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

YuVote: Using homomorphic encryption to create a verifiable voting system

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

Submitted by:

Berwin Jarret T. Yu

May 2017

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 "YuVote: Using homomorphic encryption to create a verifiable voting system" prepared and submitted by Berwin Jarret T. Yu 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. (*cand.*)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Perlita E. Gasmen, M.Sc. (*cand.*)	_____	_____
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

Privacy and verifiability are the two most important properties in voting systems. We created YuVote that satisfies these properties using homomorphic encryption. Elliptic curve cryptography is also used to provide more security. Specifically, the Elliptic ElGamal Cryptosystem is used by YuVote.

Keywords: Internet voting, homomorphic encryption, elliptic curve cryptography, Bouncy Castle, Elliptic ElGamal Cryptosystem

Contents

List of Figures

List of Tables

I. Introduction

A Background of the Study

Internet voting has been used in real elections over the recent years. It has been used in countries like Estonia and Norway. Internet voting has become a popular research topic [1]. There are many reasons why Internet voting is used. Internet voting provides a faster and a more accurate way to count the votes since machines are used and this prevents human errors. Internet voting allows people to vote from home and from out of country [2].

Despite the advantages that Internet voting provides, a lot of things have to be considered. Internet voting should satisfy properties like privacy, verifiability, unicity, fairness and other security requirements. Among these properties, privacy and verifiability are the hardest to achieve [3]. Privacy means that the vote of the voter would not be revealed to other people. Verifiability means that the voting process can be checked and it has many types. Individual verifiability means that the voters can verify that their vote was counted. Universal verifiability means that anyone can verify that the tally is correct. End-to-end verifiability means that anyone can verify the whole election process [4].

All voting systems must satisfy privacy and verifiability. This is done by using cryptography. Cryptography plays an important role in Internet voting systems. Privacy is achieved by encrypting the vote before sending it to the server [2]. However, the problem is that computations cannot be performed on encrypted votes to get the tally. The authorities could decrypt each votes and perform the tally but this would compromise the voter's privacy.

One of the approach in solving this problem is using homomorphic encryption. By using homomorphic encryption, computations can be performed on encrypted votes to get the tally. The votes does not need to be individually encrypted for them to be

included in the tally which means privacy is achieved. The result of the computation is then decrypted to get the tally.

Voterify [5] is one such Internet voting system created by Marvin Occeño that uses homomorphic encryption to satisfy security requirements like privacy and verifiability. Votes are encrypted using the Paillier cryptosystem which is an additive homomorphic cryptosystem which means that votes can be added by performing a multiplication operation on the encrypted votes.

Voterify achieves verifiability by giving voters a receipt. The receipt contains the hashes of candidates they chose. These hashes are posted on the public bulletin board so that the voters can check whether the hashes they received are the same as the ones posted. Any users can download the election results and the public bulletin board which contains the encrypted votes.

B Statement of the Problem

One problem with Voterify is that it uses the 1024-bit Paillier key [5]. Since computers are getting faster, brute-force attacks on cryptosystems are also getting faster. This means larger key sizes are needed since the larger key sizes provide more security than smaller ones [6]. However, using larger key sizes make the computations like key generation and encryption slower. It is important to choose a key size that is less vulnerable to brute-force attacks without making the computations too slow.

A solution to this problem is using elliptic curve cryptography. Elliptic curve cryptography is a type of public key cryptography that uses elliptic curves [7]. Elliptic curve cryptography requires less key sizes for the same security than Paillier [7, 8, 9]. A 160-bit elliptic curve key is as secure as a 1024-bit Paillier key. With elliptic curve cryptography, smaller key sizes can be used without compromising the security. This also means that computations would be faster and more efficient since smaller key sizes are used [7, 10].

Another problem of Voterify is that the votes are encrypted on the server side. This means that the votes are sent to the server in plaintext form before they are encrypted. The problem with this is that the votes can be intercepted while they are being sent to the server [5].

C Objectives of the Study

This work aims to create an Internet voting system using homomorphic encryption over elliptic curves which has the following functionalities:

1. Allows the voter to
    (a) Cast a vote in an election
    (b) See the hash codes based on the selected votes
    (c) Download the verification code with the hash codes
    (d) View the public bulletin board
    (e) Verify his vote based the verification code and hash codes
    (f) View election results
2. Allows the voting official to
    (a) Setup an election
        i. Manage sub-elections
        ii. Manage parties
        iii. Manage blocks
        iv. Manage positions
        v. Manage candidates
        vi. Manage voters
        vii. Start and stop an election

(b) Tally the result

(c) View the public bulletin board

(d) Manage his account

3. Allows the public user to

(a) View election results

(b) View the public bulletin board

4. Allows the system administrator to

(a) Manage the voting officials

(b) Manage his account

D Significance of the Project

Online voting systems allow fast and accurate results because it eliminates human errors especially if the number of voters is very large. Online voting systems make the voting process easier and more convenient to the voters [11]. They are also very accessible since the voter can use computers and mobile devices to vote. This would help increase the number of voter turnouts.

Verifiability and privacy are very important properties to have even in traditional voting. Privacy is important because people do not want to reveal their votes to other people. Verifiability is also important because there is a need to verify that no votes were manipulated or lost, that votes were stored correctly, and that all the votes were counted correctly [12]. In short, verifiability provides any people a way to check if the election process is correct. A voting system that has privacy and verifiability would help gain the trust of the voters [5] and help encourage more people to vote.

It is important that voting systems are secure so that other people would not be able to manipulate the results. That is why choosing the right key size is important.

As computers are getting faster, it is becoming easier to break cryptosystems. Elliptic curve cryptography allows smaller key sizes to be as secure as the larger key sizes of other algorithms. This allows the key sizes to be small without compromising the computation times.

E Scope and Limitations

1. This system should not be used in high-coercible elections.
2. This system should not be used in big elections like government elections.
3. This system does not support registration of voters. The voting official is responsible for adding the list of voters.
4. The voters can vote as many times as they want but only the last vote will be counted.
5. The system doesn't take into account accessibility to people with disabilities.
6. Write-in votes are not accepted.

F Assumptions

1. The email addresses of voters are legitimate and not compromised.
2. The voting officials do not lose or share their private key to other people.
3. The voting officials do not change the contents of the private key file.
4. The voters will not lose their username and password to other people.
5. An abstain vote will mean not voting for any candidate in a certain position.
6. The distribution and collection of the private keys will be done outside of the system.

II. Review of Related Literature

With the advances in technology, more and more elections use Internet voting. Internet voting systems provide more advantages than the traditional voting systems. Internet voting systems try to enable efficient and secure elections. It is also inexpensive because its resources are reusable since we could use the same code for different elections. Lastly, it does not require a geographical proximity of voters which means that a voter may not need to be in a certain place to be able to vote and it also provides better scalability for large elections.

Even though electronic voting systems have many advantages, they are hard to implement because they must satisfy some requirements. Yumeng et. al [3] divided the requirements or properties into three levels as shown in figure 1. The first level consists of basic requirements and it should be fulfilled in almost every voting system. The second level is more rigorous for a voting system. The third level is the advanced requirements for special systems. These requirements need to be considered when creating a voting system. Among the properties, privacy and verifiability are the hardest to resolve due their inherent conflict as privacy makes verifiability harder to achieve [3].

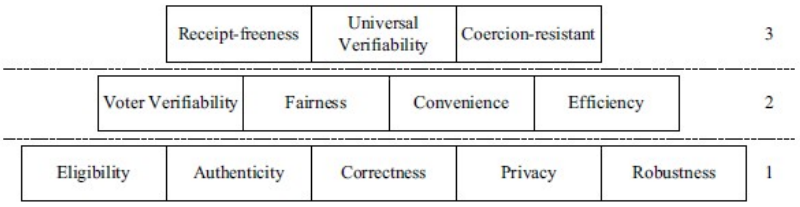

Figure 1: Requirements of a voting system

The Estonian Voting System [13] is the first Internet voting system that was used for national elections. The Estonian Voting System uses national ID cards for the authentication of the voters and for the singing of the ballots. The voter inputs a PIN associated with his authentication key to authenticate himself and then he enters

his signing key PIN to sign the ballot. The client then encrypts the vote using the election public key and signs the encrypted vote with the voter's private key. The votes are then stored to the vote storage server. After the election period, officials use a DVD to copy the encrypted votes from the vote storage server to the vote counting server. The vote counting server decrypts the votes, counts them, and outputs the official results [13]. Voters are allowed to vote as many times as they want but only the last vote is counted. This protects the voters from coercion.The vote can be verified by using a mobile application that scans the QR code displayed by the voting client and returns the candidates voted. While this may provide cast-as-intended verifiability, the voting system lacks other types of verifiability like individual and universal verifiability since the voters cannot verify that their votes are counted and they cannot verify the final tally [2].

Norway used an Internet voting system in its 2011 and 2013 local government elections [14]. The voting process starts by letting the voter choose his vote to form the ballot. The ballot is then encrypted using the ElGamal cryptosystem before sending it. A return code is then computed and given to the voter via SMS. The voter then verifies his vote by comparing the return codes he received against a list of precomputed option-return codes that is printed on his voting card. The voters are also allowed to revote to protect them from coercion.The votes are shuffled in a mix-net before they are decrypted and tallied. This protocol does not give any way for the voter to verify the final tally. However, there are auditors that checks the consistency of the ballot box and the proofs produced when the ballots are decrypted [2].

iVote is the voting system that was used to vote at the 2015 State General Election in New South Wales, Australia [15]. During the registration stage, the voter chooses a PIN and later receives an iVote number. The voter uses these two to login to the iVote System and cast a vote. After the voter chooses his vote, the voting client generates

two envelopes containing the encrypted voting options. The voter receives a receipt number after he votes. The first envelope is encrypted using the Electoral Board's public key while the second envelope is encrypted using the Verification Server public key. Both of these envelopes include the receipt number. Zero knowledge proofs are generated to prove that those two envelope contain the same voting options. The first envelope or the *envelope for counting* is used for counting the tally. It is exported to the Offline Voting-Management System where it is decrypted and tallied. The second envelope or the *envelope for verification* is sent to the Verification Server. The voter uses his receipt number, his PIN, and his iVote number to call the Verification Server. The Verification Server will then open the envelope and read aloud the votes to the voter. After the election, the receipt number from the decrypted votes are posted online so that the voters can check if their vote was counted in the tally. Despite the verifiability that iVote offers, it can only be used in low-coercion elections [15].

There are three main paradigms used in Internet voting to fulfill the security requirements in Internet voting systems: mix-net paradigm, blind signature based and homomorphic encryption.

In the mix-net paradigm, the voter encrypts the vote and submits it. After the voting period, all the ballots are shuffled and remasked. This is done to dissociate the vote from its voter [16]. After receiving the final mix, all ballots are decrypted and tallied. There are two types of mix-net protocols: decryption mix-nets and re-encryption mix-nets. In decryption mix-nets, the message is encrypted by applying a concatenation of k encryption operations [17]. Each mix servers peels off one of the encryptions and then mixes the decrypted inputs. The decryption mix-net protocol is described as an onion where the encryption is like building the onion and the decryption done by each mix servers is like peeling off one layer of the onion [17]. In re-encryption mix-net, the mix server re-encrypts the ciphertexts by adding a random number to the randomness of the encryption and then shuffles the ciphertexts by

multiplying an encryption of the number 1 [3]. To provide verifiability, the mix server produces a zero-knowledge proof that they did the operations properly [4].

In the blind signature based paradigm, the authorities sign the ballots without knowing the content of the ballot [16]. This is done by letting the voter blind the ballot before sending it to the authentication server. If the sender is a valid voter, the authentication server signs the blinded ballot with its private key and sends the ballot back to the voter. After that, the voter unblinds the ballot to get his signed vote. The voter can check whether his signed ballot contains his vote. The voter would then send the signed ballot to the tallying server. After the voting period, all ballots are decrypted and tallied. The purpose of blinding the ballot is to hide the votes from the authentication server which gives privacy. Universal verifiability is only achieved when a verifiable anonymous channel is used [4]. Eligibility verifiability is achieved since authentication servers will not sign the ballot if the sender is not a voter.

In homomorphic encryption, the voter encrypts his vote with a homomorphic cryptosystem. With a homomorphic cryptosystem, the votes could be added without decrypting the votes. This would add privacy [3]. When a voter creates the ballot, a zero-knowledge proof is also created to prove that the ballot was created properly. Individual verifiability is achieved by posting the cast ballot in a public bulletin board. The cast ballot is formed by encrypting the votes of the voter. This technique is also used by other paradigms to provide individual verifiability.

Homomorphic encryption is used in many online voting systems. One advantage homomorphic encryption has over the other paradigms is that it has a smaller computational cost for verifiable decryption [4]. In verifiable decryption, each decryption is accompanied by proofs that show correct decryption. In the homomorphic encryption, the aggregated data is the only one that is decrypted and a proof is done for that decryption. In the mix-net and blind signature based paradigm, the ballots

are individually decrypted and proofs are done for each decryption. Homomorphic encryption also provides better privacy since the individual votes are never decrypted.

Joaquim [18] proposed a way to prove the validity of a ballot encryption by using a single verification code. After the voter casts an encrypted ballot, the voting machine computes for a verification code and sends it to the voter. If the voting machine encrypts the vote correctly, the voter would be able to compute for the verification code provided by the voting machine using his selected votes. The verification code does not reveal any information about the voter's selections [18]. This proposal can be used for the mix-net paradigm and the homomorphic encryption paradigm. The disadvantage of this proposal is that it is an inconvenience to the voters. The voter would need to find a special program and input large numbers to the calculator to be able to get the verification code. The voting system could provide the special program but the user still needs to calculate those large numbers.

Galindo et. al discussed in [19] the protocol used in Neuchâtel to provide cast-as-intended verification. This protocol uses return codes which are computed and given to voters after they register. The return code is composed of k codes where each code represents a candidate and k is the number of candidates. The return codes are different per voter and per candidates. This means that if there are a voters and b candidates, there would be a total of $a \times b$ unique return codes. After voting, a set of return codes is computed from the ballot and given to the voter. If the ballot is encrypted properly, the return codes should consist of n codes where each code corresponds to the return code of the candidates selected by the voter. These codes should match the return codes given after the registration. This protocol has a phase called the confirmation phase. The purpose of this phase is to support single vote casting [19]. After the voter receives the return codes from the voting machine, he can input a confirmation code which is given after registration. When the correct confirmation code is given, the ballot will now be counted in the tally. The purpose

of this phase is to confirm that the voter received the correct return codes from the voting machine.

Joaquim's proposal [18] may seem similar to the protocol used in Neuchâtel [19]. However, they have few differences. The main difference is that the protocol in Neuchâtel has a return code for each candidate while Joaquim's proposal only has one verification code. The protocol in Neuchâtel only requires voter to compare each return code to the ones posted on the bulletin board while Joaquim's procotol requires the voter to calculate the verification code before comparing it to the one posted on the bulletin board to be able to verify his vote.

Helios [20] is a web-based open-audit voting system developed by Ben Adida. Helios is called an open-audit system because anyone can verify the election. Helios does this by using the proposal of Benaloh [21] which has a separation of ballot preparation and casting to allow the ballot to be audited. After creating a ballot, a person can either cast or audit it. Casting the ballot can only be done by legitimate voters. Auditing of the ballots can be done by anyone and its purpose is to open the ballot to check if the ballots are prepared correctly. Auditing of the ballot provides cast-as-intended verifiability. Helios provides a bulletin board where the voters and the ballots are displayed. Helios offers a way to let anyone verify the whole election tally by the Election Tallying Verification program. The programs accepts an election ID and checks the all proofs and performs re-tallying [20]. Helios provides privacy by using the mix-net paradigm. It also provides individual and universal verifiability by using the proposal of Benaloh [21]. Helios does not provide coercion-resistance which means that Helios should not be used in big elections like the government elections. Kulyk et. al [22] proposed a protocol to add eligibility verifiability and participation privacy to Helios. This is done by allowing a chosen set of people called the "posting proxies" to add null votes in behalf of a voter. These null votes have no effect on the tally and are indistinguishable from real votes. Null votes are also indistinguishable

from other null votes. All of the votes cast by a voter are published on the public bulletin board. These cast votes may be null votes cast by other people or real votes cast by the voter himself. This satisfies participation privacy because other people would not be able to determine if the votes cast are null votes are real votes.

Culnane et. al discussed in [23] the end-to-end verifiability feature of vVote which is based on Prêt à Vote. End-to-end verifiability is achieved by providing cast-as-intended verification, recorded-as-cast verification, and universally verifiable tallying. vVote uses a feature that is similar to Benaloh's proposal [21] to achieve cast-as-intended verification as discussed in the previous paragraph. After voting, the voter receives a receipt which is used to check if the information posted on the Web Bulletin Board or WBB is correct and this achieves recorded-as-cast verification. After the voting period, the votes are mixed using a mix-net and decrypted. Proofs of correct mixing and decryption are also posted on the WBB. All informations posted on the WBB can be used to verify the tally and this achieves universally verifiable tallying. vVote also achieves privacy by using mix-nets.

STAR-vote is a voting system that gives a secure, transparent, auditable, and a reliable election [24]. To vote, the voter first checks-in with a poll worker in the polling station where the voter receives a sticker with his name, precinct and a ballot style. The voter will then approach a device called the controller that scans the barcode on the sticker and then prints a 5-digit code. The voter inputs the 5-digit code in the voting terminal then the voting options will be shown. After the voter chooses the candidates, the voting terminal encrypts the vote and prints two items: a paper ballot and a take-home receipt. The paper ballot contains the voter's readable choices and a random serial number. The voter can choose to cast the ballot by letting the ballot box scan the serial number of the ballot or he can choose to spoil it by allowing it to be decrypted. This process is also known as the Benaloh challenge [21] which provides cast-as-intended verifiability. The encrypted ballots are posted

on a public bulletin board with their corresponding hash codes and non-interactive zero-knowledge proofs that the ballots are well-formed. The vote receipt contains a hash code which can be used to check if his vote is posted on the public bulletin board. This provides record-as-cast and individual verifiability. The encrypted votes do not contain any information about the voter so privacy is achieved. After the election, votes are homomorphically tallied and a number of election trustees work together to decrypt the tally. Universal verifiability is achieved since anyone can check that the homomorphic tally corresponds to the encrypted ballots on the bulletin board and since the encrypted tally is verifiably decrypted.

Voterify is an Internet voting system that was developed by Marvin Occeño [5]. Votes are encrypted using the Paillier cryptosystem which has an additive homomorphic property by multiplying the encrypted message. The plaintext of vote v, the one that is encrypted, is formed using the formula $v = C_1 \times 10^0 + C_2 \times 10^1 + ... + C_n \times 10^{n-1}$ where n is the number of candidates and C_i is equal to 1 if the voter votes for candidate i otherwise it is 0. For example, if the voter voted for candidate 2 the vote v would be equal to 10. The encrypted votes are multiplied to get the sum or the tally of the votes. When the tally is decrypted, the resulting plaintext has the same form as the formula as v. The only difference is that C_i now is equal to the number of votes for candidate i. The formula is problematic when the number of votes for a candidate exceeds 10. For example, if the number of votes for candidate 2 is 10, then the plaintext tally is equal to 100. The system would read this as 1 vote for candidate 3 which is an error. To prevent this overflowing from happening, 9 encrypted votes are aggregated at a time before they are decrypted and added to the tally. Privacy is achieved because of homomorphic encryption. Voterify offers cast-as-intended verification by giving voters a receipt which contains the hash of the candidates they choose and a verification code. Each candidates have a corresponding hash linked to them. When a voter chooses a candidate, the hash of that candidate should appear

on the receipt. These hashes are posted on the public bulletin board so that the voter can check whether the codes posted are the same codes which they received. Any users can download the election results and the public bulletin board which contains the encrypted votes.

Mateu et. al proposed a protocol in [4] which combine parts of the mix-net and homomorphic encryption paradigm together. The Elliptic ElGamal Cryptosystem which is homomorphic is used to encrypt the vote while mix-net is used to generate or create the ballot. The ballot is composed of a vector with k elements where k is the number of candidates. Each elements of the ballot is either an encryption of a vote or not a vote. An encryption of a vote means that the candidate is chosen by the voter while an encryption of not a vote means that the candidate is not chosen by the voter. Before the election starts, the polling station would create a ballot called C where the first element is an encryption of a vote and the rest of the elements are encryptions of not a vote. This ballot is posted publicly and will be used by all of the voters as a template to create their ballot. The voter creates his ballot by getting the ballot posted by the polling station and mixing the vector to match his vote. Figure 2 shows us how the template ballot is used when the voter votes for candidate 2. The ballot will then be re-encrypted and sent to the polling station. The votes are tallied by adding each components of the vectors of all the ballots to get the vector of the encrypted sum of the votes. Each components of the vector sum is then decrypted to get the final tally of each candidates. One advantage of this protocol is that the proof of correct ballot generation costs less because proof of correct mixing is used rather than range proofs used in homomorphic encryption [4]. This protocol can support multi-candidate elections where the voters can vote for more than one candidate. This is done just changing the way the polling station creates template ballot C. The template ballot now consists of $C + l$ elements where the first C elements is an encryption of not a vote while the last l elements is an encryption of a vote. Note

that l is the maximum number of candidates that a voter can vote. The voter will then mix this template ballot to match his vote. Figure 3 shows us how the template ballot is used when the voter votes for candidates 1 and 3. It is important to note that the first k elements of the vector are the one that is aggregated. Overall, the protocol has a lesser cost in proving the correct ballot generation. It also satisfies properties like authenthicity, unicity, privacy, fairness, and verifiability.

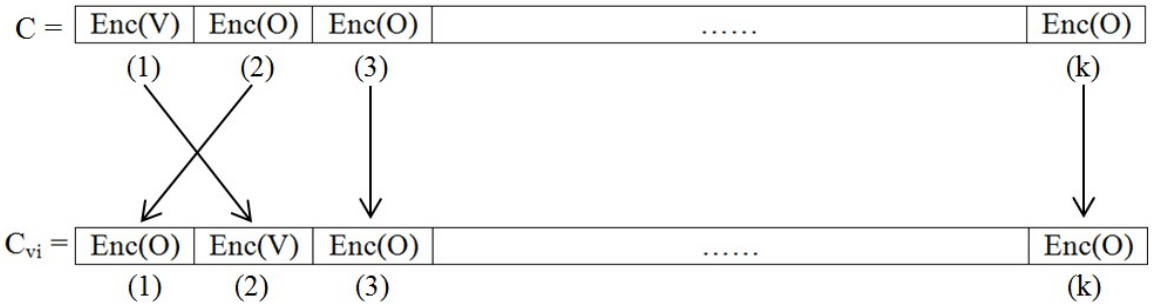

Figure 2: How the ballot for voter i is formed from the template ballot C for single elections

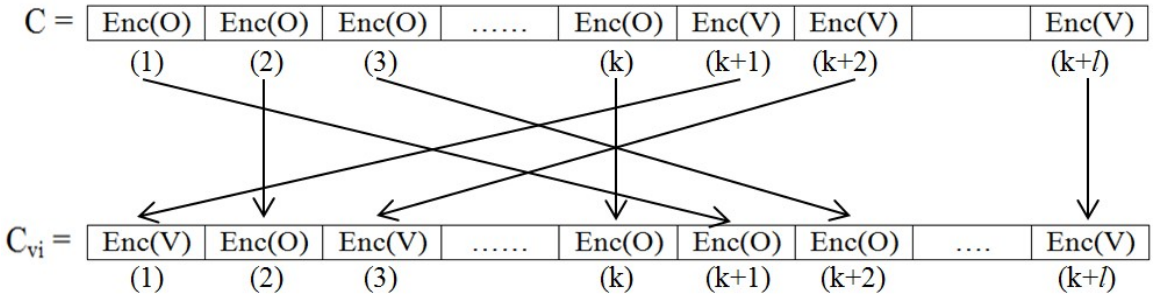

Figure 3: How the ballot for voter i is formed from the template ballot C for multi-candidate elections

III. Theoretical Framework

A Internet Voting

Internet voting is sometimes called online voting. It can be divided into two types [25]:

1. Poll site voting: This type of Internet voting requires voters to go to specific sites with voting staffs to vote. Votes are transfered through the Internet from each poll sites to a centralized site where votes are tallied.

2. Remote voting: This type of Internet voting allows voters to vote from any computer or device that is connected to the Internet.

Internet voting is not the same as electronic voting or e-voting. Electronic voting uses electronic devices to do voting procedures. It uses technologies like optical scanning, punched cards and voting kiosks where ballots are transmitted via telephone, Internet, or private computer networks [26]. This means that Internet voting is only part of electronic voting. Internet voting is the use of the Internet to transfer the vote from the voting device which is located in the voter's home to the polling station or tallying station. Devices commonly used in used Internet voting are devices with internet connection like computers and mobile phones.

Internet voting provides advantages over the traditional voting systems. However, Internet voting provides more risk of election manipulation. Thus, voting systems, for them to be secure, reliable, and trustworthy, should satisfy the following security requirements which are described in [3, 16, 4, 19, 5]:

1. Privacy: A vote is kept secret and no one can determine who others voted for

2. Eligibility: Only authorized voters are able to vote.

3. Uniqueness/Unicity: Although a person can vote for more than once, his latest vote is the one that is counted.

4. Fairness: Partial results cannot be known before the end of the voting period.

5. Verifiability: Correctness of the voting process can be checked. The different types of verifiability are:

   (a) Individual verifiability is where voters can verify that their vote was counted in the tally.

   (b) Universal verifiability is where any entity can verify that all votes have been tallied properly. Also known as counted-as-recorded verifiability.

   (c) Cast-as-intended verifiability is where the voter can check if the cast ballot contains the votes that they have selected.

   (d) Recorded-as-cast verifiability is where the voters can check that their votes have been correctly received and stored at the remote voting server.

   (e) End-to-end verifiability is where all the phases of the election can be verified by any entity.

6. Uncoercibility: Any coercer should not be able to coerce a voter to cast his vote.

7. Receipt-freeness: A voter cannot prove he voted in a particular way.

8. Robustness: No voter can disrupt the election, any invalid vote will be detected and not counted in the final tally.

B Homomorphic Encryption

Homomorphic encryption is a type of encryption that allows computations to done on ciphertexts without compromising the plaintext when it is decrypted. More precisely, homomorphic encryption allows certain operations, such as addition, to be done on ciphertexts directly so that upon decryption the same answer is obtained when operating on the plaintext [27]. An encryption scheme is said to be homomorphic for some

operations $\circ \in F_m$ acting in message space (such as addition), there are corresponding operations $\diamond \in F_c$ acting in ciphertext space satisfying the property:

$$Dec(k_s, Enc(k_p, m_1) \diamond Enc(k_p, m_1)) = m_1 \circ m_1$$

where m_1 and m_2 are the cleartext messages and k_p and k_s are the public-private key pairs [27].

Homomorphic encryption can be partially or fully homomorphic. A scheme is considered partially homomorphic if it exhibits either additive or multiplicative homomorphism. Additive homomorphism is where $F_m = \{+\}$ which means an operation done on the ciphertext corresponds to addition in cleartext. Multiplicative homomorphism is where $F_m = \{\times\}$ which means an operation done on the ciphertext corresponds to multiplication in cleartext. A scheme is considered fully homomorphic if it exhibits both additive and multiplicative homomorphism [28].

C Elliptic Curve Cryptography

Elliptic curve cryptography is a type of cryptography that uses elliptic curves. An elliptic curve E over a finite field $\mathbb{F}_p$ is defined by an equation of the form

$$y^2 = x^3 + ax + b$$

where $a, b \in \mathbb{F}_p$ satisfy $4a^3 + 27b^2 \neq 0$. A pair (x, y), where $x, y \in \mathbb{F}_p$, is a point on the curve if (x, y) satisfies the equation. The point at infinity, denoted by the symbol ∞, is also said to be on the curve [4, 8, 29]. The point at infinity is the identity element of the field.

The chord-tangent method is used to add two points in the curve. The method has two cases. The first case is when you add points P and Q to get the sum called R. To get R, a line through points P and Q is drawn in the graph. This line intersects a

third point in the graph called $-R$. The reflection of the third point about the x-axis is R which is the sum of P and Q. The second case is when P is added to itself. To get R, a tangent line to the elliptic curve at point P is drawn. This line intersects a second point in the graph called $-R$. The reflection of the second point about the x-axis is R which is the sum of P and Q. Points can be multiplied to an integer of n by adding the point to itself n times.

Elliptic cryptography is based on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECDLP states that given two points, P and Q, on an elliptic curve, find the integer l, if it exists, such that $P = lQ$ [8]. The combination of the Pohlig-Hellman algorithm and Pollard's rho algorithm is used to solve th ECDLP and it has a fully-exponential running time of $O(\sqrt{p})$ where p is the largest prime divisor of n and n is the order of point P. This means that elliptic curve parameters should be chosen so that n is divisible by a prime number p sufficiently large so that $\sqrt{p}$ steps is an infeasible amount of computation [8].

In elliptic curve cryptography, the public and private key pair is generated using algorithm 1. Algorithm 1 returns Q and d where Q is the public key and d is the private key. Elliptic curve cryptography is an approach to public-key cryptography [7] which means that the public key of the receiver is used to encrypt the message while the receiver uses his private key to decrypt the message. Other people can decrypt the message if they manage to get the private. However, getting the private key requires solving the ECDLP.

Elliptic curve cryptography have advantages over other public-key cryptography algorithm. One advantage is that ECC provide more security for the same key size. Figure 4 shows the comparison of strength of RSA and ECC. This means that an RSA key with 512 bits would provide the same security as an ECC key with 106 bits. Another advantage is that ECC requires smaller space since a smaller size of keys are required for the same security [7].

Time to break (in MIPS-years)	RSA key-size (in bits)	ECC key-size (in bits)
10^4	512	106
10^8	768	132
10^{11}	1024	160
10^{20}	2048	210
10^{78}	21000	600

Figure 4: Comparison of strength of RSA and ECC

D Elliptic ElGamal Cryptosystem

Elliptic ElGamal cryptosystem is an encryption scheme that uses elliptic curves. The key pair is generated given the parameters p, E, P, and n. These parameters are also known as domain parameters. The prime number p is the order of the finite field $\mathbb{F}_p$. E is the elliptic curve defined by the parameters a and b where a and b are integers. P is a point in the elliptic curve with order n. Domain parameters are usually chosen using named curves. Named curves are a set of elliptic curves with predefined domain parameters.The key pair is generated using algorithm 1 as presented in [8]:

Algorithm 1: Elliptic curve key pair generation

Input : Elliptic curve domain parameters (p, E, P, n)
Output: Public key Q and private key d.
1 Select $d \in_R$ [1,n - 1];
2 Compute $Q = dP$;
3 Return (Q, d);

Messages are encrypted and decrypted using algorithms 2 and 3 respectively as

presented in [8]:

Algorithm 2: Basic ElGamal elliptic curve encryption

Input : Elliptic curve domain parameters (p, E, P, n), public key Q, plaintext m

Output: Ciphertext (C_1, C_2)

1 Represent the message m as a point M in $E(\mathbb{F}_p)$;
2 Select $k \in_R$ [1,n - 1];
3 Compute $C_1 = kP$;
4 Compute $C_2 = M + kQ$;
5 Return (C_1, C_2);

Algorithm 3: Basic ElGamal elliptic curve decryption

Input : Domain parameters (p, E, P, n), private key d, ciphertext (C_1, C_2)

Output: Plaintext m

1 Compute $M = C_2 - dC_1$, and extract m from M;
2 Return m;

Elliptic ElGamal Cryptosystem is also a homomorphic cryptosystem because of its encryption function. When we encrypt a point V_1 with public key p_k, we would get $(k_1P, V_1 + k_1p_k)$ where P is the base point in the elliptic curve and k_1 is a random number. When we encrypt a point V_2 with public key p_k, we would get $(k_2P, V_2+k_2p_k)$ where P is the base point in the elliptic curve and k_2 is a random number. Adding these two ciphertexts would yield $(k_1P + k_2P, V_1 + k_1p_k + V_2 + k_2p_k)$. Factoring and rearranging the ciphertext would yield $((k_1+k_2)P, (V_1+V_2)+(k_1+k_2)p_k)$. Decrypting the ciphertext with private key s_k would yield V_1+V_2 which is why this cryptosystem is homomorphic.

Elliptic ElGamal Cryptosystem is IND-CPA secure[30, 31]. Indistinguishable under chosen plaintext attack or IND-CPA means that if an adversary chooses two plaintexts and a challenger encrypts one of the plaintexts, the adversary would have hard time telling which one of the two plaintexts was encrypted even if the adversary has access to the encryption function [32].

E Bouncy Castle

The Legion of the Bouncy Castle is an organization that provides cryptography API for Java and C# [33]. The Bouncy Castle API is also a provider for the Java Cryptography Extension(JCE) and the Java Cryptography Architecture(JCA). The Bouncy Castle API implements the elliptic curve cryptography, specifically the Elliptic ElGamal cryptosystem.

The inputs needed in algorithm 2 are generated using named curves. Named curves are elliptic curves with predefined domain parameters p, E, P, n. The reason for using named curves is because generating elliptic curve parameters from scratch is a very difficult task and not all elliptic curve parameters are secure. SafeCurves [34] is website that analyzed the security of named curves. To get the parameters of named curves, the `ECNamedCurveTable.getParameterSpec(String)` or the `TeleTrusTNamedCurves.getByName(String)` method is used where String is the name of the curve. The parameters returned by these methods are used to create a public and private key pair using the `ECKeyPairGenerator` class. The public key is used to initialize the `ECElGamalEncryptor` class while the private key is used to initialize the `ECElGamalDecryptor` class. Encryption and decryption are done using the `encrypt(ECPoint)` and `decrypt(ECPair)` method of the `ECElGamalEncryptor` and the `ECElGamalDecryptor` class respectively where `ECPoint` is a class that represents a point in the curve and `ECPair` class that represents a pair of elliptic curve point.

It is important emphasize that the `encrypt(ECPoint)` method returns a pair of elliptic curve point denoted by the class `ECPair` which means that the ciphertext is a pair of elliptic curve points. To homomorphically add the ciphertexts, the first points of the two `ECPair` are added and second points of the two `ECPair` are added using the `add(ECPoint)` method of the `ECPoint` class. The results of the two additions are then stored to another `ECPair` which is the sum of the ciphertexts.

IV. Design and Implementation

A Use Cases

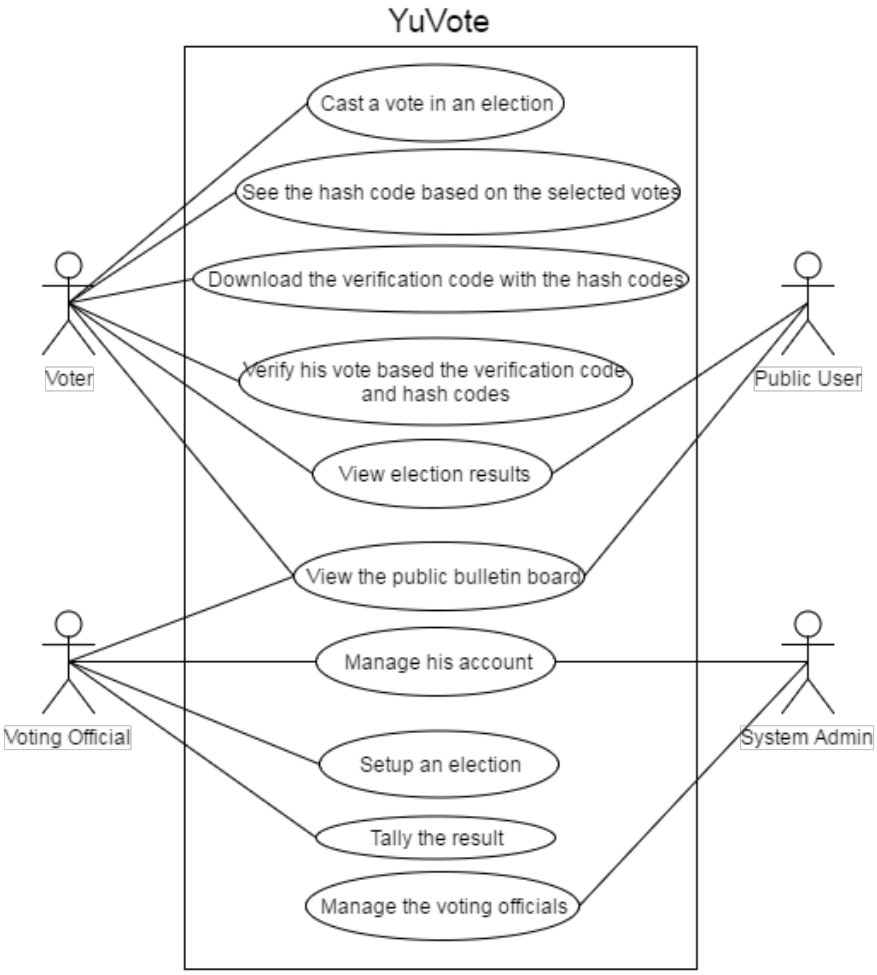

Figure 5: Top level use case diagram

The voting system has four users: the voter, the voting official, the public user and the system administrator.

