sidebar-menu" id="nav-accordion">

<p class="centered"><a href=""><img src="<s:url
 value='/dashgum/assets/img/logo.png'/>"></a></p>

<li class="mt">
<a href="home">
<i class="fa fa-dashboard"></i>
<span>Home</span>
</a>
</li>

<li class="sub-menu">
<a class="active" href="vote">
<i class="fa fa-desktop"></i>
<span>Vote</span>
</a>
</li>

<li class="sub-menu">
<a href="#" onclick="opennewtab('board')">
<i class="fa fa-th"></i>
<span>Bulletin Board</span>
</a>
</li>


</ul>
<!-- sidebar menu end-->
</div>
</aside>
<!--sidebar end-->

<section id="main-content">
<section class="wrapper">

<div class="row mt">
<div class="col-md-12">
<section class="task-panel tasks-widget">

<br>
<div align="center"><h2><s:property
 value="%{event.getName()}" />
</h2></div>
<hr />
<s:set var="counter" value="-1"/>

<s:fielderror fieldName="checker"
 cssClass="voteerror"/>
<s:if test='event.getStatus().
equals("RUNNING")'>
<s:form action="submit_vote"
method="post" theme="simple">
<s:iterator value="elections"
 var="e" status="c1">
<div class="panel-heading">
<div class="pull-left"><h4><i
 class="fa fa-tasks"></i>
<s:property value="%{#e.name}" /></h4>
</div>
<br>
</div>
<s:iterator value="%{positions.get
(#c1.index)}"
 var="p" status="c2">
<br>
<div class="panel-body">
<div class="task-content">
<ul class="task-list">
<s:set var="counter" value="
%{#counter+1}"/>
<b><s:property value="%{#p.name}" /></b>
 (Choose from <s:property value=
"%{#p.required_min}"/>
to <s:property value="
%{#p.required_max}"/>)

<s:iterator value="%{candidates.
get(#c1.index).get(#c2.index)}"
  status="c3">
<li>
<s:if test="%{#p.required_max == 1}">
<s:radio theme="simple"
name="singlePos
[%{#c1.index}]
[%{#c2.index}]"
 list="last_name+' '+first_name"
 listKey="candidate_id"
 onclick="validateR('
%{hashes.get(#c1.index).
get(#c2.index).get(#c3.index)}',
%{#counter})" />
</s:if>
<s:else>
<s:if test="%{candidates.get(#c1.index).
get(#c2.index).
get(#c3.index).getLast_name()
!= 'ABSTAIN'}">
<s:checkboxlist theme="simple"
name="multiplePos[%{#c1.index}]
[%{#c2.index}]"
 list="last_name+' '+first_name"
 listKey="candidate_id"
 onclick="validateC('%{evotes.get
(#c1.index).get(#c2.index)}',
%{#counter},
 %{#c1.index}, %{#c2.index})" />
</s:if>
<s:else>
<s:radio theme="simple"
name="multiplePos
[%{#c1.index}][%{#c2.index}]"
 list="last_name+' '+first_name"
 listKey="candidate_id"
 onclick="validateCA('%{evotes.
 get(#c1.index).get(#c2.index)}',
%{#counter}, %{#c1.index},
%{#c2.index}) />
</s:else>
</s:else>
 
<s:property value="%{
candidates.get(#c1.index).
get(#c2.index).get(#c3.index)
.getTemp()}" />
</li>
</s:iterator>
<s:fielderror fieldName="%
{fErrors[#counter]}" />
<h4><p id="demo<s:property
 value='%{#counter}'/>">
Hash Code:</p></h4>
</ul>
</div>
</div>
</s:iterator><br>
</s:iterator>
```

```
<div align="center">
<s:submit value="Submit"
cssClass="btn btn-primary"/>
</div><br>
</s:form>
</s:if>
<s:elseif test='event.getStatus().
("FINISHED")'>
<h1>EVENT IS OVER</h1>
</s:elseif>
<s:else>
<h1>NOT YET AVAILABLE</h1>
</s:else>