The voter can cast a vote in an election. After voting, the voter can view his encrypted vote and the voter receives a verification code. The voter can then view the bulletin board to check if the verification code posted on the bulletin board is the same as the one he received. After the election, the voter can view the election results. Voters can vote as many as they want but only the last vote will be counted.

The voting official is the one that setups an election. Setting up an elections includes managing the sub-elections, blocks, positions, candidates and, voters. The official is also the one that starts and stops an election. The voting official receives the election's private key when he/she starts the election.

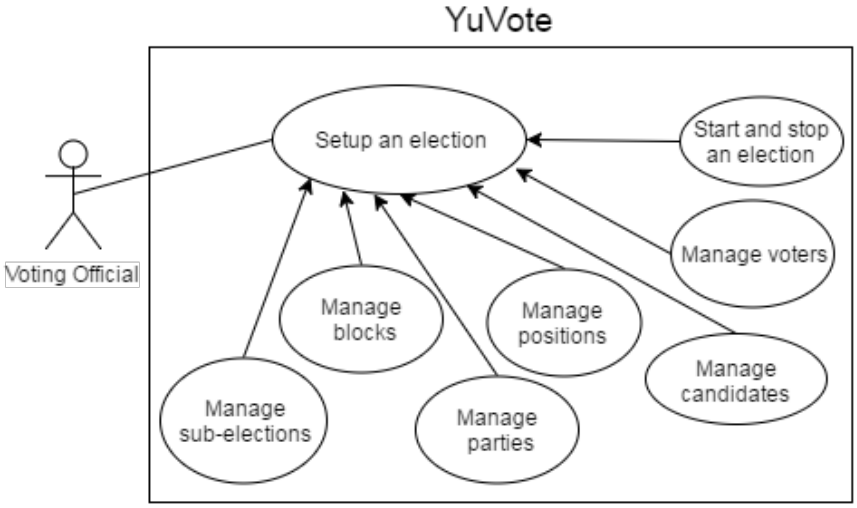

Figure 6: Use case diagram of setup an election

The voting official is the one that allow the votes to be tallied. To tally the votes, the voting official needs to upload the election's private key.

The system administration is the one that manages the voting officials.

The public user can view the public bulletin board where the votes are posted and the election results.

B Ballot Representation

The ballot is represented similar to the proposal of Mateu et. al [4]. The contents of the ballot are not the same for each voter. The contents depend on which block the voter belongs. The number of candidates, k, in the ballot is obtained by first getting all of the sub-elections the block participates in. Then, the positions for each sub-election are obtained. Finally, the candidates of each positions is obtained. The

ballot vector consists of k ciphertexts where k is the number of candidates in the ballot. The ordering of the candidates in the ballot vector are is first based on the ordering of the candidates based on the ordering of the position, then the ordering of candidates within the position. At the end of each position, a candidate named "abstain" is added and choosing this candidate represents that the voter did not vote for any candidates in the position. Abstain is treated as a candidate which means that the tally for it is also computed.

These ciphertexts are either an encryption of V or O and they represent whether the candidates are chosen or not. V is generated using the formula $V = rP$ where r is a random number from 1 to $n - 1$, n is the order of the elliptic curve, and P is the base point in the elliptic curve. The values of P and n depend on the parameters of the curve described in algorithm 1. O is represented by the point at infinity in the elliptic curve as discussed in section C. Point at infinity is used because it is the identity element of the elliptic curve which means that a point added to it will result to that point.

V represents a voter voting for a particular candidate and O representing the voter not voting for a particular candidate. This means that the ith element of the vector is an encryption of V if the voter voted for candidate i while it is an encryption of O if the voter did not vote for candidate i. For example, in a ballot vector named C that has two positions with 2 and 3 candidates, respectively, the ciphertext would look like this if the voter voted for candidate 2 on the 1st position and candidate 3 on the 2nd position:

$$C = (Enc(p_k, O), Enc(p_k, V), Enc(p_k, O), Enc(p_k, O), Enc(p_k, O), Enc(p_k, V), Enc(p_k, O))$$

where p_k is the public key of the election. The public key p_k and the private key s_k are generated during the start of the election. Section C discusses how the public and

private keys are generated.

After a voter votes, each components of his/her ballot vector are stored to the database together with the identifier of the candidate. The identifier of the candidate will determine which candidate the ciphertext corresponds to. Figure 7 shows us how the vote is stored in the database. The `vote_id` field is the identifier of the vote. The `candidate_id` field determines which candidate the vote is for. The `voter_id` field is the identifier of the voter. The time field determines time that this vote was submitted. The fields `encrypted_point1` and `encrypted_point2` are the ciphertexts of the vote. An encrypted vote is stored in these two fields because an encryption of any point in the Elliptic ElGamal cryptosystem would yield two points as shown in algorithm 2. This means a component of the ballot vector is stored in a single row. A ballot vector with 5 candidates would be stored in the database with 5 rows.

	vote_id	candidate_id	voter_id	time	encrypted_point1	encrypted_point2
Edit Copy Delete	1421	250	89	2017-04-28 13:21:39	0x040bd98d7a66a626872	0x044210334c61c5f1585c7f

Figure 7: A vote stored on the database

It is important to note that ciphertext will not be equal even if the encrypted plaintexts are the same. This is true because a random number is used in encrypting the votes as described in algorithm 2. This means that when two voters voted for the same candidate, the two ciphertexts will be different because two different random numbers were used in encrypting their votes. This also means that when a voter voted for two different candidates, the two ciphertexts will also be different since the two different random numbers were also used in encryption of the votes.

C Multiple Private Keys

Before the election starts, the voting official will be asked how many keys will be used by the election. The number of keys will determine how many private key files will be given to the voting official. All of these private key files will need to be uploaded

to the system before the tally can be computed.

Multiple private keys work by generating n public and private key pairs using algorithm 1 where each key pairs is denoted by Q_i and d_i. The public key of the election p_k is obtained with $p_k = \sum_{i=1}^{n} Q_i$. Each private key d_i is then stored in a file. The voting official is responsible for distributing these private key files. The private key of the election s_k is obtained with $s_k = \sum_{i=1}^{n} d_i$. This means to tally the election, the voting official needs to provide all of these private keys that were generated before the election.

The system provides a mechanism to check if the private keys provided by the voting officials is wrong or if the number of privates provided is wrong. This mechanism is done during the vote tallying stage which will be explained in section D.

D Vote Tallying

To tally the votes, the voting officials must first provide the private keys of the election that were given to them when the election started. The tally is computed by first getting all of candidates in the election. The tally for a candidate j, T_j, is obtained by getting all the ciphetexts corresponding to candidate j and homomorphically adding them. To get the the ciphertexts corresponding to a certain candidate, the identifier of that candidate will be used since votes are stored in the database with the identier of the candidate. T_j is then decrypted using the sum of the private keys that the voting officials provided to get R_j. R_j is of the form $a_j V$ where a_j is the number of votes for candidate j. R_j is of this form because of the homomorphic property of the cryptosystem. Since each ciphertext is either an encryption of V or O, adding these ciphertexts and decrypting it would always result to an integer multiplied by the point V. For example, adding 5 encryption of Vs and 2 encryption of Os then decrypting it would result to $5V$. Adding an encryption of O to any ciphertext would not affect the plaintext value of that ciphertext due to the fact that O is the identity

element of the curve which means that $V + O = V$ for any point V. During the setup stage of the election, a lookup table consisting of x and xV are created where x is from 1 to the number of voters. The lookup table makes it easier to get a_j from a_jV.

If there is any R_j that is not found in the lookup table created, then this means that the private keys provided by the voting officials are wrong. The reason to this is that if the correct private keys were provided, R_j would always be of the form xV as explained in the previous paragraph.

E Vote Verification

To provide verifiability to voters, voters are given a single verification code and a hash code for each position. The first step in getting the hash code for a position is to get the encrypted votes of each candidates for that position. Then, each encrypted votes are converted to `BigInteger` and added to each other. Finally, the MD5 hash of the sum is obtained and this now is the hash code for that position. The verification code is obtained similar to the hash code for each position, the only difference is that the components of the ballot are the ones that are converted to `BigInteger` and added to each other. Remember that the ballot contains the encrypted votes of all the candidates that the voter can vote for.

There are two stages in the voter verification process. The first stage is during the voting stage where the voter can see and download the verification code and hash codes. The second stage is after the tallying stage where the verification code and the hash codes are computed for each voter that voted and displayed in the public bulletin board.

While the voter is voting, the hash codes for each position are computed and shown below each position. When the voter picks a different choice, the hash code would be recomputed and changed. The voter can download the receipt which contains the verification code and the hash code for each position once he submitted his vote. It

is important to note that the verification code and the hash codes are not stored in the database during this stage.

After the tallying stage, the verification code and the hash code of each position for each voters that voted are computed and posted on the public bulletin board. During this stage, the voters can compare the contents of their receipt with the contents on the bulletin board. The reason for computing the verification code and the hash codes after the tallying stage is to detect any changes in the vote. If the votes were changed, then the voter would see the the verification code on the bulletin board is different from the one in the receipt.

F Database Design

YuVote follows the database design below.

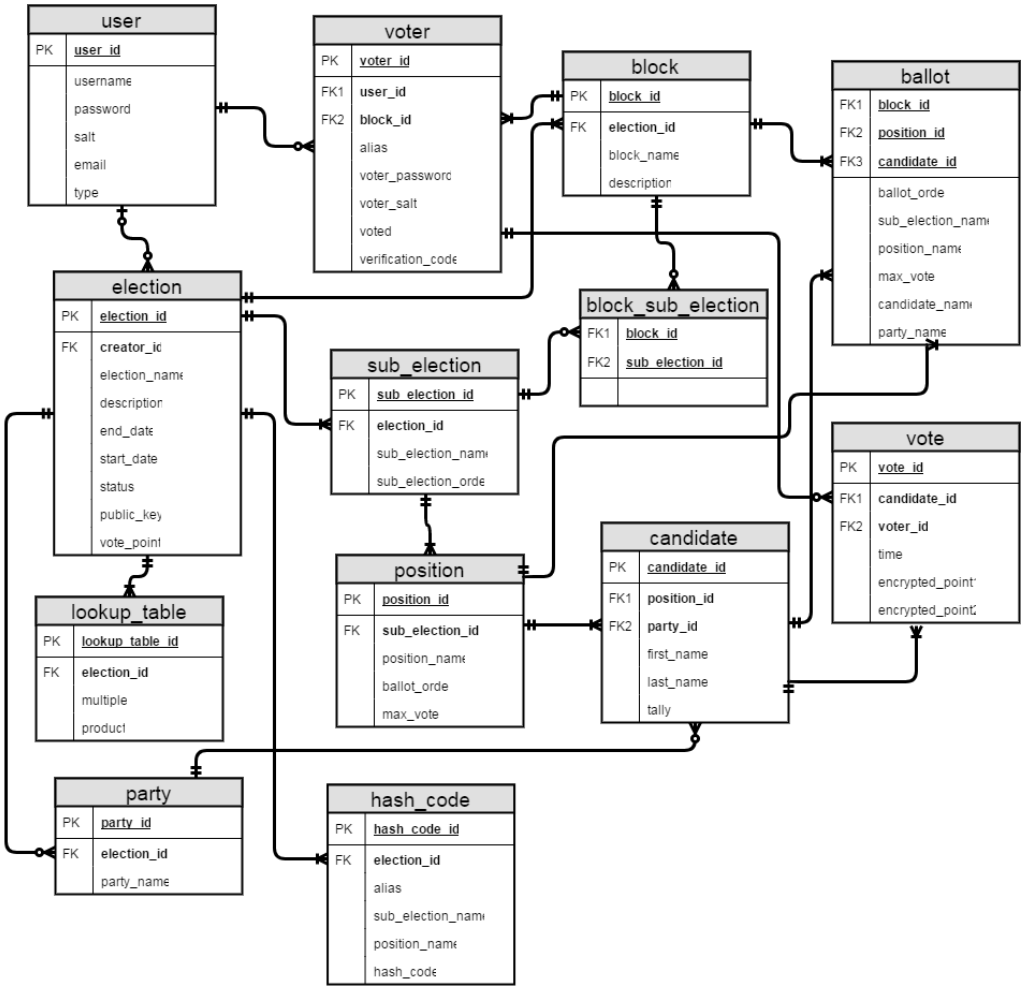

Figure 8: Entity Relationship Diagram of YuVote

G Data Dictionary

Column	Type	Description
user_id	int(11)	ID of the user
username	varchar(50)	Username of the user
password	varchar(50)	Hashed password of the user
salt	varbinary(50)	Salt of the password
email	varchar(50)	Email of the user
type	varchar(20)	Type of the user

Table 1: user table

Column	Type	Description
voter_id	int(11)	ID of the voter
user_id	int(11)	ID of the user
block_id	int(11)	ID of the block this voter belongs to
alias	varchar(50)	Alias of the voter
voter_password	varchar(50)	Hashed password of the voter
voter_salt	varbinary(50)	Salt of the voter password
voted	int(1)	Has value 1 if the voter has voted and 0 if not
verification_code	varchar(100)	Verification code of the voter

Table 2: voter table

Column	Type	Description
election_id	int(11)	ID of the election
creator_id	int(11)	ID of the user who created this election
election_name	varchar(50)	Name of the election
description	varchar(500)	Description of the election
end_date	datetime	End date of the election
start_date	datetime	Start date of the election
status	varchar(20)	Status of the election
public_key	varbinary(100)	Public key of the election
vote_point	varbinary(100)	Vote point of the election

Table 3: election table

Column	Type	Description
sub_election_id	int(11)	ID of the sub-election
election_id	int(11)	ID of the election where this sub-election belongs
sub_election_name	varchar(40)	Name of the sub-election
sub_election_order	int(11)	Order of the sub-election in the ballot

Table 4: sub_election table

Column	Type	Description
position_id	int(11)	ID of the position
sub_election_id	int(11)	ID of the sub-election where this position belongs
position_name	varchar(30)	Name of the position
ballot_order	int(11)	Order of the position in the ballot
max_vote	int(11)	Maximum number of candidate that a voter can vote

Table 5: position table

Column	Type	Description
candidate_id	int(11)	ID of the candidate
position_id	int(11)	ID of the position where this candidate belongs
party_id	int(11)	ID of the party of this candidate
first_name	varchar(30)	First name of the candidate
last_name	varchar(30)	Last name of the candidate
tally	int(11)	Total number of votes of the candidate

Table 6: candidate table

Column	Type	Description
party_id	int(11)	ID of the party
election_id	int(11)	ID of the election where this party belongs
party_name	varchar(30)	Name of the party

Table 7: party table

Column	Type	Description
block_id	int(11)	ID of the block
election_id	int(11)	ID of the election where this block belongs
block_name	varchar(30)	Name of the block
description	varchar(500)	Description of the block

Table 8: block table

Column	Type	Description
block_id	int(11)	ID of the block
sub_election_id	int(11)	ID of the sub-election

Table 9: block_sub_election table

Column	Type	Description
block_id	int(11)	ID of the block of the ballot
ballot_order	int(11)	Order of the candidate in the ballot
sub_election_name	varchar(40)	Name of the sub-election of the candidate
position_id	int(11)	ID of the position where the candidate belongs
position_name	varchar(30)	Name of the position of the candidate
max_vote	int(11)	Maximum vote of the position
candidate_id	int(11)	ID of the candidate
party_name	varchar(30)	Name of the party of the candidate
candidate_name	varchar(60)	Full name of the candidate

Table 10: ballot table

Column	Type	Description
vote_id	int(11)	ID of the vote
candidate_id	int(11)	ID of the candidate where this vote belongs
time	datetime	Time when the vote was submitted
encrypted_point1	varbinary(100)	The first point of the encrypted vote
encrypted_point2	varbinary(100)	The second point of the encrypted vote

Table 11: vote table

Column	Type	Description
hash_code_id	int(11)	ID of the vote
election_id	int(11)	ID of the candidate where this vote belongs
alias	varchar(50)	Time when the vote was submitted
sub_election_name	varchar(40)	The first point of the encrypted vote
position_name	varchar(30)	The second point of the encrypted vote
hash_code	varchar(100)	The second point of the encrypted vote

Table 12: hash_code table

Column	Type	Description
lookup_table_id	int(11)	ID of the lookup table entry
election_id	int(11)	ID of the election where this the lookup table belongs
multiple	int(50)	The integer that is multiplied to the vote point
product	varbinary(100)	The product of the multple and the vote point

Table 13: lookup_table table

H System Architecture

YuVote is divided into 3 separate components, a Java application for the system administrator and voting official, another Java application for the voters, and a web application for the public users. MariaDB is used as the database server. All of these components communicate with the MariaDB server to store or to get data.

The Java applications are developed using the JavaFX architechture. The Java applications follow the Model-View-Controller or the MVC pattern. The user interfaces are created using Scene Builder by Gluon. The reason for having 2 separate Java applications is to separate the functions of the system admin and voting official from the function of the voter. The Bouncy Castle library is used in the Java applications since they are the ones that need the cryptographic functions.

The web application is developed using the CodeIgniter framework which is a PHP framework. The purpose of the web application is mainly to show the election results and the bulletin board.

I Technical Architecture

The minimum requirements for the server machine include:

- Apache 2.4.23
- MariaDB 10
- 1GB RAM

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

- Google Chrome 57.0.2987
- Mozilla Firefox 43.0.1
- Internet Explorer 11.0.9600.18036
- PDF Viewer
- Notepad
- Windows 7
- Intel Core i5-4200U
- 4GB RAM

V. Results

The figure below shows the login page of the application for the system administrator and voting official.

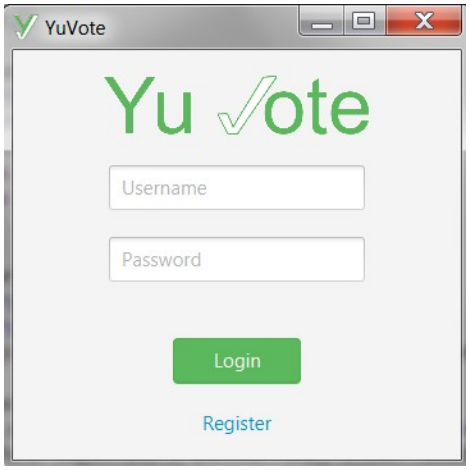

Figure 9: Login page of the system admin and voting official

After the voting official provides the correct credentials, the home page of the voting official will be shown. The home page contains all the elections the voting official manages. The voting official can add, edit, and delete an election. The voting official can also filter the elections shown based on the status. The voting official can also manage an election.

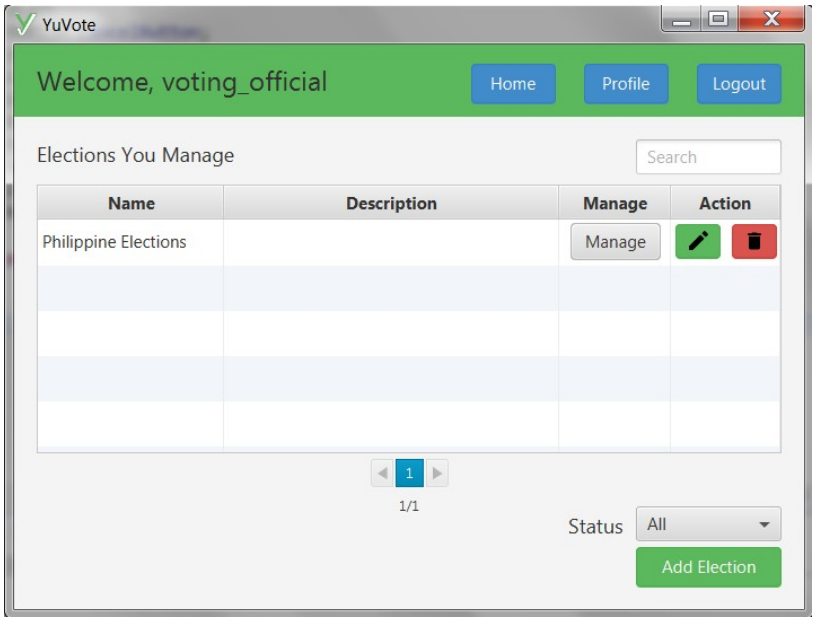

Figure 10: Home page of the voting official

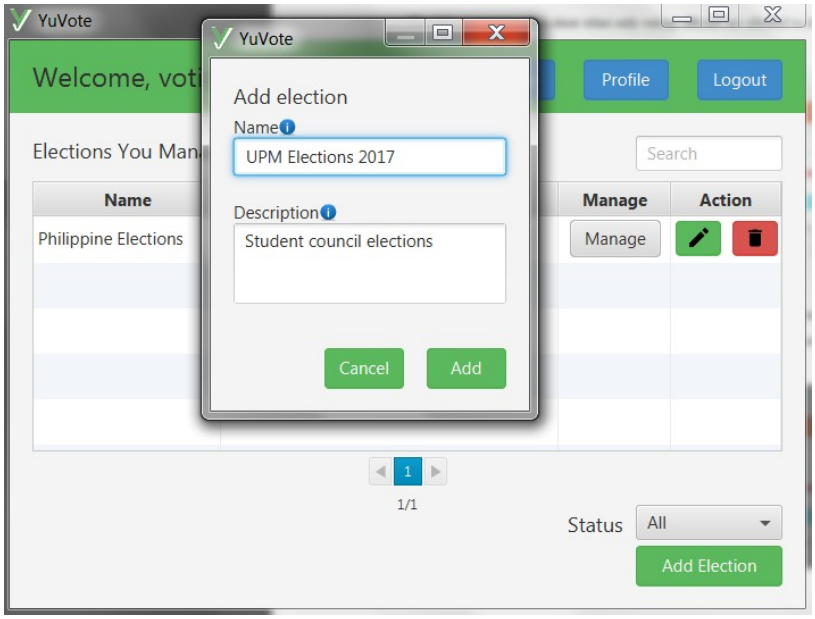

Figure 11: Add an election

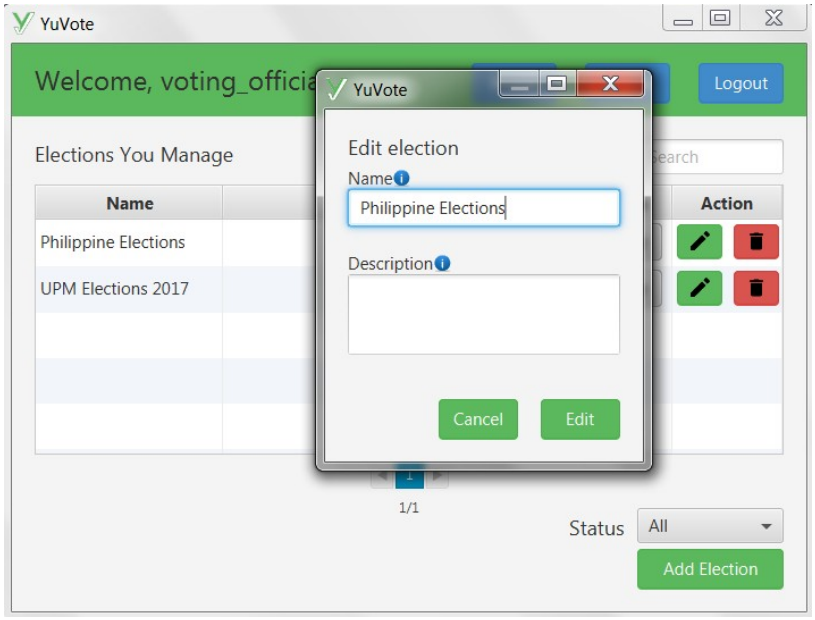

Figure 12: Edit an election

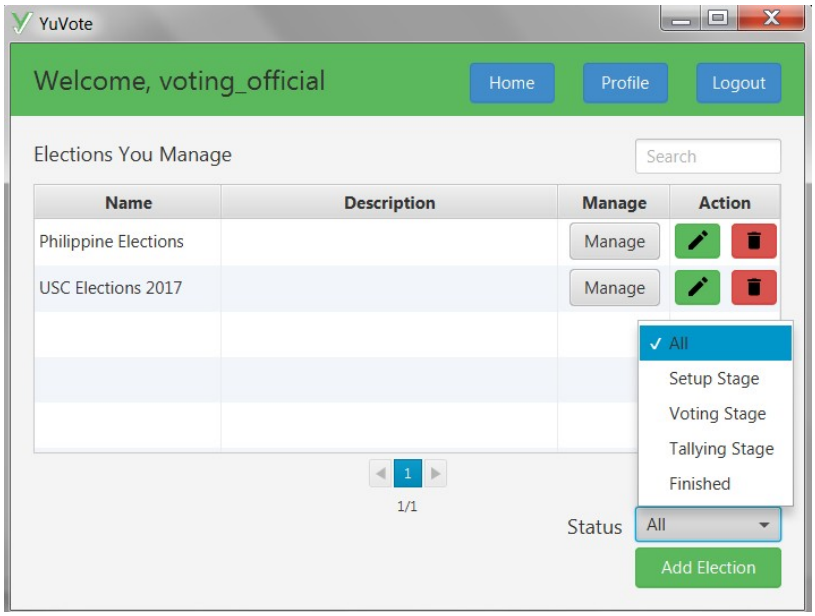

Figure 13: Filter elections

An i icon and the label beside it can be hovered to show a tooltip.

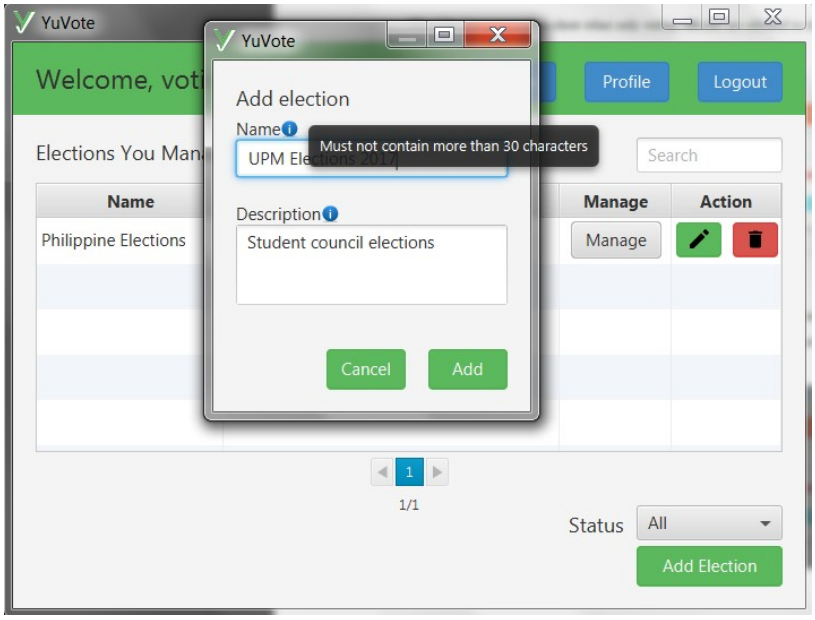

Figure 14: Tooltip

When the voting official chooses to manage an election, the main page of the election will be shown. This page contains all the requirements needed to start an election.

The sub-election page is where the voting official can add, edit, and delete sub-elections. The voting official can also reorder the sub-elections.

Figure 15: Main page of the election

Figure 16: Sub-elections page

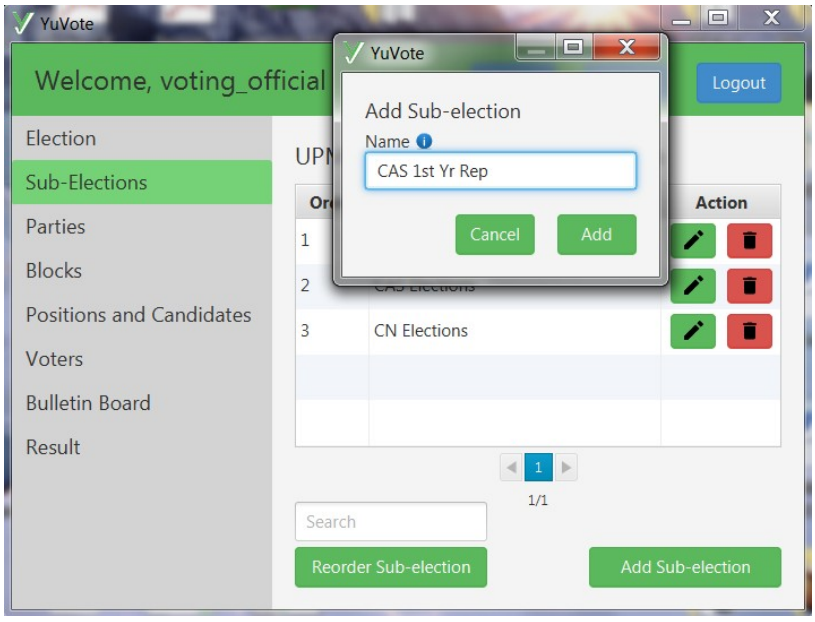

Figure 17: Add sub-election

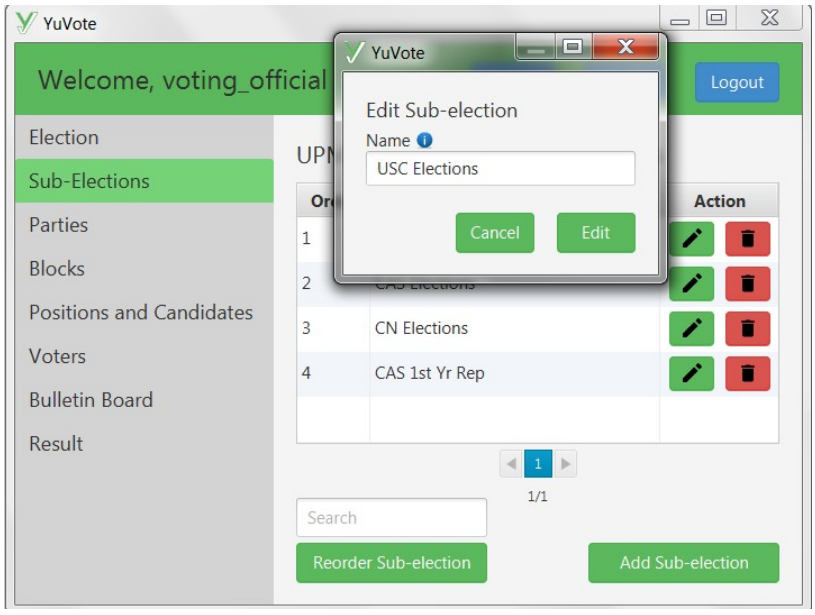

Figure 18: Edit sub-election

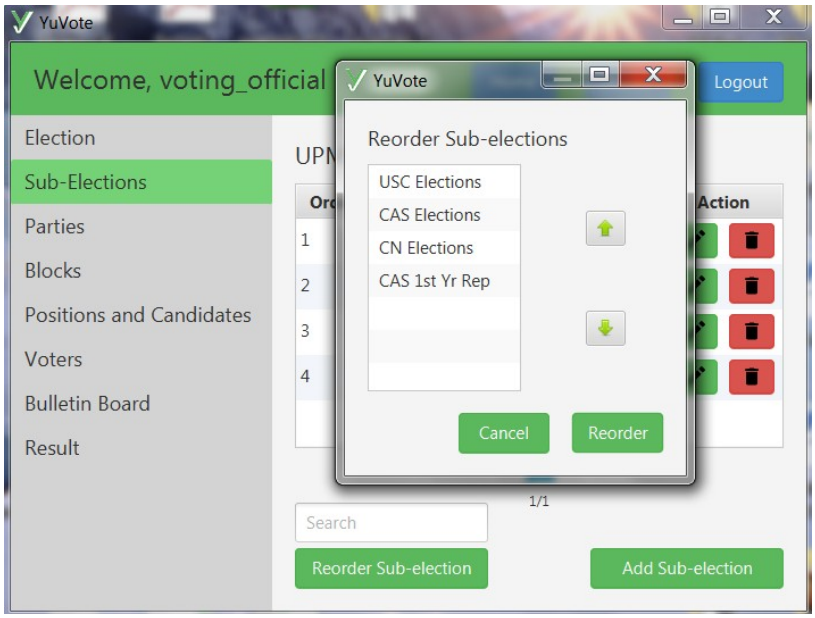

Figure 19: Reorder sub-election

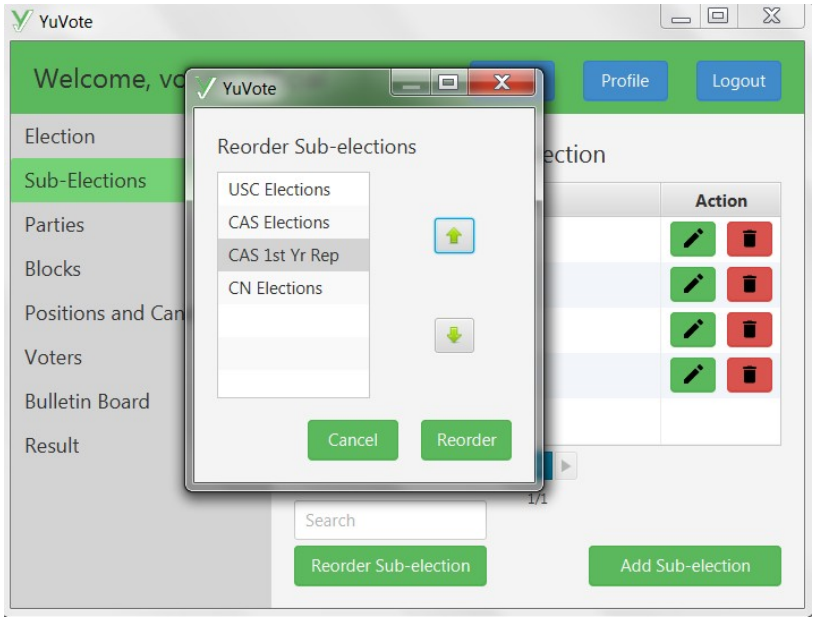

Figure 20: Reordering sub-elections

The party page is where the voting official can add, edit, and delete parties.

Figure 21: Parties page

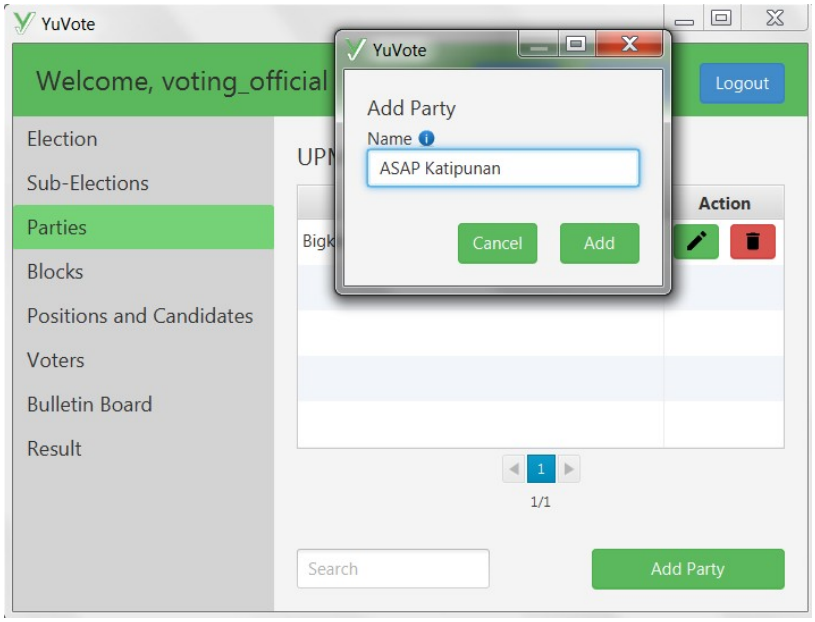

Figure 22: Add party

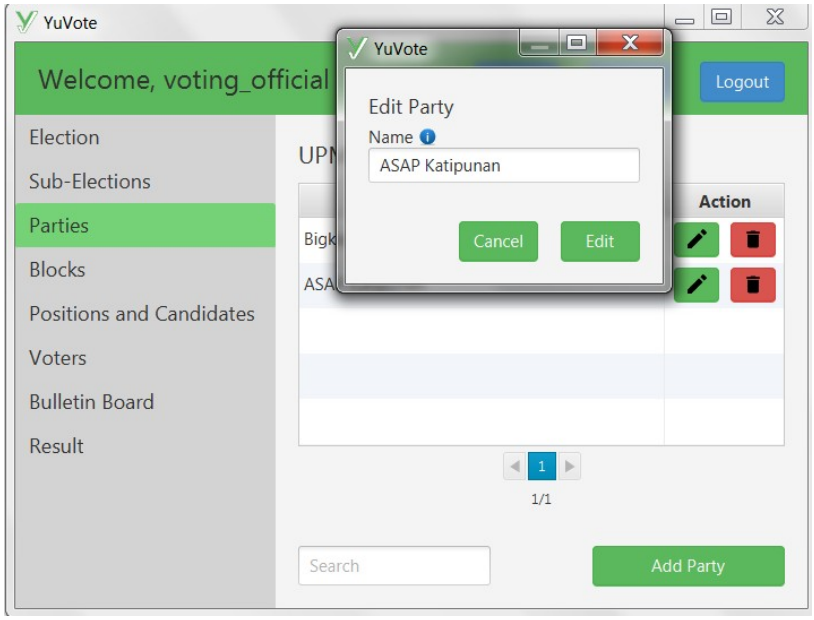

Figure 23: Edit party

The block page is where the voting official can add, edit, and delete blocks. The voting official can view the ballot per block. The voting official can also modify the participation of blocks in sub-elections.

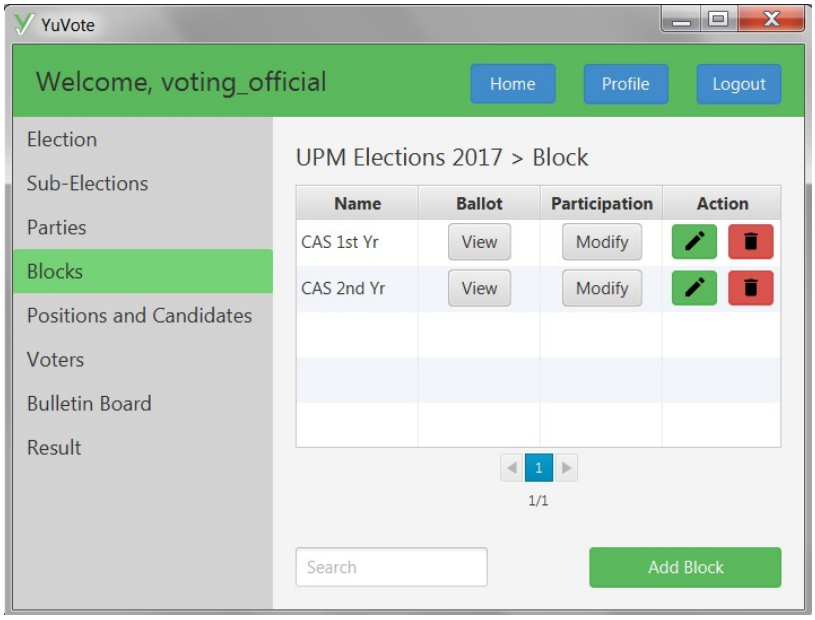

Figure 24: Blocks page

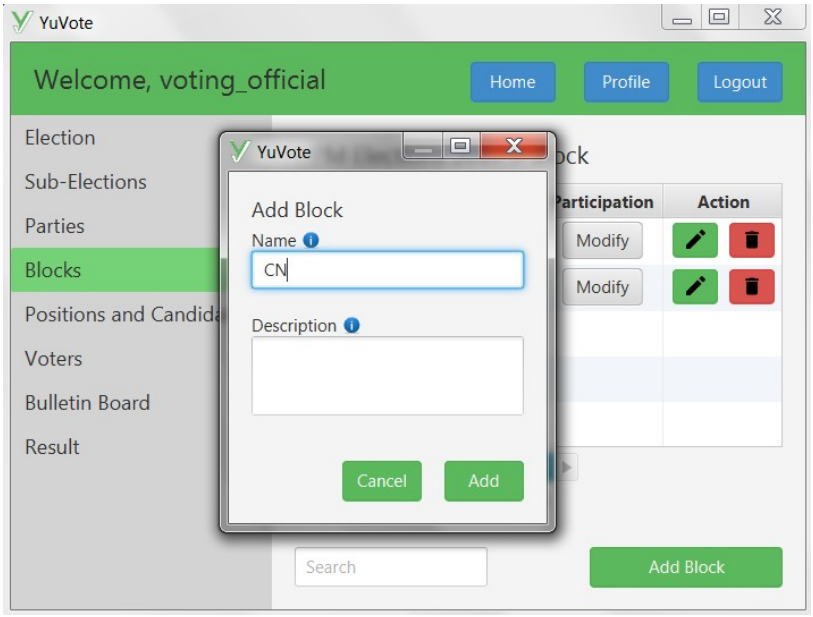

Figure 25: Add block

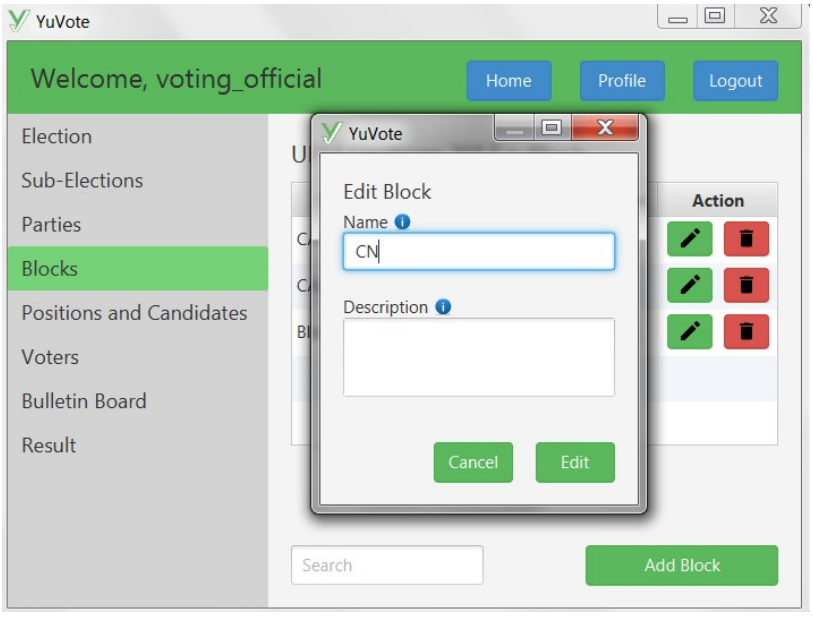

Figure 26: Edit block

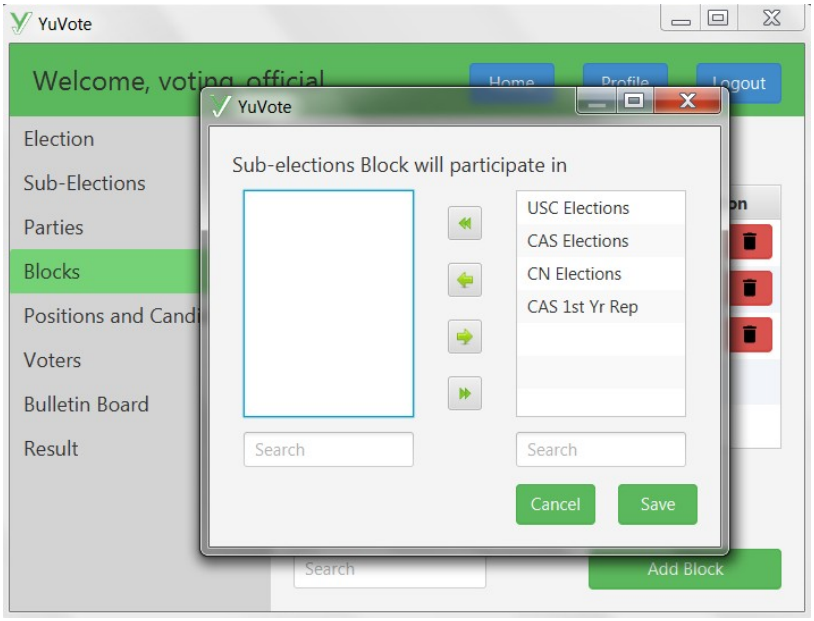

Figure 27: Modify block participation

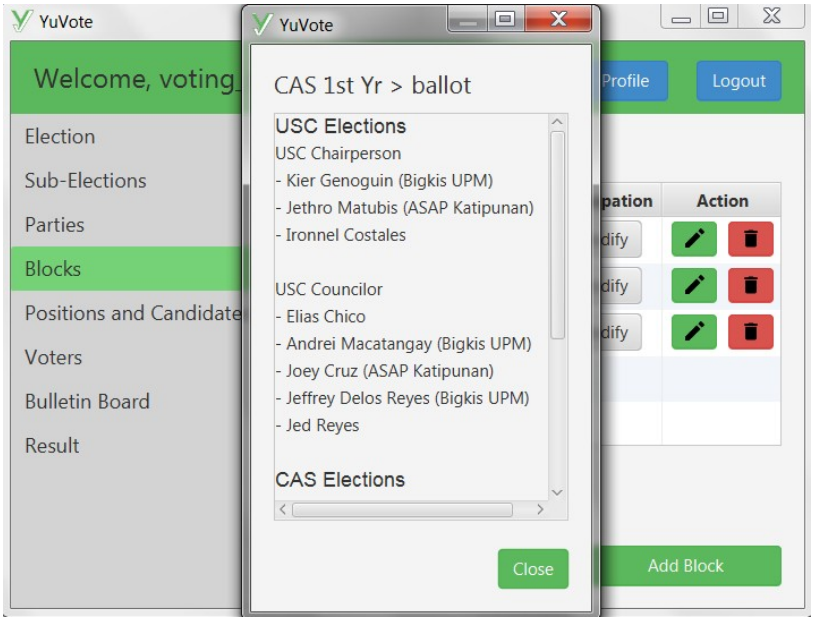

Figure 28: View Block Ballot

The positions and candidates page is where the voting official can add, edit, and delete positions per sub-election. The voting official can also view the candidate page by clicking the view button.

Figure 29: Positions page

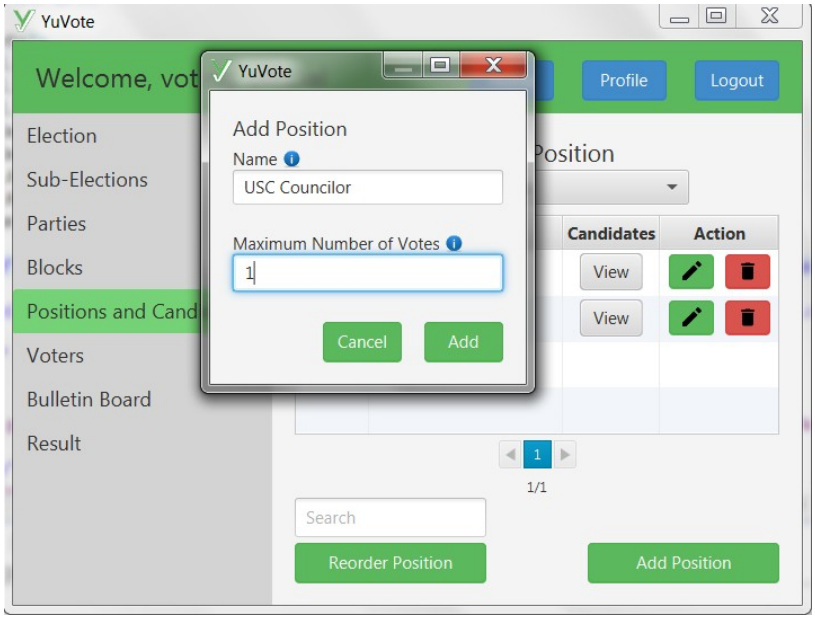

Figure 30: Add position

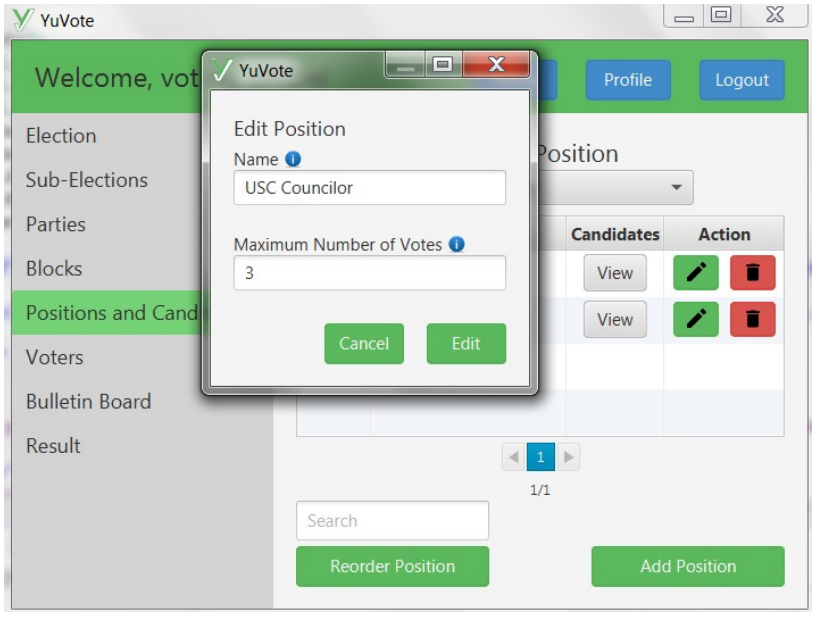

Figure 31: Edit position

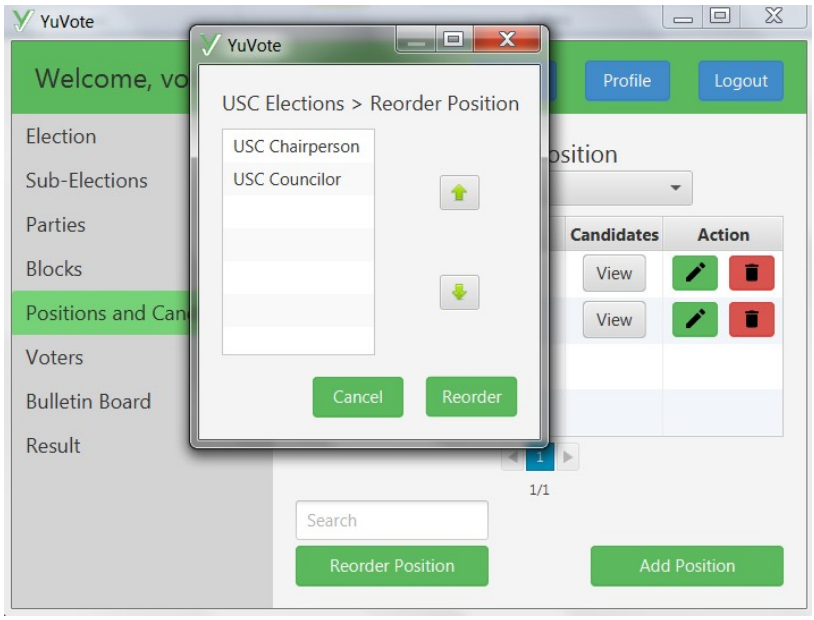

Figure 32: Reorder position

The candidates page is where the voting official can add, edit, and delete candidates per position.

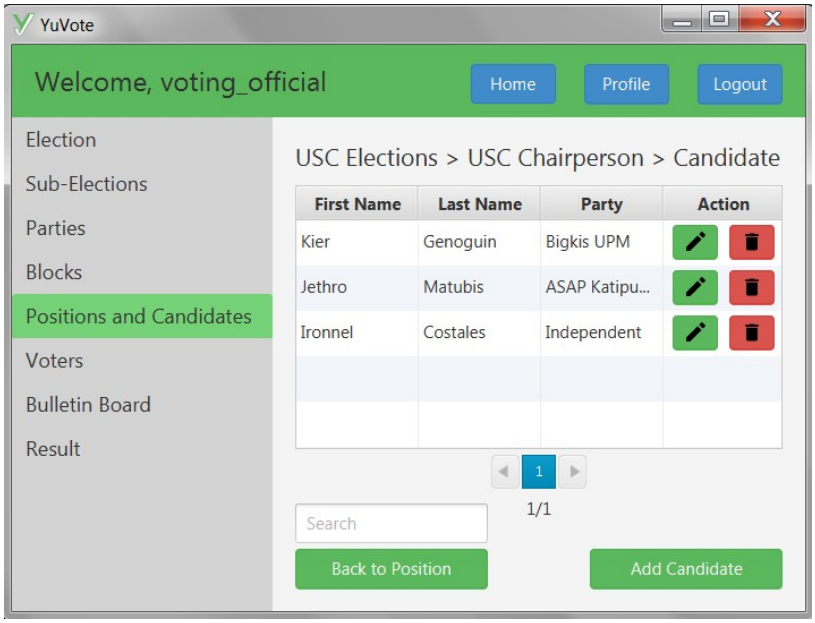

Figure 33: Candidate page

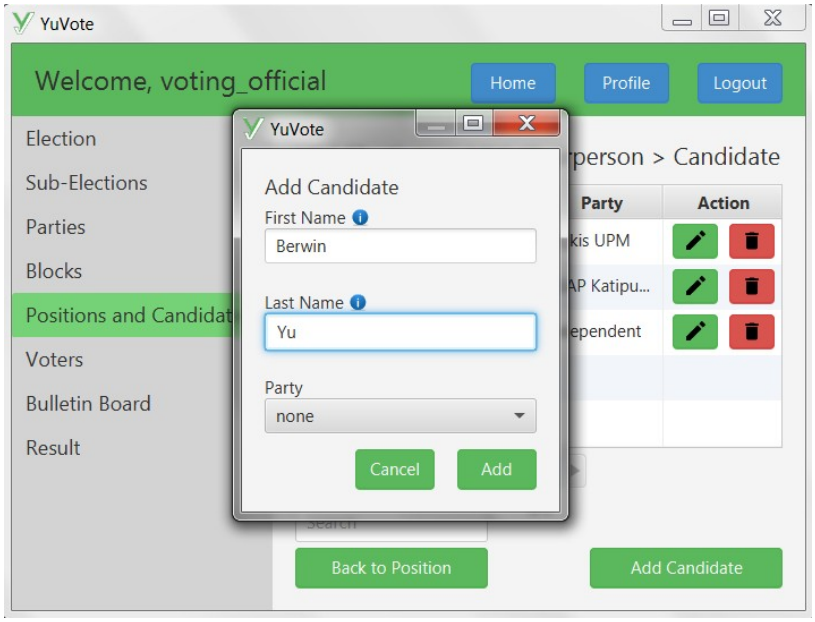

Figure 34: Add candidate

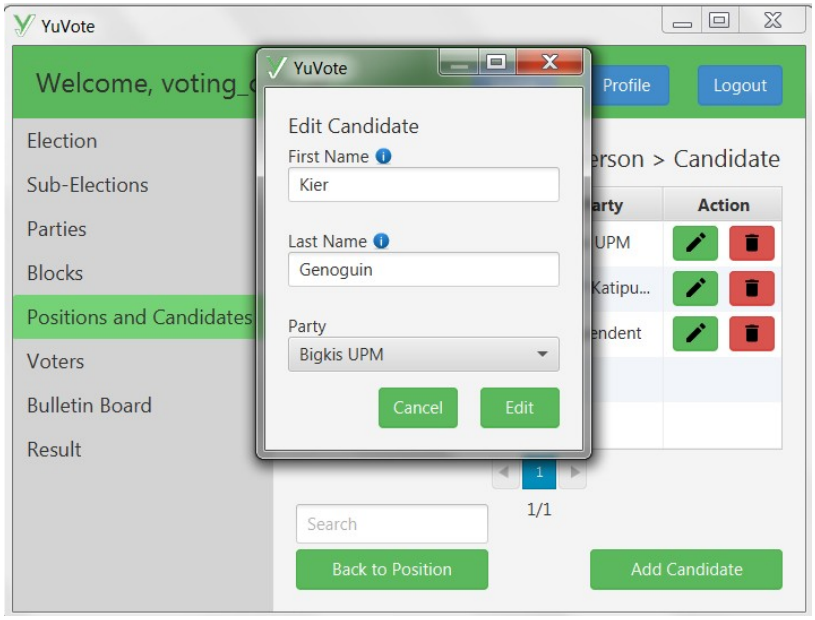

Figure 35: Edit candidate

The voter page is where the voting official can add, bulk add,edit, and delete voters. When the voting official wants to bulk add voters, he needs to add CSV file that contains usernames and emails of voters.

Figure 36: Voters page

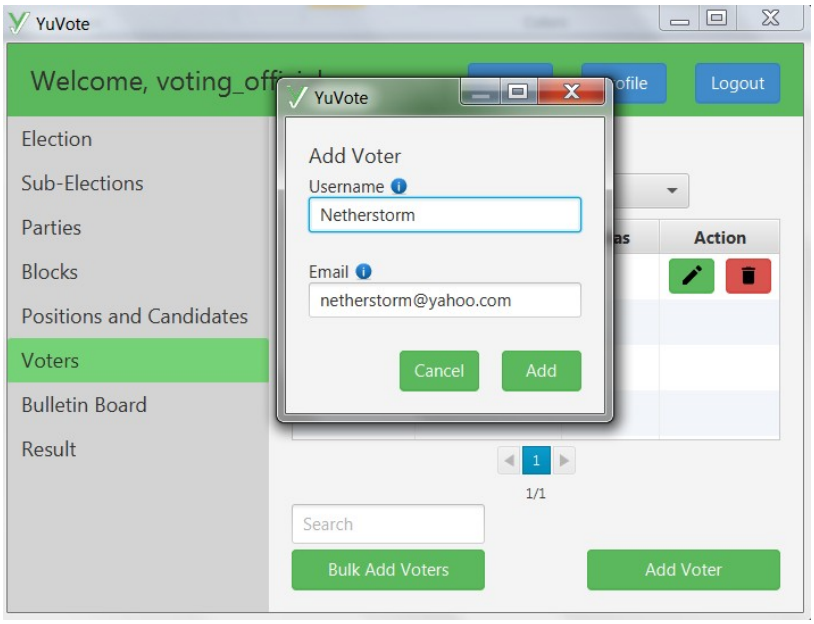

Figure 37: Add voter

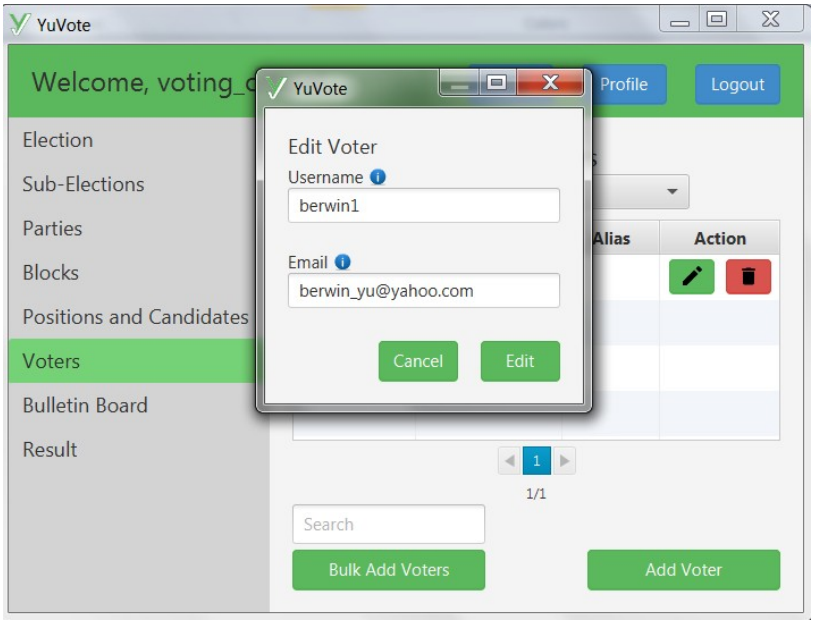

Figure 38: Edit voter

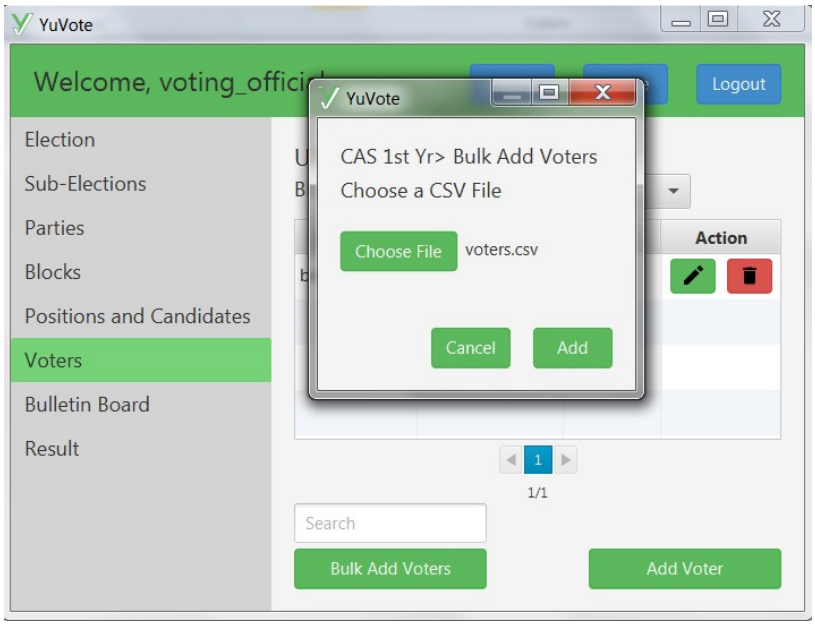

Figure 39: Bulk add voters

Before starting an election, the voting official must first input the number of keys to be used in the election. Once the election is started, the start election button will change to the stop election button.

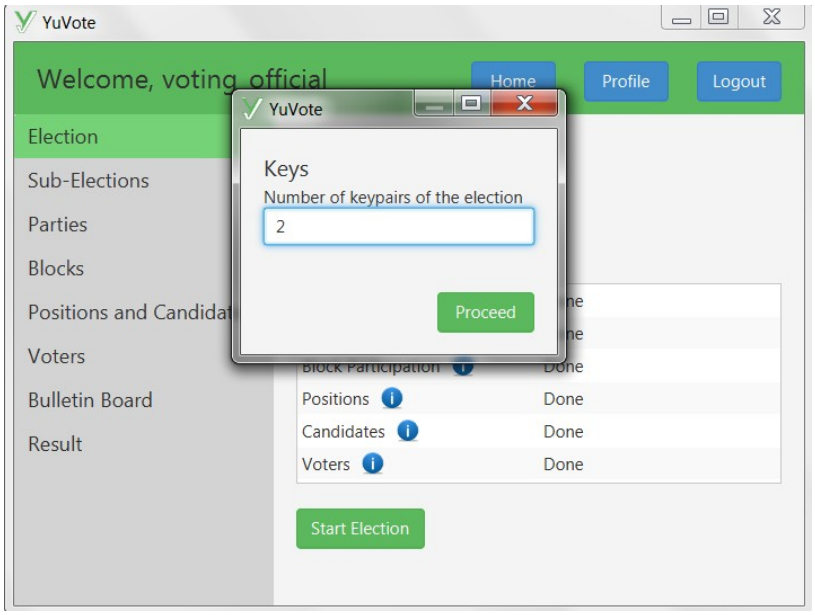

Figure 40: Input number of keys

Figure 41: Main page of the election after the election starts

Once the election is ended, the result page will first ask for the private keys of the election. The page will show a list of private keys that were added. The voter can also delete the private keys. The voting official can press the compute button once all of the private keys are added.

Figure 42: Result page before the private key is given

Figure 43: Choose a private key

Figure 44: Result after a private key is given

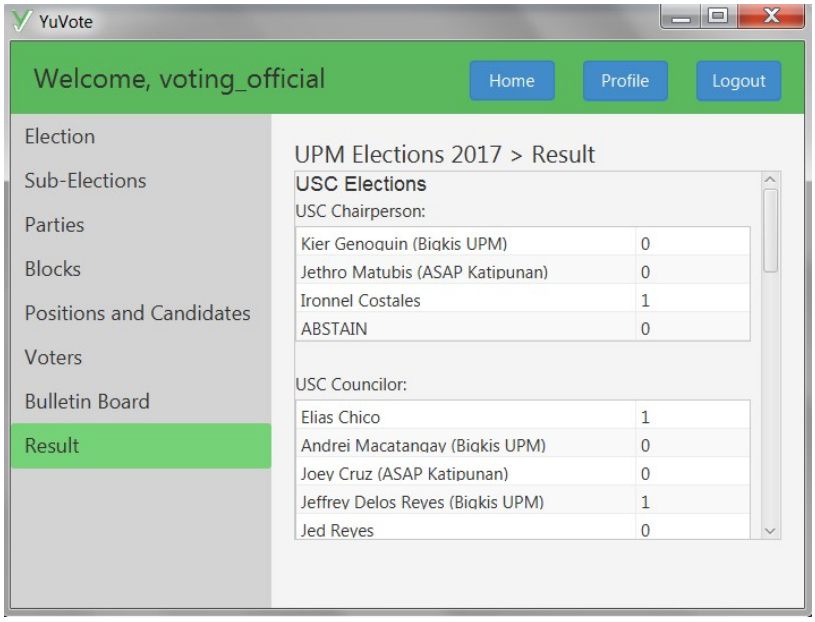

Figure 45: Result page after computing the tally

The bulletin board which contains the verification code can also be viewed after the results are tallied. The voting official can also view the hash codes per voter by clicking the view button.

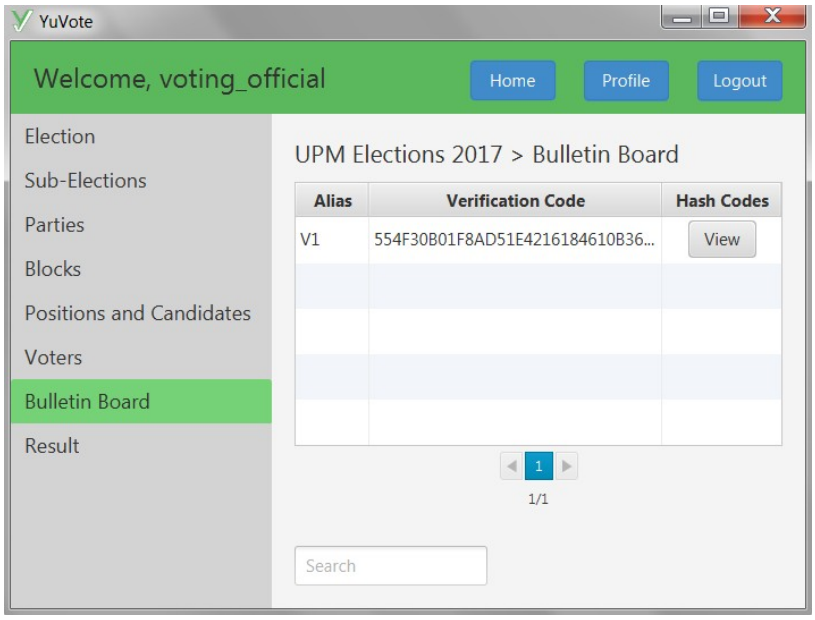

Figure 46: Bulletin board containing the verification codes

The system administrator, as well as the voting official, can edit their account by clicking on the profile button.

Figure 47: Hash codes per voter

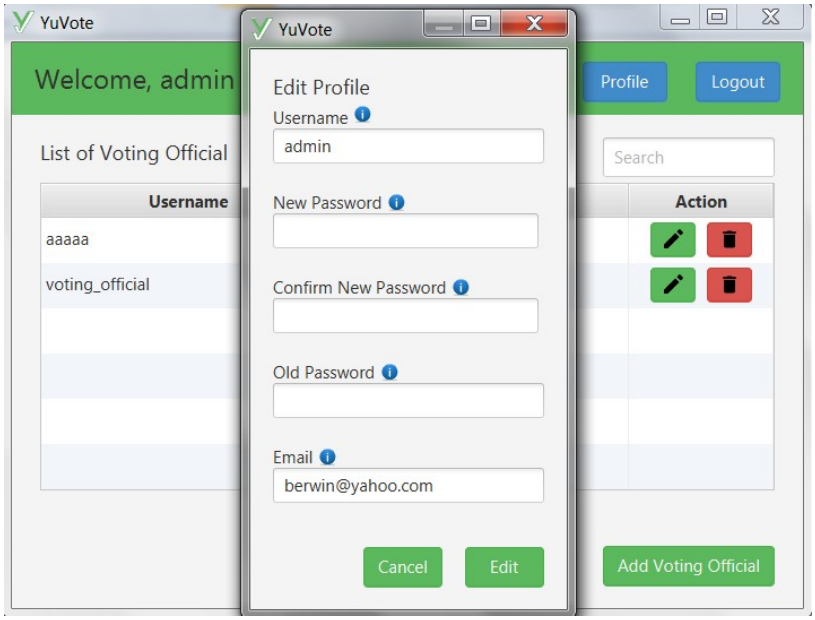

Figure 48: Edit profile

After the system administrator provides the correct credentials, the home page of the system administrator will be shown. The home page contains the list of voting officials. The voting official can add, edit, and delete a voting official.