</section>
</div><!--

</div><! --/row -->
</section>
</section>

<!--main content end-->

</section>


<s:include value="/pages/includes/scripts.jsp"/>
<script src="http://code.jquery.com/ui/1.10.3/
jquery-ui.js"></script>
<script src="<s:url value='dashgum/assets/js/
tasks.js' />"></script>

<script src="<s:url value='dashgum/assets/js/
md5.min.js' />"></script>

<script src="http://crypto-js.googlecode.com/
svn/tags/3.0.2/build/rollups/md5.js"></script>
<script>
function validateR(hash, ctr){
document.getElementById("demo"+ctr).innerHTML
= "Hash Code: " +hash;
}
function validateC(id, ctr, c1, c2){
var check = document.getElementsByName(
"multiplePos["+c1+"]["+c2+"]");
var array = id.split(", ");
array[0] = array[0].substring(1,
array[0].length);
array[check.length-1] =
array[check.length-1].
substring(0,array[check.length-1]
.length-1);
//alert(array[check.length-1]);
var str = "";
var zeroCheck = true;
for(var i=0; i<check.length; i++){
if(check[i].checked){
str += array[i];
zeroCheck = false;
}
if(i==check.length-1)
check[i].checked = false;
}
if(!zeroCheck){
document.getElementById("demo"+ctr).
innerHTML = "Hash Code: " +md5(str);
}
else
document.getElementById("demo"+ctr).
innerHTML = "";
}
function validateCA(id, ctr, c1, c2){
var check = document.getElementsByName
("multiplePos["+c1+"]["+c2+"]");
var array = id.split(", ");
array[check.length-1] = array
[check.length-1]
.substring(0,array[check.length-1]
.length-1);
//alert(array[check.length-1]);
for(var i=0; i<check.length-1; i++){
if(check[i].checked)
check[i].checked = false;
}
document.getElementById("demo"+ctr)
.innerHTML
 = "Hash Code: " +
 md5(array[check.length-1]);
}
</script>

<script>
function opennewtab(url){
var win=window.open(url, '_blank');
}
</script>


</body>
</html>
```

voting-officials.jsp

```
<html lang="en">
<s:include value="/pages/includes/head.jsp"/>
<body>
<section id="container">
<s:include value="/pages/includes/header.jsp"/>
<aside>
<div id="sidebar" class="nav-collapse">
<ul class="sidebar-menu" id="nav-accordion">
<p class="centered"><a href=""><img src="
<s:url value='/dashgum/assets/img/logo.png'/>">
</a></p>
<li class="mt">
<a href="home">
<i class="fa fa-dashboard"></i>
<span>Home</span>
</a>
</li>
<li class="sub-menu">
<a class="active" href="voting_officials">
<i class="fa fa-desktop"></i>
<span>Voting Officials</span>
</a>
</li>
</ul>
</div>
</aside>
<section id="main-content">
<section class="wrapper">
<div class="row">
<div class="col-lg-12 main-chart">
<h3><s:property value="event.getName()" /></h3>
<div class="row mt">
<div class="col-md-12">
<div class="content-panel">
<table id="datazexample" class="table
table-striped table-advance table-hover">
<h4><i class="fa fa-angle-right"></i>
Voting Officials<div class="pull-right">
<button class="btn btn-primary" data-toggle="modal"
data-target="#myModal"><i class="fa fa-plus"></i>
Add Voting Official</button>
   
</div>
</h4>
<hr>
<thead>
<tr>
<th><i class="fa fa-bullhorn"></i> Name</th>
<th><i class="fa fa-bullhorn"></i> Email</th>
<th class="hidden-phone"><i class=
"fa fa-question-circle"></i> Action</th>
</tr>
</thead>
<tbody>
<s:iterator value="voting_officials" var="v">
<tr>
<td><s:property value="%{#v.username}"/></td>
<td><s:property value="%{#v.email}"/></td>
<td>
<s:url action="voting_officials" var="editURL"
escapeAmp="false">
<s:param name="voting_official.user_id"
value="%{#v.user_id}"></s:param>
<s:param name="voting_official.username"
value="%{#v.username}"></s:param>
<s:param name="voting_official.email"
value="%{#v.email}"></s:param>
<s:param name="mode" value="'editPage'">
</s:param>
</s:url>
<s:a href="%{editURL}" data-toggle="modal"
data-target="#myeditdialog" data-backdrop="static"
data-keyboard="false" cssClass="
btn btn-primary btn-xs">
<i class="fa fa-pencil"></i></s:a>
<s:url action="voting_officials" var="deleteURL">
<s:param name="voting_official.user_id"
value="%{#v.user_id}"></s:param>
<s:param name="mode" value="'delete'"></s:param>
</s:url>
<s:a href="%{deleteURL}" cssClass="
```