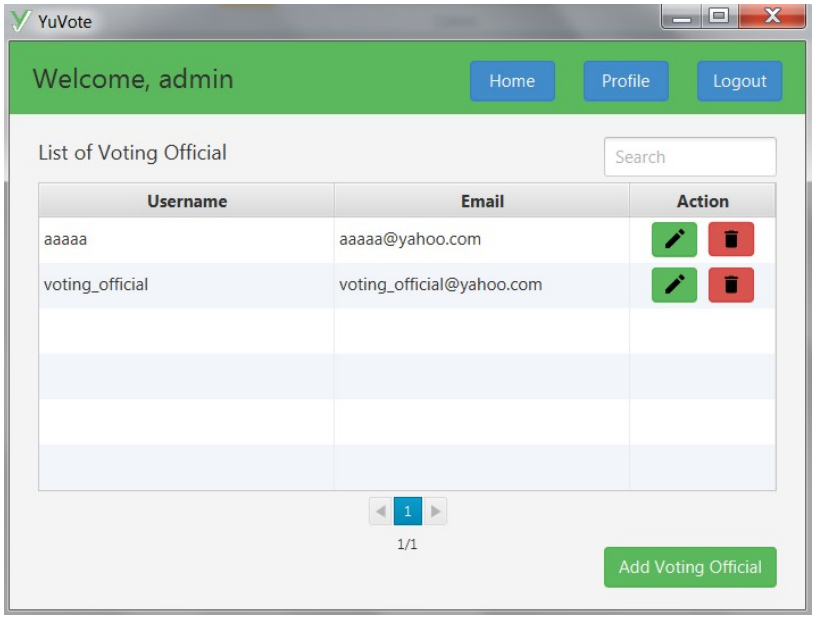

Figure 49: Home page of the system administrator

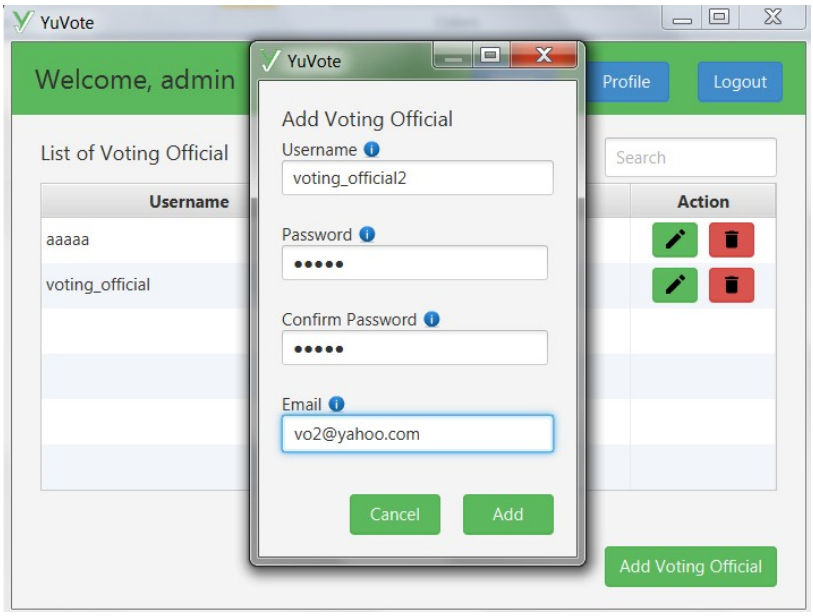

Figure 50: Add a voting official

The figure below shows the login page of the application for the voter.

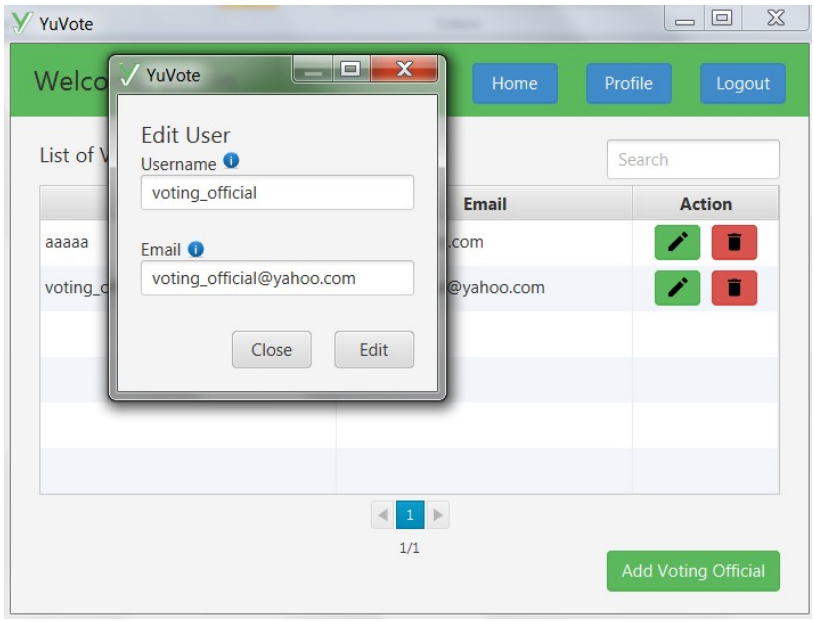

Figure 51: Edit a voting official

Figure 52: Login page of the voter

After the voter provides the correct credentials, the home page of the voter will be shown.

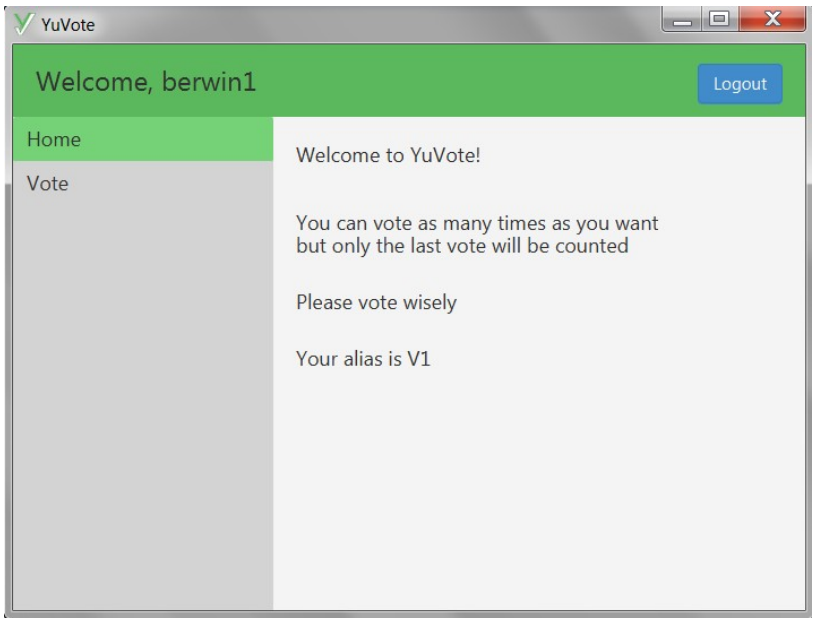

Figure 53: Home page of the voter

The vote page is where the voter votes. When a voter chooses a candidate, the hash code will change

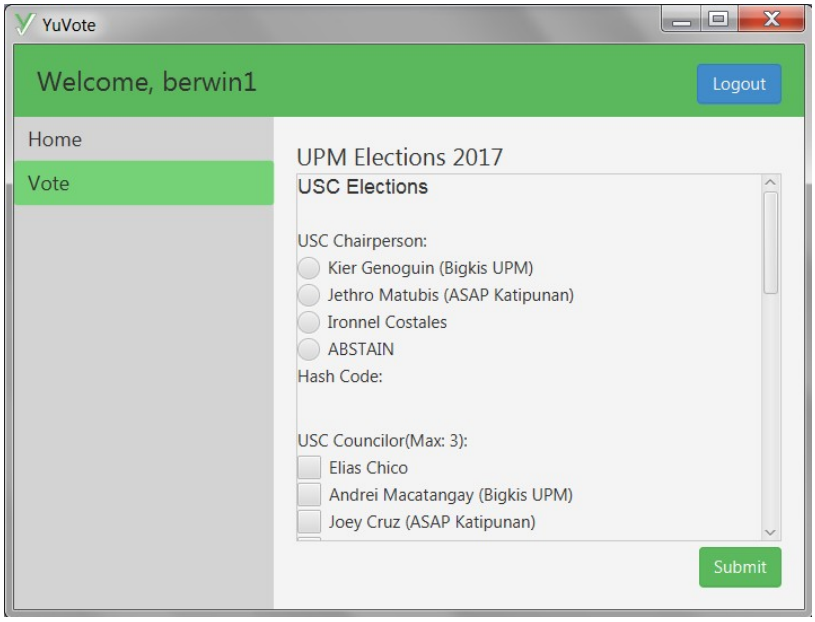

Figure 54: Vote page

When the voter submits his/her vote, he/she will be sent an email that contains the voting receipt. The voter can download the voting receipt from the email and view it.

Figure 55: Hash code changes as a candidate is chosen

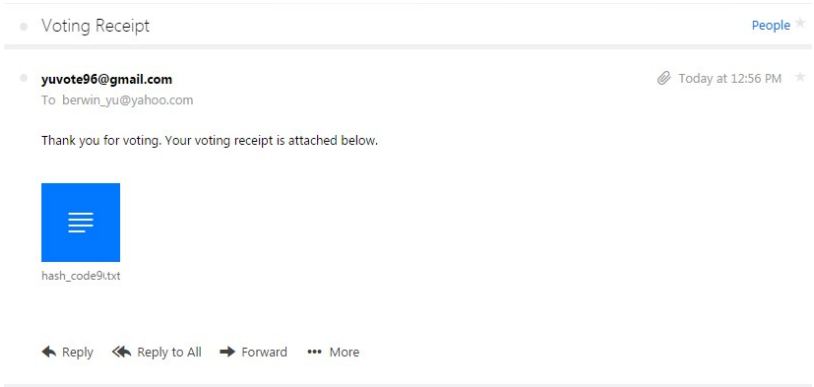

Figure 56: Email sent to the voter

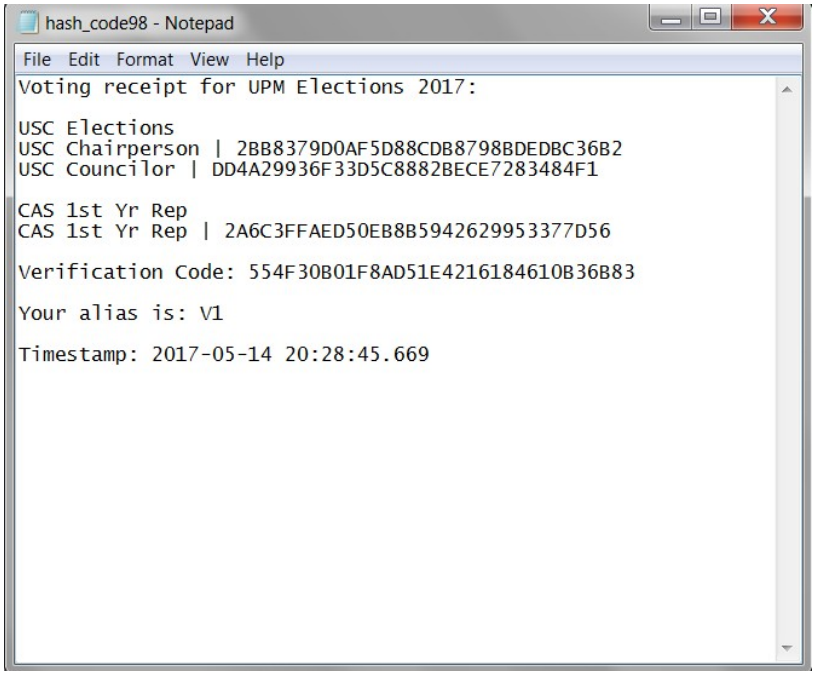

Figure 57: Vote receipt

The webpage of YuVote is a single page that lets the user choose the election name and the type of result the public user wants to see. The type can be the election result or the bulletin board. In the election result, the public user can download the election result in PDF. In the bulletin board, the verification codes are shown and the public user can click on the view button to view the hash codes. This is the page where the voter can upload his vote receipt to verify his vote.

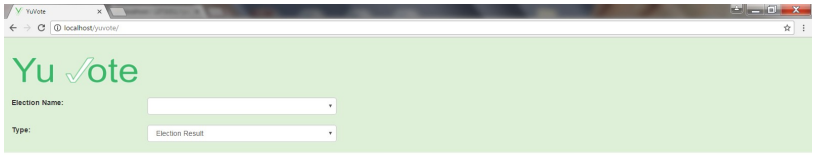

Figure 58: Home page of the web application

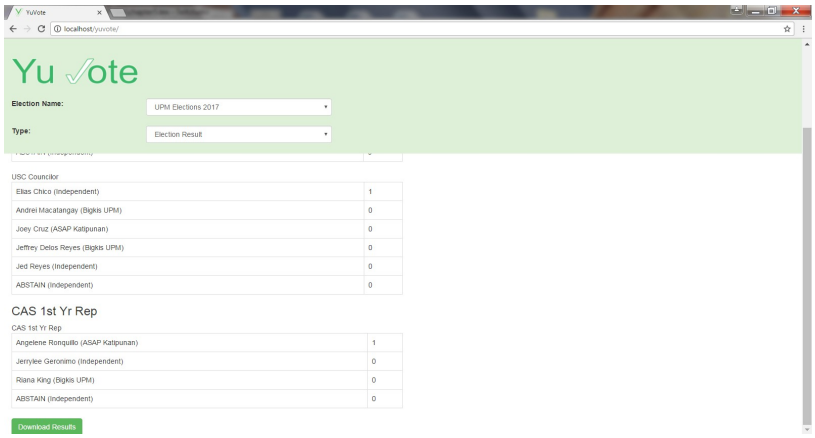

Figure 59: Election result of the web application

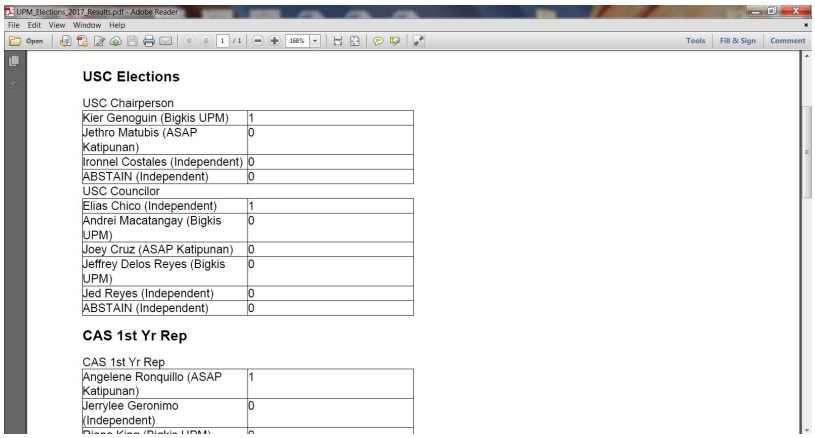

Figure 60: PDF file containing the election results

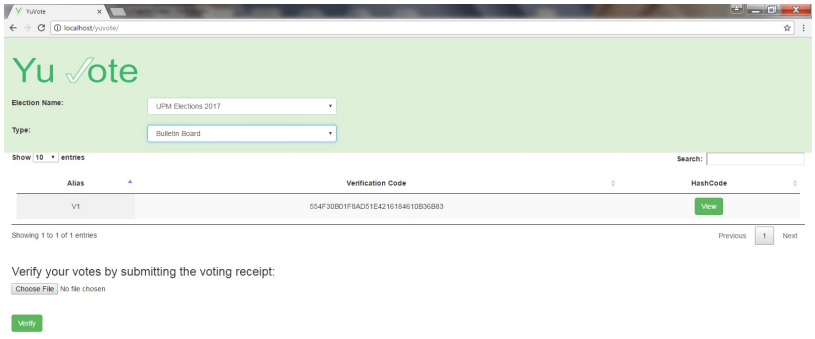

Figure 61: Bulletin board of the web application

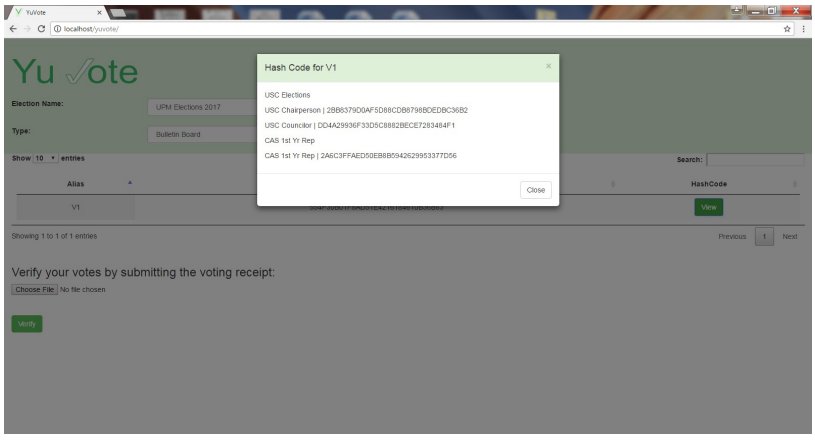

Figure 62: Hash codes of the web application

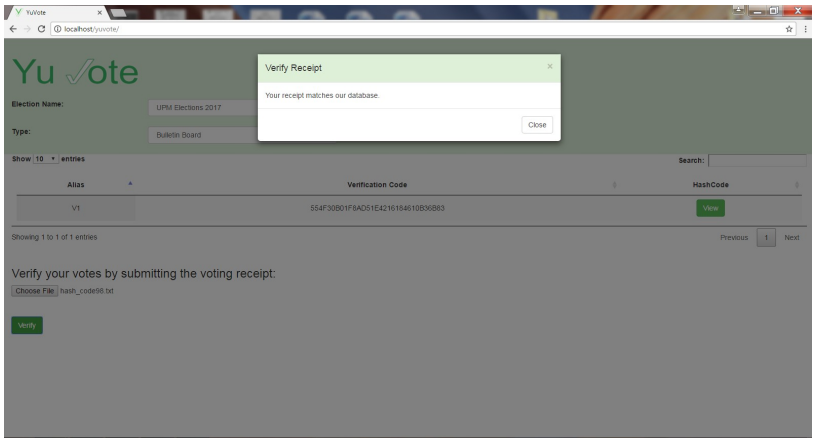

Figure 63: Verify the receipt

VI. Discussions

Initially, we considered two methods in representing the ballot. The first is per candidate vote representation which was used in this work. The second is per position vote representation which is an adaptation of the Paillier [5, 35] vote representation in elliptic curve cryptography.

In per position vote representation, we represent the ballot vector by m ciphertexts where m is the number positions in the ballot. The ciphertext is an encryption of the formula $b * V$ where $b = \sum_{i=1}^{k_j} a_i n^{i-1}$. V is the same as the V in the per candidate vote representation. a_i is 1 if the voter voted for candidate i otherwise it is 0. n is the number of voters using the ballot. k_j is the number of candidates in position j. For example, in a ballot with 10 voters and 3 positions with three candidates in position 1, five candidates in position 2, and four candidates in position 3. If the voter voted for the second candidate in position 1, the fourth candidate in position 2, and the fourth and fifth in position 3, the ballot of the voter would look like

$$C = (Enc(p_k, 10 * V), Enc(p_k, 100 * V), Enc(p_k, 1100 * V)).$$

where p_k is the public key of the election. The public key p_k and the private key s_k are generated during the start of the election using algorithm 1.

The ballot vector is stored in the database the same way as per candidate vote representation. The only difference is that the `candidate_id` field is replaced with `position_id` which is the identifier of the position and this is because a component in the ballot vector now represents the vote for a whole position. This means that a ballot vector with 3 positions would be stored in the database with 3 rows.

T_j would now be the tally for the position j and is obtained using the formula $T_j = \sum_{i=1}^{n_j} C_{j,i}$ where C_j is a vector containing all ciphertexts that correspond to position j and n_j is the number of ciphertexts that correspond to position j or simply

the number of components of C_j. T_j is then decrypted to get R_j. R_j is of the form t_jV. t_j is equal to b with the difference of a_i having different values. a_i will now be the number of votes for candidate i. Getting t_j from t_jV would require a lookup table of x and xV where x is from 1 to $n^{k_{max}}$, n is the number of voters, k_{max} is the maximum number of candidates in a position.

To help us in choosing which method is better, we look at the running time and the space requirement of both methods. The things we considered in the running time are: the homomorphic tallying of the ballots, the decryption of the ballots, and the creation of the lookup table. The space requirement is also considered by looking at the size of the ciphertexts.

In per candidate vote representation, the number of homomorphic addition is nk where n is the number of voters and k is the number of candidates in the election. k can be solved using $\sum_{j=1}^{m} k_j$ where m is the number of positions and k_j is the number of candidates in position j. In per position vote representation, the number of homomorphic addition is nm where m is the number of position. From these formulas, it can be seen that per position vote representation is faster since it requires fewer addition. Three parameters are needed to simulate the homomorphic addition: the number of voters, the number of candidates per position, and the number of position. The number of candidates in each position are assumed to be equal so that the simulation would be easier. This would make the equation $k = mk_j$ true for any j. The result of the simulation is shown in table 14. The results confirm that per position vote representation is faster. The results also show that the number of candidates per position does not affect the running time in per position vote representation.

In per candidate vote representation, the number of decryption operation is k where k is the number of candidates in the election. In per position vote representation, the number of decryption operation is m where m is the number of position. It can be seen that per position vote representation is faster since $m \leq k$. Two param-

Parameters			Per Candidate	Per Position
# of Voters	# of Candidates Per Position	# of Positions	Time(ms)	Time(ms)
1000	5	5	91	18
1000	10	5	163	19
1000	10	10	300	35
5000	5	5	331	73
5000	10	5	601	72
5000	10	10	1193	142

Table 14: Running time of homomorphic addition

eters are needed to simulate the decryption operation: the number of candidates per position and the number of position. The number of candidates in each position are also assumed to be equal so that $k = mk_j$ would be true. The result of the simulation is shown in the table below. The results confirm that per position vote representation is faster. The results also show that the number of candidates per position does not affect the running time.

Parameters		Per Candidate	Per Position
# of Candidates Per Position	# of Positions	Time(ms)	Time(ms)
5	5	8	2
5	10	19	4
10	5	17	2
10	10	33	3

Table 15: Running time of decryption

In per candidate vote representation, the lookup table consists of n rows where n is the number of voters. In per position vote representation, the lookup table consists of $n^{k_{max}}$ rows where n is the number of voters and k_{max} is the maximum number of candidates in a position. It is important to note that the lookup table is only computed once in the election even if there are many positions. Two parameters are needed to simulate the lookup table creation: the number of voters and the number of candidates. The number of candidates in each position are also assumed to be equal so that $k_{max} = k_j$ would be true for any j. One problem in generating the lookup table in per position vote representation is that it increases exponentially as k_{max} increases

and simulating this would take up too much time. To solve this problem, the rate of solving 1 row is computed instead of simulating the whole lookup table creation. To do this, 1 million rows of the lookup table is solved and the resulting time is divided by 1 million to get the rate. The rate is then multiplied to the number of rows required in the test cases. The simulation resulted with a rate of 0.034623ms/row. The result of each test cases is shown in the table below. The results show that per candidate vote representation is very fast compared to per position vote representation. The results in per position vote representation is very big even if the number of voters is little.

Parameters		Per Candidate		Per Position	
# of Voters	# of Candidates	# of Rows	Time(ms)	# of Rows	Time(ms)
100	5	1×10^{10}	3.46×10^{8}	100	3.46
100	10	1×10^{20}	3.46×10^{18}	100	3.46
500	5	3.13×10^{13}	1.08×10^{12}	500	17.31
500	10	9.77×10^{26}	3.38×10^{25}	500	17.31

Table 16: Running time of lookup table creation

It is also important to determine the space required to store a vote in a database. In per candidate vote representation, the required number of rows to store the votes is nk where n is the number of voters, k is the number of candidates. In per position vote representation, required number of rows to store the votes is nm where m is the number of position. The formulas show that more space is required in per candidate vote representation since the number of candidates is always more than the number of positions. We also simulated the size of the ciphertext in both methods. The simulation shows that the size of the ciphertext in both methods do not differ that much.

Based from the simulations above, per position vote representation is better in terms of homomorphic addition, decryption, and space requirement. However, per candidate vote representation is still used in this research due to the fact that the lookup table is a limiting factor in per position vote representation. Creating a

lookup table in per position vote representation would require too much time. An election with 100 voters and 5 candidates would already require 3.4623×10^{10}ms which is roughly 4 days. Per position vote representation would also require additional operations like solving b and solving the a_i's after decryption.

VII. Conclusions

YuVote is a voting system that uses homomorphic encryption to provide privacy and verifiability. Homomorphic encryption allows the votes to be tallied while they are encrypted. This means that the votes do not need to be individually decrypted. Verification codes are provided to voters after they vote so that they can verify that the voting system received their votes correctly.

Elliptic curve cryptography is used in this system because it provides more security with the same key size than other algorithms. Specifically, the Elliptic ElGamal Cryptosystem is used which has a homomorphic property. The system uses the Bouncy Castle which is a Java library that implemented the Elliptic ElGamal Cryptosystem.

Specifically, YuVote is composed two Java applications that use the Bouncy Castle. One is for the system administrator and the voting official, and the other one is for the voter. Java applications are used so that votes can be encrypted on the client side. YuVote is also composed of a web application and its purpose is for the public users to view the election and the public bulletin board.

To conclude, YuVote is a secure and verifiable system. This system shows that it is possible to have an Internet voting that is secure. This system can serve as an inspiration to create a better Internet voting systems for bigger elections.

VIII. Recommendations

It is suggested that the voting system would allow multiple candidates to edit a election. One problem with YuVote is that it cannot handle big elections. This is because a single voting official would have to add every position and every candidate one at a time. Allowing an election to be edited by many voting officials would make it easier to setup a large election.

It is also suggested to have a report system in case the voter receipt did not match the hash codes on the bulletin board. The purpose of this is to notify the voting official if there is a cheating on the election. There could also be a system to nullify the votes. This is to ensure that the tally contains votes that are not compromised.

The user interface can also be improved to provide better user experience. Additionally, process flow of the election can also be improved. The system can also provide statistics like voter turnouts per block, voting time distribution, etc.

IX. Bibliography

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

[2] V. Cortier, "Formal verification of e-voting: solutions and challenges," *ACM SIGLOG News*, vol. 2, pp. 26–34, January 2015.

[3] 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, August 2012.

[4] V. Mateu, J. M. Miret, and F. Sebé, "A hybrid approach to vector-based homomorphic tallying remote voting," *International Journal of Information Security*, vol. 15, pp. 211–221, April 2016.

[5] M. J. Occeño and R. B. Chua, "Using paillier scheme to provide privacy and verifiability in internet voting," in *Proceedings of the 17th Philippine Computing Science Congress*, pp. 229–232, Computing Society of the Philippines.

[6] A. W. Dent, "Choosing key sizes for cryptography," *information security technical report*, vol. 15, no. 1, pp. 21–27, 2010.

[7] V. Kapoor, V. S. Abraham, and R. Singh, "Elliptic curve cryptography," *Ubiquity*, vol. 2008, no. May, p. 7, 2008.

[8] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[9] C. Jost, H. Lam, A. Maximov, and B. J. Smeets, "Encryption performance improvements of the paillier cryptosystem.," *IACR Cryptology ePrint Archive*, vol. 2015, p. 864, 2015.

[10] N. Jansma and B. Arrendondo, "Performance comparison of elliptic curve and rsa digital signatures." http://www.nicj.net/files/performance_comparison_of_elliptic_curve_and_rsa_digital_signatures.pdf. Accessed: 2017-05-24.

[11] "The pros and cons of online voting." https://followmyvote.com/pros-cons-online-voting/. Accessed: 2016-10-2.

[12] "The future of voting end-to-end verifiable internet voting." https://www.usvotefoundation.org/sites/default/files/E2EVIV_expert_statements.pdf. Accessed: 2016-11-5.

[13] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J. A. Halderman, "Security analysis of the Estonian internet voting system," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 703–715, ACM, 2014.

[14] K. Gjøsteen, "The Norwegian internet voting protocol," in *International Conference on E-Voting and Identity*, pp. 1–18, Springer, 2011.

[15] "An overview of the iVote 2015 voting system." https://www.elections.nsw.gov.au/__data/assets/pdf_file/0019/204058/An_overview_of_the_iVote_2015_voting_system_v4.pdf. Accessed: 2016-11-9.

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

[17] "Lecture 18: Mix-net voting systems." http://courses.csail.mit.edu/6.897/spring04/L18.pdf. Accessed: 2016-9-19.

[18] R. Joaquim, "How to prove the validity of a complex ballot encryption to the voter and the public," *Journal of Information Security and Applications*, pp. 130–142, April 2014.

[19] D. Galindo, S. Guasch, and J. Puiggali, "2015 Neuchâtels cast-as-intended verification mechanism," in *International Conference on E-Voting and Identity*, pp. 3–18, Springer, 2015.

[20] B. Adida, "Helios: Web-based open-audit voting.," in *USENIX security symposium*, vol. 17, pp. 335–348, 2008.

[21] J. Benaloh, "Simple verifiable elections.," *EVT'06 Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, vol. 6, pp. 5–5, 2006.

[22] O. Kulyk, V. Teague, and M. Volkamer, "Extending Helios towards private eligibility verifiability," in *International Conference on E-Voting and Identity*, pp. 57–73, Springer, 2015.

[23] C. Culnane, P. Y. Ryan, S. Schneider, and V. Teague, "vVote: a verifiable voting system," *ACM Transactions on Information and System Security (TISSEC)*, vol. 18, no. 1, p. 3, 2015.

[24] S. Bell, J. Benaloh, M. D. Byrne, D. Debeauvoir, B. Eakin, P. Kortum, N. McBurnett, O. Pereira, P. B. Stark, D. S. Wallach, G. Fisher, J. Montoya, M. Parker, and M. Winn, "STAR-Vote: A secure, transparent, auditable, and reliable voting system," in *2013 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 13)*, (Washington, D.C.), USENIX Association, Aug. 2013.

[25] G. Z. Qadah and R. Taha, "Electronic voting systems: Requirements, design, and implementation," *Computer Standards & Interfaces*, vol. 29, no. 3, pp. 376–386, 2007.

[26] "Electronic voting technologies." http://www.bravenewballot.org/. Accessed: 2016-10-24.

[27] L. J. Aslett, P. M. Esperança, and C. C. Holmes, "A review of homomorphic encryption and software tools for encrypted statistical machine learning," *arXiv preprint arXiv:1508.06574*, 2015.

[28] L. Morris, "Analysis of partially and fully homomorphic encryption," 2013.

[29] G. C. Kessler, "An overview of cryptography," 2003.

[30] "Advanced cryptography." https://courses.maths.ox.ac.uk/node/view_material/1963, note = Accessed: 2017-5-25,.

[31] M. Joye and B. Libert, "Encoding-free elgamal-type encryption schemes on elliptic curves," in *Cryptographers Track at the RSA Conference*, pp. 19–35, Springer, 2017.

[32] "Symmetric encryption." https://cseweb.ucsd.edu/ mihir/cse107/w-se.pdf. Accessed: 2017-5-25.

[33] "The legion of the bouncy castle." http://www.bouncycastle.org/. Accessed: 2016-10-24.

[34] "Safecurves:choosing safe curves for elliptic-curve cryptography." https://safecurves.cr.yp.to/index.html. Accessed: 2017-5-20.

[35] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223–238, Springer, 1999.

X. Appendix

A Definition of Terms

1. Sub-election - a mini election that has its own set of positions and candidates
2. Block - a voter type, each block participates in different sets of sub-elections
3. Ballot - contains a list of positions and candidates that a voter can vote, it is unique per block
4. Vote receipt - a text file that contains the verification code and the hash codes per position of a voter
5. Bulletin board - contains the verification codes and hash codes of those who voted

B Source Code

B.1 System Administrator and Voting Official Application

MainApp.java

```
package ph.edu.upm.cas.dpsm.bjyu;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;

import javax.xml.bind.DatatypeConverter;

import org.bouncycastle.asn1.x9.X9ECParameters;
import org.bouncycastle.crypto.AsymmetricCipherKeyPair;
import org.bouncycastle.crypto.ec.CustomNamedCurves;
import org.bouncycastle.crypto.ec.ECElGamalDecryptor;
import org.bouncycastle.crypto.ec.ECPair;
import org.bouncycastle.crypto.generators.ECKeyPairGenerator;
import org.bouncycastle.crypto.params.ECDomainParameters;
import org.bouncycastle.crypto.params.
        ECKeyGenerationParameters;
import org.bouncycastle.crypto.params.ECPrivateKeyParameters
        ;
import org.bouncycastle.crypto.params.ECPublicKeyParameters;
import org.bouncycastle.math.ec.ECPoint;

import javafx.application.Application;
import javafx.application.Platform;
import javafx.concurrent.Task;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.BorderPane;
import javafx.stage.DirectoryChooser;
import javafx.stage.FileChooser;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ActionBlockController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ActionCandidateController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ActionElectionController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ActionPartyController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ActionPositionController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ActionSubElectionController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ActionVoterController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        AddVotingOfficialController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        EditProfileController;
import ph.edu.upm.cas.dpsm.bjyu.controller.EditUserController;
import ph.edu.upm.cas.dpsm.bjyu.controller.HomeController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        InputNumberKeysController;
import ph.edu.upm.cas.dpsm.bjyu.controller.LoginController;
import ph.edu.upm.cas.dpsm.bjyu.controller.LogoutController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ReorderPositionController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ReorderSubElectionController;
import ph.edu.upm.cas.dpsm.bjyu.controller.ResultController;
```

```
import ph.edu.upm.cas.dpsm.bjyu.controller.
        StartElectionController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        TallyElectionController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ViewBallotController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        ChoosePrivateKeyController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        BlockParticipationController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
        BulkAddVotersController;
import ph.edu.upm.cas.dpsm.bjyu.model.Block;
import ph.edu.upm.cas.dpsm.bjyu.model.BulletinBoard;
import ph.edu.upm.cas.dpsm.bjyu.model.Candidate;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Party;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.model.User;
import ph.edu.upm.cas.dpsm.bjyu.model.Voter;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.EmailHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PasswordHelper;

public class MainApp extends Application {
 public Stage primaryStage;
 private User currentUser;
 private HomeController homeController;
 private boolean correctPrivateKey;
 private boolean emailSuccess;

 @Override
 public void start(Stage primaryStage) {
  this.primaryStage = primaryStage;
  this.primaryStage.setTitle("YuVote");
  primaryStage.getIcons().add(new Image(getClass().
          getResourceAsStream("images/YuVote Logo.png")));
  try {
   FXMLLoader loader = new FXMLLoader();
   loader.setLocation(MainApp.class.getResource("view/Login.
           fxml"));
   AnchorPane loginFXML = (AnchorPane) loader.load();

   Scene scene = new Scene(loginFXML);

   primaryStage.setScene(scene);
   primaryStage.show();

   LoginController loginController = loader.getController();
   loginController.setup();
   loginController.setMainApp(this);
  } catch (IOException e) {
   e.printStackTrace();
  }
 }

 public void setCurrentUser(User user) {
  currentUser = user;
 }

 public User getCurrentUser() {
  return currentUser;
 }

 public void showLogin() {
  try {
   FXMLLoader loader = new FXMLLoader();
   loader.setLocation(MainApp.class.getResource("view/Login.
           fxml"));
   AnchorPane loginFXML = (AnchorPane) loader.load();

   Scene scene = new Scene(loginFXML);

   primaryStage.setScene(scene);
   primaryStage.show();

   LoginController loginController = loader.getController();
   loginController.setup();
   loginController.setMainApp(this);
  } catch (IOException e) {
   e.printStackTrace();
  }
 }

 public void showHome() {
  try {
   FXMLLoader loader = new FXMLLoader();
   loader.setLocation(MainApp.class.getResource("view/Home.
           fxml"));
   BorderPane homeFXML = (BorderPane) loader.load();

   Scene scene = new Scene(homeFXML);

   primaryStage.setScene(scene);
   primaryStage.show();

   homeController = loader.getController();
   setHomeControllerUser();
   homeController.setMainApp(this);
   if (currentUser.getType().equals("System Admin"))
    homeController.showSAMainView();
   if (currentUser.getType().equals("Voting Official")) {
    homeController.showVOMainView();
   }
  } catch (IOException ioe) {
   ioe.printStackTrace();
  }

 }

 public void setHomeControllerUser() {
  homeController.setUser(currentUser);
 }

 public void showLogout() {
  try {
   FXMLLoader loader = new FXMLLoader();
   loader.setLocation(MainApp.class.getResource("view/Logout.
           fxml"));
   AnchorPane logoutFXML = (AnchorPane) loader.load();

   Scene scene = new Scene(logoutFXML);
   Stage secondaryStage = new Stage();
   secondaryStage.setTitle("YuVote");
   secondaryStage.getIcons().add(new Image(getClass().
           getResourceAsStream("images/YuVote Logo.png")));
   secondaryStage.initModality(Modality.WINDOW_MODAL);
   secondaryStage.initOwner(primaryStage);
   secondaryStage.setScene(scene);
   secondaryStage.show();

   LogoutController logoutController = loader.getController();
   logoutController.setMainApp(this);
  } catch (IOException ioe) {
   ioe.printStackTrace();
  }
 }

 public void showEditUser(User editUser) {
  try {
   FXMLLoader loader = new FXMLLoader();
   loader.setLocation(MainApp.class.getResource("view/
           EditUser.fxml"));
   AnchorPane editUserFXML = (AnchorPane) loader.load();

   Scene scene = new Scene(editUserFXML);
   Stage secondaryStage = new Stage();
   secondaryStage.setTitle("YuVote");
   secondaryStage.getIcons().add(new Image(getClass().
           getResourceAsStream("images/YuVote Logo.png")));
   secondaryStage.initModality(Modality.WINDOW_MODAL);
   secondaryStage.initOwner(primaryStage);
   secondaryStage.setScene(scene);
   secondaryStage.show();

   EditUserController editUserController = loader.getController
           ();
   editUserController.setUser(editUser);
   editUserController.setup();
  } catch (IOException ioe) {
   ioe.printStackTrace();
  }
 }

 public void showAddVotingOfficial(String mode) {
  try {
   FXMLLoader loader = new FXMLLoader();
   loader.setLocation(MainApp.class.getResource("view/
           AddVotingOfficial.fxml"));
   AnchorPane addVotingOfficialFXML = (AnchorPane) loader.
           load();

   Scene scene = new Scene(addVotingOfficialFXML);
   Stage secondaryStage = new Stage();
   secondaryStage.setTitle("YuVote");
   secondaryStage.getIcons().add(new Image(getClass().
           getResourceAsStream("images/YuVote Logo.png")));
   secondaryStage.initModality(Modality.WINDOW_MODAL);
   secondaryStage.initOwner(primaryStage);
   secondaryStage.setScene(scene);
   secondaryStage.show();
```

```
  AddVotingOfficialController addVotingOfficialController =
        loader.getController();
  addVotingOfficialController.setMode(mode);
  addVotingOfficialController.setup();
  addVotingOfficialController.setMainApp(this);
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showEditProfile() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        EditProfile.fxml"));
  AnchorPane editProfileFXML = (AnchorPane) loader.load();

  Scene scene = new Scene(editProfileFXML);
  Stage secondaryStage = new Stage();
  secondaryStage.setTitle("YuVote");
  secondaryStage.getIcons().add(new Image(getClass().
        getResourceAsStream("images/YuVote Logo.png")));
  secondaryStage.initModality(Modality.WINDOW_MODAL);
  secondaryStage.initOwner(primaryStage);
  secondaryStage.setScene(scene);
  secondaryStage.show();

  EditProfileController editProfileController = loader.
        getController();
  editProfileController.setUser(currentUser);
  editProfileController.setMainApp(this);
  editProfileController.setup();
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showActionElection(Election election, String mode)
      {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        ActionElection.fxml"));
  AnchorPane editElection = (AnchorPane) loader.load();

  Scene scene = new Scene(editElection);
  Stage secondaryStage = new Stage();
  secondaryStage.setTitle("YuVote");
  secondaryStage.getIcons().add(new Image(getClass().
        getResourceAsStream("images/YuVote Logo.png")));
  secondaryStage.initModality(Modality.WINDOW_MODAL);
  secondaryStage.initOwner(primaryStage);
  secondaryStage.setScene(scene);
  secondaryStage.show();

  ActionElectionController actionElectionController = loader.
        getController();
  actionElectionController.setMainApp(this);
  actionElectionController.setup();
  if (mode.equals("edit"))
   actionElectionController.setElection(election);
  actionElectionController.setMode(mode);
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showActionParty(Party party, Election election,
      String mode) {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        ActionParty.fxml"));
  AnchorPane actionPartyFXML = (AnchorPane) loader.load();

  Scene scene = new Scene(actionPartyFXML);
  Stage secondaryStage = new Stage();
  secondaryStage.setTitle("YuVote");
  secondaryStage.getIcons().add(new Image(getClass().
        getResourceAsStream("images/YuVote Logo.png")));
  secondaryStage.initModality(Modality.WINDOW_MODAL);
  secondaryStage.initOwner(primaryStage);
  secondaryStage.setScene(scene);
  secondaryStage.show();

  ActionPartyController actionPartyController = loader.
        getController();
  actionPartyController.setMainApp(this);
  if (mode.equals("edit"))
   actionPartyController.setParty(party);
  actionPartyController.setElection(election);
  actionPartyController.setMode(mode);
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showActionBlock(Block block, Election election,
      String mode) {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        ActionBlock.fxml"));
  AnchorPane actionBlockFXML = (AnchorPane) loader.load();

  Scene scene = new Scene(actionBlockFXML);
  Stage secondaryStage = new Stage();
  secondaryStage.setTitle("YuVote");
  secondaryStage.getIcons().add(new Image(getClass().
        getResourceAsStream("images/YuVote Logo.png")));
  secondaryStage.initModality(Modality.WINDOW_MODAL);
  secondaryStage.initOwner(primaryStage);
  secondaryStage.setScene(scene);
  secondaryStage.show();

  ActionBlockController actionBlockController = loader.
        getController();
  actionBlockController.setMainApp(this);
  if (mode.equals("edit"))
   actionBlockController.setBlock(block);
  actionBlockController.setElection(election);
  actionBlockController.setMode(mode);
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showActionPosition(Position position, SubElection
      subElection, String mode) {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        ActionPosition.fxml"));
  AnchorPane actionPositionFXML = (AnchorPane) loader.load
        ();

  Scene scene = new Scene(actionPositionFXML);
  Stage secondaryStage = new Stage();
  secondaryStage.setTitle("YuVote");
  secondaryStage.getIcons().add(new Image(getClass().
        getResourceAsStream("images/YuVote Logo.png")));
  secondaryStage.initModality(Modality.WINDOW_MODAL);
  secondaryStage.initOwner(primaryStage);
  secondaryStage.setScene(scene);
  secondaryStage.show();

  ActionPositionController actionPositionController = loader.
        getController();
  actionPositionController.setMainApp(this);
  if (mode.equals("edit"))
   actionPositionController.setPosition(position);
  actionPositionController.setSubElection(subElection);
  actionPositionController.setMode(mode);
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showManageElection(Election election) {
 homeController.showVOElectionView(election);
}

public void showParticipation(Block block) {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        BlockParticipation.fxml"));
  AnchorPane blockParticipationFXML = (AnchorPane) loader.
        load();

  Scene scene = new Scene(blockParticipationFXML);
  Stage secondaryStage = new Stage();
  secondaryStage.setTitle("YuVote");
  secondaryStage.getIcons().add(new Image(getClass().
        getResourceAsStream("images/YuVote Logo.png")));
  secondaryStage.initModality(Modality.WINDOW_MODAL);
  secondaryStage.initOwner(primaryStage);
  secondaryStage.setScene(scene);
  secondaryStage.show();

  BlockParticipationController blockParticipationController =
        loader.getController();
  blockParticipationController.setMainApp(this);
  blockParticipationController.setBlock(block);
```

```
    blockParticipationController.setData();
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }
 }

 public void showCandidate(Position position, SubElection
       subElection) {
  homeController.showCandidate(position, subElection);
 }

 public void showActionCandidate(Candidate candidate,
       Position position, SubElection subElection, String mode)
       {
  try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/
          ActionCandidate.fxml"));
    AnchorPane actionCandidateFXML = (AnchorPane) loader.
          load();

    Scene scene = new Scene(actionCandidateFXML);
    Stage secondaryStage = new Stage();
    secondaryStage.setTitle("YuVote");
    secondaryStage.getIcons().add(new Image(getClass().
          getResourceAsStream("images/YuVote Logo.png")));
    secondaryStage.initModality(Modality.WINDOW_MODAL);
    secondaryStage.initOwner(primaryStage);
    secondaryStage.setScene(scene);
    secondaryStage.show();

    ActionCandidateController actionCandidateController =
          loader.getController();
    actionCandidateController.setMainApp(this);
    if (mode.equals("edit"))
     actionCandidateController.setCandidate(candidate);
    actionCandidateController.setSubElection(subElection);
    actionCandidateController.setPosition(position);
    actionCandidateController.setMode(mode);
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }
 }

 public void showActionSubElection(SubElection subElection,
       Election election, String mode) {
  try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/
          ActionSubElection.fxml"));
    AnchorPane actionSubElectionFXML = (AnchorPane) loader.
          load();

    Scene scene = new Scene(actionSubElectionFXML);
    Stage secondaryStage = new Stage();
    secondaryStage.setTitle("YuVote");
    secondaryStage.getIcons().add(new Image(getClass().
          getResourceAsStream("images/YuVote Logo.png")));
    secondaryStage.initModality(Modality.WINDOW_MODAL);
    secondaryStage.initOwner(primaryStage);
    secondaryStage.setScene(scene);
    secondaryStage.show();

    ActionSubElectionController actionSubElectionController =
          loader.getController();
    actionSubElectionController.setMainApp(this);
    if (mode.equals("edit"))
     actionSubElectionController.setSubElection(subElection);
    actionSubElectionController.setElection(election);
    actionSubElectionController.setMode(mode);
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }
 }

 public void showReorderPosition(SubElection subElection) {
  try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/
          ReorderPosition.fxml"));
    AnchorPane reorderPositionFXML = (AnchorPane) loader.
          load();

    Scene scene = new Scene(reorderPositionFXML);
    Stage secondaryStage = new Stage();
    secondaryStage.setTitle("YuVote");
    secondaryStage.getIcons().add(new Image(getClass().
          getResourceAsStream("images/YuVote Logo.png")));
    secondaryStage.initModality(Modality.WINDOW_MODAL);
    secondaryStage.initOwner(primaryStage);
    secondaryStage.setScene(scene);
    secondaryStage.show();

    ReorderPositionController reorderPositionController = loader.
          getController();
    reorderPositionController.setMainApp(this);
    reorderPositionController.setSubElection(subElection);
    reorderPositionController.setData();
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }
 }

 public void showReorderSubElection(Election election) {
  try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/
          ReorderSubElection.fxml"));
    AnchorPane reorderSubElectionFXML = (AnchorPane) loader
          .load();

    Scene scene = new Scene(reorderSubElectionFXML);
    Stage secondaryStage = new Stage();
    secondaryStage.setTitle("YuVote");
    secondaryStage.getIcons().add(new Image(getClass().
          getResourceAsStream("images/YuVote Logo.png")));
    secondaryStage.initModality(Modality.WINDOW_MODAL);
    secondaryStage.initOwner(primaryStage);
    secondaryStage.setScene(scene);
    secondaryStage.show();

    ReorderSubElectionController reorderSubElectionController =
          loader.getController();
    reorderSubElectionController.setMainApp(this);
    reorderSubElectionController.setElection(election);
    reorderSubElectionController.setData();
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }
 }

 public void showActionVoter(Voter voter, Block block, String
       mode) {
  try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/
          ActionVoter.fxml"));
    AnchorPane actionVoterFXML = (AnchorPane) loader.load();

    Scene scene = new Scene(actionVoterFXML);
    Stage secondaryStage = new Stage();
    secondaryStage.setTitle("YuVote");
    secondaryStage.getIcons().add(new Image(getClass().
          getResourceAsStream("images/YuVote Logo.png")));
    secondaryStage.initModality(Modality.WINDOW_MODAL);
    secondaryStage.initOwner(primaryStage);
    secondaryStage.setScene(scene);
    secondaryStage.show();

    ActionVoterController actionVoterController = loader.
          getController();
    actionVoterController.setMainApp(this);
    if (mode.equals("edit"))
     actionVoterController.setVoter(voter);
    actionVoterController.setBlock(block);
    actionVoterController.setMode(mode);
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }
 }

 public void showBulkAddVoters(Block block) {
  try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/
          BulkAddVoters.fxml"));
    AnchorPane bulkAddVotersFXML = (AnchorPane) loader.
          load();

    Scene scene = new Scene(bulkAddVotersFXML);
    Stage secondaryStage = new Stage();
    secondaryStage.setTitle("YuVote");
    secondaryStage.getIcons().add(new Image(getClass().
          getResourceAsStream("images/YuVote Logo.png")));
    secondaryStage.initModality(Modality.WINDOW_MODAL);
    secondaryStage.initOwner(primaryStage);
    secondaryStage.setScene(scene);
    secondaryStage.show();

    BulkAddVotersController bulkAddVotersController = loader.
          getController();
    bulkAddVotersController.setMainApp(this);
    bulkAddVotersController.setBlock(block);
  } catch (IOException ioe) {
```

```
    ioe.printStackTrace();
  }
}

public void showChooseFileBulkAdd(Block block,
        BulkAddVotersController bulkAddVotersController) {
  FileChooser fileChooser = new FileChooser();
  fileChooser.setTitle("Choose file");
  fileChooser.getExtensionFilters().addAll(new FileChooser.
          ExtensionFilter("CSV File", "*.csv"));
  fileChooser.setInitialDirectory(new File(System.getProperty
          ("user.home") + "/Desktop"));
  File file = fileChooser.showOpenDialog(primaryStage);
  if (file != null)
   bulkAddVotersController.setFileName(file);

}

public void startElection(Election election) {
  try {
   FXMLLoader loader = new FXMLLoader();
   loader.setLocation(MainApp.class.getResource("view/
           InputNumberKeys.fxml"));
   AnchorPane inputNumberKeysFXML = (AnchorPane) loader.
           load();

   Scene scene = new Scene(inputNumberKeysFXML);
   Stage secondaryStage = new Stage();
   secondaryStage.setTitle("YuVote");
   secondaryStage.getIcons().add(new Image(getClass().
           getResourceAsStream("images/YuVote Logo.png")));
   secondaryStage.initModality(Modality.WINDOW_MODAL);
   secondaryStage.initOwner(primaryStage);
   secondaryStage.setScene(scene);
   secondaryStage.showAndWait();
   InputNumberKeysController inputNumberKeysController =
           loader.getController();
   int number = inputNumberKeysController.getNumber();
   if (number < 1) {
    return;
   }

   Alert infoAlert = new Alert(AlertType.INFORMATION);
   infoAlert.setTitle("YuVote");
   infoAlert.initOwner(primaryStage);
   infoAlert.setHeaderText(null);
   infoAlert.setContentText("Please choose a directory to put
           the private key");
   infoAlert.showAndWait();

   DirectoryChooser dirChooser = new DirectoryChooser();
   File chosenDir = dirChooser.showDialog(primaryStage);
   boolean canWrite;
   try {
    if (chosenDir != null) {
     PrintWriter writer = new PrintWriter(chosenDir.toString()
             + "try.txt", "UTF-8");
     writer.close();
    }
    canWrite = true;
   } catch (IOException e) {
    canWrite = false;
   }

   if (chosenDir != null && canWrite) {
    loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/
            StartElection.fxml"));
    AnchorPane startElectionFXML = (AnchorPane) loader.load
            ();

    scene = new Scene(startElectionFXML);
    Stage secondaryStage1 = new Stage();
    secondaryStage1.setTitle("YuVote");
    secondaryStage1.getIcons().add(new Image(getClass().
            getResourceAsStream("images/YuVote Logo.png")));
    Platform.setImplicitExit(false);

    secondaryStage1.setOnCloseRequest(new EventHandler<
            WindowEvent>() {
     @Override
     public void handle(WindowEvent event) {
      event.consume();
     }
    });
    secondaryStage1.initModality(Modality.WINDOW_MODAL);
    secondaryStage1.initOwner(primaryStage);
    secondaryStage1.setScene(scene);
    secondaryStage1.show();

    StartElectionController startElectionController = loader.
            getController();

    DatabaseHelper databaseHelper = new DatabaseHelper();
    PasswordHelper passwordHelper = new PasswordHelper();
    EmailHelper emailHelper = new EmailHelper();
    emailSuccess = true;

    List<Integer> blockIdList = new ArrayList<Integer>();
    blockIdList = databaseHelper.getBlockId(election.
            getElectionId());
    List<Integer> voterIdList = new ArrayList<Integer>();
    for (int i = 0; i < blockIdList.size(); i++) {
     voterIdList.addAll(databaseHelper.getVoterId(blockIdList.
             get(i)));
    }
    startElectionController.setEmailProgress(0, voterIdList.size
            ());

    startElectionController.setGeneratingKeys("Generating Key
            Pairs..");
    X9ECParameters parameterSpec = CustomNamedCurves.
            getByName("curve25519");
    ECDomainParameters domainParams = new
            ECDomainParameters(parameterSpec.getCurve(),
            parameterSpec.getG(),
      parameterSpec.getN(), parameterSpec.getH(),
              parameterSpec.getSeed());
    ECKeyGenerationParameters keygenparam = new
            ECKeyGenerationParameters(domainParams, new
            SecureRandom());
    ECKeyPairGenerator generator = new ECKeyPairGenerator
            ();
    generator.init(keygenparam);

    BigInteger n = domainParams.getN();
    int nBitLength = n.bitLength();
    BigInteger randomNum = new BigInteger(nBitLength,
            keygenparam.getRandom());
    ECPoint votePoint = parameterSpec.getG().multiply(
            randomNum);
    databaseHelper.addVotePoint(election.getElectionId(),
            votePoint.getEncoded(false));

    MainApp mainApp = this;
    Task<Integer> task = new Task<Integer>() {
     @Override
     protected Integer call() throws Exception {
      int i;
      for (i = 0; i < voterIdList.size(); i++) {
       databaseHelper.addAlias(voterIdList.get(i), "V" + (i +
               1));
       try {
        byte[] salt = passwordHelper.getSalt();
        String password = passwordHelper.generatePassword();

        String hashedPassword = passwordHelper.
                getSecurePassword(password, salt);

        databaseHelper.addVoterPassword(voterIdList.get(i), salt
                , hashedPassword);
        Voter voter = databaseHelper.getVoter(voterIdList.get(i)
                , mainApp);
        String email = voter.getEmail();
        String subject = election.getName();
        String body = "Good day, \n" + "\tYou have been
                invited to join the election: "
          + election.getName() + "\n\n" + "Your credentials
                  are shown below: \nUsername: "
          + voter.getUsername() + "\n" + "Password: " +
                  password + "\n\nYour alias is "
          + voter.getAlias();
        if (!emailHelper.sendMail(email, subject, body)) {
         emailSuccess = false;
         return -1;
        }
        final int current = i;
        Platform.runLater(new Runnable() {
         @Override
         public void run() {
          startElectionController.setEmailProgress(current + 1,
                  voterIdList.size());
         }
        });
       } catch (Exception e) {
        e.printStackTrace();
       }
      }
      List<Integer> allSubElectionId = databaseHelper.
              getSubElectionId(election.getElectionId());
      for (i = 0; i < allSubElectionId.size(); i++) {
       List<Integer> positionIdList = databaseHelper.
               getPositionId(allSubElectionId.get(i));
       for (int j = 0; j < positionIdList.size(); j++) {
```

```
        databaseHelper.addCandidate(positionIdList.get(j), 0, "
            ABSTAIN", "");
      }
     }

     List<Integer> blockId = databaseHelper.getBlockId(
         election.getElectionId());
     List<Position> positionList;
     List<SubElection> subElection;
     List<Candidate> candidateList;
     int ballotOrder;
     for (i = 0; i < blockId.size(); i++) {
      subElection = databaseHelper.getSubElectionOfBlock(
          blockId.get(i), mainApp);
      ballotOrder = 1;
      for (int j = 0; j < subElection.size(); j++) {
       positionList = databaseHelper.getAllPosition(
           subElection.get(j).getSubElectionId(),
         mainApp);
       for (int k = 0; k < positionList.size(); k++) {
        candidateList = databaseHelper.getCandidate(
            positionList.get(k).getPositionId(),
          mainApp);
        for (int l = 0; l < candidateList.size(); l++) {
         databaseHelper.addBallot(blockId.get(i), subElection.
             get(j).getName(),
           positionList.get(k).getPositionId(), ballotOrder,
           positionList.get(k).getName(), positionList.get(k).
               getMaxVote(),
           candidateList.get(l).getCandidateId(),
           candidateList.get(l).getPartyName(), candidateList.
               get(l).getFirstName()
             + " " + candidateList.get(l).getLastName());
         ballotOrder++;
        }
       }
      }
     }
     Platform.runLater(new Runnable() {
      @Override
      public void run() {
       startElectionController.setGeneratingBallots("
           Generating ballots: Done");
      }
     });

     for (i = 0; i <= voterIdList.size(); i++) {
      ECPoint result = votePoint.multiply(new BigInteger(
          Integer.toString(i)));
      databaseHelper.addLookupTable(election.getElectionId(),
          i, result.getEncoded(false));
     }
     Platform.runLater(new Runnable() {
      @Override
      public void run() {
       startElectionController.setGeneratingLookupTable("
           Generating lookup table: Done");
      }
     });
     return i;
    }
   };
   task.setOnSucceeded(e -> {
    secondaryStage1.close();

    if (emailSuccess) {
     Alert startAlert = new Alert(AlertType.INFORMATION)
         ;
     startAlert.setTitle("YuVote");
     startAlert.initOwner(primaryStage);
     startAlert.setHeaderText(null);
     startAlert.setContentText("Election has started");
     startAlert.showAndWait();
     ECPoint totalPrivateKey = parameterSpec.getCurve().
         getInfinity();
     for (int i = 0; i < number; i++) {
      AsymmetricCipherKeyPair keypair = generator.
          generateKeyPair();

      ECPublicKeyParameters publicKey = (
          ECPublicKeyParameters) keypair.getPublic();
      ECPrivateKeyParameters privateKey = (
          ECPrivateKeyParameters) keypair.getPrivate();
      totalPrivateKey = totalPrivateKey.add(publicKey.getQ())
          ;

      String keyDirectory = chosenDir.getAbsolutePath() +
          "\\private_key"
        + election.getElectionId() + "(" + (i + 1) + ").key";
      try {
       PrintWriter writer = new PrintWriter(keyDirectory, "
           UTF-8");
       writer.println("-----BEGIN EC PRIVATE KEY
           -----");
       writer.println(Base64.getEncoder().encodeToString(
           privateKey.getD().toByteArray()));
       writer.println("-----END EC PRIVATE KEY
           -----");
       writer.close();
      } catch (IOException ioe) {
       ioe.printStackTrace();
      }
     }
     databaseHelper.addPublicKey(election.getElectionId(),
         totalPrivateKey.getEncoded(false));

     startElectionController.setGeneratingKeys("Generating
         Key Pairs: Done");
     databaseHelper.updateElectionStatus(election.
         getElectionId(), "Voting Stage");
     election.setStatus("Voting Stage");
     this.showManageElection(election);
    } else {
     Alert errorAlert = new Alert(AlertType.ERROR);
     errorAlert.initOwner(primaryStage);
     errorAlert.setTitle("YuVote");
     errorAlert.setTitle("Error");
     errorAlert.setHeaderText("Election was not started");
     errorAlert.setContentText("Email not sent, please check
         your internet connection!");
     errorAlert.showAndWait();
    }
   });
   Thread th = new Thread(task);
   th.setDaemon(true);
   th.start();
  } else if (chosenDir != null && !canWrite) {
   Alert errorAlert = new Alert(AlertType.ERROR);
   errorAlert.setTitle("YuVote");
   errorAlert.setHeaderText("Election was not started");
   errorAlert.initOwner(primaryStage);
   errorAlert.setContentText("Cannot write in this directory!")
       ;
   errorAlert.showAndWait();
  } else if (chosenDir == null) {
   Alert errorAlert = new Alert(AlertType.ERROR);
   errorAlert.setTitle("YuVote");
   errorAlert.initOwner(primaryStage);
   errorAlert.setHeaderText("Election was not started");
   errorAlert.setContentText("There was no directory chosen
       !");
   errorAlert.showAndWait();
  }
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }

}

public HomeController getHomeController() {
 return homeController;
}

public static void main(String[] args) {
 launch(args);
}

public void showChoosePrivateKey(Election election,
      ResultController resultController) {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
      ChoosePrivateKey.fxml"));
  AnchorPane choosePrivateKeyFXML = (AnchorPane) loader.
      load();

  Scene scene = new Scene(choosePrivateKeyFXML);
  Stage secondaryStage = new Stage();
  secondaryStage.setTitle("YuVote");
  secondaryStage.getIcons().add(new Image(getClass().
      getResourceAsStream("images/YuVote Logo.png")));
  secondaryStage.initModality(Modality.WINDOW_MODAL);
  secondaryStage.initOwner(primaryStage);
  secondaryStage.setScene(scene);
  secondaryStage.show();

  ChoosePrivateKeyController choosePrivateKeyController =
      loader.getController();
  choosePrivateKeyController.setMainApp(this);
  choosePrivateKeyController.setResultController(
      resultController);
  choosePrivateKeyController.setElection(election);
 } catch (IOException ioe) {
  ioe.printStackTrace();
```

```
 }
}

public void showChooseFileTally(ChoosePrivateKeyController
      choosePrivateKeyController) {
 FileChooser fileChooser = new FileChooser();
 fileChooser.setTitle("Choose file");
 fileChooser.getExtensionFilters().addAll(new FileChooser.
      ExtensionFilter("Key file", "*.key"));
 fileChooser.setInitialDirectory(new File(System.getProperty
      ("user.home") + "/Desktop"));
 File file = fileChooser.showOpenDialog(primaryStage);
 if (file != null)
  choosePrivateKeyController.setFileName(file);
}

public void showTallyProgress(Election election, BigInteger
      decodedPrivateKey) {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
      TallyElection.fxml"));
  AnchorPane tallyElectionFXML = (AnchorPane) loader.load
      ();

  Scene scene = new Scene(tallyElectionFXML);
  Stage secondaryStage = new Stage();
  secondaryStage.setTitle("YuVote");
  secondaryStage.getIcons().add(new Image(getClass().
      getResourceAsStream("images/YuVote Logo.png")));
  Platform.setImplicitExit(false);

  secondaryStage.setOnCloseRequest(new EventHandler<
      WindowEvent>() {
   @Override
   public void handle(WindowEvent event) {
    event.consume();
   }
  });
  secondaryStage.initModality(Modality.WINDOW_MODAL);
  secondaryStage.initOwner(primaryStage);
  secondaryStage.setScene(scene);
  secondaryStage.show();

  TallyElectionController tallyElectionController = loader.
      getController();

  MainApp mainApp = this;
  DatabaseHelper databaseHelper = new DatabaseHelper();

  X9ECParameters parameterSpec = CustomNamedCurves.
      getByName("curve25519");
  ECDomainParameters domainParams = new
      ECDomainParameters(parameterSpec.getCurve(),
      parameterSpec.getG(),
    parameterSpec.getN(), parameterSpec.getH(),
        parameterSpec.getSeed());
  ECKeyGenerationParameters keygenparam = new
      ECKeyGenerationParameters(domainParams, new
      SecureRandom());
  ECKeyPairGenerator generator = new ECKeyPairGenerator()
      ;
  generator.init(keygenparam);
  ECPrivateKeyParameters privateKey = new
      ECPrivateKeyParameters(decodedPrivateKey,
      domainParams);
  ECElGamalDecryptor dec = new ECElGamalDecryptor();

  dec.init(privateKey);
  correctPrivateKey = true;
  Task<Integer> task = new Task<Integer>() {
   @Override
   protected Integer call() throws Exception {
    int i = 0;
    List<Integer> allSubElectionId = databaseHelper.
        getSubElectionId(election.getElectionId());
    List<Integer> allCandidateId = new ArrayList<Integer>()
        ;
    List<Integer> blockIdList = new ArrayList<Integer>();
    blockIdList = databaseHelper.getBlockId(election.
        getElectionId());
    List<Voter> voterList = new ArrayList<Voter>();
    for (i = 0; i < blockIdList.size(); i++) {
     voterList.addAll(databaseHelper.getVoted(blockIdList.get(
         i), mainApp));
    }
    Platform.runLater(new Runnable() {
     @Override
     public void run() {
      tallyElectionController.setComputingVerificationCodes
          (0, voterList.size());
     }
    });

    for (i = 0; i < allSubElectionId.size(); i++) {
     List<Integer> positionIdList = databaseHelper.
         getPositionId(allSubElectionId.get(i));
     for (int j = 0; j < positionIdList.size(); j++) {
      allCandidateId.addAll(databaseHelper.getCandidateId(
          positionIdList.get(j)));
     }
    }

    for (i = 0; i < allCandidateId.size(); i++) {
     ResultSet candidateVotes = databaseHelper.
         getCandidateVotes(allCandidateId.get(i));
     ECPoint nullPoint = parameterSpec.getCurve().getInfinity
         ();
     ECPair sum = new ECPair(nullPoint, nullPoint);
     while (candidateVotes.next()) {
      try {

       ECPoint point1 = parameterSpec.getCurve().
           decodePoint(candidateVotes.getBytes(5));
       ECPoint point2 = parameterSpec.getCurve().
           decodePoint(candidateVotes.getBytes(6));
       sum = new ECPair(sum.getX().add(point1), sum.getY().
           add(point2));
      } catch (Exception e) {

      }
     }
     int tally = databaseHelper.getMultiple(election.
         getElectionId(),
       dec.decrypt(sum).getEncoded(false));

     if (tally == -1) {
      correctPrivateKey = false;
      return -1;
     }
     databaseHelper.setTally(allCandidateId.get(i), tally);
    }
    Platform.runLater(new Runnable() {
     @Override
     public void run() {
      tallyElectionController.setTallyProgress("Tallying
          election: Done");
     }
    });

    for (i = 0; i < voterList.size(); i++) {
     int counter = i;
     Platform.runLater(new Runnable() {
      @Override
      public void run() {
       tallyElectionController.setComputingVerificationCodes(
           counter + 1, voterList.size());
      }
     });
     ResultSet ballot = databaseHelper.getBallot(voterList.get(
         i).getBlockId());
     BigInteger sum = new BigInteger("0");
     BigInteger verificationCode = sum;
     try {
      Position currentPosition = new Position(0, 0, "", 0, 0,
          mainApp);
      String currentSubElection = "";
      while (ballot.next()) {
       if (currentPosition.getPositionId() != ballot.getInt(4))
           {
        if (!currentPosition.getName().equals("")) {
         databaseHelper.addHashCode(election.getElectionId(),
           voterList.get(i).getAlias(), currentSubElection,
           currentPosition.getName(), getMD5(sum.toString())
               );
        }
        currentPosition = new Position(ballot.getInt(4), 0,
            ballot.getString(5), 0, 0,
          mainApp);
        currentSubElection = ballot.getString(3);
        verificationCode = verificationCode.add(sum);
        sum = new BigInteger("0");
       }

       ResultSet encryptedVote = databaseHelper.getVote(
           voterList.get(i).getVoterId(),
         ballot.getInt(7));
       sum = sum.add(new BigInteger(encryptedVote.getBytes
           (5)));
       sum = sum.add(new BigInteger(encryptedVote.getBytes
           (6)));
      }
      verificationCode = verificationCode.add(sum);
```

```
        databaseHelper.setVerificationCode(voterList.get(i).
                getVoterId(),
            getMD5(verificationCode.toString()));
        databaseHelper.addHashCode(election.getElectionId(),
                voterList.get(i).getAlias(),
            currentSubElection, currentPosition.getName(),
                    getMD5(sum.toString()));

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

      return i;
     }
    };
    task.setOnSucceeded(e -> {
     if (correctPrivateKey) {
      secondaryStage.close();
      Alert startAlert = new Alert(AlertType.INFORMATION);
      startAlert.setTitle("YuVote");
      startAlert.initOwner(primaryStage);
      startAlert.setHeaderText(null);
      startAlert.setContentText("Tallying has completed!");
      startAlert.showAndWait();
      databaseHelper.updateElectionStatus(election.getElectionId
              (), "Finished");
      election.setStatus("Finished");
      this.homeController.showResult();
     } else {
      secondaryStage.close();

      Alert errorAlert = new Alert(AlertType.ERROR);
      errorAlert.setTitle("YuVote");
      errorAlert.initOwner(primaryStage);
      errorAlert.setHeaderText("Tally error");
      errorAlert.setContentText("The private key given is
              different from this election!");
      errorAlert.showAndWait();
     }
    });
    Thread th = new Thread(task);
    th.setDaemon(true);
    th.start();

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

  public String getMD5(String str) {
   try {
    MessageDigest md = MessageDigest.getInstance("MD5");
    md.update(str.getBytes());
    byte[] digest = md.digest();
    return (DatatypeConverter.printHexBinary(digest).
            toUpperCase());
   } catch (NoSuchAlgorithmException nsae) {
    return null;
   }
  }

  public void showHashCode(BulletinBoard bulletinBoard) {
   homeController.showHashCode(bulletinBoard);
  }

  public Stage getPrimaryStage() {
   return primaryStage;
  }

  public void showBallot(Block block) {
   try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/
            ViewBallot.fxml"));
    AnchorPane viewBallotFXML = (AnchorPane) loader.load();

    Scene scene = new Scene(viewBallotFXML);
    Stage secondaryStage = new Stage();
    secondaryStage.setTitle("YuVote");
    secondaryStage.getIcons().add(new Image(getClass().
            getResourceAsStream("images/YuVote Logo.png")));
    secondaryStage.initModality(Modality.WINDOW_MODAL);
    secondaryStage.initOwner(primaryStage);
    secondaryStage.setScene(scene);
    secondaryStage.show();

    ViewBallotController viewBallotController = loader.
            getController();
    viewBallotController.setMainApp(this);
    viewBallotController.setBlock(block);
    viewBallotController.setBallot();
   } catch (IOException ioe) {
    ioe.printStackTrace();
   }
  }

}
```

ActionBlockController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Block;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class ActionBlockController {
 private MainApp mainApp;
 private Election election;
 private String mode;
 private Block block;
 private DatabaseHelper databaseHelper;

 @FXML
 TextField nameField;
 @FXML
 Label nameError;
 @FXML
 Label titleLabel;
 @FXML
 TextArea descriptionField;
 @FXML
 Label descriptionError;
 @FXML
 Button actionButton;
 @FXML
 Button cancelButton;
 @FXML
 HBox nameHBox;
 @FXML
 HBox descriptionHBox;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

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

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

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getBlockController().setTable();
  stage.close();
 }

 public void setMode(String mode) {
  this.mode = mode;
  nameError.requestFocus();

  Tooltip tooltip = new Tooltip("Must not contain more than
          30 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(nameHBox, tooltip);

  tooltip = new Tooltip("Must not contain more than 500
          characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(descriptionHBox, tooltip);

  descriptionField.setOnKeyPressed(new EventHandler<
          KeyEvent>() {
   @Override
   public void handle(KeyEvent keyEvent) {
    if (keyEvent.getCode() == KeyCode.ENTER) {
```

```
      action();
     }
    }
   });
   databaseHelper = new DatabaseHelper();

   if (mode == "edit") {
    titleLabel.setText("Edit Block");
    actionButton.setText("Edit");
    nameField.setText(block.getName());
    descriptionField.setText(block.getDescription());
   } else {
    titleLabel.setText("Add Block");
    actionButton.setText("Add");
   }
  }