```
btn btn-danger btn-xs"
onclick="return confirm('Are you sure you want to
delete this voting official?')">
<i class="fa fa-trash-o"></i></s:a>
</td>
</tr>
</s:iterator>
</tbody>
</table>
</div>
</div>
</div>
<s:form action="voting_officials" theme="simple">
<div aria-hidden="true" aria-labelledby="
myModalLabel" role="dialog" tabindex="-1"
id="myModal" class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<s:a href="voting_officials" cssClass="close">
&times;</s:a>
<h4 class="modal-title">Add Voting Official</h4>
</div>
<div class="modal-body">
<p>Username</p>
<s:textfield name="voting_official.username"
cssClass="form-control" required="required" /><br>
<div class="myerror"><s:fielderror fieldName="userF" />
</div><br>
<p>Email Address</p>
<s:textfield name="voting_official.email"
cssClass="form-control" required="required" /><br>
<div class="myerror"><s:fielderror fieldName="emailF" />
</div><br>
<s:hidden name="mode" value="add" />
</div>
<div class="modal-footer">
<s:a href="voting_officials"
cssClass="btn btn-default">Cancel</s:a>
<s:submit value="Submit" cssClass="btn btn-theme"/>
</div>
</div>
</div>
</div>
</s:form>
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myeditdialog"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
</div>
</div>
</div>
<div aria-hidden="true" aria-labelledby="myModalLabel"
role="dialog" tabindex="-1" id="myeditdialog2"
class="modal fade">
<div class="modal-dialog">
<div class="modal-content">
<s:form action="voting_officials" method="post"
theme="simple">
<div class="modal-header">
<s:a href="voting_officials" cssClass="close">
&times;</s:a>
<h4 class="modal-title">Edit Voting Official</h4>
</div>
<div class="modal-body">
<p>Username</p>
<s:textfield name="voting_official.username"
value="%{voting_official.getUsername()}"
cssClass="form-control" required="required" /><br>
<div class="myerror"><s:fielderror fieldName="userF" />
</div><br>
<p>Email Address</p>
<s:textfield name="voting_official.email"
value="%{voting_official.getEmail()}"
cssClass="form-control" required="required" /><br>
<div class="myerror"><s:fielderror
fieldName="emailF" /></div><br><br>
<s:hidden name="voting_official.user_id"
value="%{voting_official.getUser_id()}" />
<s:hidden name="mode" value="edit" />
</div>
<div class="modal-footer">
<s:a href="voting_officials" cssClass="btn btn-default">
Cancel</s:a>
<s:submit value="Submit" cssClass="btn btn-theme"/>
</div>
</s:form>
</div>
</div>
</div>
<s:if test="errorDel == false">
<script>alert("You cannot delete an election
in use by positions, candidates, ...");</script>
</s:if>
</div>
</div><!--/row -->
</section>
</section>
</section>
<s:include value="/pages/includes/scripts.jsp"/>
<s:if test="hasError[0] == true">
<script type="text/javascript">$(window).load(function()
{$("#myModal").
modal("show")});</script>
</s:if>
<s:if test="hasError[1] == true">
<script type="text/javascript">$(window).load(
function(){
$("#myeditdialog2").modal("show")});</script>
</s:if>
</body>
</html>
```

struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD
Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/
struts-2.0.dtd">

<struts>
<constant name="struts.devMode"
value="false" />

<package name="default" extends="
struts-default">

<global-results>
<result name="sorry" type="redirect">
/pages/logoutaction</result>
</global-results>


<action name="rdr" class="com.sp.vote.
action.LoginAction">
<result name="official_user">/pages/
official/home_vo.jsp</result>
<result name="voter_user">
/pages/voter/home_v.jsp</result>
<result name="admin_user">/pages/
admin/home_a.jsp</result>
<result name="input">/pages/login.jsp
</result>
</action>

<action name="login" class="com.sp.vote.action.
LogoutAction">
<result name="success">/pages/login.jsp</result>
<result name="input">/pages/login.jsp</result>
</action>

<action name="about_page" class="com.sp.vote.