  @FXML
  public void action() {
   if (mode.equals("add"))
    add();
   else
    edit();
  }

  public void add() {
   String blockName = nameField.getText();
   String description = descriptionField.getText();
   nameError.setText("");
   descriptionError.setText("");

   boolean validDescription = true;
   if (description.length() > 500)
    validDescription = false;
   if (!validDescription)
    descriptionError.setText("Description is too long.");
   if (blockName.length() == 0 || blockName.length() > 30)
    nameError.setText("Block name is invalid.");
   else {
    boolean blockTaken = databaseHelper.isBlockTaken(
          blockName, election.getElectionId());

    if (!blockTaken && validDescription) {
     databaseHelper.addBlock(blockName, election.getElectionId
           (), description);
     cancel();
    }
    if (blockTaken)
     nameError.setText("Block name already taken.");
   }
  }

  public void edit() {
   String blockName = nameField.getText();
   String description = descriptionField.getText();
   nameError.setText("");
   descriptionError.setText("");

   boolean validDescription = true;
   if (description.length() > 500)
    validDescription = false;
   if (blockName.length() == 0 || blockName.length() > 30)
    nameError.setText("Block name is invalid.");
   else {
    if (blockName.equals(block.getName()) && description.
          equals(block.getDescription())) {
     cancel();
    } else if (blockName.equals(block.getName()) && !
          description.equals(block.getDescription())) {
     databaseHelper.updateBlock(block.getBlockId(), block.
           getElectionId(), blockName, description);
     cancel();
    } else {
     boolean blockTaken = databaseHelper.isBlockTaken(
           blockName, block.getElectionId());

     if (!blockTaken && validDescription) {
      databaseHelper.updateBlock(block.getBlockId(), block.
            getElectionId(), blockName, description);
      cancel();
     }
     if (blockTaken)
      nameError.setText("Block name is already taken.");
    }
   }
  }
}
```

ActionCandidateController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Candidate;
import ph.edu.upm.cas.dpsm.bjyu.model.Party;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class ActionCandidateController {
 private MainApp mainApp;
 private SubElection subElection;
 private String mode;
 private Candidate candidate;
 private Position position;
 private ObservableList<Party> data;
 private DatabaseHelper databaseHelper;

 @FXML
 TextField firstNameField;
 @FXML
 TextField lastNameField;
 @FXML
 Label firstNameError;
 @FXML
 Label lastNameError;
 @FXML
 Label titleLabel;
 @FXML
 ComboBox<String> partyComboBox;
 @FXML
 Button actionButton;
 @FXML
 Button cancelButton;
 @FXML
 HBox firstNameHBox;
 @FXML
 HBox lastNameHBox;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setSubElection(SubElection subElection) {
  this.subElection = subElection;
 }

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

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

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getCandidateController().setTable()
        ;
  stage.close();
 }

 public void setMode(String mode) {
  this.mode = mode;
  firstNameError.requestFocus();
  databaseHelper = new DatabaseHelper();
  Tooltip tooltip = new Tooltip("Must not contain more than
        30 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(firstNameHBox, tooltip);

  tooltip = new Tooltip("Must not contain more than 30
        characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(lastNameHBox, tooltip);

  data = FXCollections.observableArrayList();
  data.add(new Party(0, 0, "none", mainApp));
  data.addAll(databaseHelper.searchParty("", subElection.
        getElectionId(), mainApp));
  ObservableList<String> partyData = FXCollections.
        observableArrayList();
```

```
 int partyIndex = 0;
 for (int i = 0; i < data.size(); i++) {
  partyData.add(data.get(i).getName());
 }
 partyComboBox.setItems(partyData);

 if (mode == "edit") {
  titleLabel.setText("Edit Candidate");
  actionButton.setText("Edit");
  firstNameField.setText(candidate.getFirstName());
  lastNameField.setText(candidate.getLastName());
  for (int i = 0; i < data.size(); i++) {
   if (data.get(i).getPartyId() == candidate.getPartyId())
    partyComboBox.getSelectionModel().select(i);
  }
 } else {
  titleLabel.setText("Add Candidate");
  actionButton.setText("Add");
  partyComboBox.getSelectionModel().select(partyIndex);
 }
}

@FXML
public void action() {
 if (mode.equals("add"))
  add();
 else
  edit();
}

public void add() {
 String firstName = firstNameField.getText();
 String lastName = lastNameField.getText();
 firstNameError.setText("");
 lastNameError.setText("");

 boolean validFirstName = true;
 boolean validLastName = true;
 if (firstName.length() > 30 || firstName.length() < 1 ||
       firstName.trim().equals("ABSTAIN"))
  validFirstName = false;
 if (lastName.length() > 30 || lastName.length() < 1)
  validLastName = false;
 if (!validFirstName)
  firstNameError.setText("First name is not valid.");
 if (!validLastName)
  lastNameError.setText("Last name is not valid.");

 boolean nameTaken = databaseHelper.isNameTaken(position.
       getPositionId(), firstName, lastName);
 if (nameTaken)
  lastNameError.setText("The name is already taken.");

 if (!nameTaken && validFirstName && validLastName) {
  databaseHelper.addCandidate(position.getPositionId(),
    data.get(partyComboBox.getSelectionModel().
          getSelectedIndex()).getPartyId(), firstName, lastName
          );
  cancel();
 }
}

public void edit() {
 String firstName = firstNameField.getText();
 String lastName = lastNameField.getText();
 firstNameError.setText("");
 lastNameError.setText("");

 boolean validFirstName = true;
 boolean validLastName = true;
 if (firstName.length() > 30 || firstName.length() < 1 ||
       firstName.trim().equals("ABSTAIN"))
  validFirstName = false;
 if (lastName.length() > 30 || lastName.length() < 1)
  validLastName = false;
 if (!validFirstName)
  firstNameError.setText("First name is not valid.");
 if (!validLastName)
  lastNameError.setText("Last name is not valid.");
 if (validFirstName && validLastName) {
  if (firstName.equals(candidate.getFirstName()) && lastName
        .equals(candidate.getLastName())) {
   databaseHelper.updateCandidate(candidate.getCandidateId()
         ,
     data.get(partyComboBox.getSelectionModel().
           getSelectedIndex()).getPartyId(), firstName,
     lastName);
   cancel();
  } else {
   boolean nameTaken = databaseHelper.isNameTaken(
         candidate.getPositionId(), firstName, lastName);
   if (!nameTaken) {
    databaseHelper.updateCandidate(candidate.getCandidateId
          (),
      data.get(partyComboBox.getSelectionModel().
            getSelectedIndex()).getPartyId(), firstName,
      lastName);
    cancel();
   }
   if (nameTaken)
    lastNameError.setText("The name is already taken.");
  }
 }
}
```

ActionElectionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class ActionElectionController {
 private MainApp mainApp;
 private String mode;
 private Election election;
 private DatabaseHelper databaseHelper;
 @FXML
 TextField nameField;
 @FXML
 TextArea descriptionField;
 @FXML
 Label nameError;
 @FXML
 Label descriptionError;
 @FXML
 Button cancelButton;
 @FXML
 Label titleLabel;
 @FXML
 Button actionButton;
 @FXML
 HBox nameHBox;
 @FXML
 HBox descriptionHBox;

 public void setup() {
  databaseHelper = new DatabaseHelper();

  Tooltip tooltip = new Tooltip("Must not contain more than
        50 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(nameHBox, tooltip);

  tooltip = new Tooltip("Must not contain more than 500
        characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(descriptionHBox, tooltip);

  nameError.requestFocus();
  descriptionField.setOnKeyPressed(new EventHandler<
        KeyEvent>() {
   @Override
   public void handle(KeyEvent keyEvent) {
    if (keyEvent.getCode() == KeyCode.ENTER) {
     action();
    }
   }
  });
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

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

 @FXML
```

```
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getMainVOController().setTable();
  stage.close();
 }

 @FXML
 public void action() {
  if (mode.equals("add"))
   add();
  else
   edit();
 }

 void add() {
  String electionName = nameField.getText();
  String description = descriptionField.getText();
  nameError.setText("");
  descriptionError.setText("");

  boolean electionTaken = databaseHelper.isElectionTaken(
          electionName);
  boolean validElectionName = true;
  boolean validDescription = true;
  if (electionName.length() > 50 || electionName.length() ==
          0)
   validElectionName = false;
  if (description.length() > 500)
   validDescription = false;
  if (!validElectionName)
   nameError.setText("Election name is not valid.");
  if (!validDescription)
   descriptionError.setText("Description is empty.");
  if (electionTaken)
   nameError.setText("Election name is already taken.");
  if (!electionTaken && validDescription && validElectionName
          ) {
   databaseHelper.addElection(mainApp.getCurrentUser().
           getUserId(), electionName, description);
   cancel();
  }
 }

 public void edit() {
  String electionName = nameField.getText();
  String description = descriptionField.getText();
  nameError.setText("");
  descriptionError.setText("");

  boolean electionTaken = false;
  boolean validElectionName = true;
  boolean validDescription = true;
  if (electionName.length() > 50 || electionName.length() ==
          0)
   validElectionName = false;
  if (description.length() > 500)
   validDescription = false;
  if (!validElectionName)
   nameError.setText("Election name is not valid.");
  if (!validDescription)
   descriptionError.setText("Description is empty.");
  if (!election.getName().equals(electionName)) {
   electionTaken = databaseHelper.isElectionTaken(
           electionName);
  }
  if (electionTaken)
   nameError.setText("Election name is already taken.");
  if (!electionTaken && validDescription && validElectionName
          ) {
   databaseHelper.updateElection(electionName, description,
           election.getElectionId());
   cancel();
  } else if (election.getName().equals(electionName) &&
          election.getDescription().equals(description)) {
   cancel();
  }
 }

 public void setMode(String mode) {
  this.mode = mode;
  if (mode == "edit") {
   titleLabel.setText("Edit election");
   actionButton.setText("Edit");
   nameField.setText(election.getName());
   descriptionField.setText(election.getDescription());
  } else {
   titleLabel.setText("Add election");
   actionButton.setText("Add");
  }
 }
}
```

ActionPartyController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Party;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class ActionPartyController {
 private MainApp mainApp;
 private Election election;
 private String mode;
 private Party party;
 private DatabaseHelper databaseHelper;

 @FXML
 TextField nameField;
 @FXML
 Label nameError;
 @FXML
 Label titleLabel;
 @FXML
 Button actionButton;
 @FXML
 Button cancelButton;
 @FXML
 HBox nameHBox;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

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

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

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getPartyController().setTable();
  stage.close();
 }

 public void setMode(String mode) {
  this.mode = mode;
  nameError.requestFocus();
  databaseHelper = new DatabaseHelper();

  Tooltip tooltip = new Tooltip("Must not contain more than
          30 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(nameHBox, tooltip);

  if (mode == "edit") {
   titleLabel.setText("Edit Party");
   actionButton.setText("Edit");
   nameField.setText(party.getName());
  } else {
   titleLabel.setText("Add Party");
   actionButton.setText("Add");
  }
 }

 @FXML
 public void action() {
  if (mode.equals("add"))
   add();
  else
   edit();
 }

 public void add() {
  String partyName = nameField.getText();
  nameError.setText("");

  if (partyName.length() == 0 || partyName.length() > 30)
   nameError.setText("Party name is invalid");
  else {
   boolean partyTaken = databaseHelper.isPartyTaken(
```

```
            partyName, election.getElectionId());

    if (!partyTaken) {
     databaseHelper.addParty(partyName, election.getElectionId
            ());
     cancel();
    } else
     nameError.setText("Party name already taken.");
   }
  }

  public void edit() {
   String partyName = nameField.getText();
   nameError.setText("");

   if (partyName.length() == 0 || partyName.length() > 30)
    nameError.setText("Party name is invalid");
   else {
    if (partyName.equals(party.getName())) {
     cancel();
    } else {
     boolean partyTaken = databaseHelper.isPartyTaken(
             partyName, party.getElectionId());

     if (!partyTaken) {
      databaseHelper.updateParty(party.getPartyId(), party.
              getElectionId(), partyName);

      cancel();
     } else
      nameError.setText("Party name already taken.");
    }
   }
  }
}
```

ActionPositionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class ActionPositionController {
 private MainApp mainApp;
 private SubElection subElection;
 private String mode;
 private Position position;
 private DatabaseHelper databaseHelper;

 @FXML
 TextField nameField;
 @FXML
 TextField maxVoteField;
 @FXML
 Label nameError;
 @FXML
 Label maxVoteError;
 @FXML
 Label titleLabel;
 @FXML
 Button actionButton;
 @FXML
 Button cancelButton;
 @FXML
 HBox nameHBox;
 @FXML
 HBox maxVoteHBox;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setSubElection(SubElection subElection) {
  this.subElection = subElection;
 }

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

 }

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getPositionController().setTable();
  stage.close();
 }

 public void setMode(String mode) {
  this.mode = mode;
  databaseHelper = new DatabaseHelper();
  nameError.requestFocus();

  Tooltip tooltip = new Tooltip("Must not contain more than
         30 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(nameHBox, tooltip);

  tooltip = new Tooltip("Must be a counting number\nMust be
         less than the number of candidates in this position");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(maxVoteHBox, tooltip);

  if (mode == "edit") {
   titleLabel.setText("Edit Position");
   actionButton.setText("Edit");
   nameField.setText(position.getName());
   maxVoteField.setText(Integer.toString(position.getMaxVote()
          ));

  } else {
   titleLabel.setText("Add Position");
   actionButton.setText("Add");
  }
 }

 @FXML
 public void action() {
  if (mode.equals("add"))
   add();
  else
   edit();
 }

 public void add() {
  String positionName = nameField.getText();
  String maxVote = maxVoteField.getText();
  nameError.setText("");
  maxVoteError.setText("");

  boolean positionTaken = databaseHelper.isPositionTaken(
         positionName, subElection.getElectionId());
  boolean validPositionName = true;
  boolean validMinVote = true;
  boolean validMaxVote = true;
  if (!isInt(maxVote))
   validMaxVote = false;
  if (positionName.length() == 0 || positionName.length() >
         30)
   validPositionName = false;

  if (!validPositionName)
   nameError.setText("Position name is not valid.");
  if (!validMaxVote)
   maxVoteError.setText("The input is not valid.");
  if (positionTaken)
   nameError.setText("Position name is already taken.");
  if (validPositionName && validMinVote && validMaxVote
         && !positionTaken) {
   if (Integer.parseInt(maxVote) > 0) {
    databaseHelper.addPosition(subElection.getSubElectionId(),
           positionName, Integer.parseInt(maxVote));
    cancel();
   } else
    maxVoteError.setText("Value too low.");
  }
 }

 boolean isInt(String str) {
  try {
   if (Integer.parseInt(str) >= 0)
    return true;
   else
    return false;
  } catch (Exception e) {
   return false;
  }
 }

 public void edit() {
  String positionName = nameField.getText();
  String maxVote = maxVoteField.getText();
  nameError.setText("");
```

```
  maxVoteError.setText("");

  boolean validPositionName = true;
  boolean validMinVote = true;
  boolean validMaxVote = true;
  if (! isInt(maxVote))
   validMaxVote = false;
  if (positionName.length() == 0 || positionName.length() >
        30)
   validPositionName = false;
  boolean positionTaken = false;

  if (!validPositionName)
   nameError.setText("Position name is not valid.");
  if (!validMaxVote)
   maxVoteError.setText("Ballot order is not valid.");
  if (!position.getName().equals(positionName)) {
   positionTaken = databaseHelper.isPositionTaken(
         positionName, subElection.getElectionId());
  }
  if (positionTaken)
   nameError.setText("Position name is already taken.");
  if (validPositionName && validMinVote && validMaxVote
        && !positionTaken) {
   if (Integer.parseInt(maxVote) > 0) {
    databaseHelper.updatePosition(position.getPositionId(),
          positionName, Integer.parseInt(maxVote));
    cancel();
   } else
    maxVoteError.setText("Value cannot be less than 1.");
  }
 }
}
```

ActionSubElectionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class ActionSubElectionController {
 private MainApp mainApp;
 private Election election;
 private String mode;
 private SubElection subElection;
 private DatabaseHelper databaseHelper;

 @FXML
 TextField nameField;
 @FXML
 Label nameError;
 @FXML
 Label titleLabel;
 @FXML
 Button actionButton;
 @FXML
 Button cancelButton;
 @FXML
 HBox nameHBox;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

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

 public void setSubElection(SubElection subElection) {
  this.subElection = subElection;
 }

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getSubElectionController().setTable
        ();
  stage.close();
 }

 public void setMode(String mode) {
  nameError.requestFocus();
  databaseHelper = new DatabaseHelper();

  Tooltip tooltip = new Tooltip("Must not contain more than
        40 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(nameHBox, tooltip);

  this.mode = mode;
  if (mode == "edit") {
   titleLabel.setText("Edit Sub-election");
   actionButton.setText("Edit");
   nameField.setText(subElection.getName());
  } else {
   titleLabel.setText("Add Sub-election");
   actionButton.setText("Add");
  }
 }

 @FXML
 public void action() {
  if (mode.equals("add"))
   add();
  else
   edit();
 }

 public void add() {
  String subElectionName = nameField.getText();
  nameError.setText("");

  if (subElectionName.length() == 0 || subElectionName.length
        () > 40)
   nameError.setText("Sub-election name is invalid.");
  else {
   boolean subElectionTaken = databaseHelper.
         isSubElectionTaken(subElectionName, election.
         getElectionId());

   if (!subElectionTaken) {
    databaseHelper.addSubElection(subElectionName, election.
          getElectionId());
    cancel();
   } else
    nameError.setText("Sub-election name already taken.");

  }
 }

 public void edit() {
  String subElectionName = nameField.getText();
  nameError.setText("");

  boolean validSubElectionName = true;
  if (subElectionName.length() == 0 || subElectionName.length
        () > 40)
   validSubElectionName = false;
  if (!validSubElectionName)
   nameError.setText("Sub-election name is invalid.");
  if (subElectionName.equals(subElection.getName()))
   cancel();
  else {
   boolean subElectionTaken = databaseHelper.
         isSubElectionTaken(subElectionName, subElection.
         getElectionId());

   if (!subElectionTaken && validSubElectionName) {
    databaseHelper.updateSubElection(subElection.
          getSubElectionId(), subElectionName);
    cancel();
   } else if (subElectionTaken)
    nameError.setText("Sub-election name already taken.");
  }
 }
}
```

ActionVoterController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Block;
import ph.edu.upm.cas.dpsm.bjyu.model.Voter;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;
```

```
public class ActionVoterController {
 private MainApp mainApp;
 private String mode;
 private Block block;
 private Voter voter;
 private DatabaseHelper databaseHelper;

 @FXML
 TextField usernameField;
 @FXML
 Label usernameError;
 @FXML
 Label titleLabel;
 @FXML
 TextField emailField;
 @FXML
 Label emailError;
 @FXML
 Button actionButton;
 @FXML
 Button cancelButton;
 @FXML
 HBox usernameHBox;
 @FXML
 HBox emailHBox;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
  titleLabel.requestFocus();
 }

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

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

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getVoterController().setTable();
  stage.close();
 }

 public void setMode(String mode) {
  this.mode = mode;
  databaseHelper = new DatabaseHelper();
  Tooltip tooltip = new Tooltip(
    "Must contain 5-20 characters only\nMust contain
         alphanumeric characters only\nCan contain dashes,
         dots, and underscores\nbut it cannot start with those
         characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(usernameHBox, tooltip);

  tooltip = new Tooltip("Must follow the format: ____@___.___\
       nMust not contain more than 50 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(emailHBox, tooltip);

  if (mode == "edit") {
   titleLabel.setText("Edit Voter");
   actionButton.setText("Edit");
   usernameField.setText(voter.getUsername());
   emailField.setText(voter.getEmail());
  } else {
   titleLabel.setText("Add Voter");
   actionButton.setText("Add");
  }
 }

 @FXML
 public void action() {
  if (mode.equals("add"))
   add();
  else
   edit();
 }

 public void add() {
  String email = emailField.getText();
  String username = usernameField.getText();
  usernameError.setText("");
  emailError.setText("");
  DatabaseHelper databaseHelper = new DatabaseHelper();

  boolean usernameTaken = databaseHelper.isUsernameTaken(
       usernameField.getText());
  boolean emailTaken = databaseHelper.isEmailTaken(
       emailField.getText());
  boolean validEmail = email.matches("^[A-Za-z][A-Za-z0
       -9._]+[A-Za-z0-9]+@[A-Za-z0-9.-]+\\.[A-Za-z
       ]{2,4}")
    && email.length() < 50;
  boolean validUsername = username.matches("^[A-Za-z][A-
       Za-z0-9._]+[A-Za-z0-9]{1,}");
  if (username.length() > 20)
   validUsername = false;
  if (!validUsername)
   usernameError.setText("Username is not valid.");
  if (!validEmail)
   emailError.setText("Email is not valid.");
  if (usernameTaken)
   usernameError.setText("That username is already taken.");
  if (emailTaken)
   emailError.setText("That email is already taken.");

  if (!usernameTaken && !emailTaken && validUsername &&
       validEmail) {
   databaseHelper.addVoterAndAccount(username, email, block.
       getBlockId());
   cancel();
  }

  if (validUsername && validEmail
    && databaseHelper.accountExistOnDifferentElection(
       username, email, block.getElectionId())
    && !databaseHelper.accountExistOnThisElection(username,
       email, block.getElectionId())) {
   databaseHelper.addVoter(username, block.getBlockId());
   cancel();
  }
 }

 public void edit() {
  boolean usernameTaken = false;
  boolean emailTaken = false;
  String email = emailField.getText();
  String username = usernameField.getText();
  usernameError.setText("");
  emailError.setText("");
  if (!username.equals(voter.getUsername()) || !email.equals(
       voter.getEmail())) {
   try {

    if (!username.equals(voter.getUsername())) {
     usernameTaken = databaseHelper.isUsernameTaken(
       usernameField.getText());
    }
    if (!email.equals(voter.getEmail())) {

     emailTaken = databaseHelper.isEmailTaken(emailField.
       getText());
    }

   } catch (Exception e) {
   }
  }
  if (username.equals(voter.getUsername()) && email.equals(
       voter.getEmail())) {
   cancel();
  } else {
   boolean validEmail = email.matches("^[A-Za-z][A-Za-z0
       -9._]+[A-Za-z0-9]+@[A-Za-z0-9.-]+\\.[A-Za-z
       ]{2,4}")
     && email.length() < 50;
   boolean validUsername = username.matches("^[A-Za-z][A-
       Za-z0-9._]+[A-Za-z0-9]{3,5}");
   if (username.length() > 20)
    validUsername = false;
   if (!validUsername)
    usernameError.setText("Username is not valid.");
   if (!validEmail)
    emailError.setText("Email is not valid.");
   if (usernameTaken)
    usernameError.setText("That username is already taken.");
   if (emailTaken)
    emailError.setText("That email is already taken.");
   if (!usernameTaken && !emailTaken && validUsername &&
       validEmail) {
    databaseHelper.updateUsernameEmail(username, email,
       voter.getUserId());
    cancel();
   }
  }
 }
}
```

AddVotingOfficialController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;
```

```
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class AddVotingOfficialController {
 private MainApp mainApp;
 private DatabaseHelper databaseHelper;
 private String mode;

 @FXML
 Label usernameError;
 @FXML
 Label passwordError;
 @FXML
 Label confirmPasswordError;
 @FXML
 Label emailError;
 @FXML
 TextField usernameField;
 @FXML
 PasswordField passwordField;
 @FXML
 PasswordField confirmPasswordField;
 @FXML
 TextField emailField;
 @FXML
 Button cancelButton;
 @FXML
 HBox usernameHBox;
 @FXML
 HBox passwordHBox;
 @FXML
 HBox confirmPasswordHBox;
 @FXML
 HBox emailHBox;
 @FXML
 Label titleLabel;
 @FXML
 Button actionButton;


 public void setMode(String mode){
  this.mode = mode;
  if(mode.equals("add")){
   titleLabel.setText("Add Voting Official");
   actionButton.setText("Add");
  }
  else{
   titleLabel.setText("Register Voting Official");
   actionButton.setText("Register");
  }
 }


 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setup() {
  databaseHelper = new DatabaseHelper();

  Tooltip tooltip = new Tooltip(
    "Must contain 5-20 characters only\nMust contain
          alphanumeric characters only\nCan contain dashes,
          dots, and underscores\nbut it cannot start with those
          characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(usernameHBox, tooltip);

  tooltip = new Tooltip("Must have more than 5 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(passwordHBox, tooltip);

  tooltip = new Tooltip("Must match the password");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(confirmPasswordHBox, tooltip);

  tooltip = new Tooltip("Must follow the format: ____@___.___\
        nMust not contain more than 50 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(emailHBox, tooltip);

  usernameError.requestFocus();
 }

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  if(mode.equals("add"))
   mainApp.getCurrentUser().getMainSAController().search();
  stage.close();
 }

 @FXML
 public void add() {
  String email = emailField.getText();
  String username = usernameField.getText();
  String password = passwordField.getText();
  String confirmPassword = confirmPasswordField.getText();
  usernameError.setText("");
  emailError.setText("");
  passwordError.setText("");
  confirmPasswordError.setText("");

  boolean usernameTaken = databaseHelper.isUsernameTaken(
        usernameField.getText());
  boolean emailTaken = databaseHelper.isEmailTaken(
        emailField.getText());
  boolean validEmail = email.matches("^[A-Za-z][A-Za-z0
        -9._]+[A-Za-z0-9]+@[A-Za-z0-9.-]+\\.[A-Za-z
        ]{2,4}")
    && email.length() < 50;
  boolean validUsername = username.matches("^[A-Za-z][A-
        Za-z0-9._]+[A-Za-z0-9]{1,}");
  boolean validPassword = true;

  if (username.length() > 20 || username.length() < 5)
   validUsername = false;
  if (!validUsername)
   usernameError.setText("Username is not valid.");
  if (!validEmail)
   emailError.setText("Email is not valid.");
  if (usernameTaken)
   usernameError.setText("That username is already taken.");
  if (emailTaken)
   emailError.setText("That email is already taken.");
  if (password.length() < 5) {
   validPassword = false;
   passwordError.setText("That password is too short.");
  }
  if (!password.equals(confirmPassword)) {
   validPassword = false;
   confirmPasswordError.setText("The passwords do not match
         .");
  }
  if (!usernameTaken && !emailTaken && validUsername &&
        validEmail && validPassword) {
   databaseHelper.addVotingOfficial(username, password, email);
   cancel();
  }
 }

}
```

BlockController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Arrays;

import javafx.application.Platform;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Block;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class BlockController {
 private MainApp mainApp;
 private Pagination pagination;
 private Election election;
 private ObservableList<Block> tableData;
 private TableView<Block> table;
 private DatabaseHelper databaseHelper;
```

```
 @FXML
 Label titleLabel;
 @FXML
 TextField searchField;
 @FXML
 AnchorPane tablePane;
 @FXML
 Button addBlockButton;

 public BlockController() {
  databaseHelper = new DatabaseHelper();
  table = new TableView<Block>();

  TableColumn<Block, String> nameColumn = new
         TableColumn<Block, String>("Name");
  TableColumn<Block, Button> viewColumn = new
         TableColumn<Block, Button>("Ballot");
  TableColumn<Block, Button> participationColumn = new
         TableColumn<Block, Button>("Participation");
  TableColumn<Block, HBox> actionColumn = new
         TableColumn<Block, HBox>("Action");
  viewColumn.setStyle("-fx-alignment: CENTER;");
  participationColumn.setStyle("-fx-alignment: CENTER;");

  nameColumn.prefWidthProperty().bind(table.widthProperty().
         multiply(0.25));
  viewColumn.prefWidthProperty().bind(table.widthProperty().
         multiply(0.25));
  participationColumn.prefWidthProperty().bind(table.
         widthProperty().multiply(0.25));
  actionColumn.prefWidthProperty().bind(table.widthProperty()
         .multiply(0.245));

  nameColumn.setSortable(false);
  viewColumn.setSortable(false);
  participationColumn.setSortable(false);
  actionColumn.setSortable(false);
  table.getColumns().addAll(Arrays.asList(nameColumn,
         viewColumn, participationColumn, actionColumn));
  table.setFocusTraversable(false);

  nameColumn.setCellValueFactory(new PropertyValueFactory<
         Block, String>("name"));
  participationColumn.setCellValueFactory(new
         PropertyValueFactory<Block, Button>("modifyButton
         "));
  viewColumn.setCellValueFactory(new PropertyValueFactory<
         Block, Button>("viewButton"));
  actionColumn.setCellValueFactory(new PropertyValueFactory
         <Block, HBox>("hBox"));
 }

 @FXML
 public void setTable() {
  tableData = databaseHelper.searchBlock(searchField.getText()
         , election.getElectionId(), mainApp);
  Platform.runLater(() -> table.refresh());
  PaginationHelper<Block> paginationHelper = new
         PaginationHelper<Block>(table, tableData, 5, 1);

  pagination = paginationHelper.getPagination();
  tablePane.getChildren().clear(); // remove all children
  tablePane.getChildren().addAll(pagination);
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
  if (!mainApp.getHomeController().getElection().getStatus().
         equals("Setup Stage")) {
   addBlockButton.setDisable(true);
  }
 }

 public void setElection(Election election) {
  this.election = election;
  titleLabel.setText(election.getName() + " > Block");
 }

 @FXML
 public void addBlock() {
  mainApp.showActionBlock(null, election, "add");
 }
}
```

BlockParticipationController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Block;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class BlockParticipationController {
 private Block block;
 private DatabaseHelper databaseHelper;
 private ObservableList<String> participateData;
 private ObservableList<String> notParticipateData;
 private MainApp mainApp;

 @FXML
 private ListView<String> participatingList;
 @FXML
 private ListView<String> notParticipatingList;
 @FXML
 private TextField searchParticipatingField;
 @FXML
 private TextField searchNotParticipatingField;
 @FXML
 private Label titleLabel;
 @FXML
 Button cancelButton;
 @FXML
 Button saveButton;
 @FXML
 Button addButton;
 @FXML
 Button addAllButton;
 @FXML
 Button removeButton;
 @FXML
 Button removeAllButton;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

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

 public void setData() {
  titleLabel.setText("Sub-elections " + block.getName() + "
         will participate in");
  databaseHelper = new DatabaseHelper();
  ImageView imageView = new ImageView(new Image(MainApp
         .class.getResourceAsStream("images/add.png")));
  imageView.setFitHeight(15);
  imageView.setFitWidth(15);
  addButton.setGraphic(imageView);

  imageView = new ImageView(new Image(MainApp.class.
         getResourceAsStream("images/add all.png")));
  imageView.setFitHeight(15);
  imageView.setFitWidth(15);
  addAllButton.setGraphic(imageView);

  imageView = new ImageView(new Image(MainApp.class.
         getResourceAsStream("images/remove.png")));
  imageView.setFitHeight(15);
  imageView.setFitWidth(15);
  removeButton.setGraphic(imageView);

  imageView = new ImageView(new Image(MainApp.class.
         getResourceAsStream("images/remove all.png")));
  imageView.setFitHeight(15);
  imageView.setFitWidth(15);
  removeAllButton.setGraphic(imageView);

  participateData = FXCollections.observableArrayList();
  participateData = databaseHelper.searchParticipatingBlock(
         block.getBlockId());
  participatingList.setItems(participateData);

  notParticipateData = FXCollections.observableArrayList();
  notParticipateData = databaseHelper.searchAllSubElection(
         block.getElectionId());
  notParticipateData.removeAll(participateData);
  notParticipatingList.setItems(notParticipateData);

  if (!mainApp.getHomeController().getElection().getStatus().
         equals("Setup Stage")) {
   saveButton.setDisable(true);
   addButton.setDisable(true);
```

```
  addAllButton.setDisable(true);
  removeButton.setDisable(true);
  removeAllButton.setDisable(true);
 }
}

@FXML
public void add() {
 String selectedItem = notParticipatingList.getSelectionModel
       ().getSelectedItem();
 if (selectedItem != null) {
  notParticipateData.remove(selectedItem);
  participateData.add(selectedItem);
  searchParticipatingField.setText("");
  searchParticipating();
  searchNotParticipating();
 }
}

@FXML
public void addAll() {
 String keyword = searchNotParticipatingField.getText();
 ObservableList<String> searchNotParticipateData =
       FXCollections.observableArrayList();
 for (int i = 0; i < notParticipateData.size(); i++) {
  if (notParticipateData.get(i).toLowerCase().contains(
        keyword.toLowerCase()))
   searchNotParticipateData.add(notParticipateData.get(i));
 }
 participateData.addAll(searchNotParticipateData);
 notParticipateData.removeAll(participateData);
 searchParticipatingField.setText("");

 searchParticipating();
 searchNotParticipating();
}

@FXML
public void remove() {
 String selectedItem = participatingList.getSelectionModel().
       getSelectedItem();
 if (selectedItem != null) {
  participateData.remove(selectedItem);
  notParticipateData.add(selectedItem);
  searchNotParticipatingField.setText("");

  searchParticipating();
  searchNotParticipating();
 }
}

@FXML
public void removeAll() {
 String keyword = searchParticipatingField.getText();
 ObservableList<String> searchParticipateData =
       FXCollections.observableArrayList();
 for (int i = 0; i < participateData.size(); i++) {
  if (participateData.get(i).toLowerCase().contains(keyword.
        toLowerCase()))
   searchParticipateData.add(participateData.get(i));
 }
 participatingList.setItems(searchParticipateData);
 notParticipateData.addAll(searchParticipateData);
 participateData.removeAll(notParticipateData);
 searchNotParticipatingField.setText("");

 searchParticipating();
 searchNotParticipating();
}

@FXML
public void save() {
 for (int i = 0; i < participateData.size(); i++) {
  databaseHelper.addParticipation(block.getBlockId(), block.
        getElectionId(), participateData.get(i));
 }
 for (int i = 0; i < notParticipateData.size(); i++) {
  databaseHelper.deleteParticipation(block.getBlockId(), block.
        getElectionId(), notParticipateData.get(i));
 }
 cancel();
}

@FXML
public void cancel() {
 Stage stage = (Stage) cancelButton.getScene().getWindow();
 stage.close();
}

@FXML
public void searchParticipating() {
 String keyword = searchParticipatingField.getText();
 ObservableList<String> searchParticipateData =
       FXCollections.observableArrayList();
 for (int i = 0; i < participateData.size(); i++) {
  if (participateData.get(i).toLowerCase().contains(keyword.
        toLowerCase()))
   searchParticipateData.add(participateData.get(i));
 }
 participatingList.setItems(searchParticipateData);

}

@FXML
public void searchNotParticipating() {
 String keyword = searchNotParticipatingField.getText();
 ObservableList<String> searchNotParticipateData =
       FXCollections.observableArrayList();
 for (int i = 0; i < notParticipateData.size(); i++) {
  if (notParticipateData.get(i).toLowerCase().contains(
        keyword.toLowerCase()))
   searchNotParticipateData.add(notParticipateData.get(i));
 }
 notParticipatingList.setItems(searchNotParticipateData);
}
}
```

BulkAddVotersController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Block;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class BulkAddVotersController {
 private MainApp mainApp;
 private Block block;
 private File file;

 @FXML
 private Label titleLabel;
 @FXML
 Button cancelButton;
 @FXML
 Label fileNameLabel;
 @FXML
 Label addError;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setBlock(Block block) {
  this.block = block;
  titleLabel.setText(block.getName() + "> Bulk Add Voters");
 }

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getVoterController().setTable();
  stage.close();
 }

 @FXML
 public void add() {
  DatabaseHelper databaseHelper = new DatabaseHelper();
  if (file == null) {
   addError.setText("File not chosen");
  } else {
   BufferedReader br;
   try {
    br = new BufferedReader(new FileReader(file.
          getAbsolutePath()));
    String line;
    while ((line = br.readLine()) != null) {
     String[] user = line.split(",");
     if (user.length >= 2) {
      boolean validUsername = user[0].matches("^[A-Za-z][A-
            Za-z0-9._]+[A-Za-z0-9]{1,}")
        && user[0].length() < 20 && user[0].length() > 5;
      boolean validEmail = user[1]
        .matches("^[A-Za-z][A-Za-z0-9._]+[A-Za-z0-9]+@
              [A-Za-z0-9.-]+\\.[A-Za-z]{2,4}")
```

```
            && user[1].length() < 50;
        if (validUsername && validEmail && !databaseHelper.
              isUsernameTaken(user[0])
          && !databaseHelper.isEmailTaken(user[1])) {
         databaseHelper.addVoterAndAccount(user[0], user[1],
               block.getBlockId());
        }
        if (validUsername && validEmail
          && databaseHelper.accountExistOnDifferentElection(
                user[0], user[1],
            block.getElectionId())
          && !databaseHelper.accountExistOnThisElection(user[0],
                user[1],
            block.getElectionId())) {
         databaseHelper.addVoter(user[0], block.getBlockId());
         cancel();
        }

       }
      }
      br.close();
     } catch (FileNotFoundException e) {
      e.printStackTrace();
     } catch (IOException e) {
      e.printStackTrace();
     }
     cancel();
    }

   }

   @FXML
   public void chooseFile() {
    mainApp.showChooseFileBulkAdd(block, this);
   }

   public void setFileName(File file) {
    this.file = file;
    fileNameLabel.setText(file.getName());
   }
}
```

BulletinBoardController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Arrays;

import javafx.application.Platform;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.BulletinBoard;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class BulletinBoardController {
 private MainApp mainApp;
 private Pagination pagination;
 private Election election;
 private ObservableList<BulletinBoard> tableData;
 private TableView<BulletinBoard> table;
 private DatabaseHelper databaseHelper;

 @FXML
 Label titleLabel;
 @FXML
 TextField searchField;
 @FXML
 AnchorPane tablePane;
 @FXML
 Label infoLabel;

 public BulletinBoardController() {
  databaseHelper = new DatabaseHelper();
  table = new TableView<BulletinBoard>();

  TableColumn<BulletinBoard, String> aliasColumn = new
        TableColumn<BulletinBoard, String>("Alias");
  TableColumn<BulletinBoard, String> verificationCodeColumn
        = new TableColumn<BulletinBoard, String>(
    "Verification Code");
  TableColumn<BulletinBoard, Button> hashCodeColumn =
        new TableColumn<BulletinBoard, Button>("Hash
        Codes");

  aliasColumn.prefWidthProperty().bind(table.widthProperty().
        multiply(0.15));
  verificationCodeColumn.prefWidthProperty().bind(table.
        widthProperty().multiply(0.6));
  hashCodeColumn.prefWidthProperty().bind(table.
        widthProperty().multiply(0.245));
  hashCodeColumn.setStyle("-fx-alignment: CENTER;");

  aliasColumn.setSortable(false);
  verificationCodeColumn.setSortable(false);
  hashCodeColumn.setSortable(false);
  table.getColumns().addAll(Arrays.asList(aliasColumn,
        verificationCodeColumn, hashCodeColumn));
  table.setFocusTraversable(false);

  aliasColumn.setCellValueFactory(new PropertyValueFactory<
        BulletinBoard, String>("alias"));
  verificationCodeColumn.setCellValueFactory(new
        PropertyValueFactory<BulletinBoard, String>("
        verificationCode"));
  hashCodeColumn.setCellValueFactory(new
        PropertyValueFactory<BulletinBoard, Button>("
        viewButton"));
 }

 @FXML
 public void setTable() {
  if (election.getStatus().equals("Finished")) {
   tableData = databaseHelper.searchBulletinBoard(searchField.
         getText(), election.getElectionId(), mainApp);
   Platform.runLater(() -> table.refresh());
   PaginationHelper<BulletinBoard> paginationHelper = new
         PaginationHelper<BulletinBoard>(table, tableData, 5,
     1);

   pagination = paginationHelper.getPagination();
   tablePane.getChildren().clear();
   tablePane.getChildren().addAll(pagination);
  } else {
   infoLabel.setText("This election has not been tallied yet.");
   tablePane.setVisible(false);
   searchField.setVisible(false);
  }
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;

 }

 public void setElection(Election election) {
  this.election = election;
  titleLabel.setText(election.getName() + " > Bulletin Board
        ");

 }
}
```

CandidateController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Arrays;

import javafx.application.Platform;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Candidate;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class CandidateController {
 private MainApp mainApp;
 private Pagination pagination;
 private SubElection subElection;
 private HomeController homeController;
 private Position position;
```

```
private ObservableList<Candidate> tableData;
private TableView<Candidate> table;
private DatabaseHelper databaseHelper;

@FXML
Label titleLabel;
@FXML
TextField searchField;
@FXML
AnchorPane tablePane;
@FXML
Button addCandidateButton;

public CandidateController() {
 databaseHelper = new DatabaseHelper();
 table = new TableView<Candidate>();

 TableColumn<Candidate, String> firstNameColumn = new
        TableColumn<Candidate, String>("First Name");
 TableColumn<Candidate, String> lastNameColumn = new
        TableColumn<Candidate, String>("Last Name");
 TableColumn<Candidate, String> partyColumn = new
        TableColumn<Candidate, String>("Party");
 TableColumn<Candidate, HBox> actionColumn = new
        TableColumn<Candidate, HBox>("Action");

 firstNameColumn.prefWidthProperty().bind(table.
        widthProperty().multiply(0.25));
 lastNameColumn.prefWidthProperty().bind(table.
        widthProperty().multiply(0.25));
 partyColumn.prefWidthProperty().bind(table.widthProperty().
        multiply(0.25));
 actionColumn.prefWidthProperty().bind(table.widthProperty()
        .multiply(0.245));

 firstNameColumn.setSortable(false);
 lastNameColumn.setSortable(false);
 partyColumn.setSortable(false);
 actionColumn.setSortable(false);
 table.getColumns().addAll(Arrays.asList(firstNameColumn,
        lastNameColumn, partyColumn, actionColumn));
 table.setFocusTraversable(false);

 firstNameColumn.setCellValueFactory(new
        PropertyValueFactory<Candidate, String>("firstName")
        );
 lastNameColumn.setCellValueFactory(new
        PropertyValueFactory<Candidate, String>("lastName")
        );
 partyColumn.setCellValueFactory(new PropertyValueFactory<
        Candidate, String>("partyName"));
 actionColumn.setCellValueFactory(new PropertyValueFactory
        <Candidate, HBox>("hBox"));
}

@FXML
public void setTable() {
 tableData = databaseHelper.searchCandidate(searchField.
        getText(), position.getPositionId(), mainApp,
    subElection);
 Platform.runLater(() -> table.refresh());
 PaginationHelper<Candidate> paginationHelper = new
        PaginationHelper<Candidate>(table, tableData, 5, 1);

 pagination = paginationHelper.getPagination();
 tablePane.getChildren().clear();
 tablePane.getChildren().addAll(pagination);
}

public void setMainApp(MainApp mainApp) {
 this.mainApp = mainApp;
 if (!mainApp.getHomeController().getElection().getStatus().
        equals("Setup Stage")) {
  addCandidateButton.setDisable(true);
 }
}

public void setHomeController(HomeController homeController)
        {
 this.homeController = homeController;
}

public void setSubElection(SubElection subElection) {
 this.subElection = subElection;
}

public void setPosition(Position position) {
 this.position = position;
 titleLabel.setText(subElection.getName() + " > " + position.
        getName() + " > Candidate");
}

@FXML
public void addCandidate() {
 mainApp.showActionCandidate(null, position, subElection, "
        add");
}

@FXML
public void backToPosition() {
 homeController.showPosition(subElection);
}
}
```

ChoosePrivateKeyController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.math.BigInteger;
import java.util.Base64;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;

public class ChoosePrivateKeyController {
 private MainApp mainApp;
 private File privateKeyDirectory;
 private ResultController resultController;

 @FXML
 private Label titleLabel;
 @FXML
 Button cancelButton;
 @FXML
 Label fileNameLabel;
 @FXML
 Label tallyError;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setElection(Election election) {
  titleLabel.setText(election.getName() + " > Tally ");
 }

 public void setResultController(ResultController
        resultController) {
  this.resultController = resultController;
 }

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  stage.close();
 }

 @FXML
 public void add() {
  if (privateKeyDirectory == null) {
   tallyError.setText("File not chosen");
  } else {
   boolean correctKeyFileFormat = true;
   BigInteger decodedPrivateKey = new BigInteger("0");
   try {
    BufferedReader br = new BufferedReader(new FileReader(
           privateKeyDirectory));

    String line = br.readLine();
    if (!line.equals("-----BEGIN EC PRIVATE KEY
           -----"))
     correctKeyFileFormat = false;

    line = br.readLine();
    decodedPrivateKey = new BigInteger(Base64.getDecoder().
           decode(line));

    line = br.readLine();
    if (!line.equals("-----END EC PRIVATE KEY
           -----"))
     correctKeyFileFormat = false;
    line = br.readLine();
    if (line != null)
     correctKeyFileFormat = false;

    br.close();
   } catch (Exception e) {
```

```
    correctKeyFileFormat = false;
   }
   if (!correctKeyFileFormat)
    tallyError.setText("The key file does not follow \nthe
        correct format");
   else {
    tallyError.setText("");
    cancel();
    resultController.addKey(privateKeyDirectory.getName(),
        decodedPrivateKey);
   }

  }
 }

 @FXML
 public void chooseFile() {
  mainApp.showChooseFileTally(this);
 }

 public void setFileName(File file) {
  this.privateKeyDirectory = file;
  fileNameLabel.setText(file.getName());
 }
}
```

EditProfileController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.User;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class EditProfileController {
 private MainApp mainApp;
 private User user;
 private DatabaseHelper databaseHelper;

 @FXML
 Label usernameError;
 @FXML
 Label newPasswordError;
 @FXML
 Label confirmNewPasswordError;
 @FXML
 Label oldPasswordError;
 @FXML
 Label emailError;
 @FXML
 TextField usernameField;
 @FXML
 PasswordField newPasswordField;
 @FXML
 PasswordField confirmNewPasswordField;
 @FXML
 PasswordField oldPasswordField;
 @FXML
 TextField emailField;
 @FXML
 Button cancelButton;
 @FXML
 HBox usernameHBox;
 @FXML
 HBox newPasswordHBox;
 @FXML
 HBox confirmNewPasswordHBox;
 @FXML
 HBox oldPasswordHBox;
 @FXML
 HBox emailHBox;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setup() {
  usernameError.requestFocus();
  usernameField.setText(user.getUsername());
  emailField.setText(user.getEmail());
  databaseHelper = new DatabaseHelper();

  Tooltip tooltip = new Tooltip(
    "Must contain 5-20 characters only\nMust contain
        alphanumeric characters only\nCan contain dashes,
        dots, and underscores\nbut it cannot start with those
        characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(usernameHBox, tooltip);

  tooltip = new Tooltip(
    "Must not be blank if other password fields are not blank\
        nMust have more than 5 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(newPasswordHBox, tooltip);

  tooltip = new Tooltip("Must not be blank if other password
      fields are not blank\nMust match the new password");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(confirmNewPasswordHBox, tooltip);

  tooltip = new Tooltip("Must not be blank if other password
      fields are not blank\nMust match the old password");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(oldPasswordHBox, tooltip);

  tooltip = new Tooltip("Must follow the format: ____@___.___\
      nMust not contain more than 50 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(emailHBox, tooltip);
 }

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

 }

 @FXML
 public void edit() {
  boolean usernameTaken = false;
  boolean emailTaken = false;
  String email = emailField.getText();
  String username = usernameField.getText();
  String newPassword = newPasswordField.getText();
  String confirmNewPassword = confirmNewPasswordField.
      getText();
  String oldPassword = oldPasswordField.getText();
  usernameError.setText("");
  emailError.setText("");
  newPasswordError.setText("");
  confirmNewPasswordError.setText("");
  oldPasswordError.setText("");

  if (!username.equals(user.getUsername()) || !email.equals(user
      .getEmail())) {
   if (!username.equals(user.getUsername())) {
    usernameTaken = databaseHelper.isUsernameTaken(
        usernameField.getText());
   }
   if (!email.equals(user.getEmail())) {
    emailTaken = databaseHelper.isEmailTaken(emailField.
        getText());
   }
  }
  boolean changePass = true;
  if (newPassword.equals("") && confirmNewPassword.equals
      ("") && oldPassword.equals("")) {
   changePass = false;
  }
  if (username.equals(user.getUsername()) && email.equals(user
      .getEmail()) && !changePass) {
   cancel();
  } else {
   boolean validEmail = email.matches("^[A-Za-z][A-Za-z0
       -9._]+[A-Za-z0-9]+@[A-Za-z0-9.-]+\\.[A-Za-z
       ]{2,4}")
     && email.length() < 50;
   boolean validUsername = username.matches("^[A-Za-z][A-
       Za-z0-9._]+[A-Za-z0-9]{1,}");
   if (username.length() > 20 || username.length() < 5)
    validUsername = false;
   if (!validUsername)
    usernameError.setText("Username is not valid.");
   if (!validEmail)
    emailError.setText("Email is not valid.");
   if (usernameTaken)
    usernameError.setText("That username is already taken.");
   if (emailTaken)
    emailError.setText("That email is already taken.");
   boolean passChangeCorrect = true;

   if (!newPassword.equals("") && newPassword.length() < 5)
       {
    newPasswordError.setText("That password is too short.");
    passChangeCorrect = false;
```

```
    }
    if ((!newPassword.equals("") || !newPassword.equals(""))
          && !newPassword.equals(confirmNewPassword)) {
     confirmNewPasswordError.setText("The new passwords do
           not match.");
     passChangeCorrect = false;
    }
    if (oldPassword.equals("") && changePass) {
     oldPasswordError.setText("The old password is blank.");
     passChangeCorrect = false;
    }
    if (!oldPassword.equals("")
      && databaseHelper.usernameMatchesPassword(user.
           getUsername(), oldPassword) == null) {
     oldPasswordError.setText("The old password is incorrect.");
     passChangeCorrect = false;
    }
    if (!oldPassword.equals("") && newPassword.equals("")) {
     newPasswordError.setText("The new password cannot be
           blank.");
     passChangeCorrect = false;
    }
    if (!usernameTaken && !emailTaken && validUsername &&
          validEmail && !changePass) {
     databaseHelper.updateUsernameEmail(username, email, user
          .getUserId());
     user.setUsername(username);
     user.setEmail(email);
     mainApp.setCurrentUser(user);
     mainApp.setHomeControllerUser();
     cancel();
    }
    if (!usernameTaken && !emailTaken && validUsername &&
          validEmail && changePass) {
     if (passChangeCorrect) {
      databaseHelper.updateUsernameEmailPassword(username,
           email, user.getSalt(), newPassword,
        user.getUserId());
      user.setUsername(username);
      user.setEmail(email);
      mainApp.setCurrentUser(user);
      mainApp.setHomeControllerUser();
      cancel();
     }
    }
   }
  }

  @FXML
  public void cancel() {
   Stage stage = (Stage) cancelButton.getScene().getWindow();
   stage.close();
  }

}
```

EditUserController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Tooltip;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.model.User;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class EditUserController {
 private User user;
 private DatabaseHelper databaseHelper;
 @FXML
 TextField usernameField;
 @FXML
 TextField emailField;
 @FXML
 Label usernameError;
 @FXML
 Label emailError;
 @FXML
 Button closeButton;
 @FXML
 HBox usernameHBox;
 @FXML
 HBox emailHBox;

 public void setUser(User user) {
  this.user = user;
  Tooltip tooltip = new Tooltip(
    "Must contain 5-20 characters only\nMust contain
         alphanumeric characters only\nCan contain dashes,
         dots, and underscores\nbut it cannot start with those
         characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(usernameHBox, tooltip);

  tooltip = new Tooltip("Must follow the format: ____@___.___\
       nMust not contain more than 50 characters");
  TooltipHelper.hackTooltipStartTiming(tooltip);
  Tooltip.install(emailHBox, tooltip);
 }

 public void setup() {
  databaseHelper = new DatabaseHelper();
  usernameField.setText(user.getUsername());
  emailField.setText(user.getEmail());
  emailError.requestFocus();
 }

 @FXML
 public void close() {
  Stage stage = (Stage) closeButton.getScene().getWindow();
  user.getMainSAController().search();
  stage.close();
 }

 @FXML
 public void edit() {
  boolean usernameTaken = false;
  boolean emailTaken = false;
  String email = emailField.getText();
  String username = usernameField.getText();
  usernameError.setText("");
  emailError.setText("");

  if (!username.equals(user.getUsername()) || !email.equals(user
       .getEmail())) {
   try {
    if (!username.equals(user.getUsername())) {
     usernameTaken = databaseHelper.isUsernameTaken(
          usernameField.getText());
    }
    if (!email.equals(user.getEmail())) {
     emailTaken = databaseHelper.isEmailTaken(emailField.
          getText());
    }
   } catch (Exception e) {
   }
  }
  if (username.equals(user.getUsername()) && email.equals(user
       .getEmail())) {
   close();
  } else {
   boolean validEmail = email.matches("^[A-Za-z][A-Za-z0
        -9._]+[A-Za-z0-9]+@[A-Za-z0-9.-]+\\.[A-Za-z
        ]{2,4}")
     && email.length() < 50;
   boolean validUsername = username.matches("^[A-Za-z][A-
        Za-z0-9._]+[A-Za-z0-9]{1,}");

   if (username.length() > 20 || username.length() < 5)
    validUsername = false;
   if (!validUsername)
    usernameError.setText("Username is not valid.");
   if (!validEmail)
    emailError.setText("Email is not valid.");
   if (usernameTaken)
    usernameError.setText("That username is already taken.");
   if (emailTaken)
    emailError.setText("That email is already taken.");
   if (!usernameTaken && !emailTaken && validUsername &&
        validEmail) {
    databaseHelper.updateUsernameEmail(username, email, user
         .getUserId());
    close();
   }
  }
 }
}
```

ElectionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

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

import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
```

```
import javafx. collections .ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control. cell .PropertyValueFactory;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Requirements;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class ElectionController {
 private MainApp mainApp;
 private Election election;
 private DatabaseHelper databaseHelper;
 private ObservableList<Requirements> requirements;
 @FXML
 Label nameLabel;
 @FXML
 Label descriptionLabel;
 @FXML
 TableView<Requirements> requirementsTable;
 @FXML
 Label statusLabel;
 @FXML
 Button electionActionButton;

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

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setup() {
  databaseHelper = new DatabaseHelper();
  nameLabel.setText(election.getName());
  statusLabel.setText("Status: " + election.getStatus());
  descriptionLabel.setText(election.getDescription());
  if (election.getStatus().equals("Setup Stage"))
   electionActionButton.setText("Start Election");
  else if (election.getStatus().equals("Voting Stage"))
   electionActionButton.setText("Stop Election");
  else
   electionActionButton.setDisable(true);

  requirements = FXCollections.observableArrayList();

  TableColumn<Requirements, HBox> requirementsColumn =
        new TableColumn<Requirements, HBox>();
  TableColumn<Requirements, String> statusColumn = new
        TableColumn<Requirements, String>();

  requirementsTable.setColumnResizePolicy(TableView.
        CONSTRAINED_RESIZE_POLICY);

  requirementsColumn.setSortable(false);
  statusColumn.setSortable(false);
  requirementsTable.getColumns().addAll(Arrays.asList(
        requirementsColumn, statusColumn));
  requirementsTable.setFocusTraversable(false);
  requirementsTable.setStyle("-fx-table-cell-border-color:
        transparent;");

  requirementsTable.widthProperty().addListener(new
        ChangeListener<Number>() {
   @Override
   public void changed(ObservableValue<? extends Number>
         source, Number oldWidth, Number newWidth) {
    Pane header = (Pane) requirementsTable.lookup("
          TableHeaderRow");
    if (header.isVisible()) {
     header.setMaxHeight(0);
     header.setMinHeight(0);
     header.setPrefHeight(0);
     header.setVisible(false);
    }
   }
  });

  List<Integer> blockIdList = new ArrayList<Integer>();
  blockIdList = databaseHelper.getBlockId(election.
        getElectionId());
  List<Integer> positionIdList = new ArrayList<Integer>();
  List<Integer> subElectionIdList = new ArrayList<Integer>();
  subElectionIdList = databaseHelper.getSubElectionId(election.
        getElectionId());
  for (int i = 0; i < subElectionIdList.size(); i++) {
   positionIdList.addAll(databaseHelper.getPositionId(
         subElectionIdList.get(i)));
  }

  String status;
  if (databaseHelper.hasSubElection(election.getElectionId()))
   status = "Done";
  else
   status = "Not Done";
  requirements.add(new Requirements("Sub-elections", status));

  if (databaseHelper.hasBlock(election.getElectionId()))
   status = "Done";
  else
   status = "Not Done";
  requirements.add(new Requirements("Blocks", status));

  if (blockIdList.size() == 0)
   status = "Not Done";
  else
   status = "Done";
  for (int i = 0; i < blockIdList.size() && status.equals("
        Done"); i++) {
   if (!databaseHelper.blockHasBlockParticipation(blockIdList.
         get(i)))
    status = "Not Done";
  }
  requirements.add(new Requirements("Block Participation",
        status));

  if (subElectionIdList.size() == 0)
   status = "Not Done";
  else
   status = "Done";
  for (int i = 0; i < subElectionIdList.size() && status.equals
        ("Done"); i++) {
   if (!databaseHelper.hasPosition(subElectionIdList.get(i)))
    status = "Not Done";
  }
  requirements.add(new Requirements("Positions", status));

  if (positionIdList.size() == 0)
   status = "Not Done";
  else
   status = "Done";
  for (int i = 0; i < positionIdList.size() && status.equals("
        Done"); i++) {
   if (!databaseHelper.hasValidCandidates(positionIdList.get(i))
         )
    status = "Not Done";
  }
  requirements.add(new Requirements("Candidates", status));

  if (blockIdList.size() == 0)
   status = "Not Done";
  else
   status = "Done";
  for (int i = 0; i < blockIdList.size() && status.equals("
        Done"); i++) {
   if (!databaseHelper.hasVoters(blockIdList.get(i)))
    status = "Not Done";
  }
  requirements.add(new Requirements("Voters", status));
  requirementsTable.setItems(requirements);

  requirementsColumn.setCellValueFactory(new
        PropertyValueFactory<Requirements, HBox>("hBox"));
  statusColumn.setCellValueFactory(new PropertyValueFactory<
        Requirements, String>("status"));

 }

 @FXML
 public void electionAction() {
  if (election.getStatus().equals("Setup Stage")) {
   boolean requirementsDone = true;
   for (int i = 0; i < requirements.size() &&
         requirementsDone; i++) {
    if (requirements.get(i).getStatus().equals("Not Done"))
     requirementsDone = false;
   }
   if (requirementsDone) {
    Alert alert = new Alert(AlertType.CONFIRMATION, "Are
          you sure you want to start this election?",
      ButtonType.YES, ButtonType.NO);
    alert.setTitle("YuVote");
    alert.initOwner(mainApp.getPrimaryStage());
    alert.showAndWait();
```

```
      if (alert.getResult() == ButtonType.YES) {
       mainApp.startElection(election);
      }
     } else {
      Alert alert = new Alert(AlertType.INFORMATION);
      alert.setTitle("YuVote");
      alert.initOwner(mainApp.getPrimaryStage());
      alert.setHeaderText("Cannot start election!");
      alert.setContentText("There are incomplete requirements.")
            ;
      alert.showAndWait();
     }
    } else if (election.getStatus().equals("Voting Stage")) {
     Alert alert = new Alert(AlertType.CONFIRMATION, "Are
           you sure you want to stop this election?",
       ButtonType.YES, ButtonType.NO);
     alert.setTitle("YuVote");
     alert.initOwner(mainApp.getPrimaryStage());
     alert.showAndWait();

     if (alert.getResult() == ButtonType.YES) {
      databaseHelper.updateElectionStatus(election.getElectionId
            (), "Tallying Stage");
      election.setStatus("Tallying Stage");
      mainApp.showManageElection(election);
     }
    }
   }

}
```

HashCodeController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.ResultSet;

import javax.xml.bind.DatatypeConverter;

import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.ScrollPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import ph.edu.upm.cas.dpsm.bjyu.model.BulletinBoard;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class HashCodeController {
 private HomeController homeController;
 private DatabaseHelper databaseHelper;
 private Election election;
 private BulletinBoard bulletinBoard;
 @FXML
 Label titleLabel;
 @FXML
 ScrollPane scrollPane;

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

 public void setBulletinBoard(BulletinBoard bulletinBoard) {
  this.bulletinBoard = bulletinBoard;
 }

 public void setHomeController(HomeController homeController)
       {
  this.homeController = homeController;
 }

 public void setup() {
  databaseHelper = new DatabaseHelper();
  titleLabel.setText(election.getName() + " > " +
        bulletinBoard.getAlias() + " > Hash Code");

  VBox vbox = new VBox();

  ResultSet hashCode = databaseHelper.getHashCode(election.
        getElectionId(), bulletinBoard.getAlias());
  String currentSubElection = "";
  try {
   while (hashCode.next()) {
    if (!currentSubElection.equals(hashCode.getString(4))) {
     if (!currentSubElection.equals("")) {
      Label blankLabel = new Label("");
      vbox.getChildren().addAll(blankLabel);
     }
     currentSubElection = hashCode.getString(4);
     Label subElectionLabel = new Label(currentSubElection);
     subElectionLabel.setFont(new Font("Arial", 18));
     vbox.getChildren().addAll(subElectionLabel);
    }
    Label positionLabel = new Label(hashCode.getString(5) + "
          | " + hashCode.getString(6));
    vbox.getChildren().addAll(positionLabel);
   }
  } catch (Exception e) {

  }

  vbox.setSpacing(3);

  scrollPane.setContent(vbox);

 }

 @FXML
 public void backToBulletinBoard() {
  homeController.showBulletinBoard();
 }

 public String getMD5(String str) {
  try {
   MessageDigest md = MessageDigest.getInstance("MD5");
   md.update(str.getBytes());
   byte[] digest = md.digest();
   return (DatatypeConverter.printHexBinary(digest).
         toUpperCase());
  } catch (NoSuchAlgorithmException nsae) {
   return null;
  }
 }
}
```

HomeController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.io.IOException;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Insets;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.BorderPane;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.BulletinBoard;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.model.User;

public class HomeController {
 private MainApp mainApp;
 private Election election;
 private User user;

 @FXML
 Label usernameLabel;
 @FXML
 private AnchorPane middlePane;
 @FXML
 private AnchorPane leftPane;
 @FXML
 private BorderPane borderPane;
 @FXML
 private Button homeButton;
 @FXML
 private Button logoutButton;
 @FXML
 private Button profileButton;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setup() {
  usernameLabel.requestFocus();
  homeButton.getStyleClass().add("success");
  logoutButton.getStyleClass().add("success");
  profileButton.getStyleClass().add("success");
 }

 public Election getElection() {
  return election;
 }

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

```
 usernameLabel.setText("Welcome, " + user.getUsername());
}

public void showSAMainView() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/MainSA
        .fxml"));
  AnchorPane mainSAFXML = (AnchorPane) loader.load();

  borderPane.setLeft(null);
  borderPane.setRight(null);
  borderPane.setBottom(null);
  middlePane.getChildren().setAll(mainSAFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);

  MainSAController mainSAController = loader.getController();
  mainSAController.setMainApp(mainApp);
  mainSAController.setTable();
  mainApp.getCurrentUser().setMainSAController(
        mainSAController);
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showVOElectionView(Election election) {
 this.election = election;
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        NavBarVO.fxml"));
  AnchorPane navBarVOFXML = (AnchorPane) loader.load();

  borderPane.setLeft(leftPane);
  borderPane.setBottom(null);
  borderPane.setCenter(null);
  leftPane.setPadding(new Insets(0, 0, 0, 0));

  leftPane.setPrefHeight(middlePane.getHeight());
  leftPane.getChildren().setAll(navBarVOFXML);
  AnchorPane.setTopAnchor(leftPane.getChildren().get(0), 0.0);
  AnchorPane.setRightAnchor(leftPane.getChildren().get(0),
        0.0);
  AnchorPane.setLeftAnchor(leftPane.getChildren().get(0), 0.0);
  AnchorPane.setBottomAnchor(leftPane.getChildren().get(0),
        0.0);

  NavBarVOController navBarVOController = loader.
        getController();
  navBarVOController.setHomeController(this);
  navBarVOController.setButtonAlignment();
  showElectionMain();
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showElectionMain() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/Election
        .fxml"));
  AnchorPane electionFXML = (AnchorPane) loader.load();

  borderPane.setCenter(null);
  borderPane.setCenter(middlePane);
  middlePane.getChildren().setAll(electionFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);

  ElectionController electionController = loader.getController
        ();
  electionController.setElection(election);
  electionController.setMainApp(mainApp);
  electionController.setup();
 } catch (IOException ioe) {
 }
}

public void showVOMainView() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        MainVO.fxml"));
  AnchorPane mainVOFXML = (AnchorPane) loader.load();

  borderPane.setLeft(null);
  borderPane.setRight(null);
  borderPane.setBottom(null);
  middlePane.getChildren().setAll(mainVOFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);

  MainVOController mainVOController = loader.getController
        ();
  mainVOController.setMainApp(mainApp);
  mainVOController.setStatusFilter();
  mainVOController.setTable();
  mainApp.getCurrentUser().setMainVOController(
        mainVOController);
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showSubElections() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        SubElection.fxml"));
  AnchorPane subElectionFXML = (AnchorPane) loader.load();

  borderPane.setCenter(null);
  borderPane.setCenter(middlePane);
  middlePane.getChildren().setAll(subElectionFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);
  SubElectionController subElectionController = loader.
        getController();

  mainApp.getCurrentUser().setSubElectionController(
        subElectionController);
  subElectionController.setElection(election);
  subElectionController.setMainApp(mainApp);
  subElectionController.setTable();
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showParty() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/Party.
        fxml"));
  AnchorPane partyFXML = (AnchorPane) loader.load();

  borderPane.setCenter(null);
  borderPane.setCenter(middlePane);
  middlePane.getChildren().setAll(partyFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);
  PartyController partyController = loader.getController();

  mainApp.getCurrentUser().setPartyController(partyController
        );
  partyController.setElection(election);
  partyController.setMainApp(mainApp);
  partyController.setTable();
 } catch (IOException ioe) {
  ioe.printStackTrace();
```

```
    }
}

public void showPosition(SubElection subElection) {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/Position
        .fxml"));
  AnchorPane positionFXML = (AnchorPane) loader.load();

  borderPane.setCenter(null);
  borderPane.setCenter(middlePane);
  middlePane.getChildren().setAll(positionFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);

  PositionController positionController = loader.getController
        ();
  mainApp.getCurrentUser().setPositionController(
        positionController);
  positionController.setMainApp(mainApp);
  positionController.setElection(election);
  positionController.setSubElectionComboBox(0);
  if (subElection == null)
   positionController.setTable();
  else
   positionController.setTable(subElection);
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showBlock() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/Block.
        fxml"));
  AnchorPane blockFXML = (AnchorPane) loader.load();

  borderPane.setCenter(null);
  borderPane.setCenter(middlePane);
  middlePane.getChildren().setAll(blockFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);

  BlockController blockController = loader.getController();
  mainApp.getCurrentUser().setBlockController(blockController
        );
  blockController.setElection(election);
  blockController.setMainApp(mainApp);
  blockController.setTable();
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showCandidate(Position position, SubElection
      subElection) {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        Candidate.fxml"));
  AnchorPane candidateFXML = (AnchorPane) loader.load();

  borderPane.setCenter(null);
  borderPane.setCenter(middlePane);
  middlePane.getChildren().setAll(candidateFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);

  CandidateController candidateController = loader.
        getController();
  mainApp.getCurrentUser().setCandidateController(
        candidateController);
  candidateController.setSubElection(subElection);
  candidateController.setPosition(position);
  candidateController.setMainApp(mainApp);
  candidateController.setHomeController(this);
  candidateController.setTable();

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

public void showVoter() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/Voter.
        fxml"));
  AnchorPane voterFXML = (AnchorPane) loader.load();

  borderPane.setCenter(null);
  borderPane.setCenter(middlePane);
  middlePane.getChildren().setAll(voterFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);

  VoterController voterController = loader.getController();
  mainApp.getCurrentUser().setVoterController(voterController
        );
  voterController.setMainApp(mainApp);
  voterController.setElection(election);
  voterController.setSubElectionComboBox(0);
  voterController.setTable();
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showResult() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/Result.
        fxml"));
  AnchorPane resultFXML = (AnchorPane) loader.load();

  borderPane.setCenter(null);
  borderPane.setCenter(middlePane);
  middlePane.getChildren().setAll(resultFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);
  ResultController resultController = loader.getController();
  mainApp.getCurrentUser().setResultController(
        resultController);

  resultController.setElection(election);
  resultController.setMainApp(mainApp);
  resultController.setup();
 } catch (IOException ioe) {
  ioe.printStackTrace();
 }
}

public void showBulletinBoard() {
 try {
  FXMLLoader loader = new FXMLLoader();
  loader.setLocation(MainApp.class.getResource("view/
        BulletinBoard.fxml"));
  AnchorPane bulletinBoardFXML = (AnchorPane) loader.load
        ();

  borderPane.setCenter(null);
  borderPane.setCenter(middlePane);
  middlePane.getChildren().setAll(bulletinBoardFXML);
  AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
  AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
  AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);
  BulletinBoardController bulletinBoardController = loader.
```

```
            getController();

    bulletinBoardController.setElection(election);
    bulletinBoardController.setMainApp(mainApp);
    bulletinBoardController.setTable();
   } catch (IOException ioe) {
    ioe.printStackTrace();
   }
  }

  public void showHashCode(BulletinBoard bulletinBoard) {
   try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/
            HashCode.fxml"));
    AnchorPane hashCodeFXML = (AnchorPane) loader.load();

    borderPane.setCenter(null);
    borderPane.setCenter(middlePane);
    middlePane.getChildren().setAll(hashCodeFXML);
    AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
            0.0);
    AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
             0.0);
    AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
            0.0);
    AnchorPane.setBottomAnchor(middlePane.getChildren().get
            (0), 0.0);

    HashCodeController hashCodeController = loader.
            getController();
    hashCodeController.setHomeController(this);
    hashCodeController.setElection(election);
    hashCodeController.setBulletinBoard(bulletinBoard);
    hashCodeController.setup();
   } catch (IOException ioe) {
    ioe.printStackTrace();
   }
  }

  @FXML
  public void logout() {
   mainApp.showLogout();
  }

  @FXML
  public void editProfile() {
   mainApp.showEditProfile();
  }

  @FXML
  public void home() {
   if (user.getType().equals("System Admin"))
    showSAMainView();
   if (user.getType().equals("Voting Official")) {
    showVOMainView();
   }
  }
 }
```

InputNumberKeysController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

public class InputNumberKeysController {

 @FXML
 TextField numberField;
 @FXML
 Label numberError;
 @FXML
 Button proceedButton;
 int number;