action.BlankAction">
<result>/pages/about.jsp</result>
</action>

<action name="signup_page" class="com.sp.vote.
action.BlankAction">
<result>/pages/signup.jsp</result>
</action>

<action name="signup" class="com.sp.vote.
action.SignupAction">
<result name="success">/pages/login.jsp</result>
<result name="input">/pages/signup.jsp</result>
</action>

<action name="index"
class="com.sp.vote.
action.BlankAction">
<result>/pages/login.jsp</result>
</action>
<action name="public_events"
class="com.sp.
vote.action.EventAction" method="viewPublicEvents">
<result name="success">/pages/public/
publicevents.jsp</result>
</action>

<action name="public_elections"
```

```
class="com.sp.vote.action.
EventAction" method="viewPublicElections">
<result name="success">/pages/public
/publicevents2.jsp
</result>
</action>

<action name="home"
class="com.sp.vote.action.HomeAction" >
<result name="success">
/pages/hello.jsp</result>
<result name="error">
/pages/login.jsp</result>
</action>

<action name="vote"
class="com.sp.vote.action.BallotFormAction">
<result name="success">
/pages/voter/vote.jsp</result>
</action>

<action name="submit_vote"
class="com.sp.vote.action.VoteAction">
<result name="success">/pages/voter/
result_aftervote.jsp</result>
<result name="input">/pages/voter/
vote.jsp</result>
</action>

<action name="events"
class="com.sp.vote.action.EventAction">
<result name="viewEvents">
/pages/official/event/events.jsp</result>
<result name="addPage">
/pages/official/event/addevent.jsp</result>
<result name="addSuccess">
/pages/official/home_vo.jsp</result>
<result name="input">
/pages/official/home_vo.jsp</result>
<result name="input2">
/pages/official/event/event_home.jsp</result>
<result name="editPage">
/pages/official/event/editevent.jsp</result>
<result name="editSuccess">
/pages/official/event/event_home.jsp</result>
<result name="deleteSuccess">
/pages/official/event/events.jsp</result>
<result name="startSuccess"
type="stream">
<param name="contentDisposition">
attachment;filename=${filename}</param>
<param name="contentType">
application/octet-stream</param>
<param name="inputName">
fileInputStream</param>
<param name="bufferSize">1024</param>
<param name="location">/
pages/official/event/event_keys.jsp</param>
</result>
<result name="stopSuccess">/
pages/official/event/event_home.jsp</result>
<result name="archiveSuccess">
/pages/official/event/event_home.jsp</result>
<result name="goToEvent">/
pages/official/event/event_home.jsp</result>
</action>

<action name="elections"
class="com.sp.vote.action.ElectionAction">
<result name="viewElections">
/pages/official/election/elections.jsp
</result>
<result name="addPage">
/pages/official/election/addelection.jsp
</result>
<result name="addSuccess">
/pages/official/election/elections.jsp
</result>
<result name="editPage">
/pages/official/election/editelection.
</result>
<result name="editSuccess">
/pages/official/election/elections.jsp
</result>
<result name="deleteSuccess">
/pages/official/election/elections.jsp
</result>
<result name="input">
/pages/official/election/elections.jsp
</result>
</action>

<action name="parties" class="com.sp.vote.action.
PartyAction">
<result name="viewParties">
/pages/official/party/parties.jsp</result>
<result name="addPage">
/pages/official/party/addparty.jsp</result>
<result name="addSuccess">
/pages/official/party/parties.jsp</result>
<result name="editPage">
/pages/official/party/editparty.jsp</result>
<result name="editSuccess">
/pages/official/party/parties.jsp</result>
<result name="deleteSuccess">
/pages/official/party/parties.jsp</result>
<result name="input">
/pages/official/party/parties.jsp</result>
</action>

<action name="blocks"
class="com.sp.vote.action.BlockAction">
<result name="viewBlocks">
/pages/official/block/blocks.jsp</result>
<result name="addPage">
/pages/official/block/addblock.jsp</result>
<result name="addSuccess">
/pages/official/block/blocks.jsp</result>
<result name="input">
/pages/official/block/blocks.jsp</result>
<result name="input2">
/pages/official/block/blocks.jsp</result>
<result name="editPage">
/pages/official/block/editblock.jsp</result>
<result name="editSuccess">
/pages/official/block/blocks.jsp</result>
<result name="deleteSuccess">
/pages/official/block/blocks.jsp</result>
</action>

<action name="positions"
  class="com.sp.vote.action.
PositionAction">
<result name="positionHome">
/pages/official/position/position_home.jsp
</result>
<result name="viewPositions">
/pages/official/position/positions.jsp
</result>
<result name="dragPositions">
/pages/official/position/positions2.jsp
</result>
<result name="addSuccess">
/pages/official/position/positions.jsp
</result>
<result name="editSuccess">
/pages/official/position/positions.jsp
</result>
<result name="deleteSuccess">
/pages/official/position/positions.jsp
</result>
<result name="editPage">
/pages/official/position/editposition.jsp
</result>
<result name="input">/
/official/position/positions.jsp
</result>
</action>

<action name="reorder_positions"
class="com.sp.vote.
action.PositionAction" method="reorderPos">
<result name="reorderSuccess">
/pages/official/position/position_home.jsp
</result>
</action>

<action name="candidates" class="
com.sp.vote.action.CandidateAction">
<result name="viewCandidates">
/pages/official/candidate/candidates.jsp
</result>
<result name="addSuccess">/
pages/official/candidate/candidates.jsp
</result>
<result name="editPage">/
pages/official/candidate/editcandidate.jsp
</result>
<result name="editSuccess">/
pages/official/candidate/candidates.jsp
</result>
<result name="deleteSuccess">/
pages/official/candidate/candidates.jsp
</result>
```

```
<result name="input">/
pages/official/candidate/candidates.jsp
</result>
</action>

<action name="voters" class=
"com.sp.vote.action.VoterAction">
<result name="viewVotersHome">/
pages/official/voter/voters_home.jsp
</result>
<result name="viewVoters">
/pages/official/voter/voters.jsp
</result>
<result name="addPage">/
pages/official/voter/addvoter.jsp
</result>
<result name="addSuccess">/
pages/official/voter/voters.jsp
</result>
<result name="input">
/pages/official/voter/voters.jsp
</result>
<result name="importPage">
/pages/official/voter/voters.jsp
</result>
<result name="importSuccess">
/pages/official/voter/voters.jsp
</result>
<result name="editPage">/
pages/official/voter/editvoter.jsp
</result>
<result name="editSuccess">
/pages/official/voter/voters.jsp
</result>
<result name="deleteSuccess">
/pages/official/voter/voters.jsp
</result>
</action>

<action name="voting_officials"
class="com.sp.vote.action.Voting
OfficialAction">
<result name="viewVotingOfficials">
/pages/admin/voting_official/
voting_officials.jsp</result>
<result name="addPage">/
pages/admin/voting_official/
addvoting_official.jsp</result>
<result name="editPage">
/pages/admin/voting_official/
editvoting_official.jsp</result>
<result name="addSuccess">
/pages/admin/voting_official/
voting_officials.jsp</result>
<result name="editSuccess">
/pages/admin/voting_official/
voting_officials.jsp</result>
<result name="input">
/pages/admin/voting_official/
voting_officials.jsp</result>
</action>

<action name="upload_voters"
  class="com.sp.vote.action.
  UploadVotersAction">
<result name="success">
/pages/official/voter/uploadvotersuccess.
jsp</result>
<result name="input">
pages/official/voter/voters.jsp</result>
</action>

<action name="upload_key"
class="com.sp.vote.action.UploadKeyAction">
<result name="success">/
pages/official/result/uploadkeysuccess
.jsp</result>
</action>

<action name="results"
class="com.sp.vote.action.ResultsAction">
<result name="display">
/pages/official/result/results_home.jsp
</result>
<result name="computePage"
>/pages/official/result/input_sk.jsp
</result>
<result name="computeSuccess">
/pages/official/result/electionresults.jsp
</result>
<result name="computeFailure">
/pages/official/result/input_sk.jsp
</result>
<result name="input">/
pages/official/result/input_sk.jsp
</result>
</action>

<action name="board"
class="com.sp.vote.action.
BulletinBoardAction">
<result name="publishBoardSuccess">
/pages/board/board_main.jsp</result>
<result name="publishBoardFailure">
/pages/board/board_error.jsp</result>
<result name="viewVoteSuccess">
/pages/board/board_ind.jsp</result>
</action>

<action name="key_download"
class="com.sp.vote.action.
DownloadFileAction">
<result name="success" type="stream">
<param name="contentDisposition">
attachment;filename=${filename}</param>
<param name="contentType">
application/octet-stream</param>
<param name="inputName">
fileInputStream</param>
<param name="bufferSize">1024</param>
</result>

</action>

<action name="account"
class="com.sp.vote.action.AccountAction">
<result name="display">
/pages/account/account_home.jsp</result>
<result name="input">/
pages/account/account_home.jsp</result>
<result name="editPassSuccess">
/pages/account/account_home.jsp</result>
</action>

<action name="ticket"
class="com.sp.vote.action.TicketAction">
<result name="NONE">/pages/voter/ticket.jsp
</result>
<result name="raiseSuccess">
/pages/voter/ticketsuccess.jsp</result>
<result name="displaySuccess">
/pages/official/ticket/ticket_vo.jsp</result>
</action>

<action name="*">
<result>/pages/misc/sorry.jsp</result>
</action>

</package>
</struts>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/
ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_
  3_0.xsd"
  id="WebApp_ID" version="3.0">

<display-name>Voterify</display-name>

<filter>
<filter-name>struts2</filter-name>
<filter-class>org.apache.struts2.dispatcher.
ng.filter.StrutsPrepareAndExecuteFilter
</filter-class>
</filter>

<filter-mapping>
<filter-name>struts2</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>

<security-constraint>
<display-name>No direct JSP access
</display-name>
<web-resource-collection>
<web-resource-name>No-JSP
</web-resource-name>
<url-pattern>*.jsp</url-pattern>
```

```
</web-resource-collection>
<auth-constraint>
<role-name>no-users</role-name>
</auth-constraint>
</security-constraint>

<session-config>
<session-timeout>30</session-timeout>
</session-config>

<welcome-file-list>
<welcome-file>index</welcome-file>
</welcome-file-list>

</web-app>
```

XI. Acknowledgement

I may not be a man of many words but deep in my heart, I am very, very thankful to the following. I truly am.

- **Lord**. Thank you so much Lord for the infinitely many blessings that you have given me and my family. Thank you for continuously guiding me through the years. Without you, I am nothing and I won't be able to finish this SP.
- **Family**. Thank you Mom and Dad for your never ending love and support. Thank you for raising me well for me to become a better person. I can't thank you enough. Also, thank you Ate for being my sister!
- **Friends**. Thank you Block12, especially the Hotties, for all the moments we've shared. It's an honor to be with you guys. You'll always have a place in my heart.
- **SP Adviser**. Thank you, Mr. Richard Bryann Chua, for giving me a cool SP topic and for guiding me 'til the end.
- **Ate Eden**. Thank you, Ate Eden, for being so accomodating and friendly. We (ComSci students) all love you!
- **UP Professors**. Thank you all for giving us students such great knowledge.
- **UP itself**. Thank you for making me experience being an Iskolar ng Bayan.
- **Kuya Guards**. Thank you, Kuya Guard of CAS and Kuya Guard of ULib also for being so accomodating and friendly. I may not know your names but I know your faces.
- **To all other people out there who care about me**. Thank you so much! You guys mean to me.