 @FXML
 public void proceed() {
  String numberString = numberField.getText();

  if (isInt(numberString)) {
   number = Integer.parseInt(numberString);
   if (number < 1)
    numberError.setText("Number cannot be less than 1.");
   else {
    Stage stage = (Stage) proceedButton.getScene().getWindow
            ();
    stage.close();
   }
  } else {
   numberError.setText("The input should be number.");
  }
  if (numberField.getText().equals("")) {
   numberError.setText("Field is empty.");
  }
 }

 public int getNumber() {
  return number;
 }

 public boolean isInt(String number) {
  try {
   Integer.parseInt(number);
   return true;
  } catch (NumberFormatException e) {
   return false;
  }
 }
}
```

LoginController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.User;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class LoginController {
 private MainApp mainApp;
 DatabaseHelper databaseHelper;

 @FXML
 private Label errorLabel;
 @FXML
 private TextField usernameField;
 @FXML
 private PasswordField passwordField;

 public void setup() {
  databaseHelper = new DatabaseHelper();
  errorLabel.requestFocus();
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 @FXML
 public void login() {
  errorLabel.setText("");
  if (usernameField.getText().isEmpty())
   errorLabel.setText("The username is empty.");
  else if (passwordField.getText().isEmpty())
   errorLabel.setText("The password is empty.");
  else {
   User user = databaseHelper.usernameMatchesPassword(
           usernameField.getText(), passwordField.getText());
   if (user != null) {
    mainApp.setCurrentUser(user);
    mainApp.showHome();
   } else
    errorLabel.setText("Username or password is incorrect.");
  }
 }

 @FXML
 public void register() {
  mainApp.showAddVotingOfficial("register");
 }
}
```

LogoutController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;

public class LogoutController {
 private MainApp mainApp;

 @FXML
 Button noButton;
```

```
 @FXML
 Button yesButton;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 @FXML
 public void no() {
  Stage stage = (Stage) noButton.getScene().getWindow();
  stage.close();
 }

 @FXML
 public void yes() {
  no();
  mainApp.showLogin();
 }

}
```

MainSAController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Arrays;

import javafx.application.Platform;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.User;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class MainSAController {
 private MainApp mainApp;
 private DatabaseHelper databaseHelper;
 private ObservableList<User> tableData;
 private Pagination pagination;
 private TableView<User> table;

 @FXML
 private TextField searchField;
 @FXML
 private AnchorPane tablePane;

 public MainSAController() {
  databaseHelper = new DatabaseHelper();
  table = new TableView<User>();

  TableColumn<User, String> usernameColumn = new
        TableColumn<User, String>("Username");
  TableColumn<User, String> emailColumn = new
        TableColumn<User, String>("Email");
  TableColumn<User, HBox> actionColumn = new
        TableColumn<User, HBox>("Action");

  usernameColumn.prefWidthProperty().bind(table.
        widthProperty().multiply(0.4));
  emailColumn.prefWidthProperty().bind(table.widthProperty().
        multiply(0.4));
  actionColumn.prefWidthProperty().bind(table.widthProperty()
        .multiply(0.1965));

  usernameColumn.setSortable(false);
  emailColumn.setSortable(false);
  actionColumn.setSortable(false);

  table.getColumns().addAll(Arrays.asList(usernameColumn,
        emailColumn, actionColumn));
  table.setFocusTraversable(false);

  usernameColumn.setCellValueFactory(new
        PropertyValueFactory<User, String>("username"));
  emailColumn.setCellValueFactory(new PropertyValueFactory<
        User, String>("email"));
  actionColumn.setCellValueFactory(new PropertyValueFactory
        <User, HBox>("hBox"));
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setTable() {
  tableData = databaseHelper.searchVotingOfficial(searchField.
        getText(), mainApp, this);
  Platform.runLater(() -> table.refresh());
  PaginationHelper<User> paginationHelper = new
        PaginationHelper<User>(table, tableData, 6, 1);

  pagination = paginationHelper.getPagination();
  tablePane.getChildren().clear();
  tablePane.getChildren().addAll(pagination);
 }

 public void search() {
  setTable();
 }

 @FXML
 public void addVotingOfficial() {
  mainApp.showAddVotingOfficial("add");
 }
}
```

MainVOController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Arrays;

import javafx.application.Platform;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class MainVOController {
 private MainApp mainApp;
 private DatabaseHelper databaseHelper;
 private ObservableList<Election> tableData;
 private Pagination pagination;
 private TableView<Election> table;

 @FXML
 private TextField searchField;
 @FXML
 private AnchorPane tablePane;
 @FXML
 ChoiceBox<String> statusFilter;

 public MainVOController() {
  databaseHelper = new DatabaseHelper();
  table = new TableView<Election>();

  TableColumn<Election, String> nameColumn = new
        TableColumn<Election, String>("Name");
  TableColumn<Election, String> descriptionColumn = new
        TableColumn<Election, String>("Description");
  TableColumn<Election, Button> manageColumn = new
        TableColumn<Election, Button>("Manage");
  TableColumn<Election, HBox> actionColumn = new
        TableColumn<Election, HBox>("Action");

  nameColumn.prefWidthProperty().bind(table.widthProperty().
        multiply(0.25));
  descriptionColumn.prefWidthProperty().bind(table.
        widthProperty().multiply(0.45));
  manageColumn.prefWidthProperty().bind(table.widthProperty
        ().multiply(0.15));
  actionColumn.prefWidthProperty().bind(table.widthProperty()
        .multiply(0.1465));
  manageColumn.setStyle("-fx-alignment: CENTER;");

  nameColumn.setSortable(false);
  descriptionColumn.setSortable(false);
  actionColumn.setSortable(false);
  manageColumn.setSortable(false);
  table.getColumns().addAll(Arrays.asList(nameColumn,
        descriptionColumn, manageColumn, actionColumn));
  table.setFocusTraversable(false);
```

```
  nameColumn.setCellValueFactory(new PropertyValueFactory<
        Election, String>("name"));
  descriptionColumn.setCellValueFactory(new
        PropertyValueFactory<Election, String>("description"))
        ;
  manageColumn.setCellValueFactory(new PropertyValueFactory
        <Election, Button>("manageButton"));
  actionColumn.setCellValueFactory(new PropertyValueFactory
        <Election, HBox>("hBox"));
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 @FXML
 public void setStatusFilter() {
  statusFilter.setItems(
    FXCollections.observableArrayList("All", "Setup Stage", "
          Voting Stage", "Tallying Stage", "Finished"));
  statusFilter.getSelectionModel().selectFirst();
  statusFilter.getSelectionModel().selectedIndexProperty().
        addListener(new ChangeListener<Number>() {
   @Override
   public void changed(ObservableValue<? extends Number>
         observableValue, Number number, Number number2) {
    statusFilter.getSelectionModel().select(number2.intValue())
          ;
    setTable();
   }
  });
 }

 @FXML
 public void setTable() {
  tableData = databaseHelper.searchElection(searchField.
        getText(), mainApp.getCurrentUser().getUserId(),
    statusFilter.getValue(), mainApp);
  Platform.runLater(() -> table.refresh());
  PaginationHelper<Election> paginationHelper = new
        PaginationHelper<Election>(table, tableData, 5, 1);

  pagination = paginationHelper.getPagination();
  tablePane.getChildren().clear(); // remove all children
  tablePane.getChildren().addAll(pagination);
 }

 @FXML
 public void addElection() {
  mainApp.showActionElection(null, "add");
 }
}
```

NavBarVOController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.geometry.Pos;
import javafx.scene.control.Button;
import javafx.scene.input.MouseEvent;

public class NavBarVOController {
 private HomeController homeController;

 @FXML
 Button electionButton;
 @FXML
 Button subElectionButton;
 @FXML
 Button partyButton;
 @FXML
 Button blockButton;
 @FXML
 Button positionButton;
 @FXML
 Button voterButton;
 @FXML
 Button bulletinBoardButton;
 @FXML
 Button resultButton;

 public void setHomeController(HomeController homeController)
       {
  this.homeController = homeController;
 }

 public void setButtonAlignment() {
  electionButton.setAlignment(Pos.CENTER_LEFT);
  subElectionButton.setAlignment(Pos.CENTER_LEFT);
  partyButton.setAlignment(Pos.CENTER_LEFT);
  blockButton.setAlignment(Pos.CENTER_LEFT);
  positionButton.setAlignment(Pos.CENTER_LEFT);
  voterButton.setAlignment(Pos.CENTER_LEFT);
  resultButton.setAlignment(Pos.CENTER_LEFT);
  bulletinBoardButton.setAlignment(Pos.CENTER_LEFT);
  addHoverColor(electionButton);
  addHoverColor(subElectionButton);
  addHoverColor(partyButton);
  addHoverColor(blockButton);
  addHoverColor(positionButton);
  addHoverColor(voterButton);
  addHoverColor(resultButton);
  addHoverColor(bulletinBoardButton);
  resetColor();
  electionButton.setStyle("-fx-background-color: #76d276");
  electionButton.setUserData(true);
 }

 public void resetColor() {
  boolean clicked = false;
  electionButton.setStyle("-fx-background-color: #d3d3d3");
  subElectionButton.setStyle("-fx-background-color: #d3d3d3
        ");
  partyButton.setStyle("-fx-background-color: #d3d3d3");
  blockButton.setStyle("-fx-background-color: #d3d3d3");
  positionButton.setStyle("-fx-background-color: #d3d3d3");
  voterButton.setStyle("-fx-background-color: #d3d3d3");
  resultButton.setStyle("-fx-background-color: #d3d3d3");
  bulletinBoardButton.setStyle("-fx-background-color: #
        d3d3d3");

  electionButton.setUserData(clicked);
  subElectionButton.setUserData(clicked);
  partyButton.setUserData(clicked);
  blockButton.setUserData(clicked);
  positionButton.setUserData(clicked);
  voterButton.setUserData(clicked);
  resultButton.setUserData(clicked);
  bulletinBoardButton.setUserData(clicked);
 }

 @FXML
 public void election() {
  homeController.showElectionMain();
  resetColor();
  electionButton.setStyle("-fx-background-color: #76d276");
  electionButton.setUserData(true);
 }

 @FXML
 public void subElections() {
  homeController.showSubElections();
  resetColor();
  subElectionButton.setStyle("-fx-background-color: #76d276
        ");
  subElectionButton.setUserData(true);
 }

 @FXML
 public void parties() {
  homeController.showParty();
  resetColor();
  partyButton.setStyle("-fx-background-color: #76d276");
  partyButton.setUserData(true);
 }

 @FXML
 public void blocks() {
  homeController.showBlock();
  resetColor();
  blockButton.setStyle("-fx-background-color: #76d276");
  blockButton.setUserData(true);
 }

 @FXML
 public void positionsAndCandidates() {
  homeController.showPosition(null);
  resetColor();
  positionButton.setStyle("-fx-background-color: #76d276");
  positionButton.setUserData(true);
 }

 @FXML
 public void voters() {
  homeController.showVoter();
  resetColor();
  voterButton.setStyle("-fx-background-color: #76d276");
  voterButton.setUserData(true);
 }

 @FXML
```

```
 public void bulletinBoard() {
  homeController.showBulletinBoard();
  resetColor();
  bulletinBoardButton.setStyle("-fx-background-color: #76
      d276");
  bulletinBoardButton.setUserData(true);
 }

 @FXML
 public void result() {
  homeController.showResult();
  resetColor();
  resultButton.setStyle("-fx-background-color: #76d276");
  resultButton.setUserData(true);
 }

 public void addHoverColor(Button button) {
  button.addEventHandler(MouseEvent.MOUSE_ENTERED,
      new EventHandler<MouseEvent>() {
   @Override
   public void handle(MouseEvent e) {
    button.setStyle("-fx-background-color: #76d276");
   }// #808080
  });

  button.addEventHandler(MouseEvent.MOUSE_EXITED, new
      EventHandler<MouseEvent>() {
   @Override
   public void handle(MouseEvent e) {
    if (!(Boolean) button.getUserData())
     button.setStyle("-fx-background-color: #d3d3d3");
   }
  });
 }
}
```

PartyController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Arrays;

import javafx.application.Platform;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Party;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class PartyController {
 private MainApp mainApp;
 private Pagination pagination;
 private Election election;
 private ObservableList<Party> tableData;
 private TableView<Party> table;
 private DatabaseHelper databaseHelper;

 @FXML
 Label titleLabel;
 @FXML
 TextField searchField;
 @FXML
 AnchorPane tablePane;
 @FXML
 Button addPartyButton;

 public PartyController() {
  databaseHelper = new DatabaseHelper();
  table = new TableView<Party>();

  TableColumn<Party, String> nameColumn = new
      TableColumn<Party, String>("Name");
  TableColumn<Party, HBox> actionColumn = new
      TableColumn<Party, HBox>("Action");

  nameColumn.prefWidthProperty().bind(table.widthProperty().
      multiply(0.75));
  actionColumn.prefWidthProperty().bind(table.widthProperty()
      .multiply(0.245));

  nameColumn.setSortable(false);
  actionColumn.setSortable(false);
  table.getColumns().addAll(Arrays.asList(nameColumn,
      actionColumn));
  table.setFocusTraversable(false);

  nameColumn.setCellValueFactory(new PropertyValueFactory<
      Party, String>("name"));
  actionColumn.setCellValueFactory(new PropertyValueFactory
      <Party, HBox>("hBox"));
 }

 @FXML
 public void setTable() {
  tableData = databaseHelper.searchParty(searchField.getText(),
      election.getElectionId(), mainApp);
  Platform.runLater(() -> table.refresh());
  PaginationHelper<Party> paginationHelper = new
      PaginationHelper<Party>(table, tableData, 5, 1);

  pagination = paginationHelper.getPagination();
  tablePane.getChildren().clear();
  tablePane.getChildren().addAll(pagination);
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
  if (!mainApp.getHomeController().getElection().getStatus().
      equals("Setup Stage")) {
   addPartyButton.setDisable(true);
  }
 }

 public void setElection(Election election) {
  this.election = election;
  titleLabel.setText(election.getName() + " > Party");
 }

 @FXML
 public void addParty() {
  mainApp.showActionParty(null, election, "add");
 }
}
```

PositionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Arrays;

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class PositionController {
 private MainApp mainApp;
 private Pagination pagination;
 private Election election;
 private ObservableList<Position> tableData;
 private ObservableList<SubElection> subElectionData;
 private ObservableList<String> subElectionName;
 private TableView<Position> table;
 private DatabaseHelper databaseHelper;

 @FXML
 Label titleLabel;
 @FXML
 TextField searchField;
 @FXML
 AnchorPane tablePane;
 @FXML
 ComboBox<String> subElectionComboBox;
 @FXML
 Button addPositionButton;
 @FXML
 Button reorderPositionButton;
```

```
public PositionController() {
 databaseHelper = new DatabaseHelper();
 table = new TableView<Position>();

 TableColumn<Position, String> orderColumn = new
        TableColumn<Position, String>("Order");
 TableColumn<Position, String> nameColumn = new
        TableColumn<Position, String>("Name");
 TableColumn<Position, Button> candidateColumn = new
        TableColumn<Position, Button>("Candidates");
 TableColumn<Position, HBox> actionColumn = new
        TableColumn<Position, HBox>("Action");

 orderColumn.prefWidthProperty().bind(table.widthProperty().
        multiply(0.15));
 nameColumn.prefWidthProperty().bind(table.widthProperty().
        multiply(0.4));
 candidateColumn.prefWidthProperty().bind(table.
        widthProperty().multiply(0.2));
 actionColumn.prefWidthProperty().bind(table.widthProperty()
        .multiply(0.245));
 candidateColumn.setStyle("-fx-alignment: CENTER;");

 orderColumn.setSortable(false);
 nameColumn.setSortable(false);
 candidateColumn.setSortable(false);
 actionColumn.setSortable(false);
 table.getColumns().addAll(Arrays.asList(orderColumn,
        nameColumn, candidateColumn, actionColumn));
 table.setFocusTraversable(false);

 orderColumn.setCellValueFactory(new PropertyValueFactory<
        Position, String>("ballotOrder"));
 nameColumn.setCellValueFactory(new PropertyValueFactory<
        Position, String>("name"));
 candidateColumn.setCellValueFactory(new
        PropertyValueFactory<Position, Button>("
        candidateButton"));
 actionColumn.setCellValueFactory(new PropertyValueFactory
        <Position, HBox>("hBox"));
}

public void setSubElectionComboBox(int index) {
 subElectionName = FXCollections.observableArrayList();
 subElectionData = databaseHelper.searchSubElection("",
        election.getElectionId(), mainApp);

 for (int i = 0; i < subElectionData.size(); i++) {
  subElectionName.add(subElectionData.get(i).getName());
 }

 subElectionComboBox.setItems(subElectionName);
 subElectionComboBox.getSelectionModel().select(index);
}

@FXML
public void setTable() {
 if (subElectionData.size() > 0) {
  tableData = databaseHelper.searchPosition(searchField.
         getText(),
    subElectionData.get(subElectionComboBox.
          getSelectionModel().getSelectedIndex()), mainApp);
  Platform.runLater(() -> table.refresh());
  PaginationHelper<Position> paginationHelper = new
         PaginationHelper<Position>(table, tableData, 4, 1);

  pagination = paginationHelper.getPagination();
  tablePane.getChildren().clear();
  tablePane.getChildren().addAll(pagination);
 } else {
  tablePane.getChildren().addAll(new Label("There are no sub
         -elections yet."));
 }
}

public void setTable(SubElection subElection) {
 for (int i = 0; i < subElectionData.size(); i++) {
  if (subElectionData.get(i).getSubElectionId() ==
         subElection.getSubElectionId())
   setSubElectionComboBox(i);
 }
 setTable();
}

public void setMainApp(MainApp mainApp) {
 this.mainApp = mainApp;
 if (!mainApp.getHomeController().getElection().getStatus().
        equals("Setup Stage")) {
  addPositionButton.setDisable(true);
  reorderPositionButton.setDisable(true);
 }
}

public void setElection(Election election) {
 this.election = election;
 titleLabel.setText(election.getName() + " > Position");
}

@FXML
public void addPosition() {
 mainApp.showActionPosition(null,
   subElectionData.get(subElectionComboBox.
         getSelectionModel().getSelectedIndex()), "add");
}

@FXML
public void reorderPosition() {
 mainApp.showReorderPosition(subElectionData.get(
        subElectionComboBox.getSelectionModel().
        getSelectedIndex()));
}
}
```

ReorderPositionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Collections;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class ReorderPositionController {
 private MainApp mainApp;
 private SubElection subElection;
 private DatabaseHelper databaseHelper;
 ObservableList<String> positionName;
 ObservableList<Position> positionData;

 @FXML
 private ListView<String> positionListView;
 @FXML
 private Label titleLabel;
 @FXML
 Button cancelButton;
 @FXML
 Button upButton;
 @FXML
 Button downButton;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setSubElection(SubElection subElection) {
  this.subElection = subElection;
 }

 public void setData() {
  databaseHelper = new DatabaseHelper();
  ImageView imageView = new ImageView(new Image(MainApp
         .class.getResourceAsStream("images/up.png")));
  imageView.setFitHeight(15);
  imageView.setFitWidth(15);
  upButton.setGraphic(imageView);

  imageView = new ImageView(new Image(MainApp.class.
         getResourceAsStream("images/down.png")));
  imageView.setFitHeight(15);
  imageView.setFitWidth(15);
  downButton.setGraphic(imageView);

  titleLabel.setText(subElection.getName() + " > Reorder
         Position");

  positionData = databaseHelper.searchPosition("", subElection,
         mainApp);
  positionName = FXCollections.observableArrayList();
  for (int i = 0; i < positionData.size(); i++) {
   positionName.add(positionData.get(i).getName());
  }
  positionListView.setItems(positionName);
```

```
 }

 @FXML
 public void save() {
  for (int i = 0; i < positionData.size(); i++) {
   databaseHelper.updatePositionOrder(positionData.get(i).
          getPositionId(), i + 1);
  }
  cancel();
 }

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getPositionController().setTable();
  stage.close();
 }

 @FXML
 public void up() {
  int index = positionListView.getSelectionModel().
         getSelectedIndex();
  if (index != 0 && index != -1) {
   Collections.swap(positionName, index, index - 1);
   Collections.swap(positionData, index, index - 1);
   positionListView.setItems(positionName);
   positionListView.getSelectionModel().select(index - 1);
  }
 }

 public void down() {
  int index = positionListView.getSelectionModel().
         getSelectedIndex();
  if (index != positionName.size() - 1 && index != -1) {
   Collections.swap(positionName, index, index + 1);
   Collections.swap(positionData, index, index + 1);
   positionListView.setItems(positionName);
   positionListView.getSelectionModel().select(index + 1);
  }
 }
}
```

ReorderSubElectionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Collections;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class ReorderSubElectionController {
 private MainApp mainApp;
 private Election election;
 private DatabaseHelper databaseHelper;
 ObservableList<String> subElectionName;
 ObservableList<SubElection> subElectionData;

 @FXML
 private ListView<String> subElectionListView;
 @FXML
 private Label titleLabel;
 @FXML
 Button cancelButton;
 @FXML
 Button upButton;
 @FXML
 Button downButton;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

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

 public void setData() {
  databaseHelper = new DatabaseHelper();
  ImageView imageView = new ImageView(new Image(MainApp
         .class.getResourceAsStream("images/up.png")));
  imageView.setFitHeight(15);
  imageView.setFitWidth(15);
  upButton.setGraphic(imageView);

  imageView = new ImageView(new Image(MainApp.class.
         getResourceAsStream("images/down.png")));
  imageView.setFitHeight(15);
  imageView.setFitWidth(15);
  downButton.setGraphic(imageView);

  titleLabel.setText("Reorder Sub-elections");

  subElectionData = databaseHelper.searchSubElection("",
         election.getElectionId(), mainApp);
  subElectionName = FXCollections.observableArrayList();
  for (int i = 0; i < subElectionData.size(); i++) {
   subElectionName.add(subElectionData.get(i).getName());
  }
  subElectionListView.setItems(subElectionName);
 }

 @FXML
 public void save() {
  for (int i = 0; i < subElectionData.size(); i++) {
   databaseHelper.updateSubElectionOrder(subElectionData.get(
          i).getSubElectionId(), i + 1);
  }
  cancel();
 }

 @FXML
 public void cancel() {
  Stage stage = (Stage) cancelButton.getScene().getWindow();
  mainApp.getCurrentUser().getSubElectionController().setTable
         ();
  stage.close();
 }

 @FXML
 public void up() {
  int index = subElectionListView.getSelectionModel().
         getSelectedIndex();
  if (index != 0 && index != -1) {
   Collections.swap(subElectionName, index, index - 1);
   Collections.swap(subElectionData, index, index - 1);
   subElectionListView.setItems(subElectionName);
   subElectionListView.getSelectionModel().select(index - 1);
  }
 }

 public void down() {
  int index = subElectionListView.getSelectionModel().
         getSelectedIndex();
  if (index != subElectionName.size() - 1 && index != -1) {
   Collections.swap(subElectionName, index, index + 1);
   Collections.swap(subElectionData, index, index + 1);
   subElectionListView.setItems(subElectionName);
   subElectionListView.getSelectionModel().select(index + 1);
  }
 }
}
```

ResultController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.math.BigInteger;
import java.util.Arrays;
import java.util.List;

import javafx.application.Platform;
import javafx.beans.binding.Bindings;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Pagination;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Candidate;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
```

```
import ph.edu.upm.cas.dpsm.bjyu.model.PrivKey;
import ph.edu.upm.cas.dpsm.bjyu.model.Result;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class ResultController {
 private MainApp mainApp;
 private Election election;
 private Pagination pagination;

 private ObservableList<PrivKey> tableData;
 private TableView<PrivKey> table;

 @FXML
 Button uploadButton;
 @FXML
 Label titleLabel;
 @FXML
 Label infoLabel;
 @FXML
 ScrollPane scrollPane;
 @FXML
 Button computeButton;
 @FXML
 AnchorPane tablePane;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setElection(Election election) {
  this.election = election;
  titleLabel.setText(election.getName() + " > Result");
 }

 public void setup() {
  if (election.getStatus().equals("Tallying Stage")) {
   infoLabel.setText("Pleasue upload the private key of the
         election \nto tally the result");
   scrollPane.setVisible(false);

   table = new TableView<PrivKey>();

   TableColumn<PrivKey, String> fileNameColumn = new
         TableColumn<PrivKey, String>("File Name");
   TableColumn<PrivKey, Button> actionColumn = new
         TableColumn<PrivKey, Button>("Action");

   fileNameColumn.prefWidthProperty().bind(table.
         widthProperty().multiply(0.75));
   actionColumn.prefWidthProperty().bind(table.widthProperty
         ().multiply(0.245));
   actionColumn.setStyle("-fx-alignment: CENTER;");

   fileNameColumn.setSortable(false);
   actionColumn.setSortable(false);
   table.getColumns().addAll(Arrays.asList(fileNameColumn,
         actionColumn));
   table.setFocusTraversable(false);

   fileNameColumn.setCellValueFactory(new
         PropertyValueFactory<PrivKey, String>("fileName"));
   actionColumn.setCellValueFactory(new PropertyValueFactory
         <PrivKey, Button>("deleteButton"));

   tableData = FXCollections.observableArrayList();

  } else if (election.getStatus().equals("Setup Stage") ||
        election.getStatus().equals("Voting Stage")) {
   infoLabel.setText("Election has not ended yet.");
   uploadButton.setVisible(false);
   scrollPane.setVisible(false);
   computeButton.setVisible(false);
   tablePane.setVisible(false);
  } else if (election.getStatus().equals("Finished")) {
   infoLabel.setVisible(false);
   uploadButton.setVisible(false);
   computeButton.setVisible(false);
   tablePane.setVisible(false);
   DatabaseHelper databaseHelper = new DatabaseHelper();
   VBox vbox = new VBox();

   List<SubElection> allSubElection = databaseHelper.
         searchSubElection("", election.getElectionId(),
         mainApp);

   for (int i = 0; i < allSubElection.size(); i++) {
    Label subElectionLabel = new Label(allSubElection.get(i).
          getName());
    subElectionLabel.setFont(new Font("Arial", 18));
    vbox.getChildren().addAll(subElectionLabel);

    List<Position> positionList = databaseHelper.
         searchPosition("", allSubElection.get(i), mainApp);
    for (int j = 0; j < positionList.size(); j++) {
     ObservableList<Result> result = FXCollections.
          observableArrayList();

     Label positionLabel = new Label(positionList.get(j).
          getName() + ":");
     vbox.getChildren().addAll(positionLabel);
     List<Candidate> candidateList = databaseHelper.
          getCandidate(positionList.get(j).getPositionId(),
       mainApp);
     for (int k = 0; k < candidateList.size(); k++) {
      String candidateName = candidateList.get(k).
           getFirstName() + " "
        + candidateList.get(k).getLastName() + getParty(
             candidateList.get(k).getPartyName());
      result.add(new Result(candidateName, candidateList.get(
           k).getTally()));
     }
     TableView<Result> resultTable = new TableView<Result
          >();
     TableColumn<Result, String> candidateColumn = new
          TableColumn<Result, String>();
     TableColumn<Result, Integer> tallyColumn = new
          TableColumn<Result, Integer>();
     resultTable.getColumns().addAll(Arrays.asList(
          candidateColumn, tallyColumn));
     resultTable.setItems(result);

     candidateColumn.setCellValueFactory(new
          PropertyValueFactory<Result, String>("
          candidateName"));
     tallyColumn.setCellValueFactory(new PropertyValueFactory
          <Result, Integer>("tally"));

     resultTable.setFixedCellSize(25);
     resultTable.setColumnResizePolicy(TableView.
          CONSTRAINED_RESIZE_POLICY);
     candidateColumn.setPrefWidth(300);
     tallyColumn.setPrefWidth(100);
     resultTable.prefHeightProperty().bind(
       Bindings.size(resultTable.getItems()).multiply(
            resultTable.getFixedCellSize()).add(1.01));
     resultTable.widthProperty().addListener(new
          ChangeListener<Number>() {
      @Override
      public void changed(ObservableValue<? extends Number>
           source, Number oldWidth,
        Number newWidth) {
       Pane header = (Pane) resultTable.lookup("
            TableHeaderRow");
       if (header.isVisible()) {
        header.setMaxHeight(0);
        header.setMinHeight(0);
        header.setPrefHeight(0);
        header.setVisible(false);
       }
      }
     });

     vbox.getChildren().addAll(resultTable, new Label(""));
    }
   }
   vbox.setSpacing(3);

   scrollPane.setContent(vbox);

  }
 }

 @FXML
 public void compute() {
  BigInteger totalPrivateKey = new BigInteger("0");
  for (int i = 0; i < tableData.size(); i++) {
   totalPrivateKey = totalPrivateKey.add(tableData.get(i).
         getPrivateKey());
  }
  mainApp.showTallyProgress(election, totalPrivateKey);
 }

 @FXML
 public void upload() {
  mainApp.showChoosePrivateKey(election, this);
 }

 public String getParty(String partyName) {
  if (partyName.equals("Independent"))
   return "";
  else
   return " (" + partyName + ")";
```

```
 }

 public void addKey(String fileName, BigInteger privKey) {
  tableData.add(new PrivKey(fileName, privKey, mainApp));
  updateTable();
 }

 public void deleteKey(PrivKey privKey) {
  tableData.remove(privKey);

  updateTable();
 }

 public void updateTable() {
  Platform.runLater(() -> table.refresh());
  PaginationHelper<PrivKey> paginationHelper = new
        PaginationHelper<PrivKey>(table, tableData, 5, 1);

  pagination = paginationHelper.getPagination();
  tablePane.getChildren().clear();
  tablePane.getChildren().addAll(pagination);
 }
}
```

StartElectionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Label;

public class StartElectionController {
 @FXML
 Label emailProgress;
 @FXML
 Label generatingKeys;
 @FXML
 Label generatingBallots;
 @FXML
 Label generatingLookupTable;

 public void setEmailProgress(int current, int max) {
  emailProgress.setText("Sending emails: " + current + " out of
        " + max);
 }

 public void setGeneratingKeys(String status) {
  generatingKeys.setText(status);
 }

 public void setGeneratingBallots(String status) {
  generatingBallots.setText(status);
 }

 public void setGeneratingLookupTable(String status) {
  generatingLookupTable.setText(status);
 }
}
```

SubElectionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Arrays;

import javafx.application.Platform;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class SubElectionController {
 private MainApp mainApp;
 private Pagination pagination;
 private Election election;
 private ObservableList<SubElection> tableData;
 private TableView<SubElection> table;
 private DatabaseHelper databaseHelper;

 @FXML
 Label titleLabel;
 @FXML
 TextField searchField;
 @FXML
 AnchorPane tablePane;
 @FXML
 Button addSubElectionButton;
 @FXML
 Button reorderSubElectionButton;

 public SubElectionController() {
  databaseHelper = new DatabaseHelper();
  table = new TableView<SubElection>();
  TableColumn<SubElection, Integer> subElectionOrderColumn
        = new TableColumn<SubElection, Integer>("Order");
  TableColumn<SubElection, String> nameColumn = new
        TableColumn<SubElection, String>("Name");
  TableColumn<SubElection, HBox> actionColumn = new
        TableColumn<SubElection, HBox>("Action");

  subElectionOrderColumn.prefWidthProperty().bind(table.
        widthProperty().multiply(0.15));
  nameColumn.prefWidthProperty().bind(table.widthProperty().
        multiply(0.6));
  actionColumn.prefWidthProperty().bind(table.widthProperty()
        .multiply(0.245));

  subElectionOrderColumn.setSortable(false);
  nameColumn.setSortable(false);
  actionColumn.setSortable(false);
  table.getColumns().addAll(Arrays.asList(
        subElectionOrderColumn, nameColumn, actionColumn));
  table.setFocusTraversable(false);

  subElectionOrderColumn.setCellValueFactory(new
        PropertyValueFactory<SubElection, Integer>("
        subElectionOrder"));
  nameColumn.setCellValueFactory(new PropertyValueFactory<
        SubElection, String>("name"));
  actionColumn.setCellValueFactory(new PropertyValueFactory
        <SubElection, HBox>("hBox"));
 }

 @FXML
 public void setTable() {
  tableData = databaseHelper.searchSubElection(searchField.
        getText(), election.getElectionId(), mainApp);
  Platform.runLater(() -> table.refresh());
  PaginationHelper<SubElection> paginationHelper = new
        PaginationHelper<SubElection>(table, tableData, 5, 1);

  pagination = paginationHelper.getPagination();
  tablePane.getChildren().clear();
  tablePane.getChildren().addAll(pagination);
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
  if (!mainApp.getHomeController().getElection().getStatus().
        equals("Setup Stage")) {
   addSubElectionButton.setDisable(true);
   reorderSubElectionButton.setDisable(true);
  }
 }

 public void setElection(Election election) {
  this.election = election;
  titleLabel.setText(election.getName() + " > Sub-election");

 }

 @FXML
 public void addSubElection() {
  mainApp.showActionSubElection(null, election, "add");
 }

 @FXML
 public void reorderSubElection() {
  mainApp.showReorderSubElection(election);
 }
}
```

TallyElectionController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Label;

public class TallyElectionController {
 @FXML
 Label tallyingVotes;
 @FXML
```

```
 Label computingVerificationCodes;

 public void setTallyProgress(String status) {
  tallyingVotes.setText(status);
 }

 public void setComputingVerificationCodes(int current, int max
       ) {
  computingVerificationCodes.setText("Computing verification
        codes: " + current + " out of " + max);
 }
}
```

ViewBallotController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.List;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ScrollPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Block;
import ph.edu.upm.cas.dpsm.bjyu.model.Candidate;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class ViewBallotController {
 private Block block;
 private DatabaseHelper databaseHelper;
 private MainApp mainApp;

 @FXML
 ScrollPane scrollPane;
 @FXML
 Button closeButton;
 @FXML
 Label titleLabel;

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

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 @FXML
 public void close() {
  Stage stage = (Stage) closeButton.getScene().getWindow();
  stage.close();
 }

 public void setBallot() {
  titleLabel.setText(block.getName() + " > ballot");
  databaseHelper = new DatabaseHelper();
  List<Position> positionList;
  List<SubElection> subElection;
  List<Candidate> candidateList;
  subElection = databaseHelper.getSubElectionOfBlock(block.
        getBlockId(), mainApp);
  VBox vbox = new VBox();

  for (int i = 0; i < subElection.size(); i++) {
   Label subElectionLabel = new Label(subElection.get(i).
         getName());
   subElectionLabel.setFont(new Font("Arial", 18));
   vbox.getChildren().addAll(subElectionLabel);

   positionList = databaseHelper.getAllPosition(subElection.get
         (i).getSubElectionId(), mainApp);
   for (int j = 0; j < positionList.size(); j++) {
    Label positionLabel = new Label(positionList.get(j).
          getName());
    vbox.getChildren().addAll(positionLabel);
    candidateList = databaseHelper.getCandidate(positionList.
          get(j).getPositionId(), mainApp);
    for (int k = 0; k < candidateList.size(); k++) {
     Label candidateLabel = new Label("- " +
       candidateList.get(k).getFirstName() + " " +
            candidateList.get(k).getLastName() + getParty(
            candidateList.get(k).getPartyName()));
     vbox.getChildren().addAll(candidateLabel);
    }
    Label blankLabel = new Label("");
    vbox.getChildren().addAll(blankLabel);
   }
   vbox.setSpacing(3);

   scrollPane.setContent(vbox);
  }
 }

 public String getParty(String partyName) {
  if (partyName.equals("Independent"))
   return "";
  else
   return " (" + partyName + ")";
 }
}
```

VoterController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.util.Arrays;

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Block;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Voter;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.PaginationHelper;

public class VoterController {
 private MainApp mainApp;
 private Pagination pagination;
 private Election election;
 private ObservableList<Voter> tableData;
 private ObservableList<Block> blockData;
 private ObservableList<String> blockName;
 private TableView<Voter> table;
 private DatabaseHelper databaseHelper;

 @FXML
 Label titleLabel;
 @FXML
 TextField searchField;
 @FXML
 AnchorPane tablePane;
 @FXML
 ComboBox<String> blockComboBox;
 @FXML
 Button bulkAddVotersButton;
 @FXML
 Button addVoterButton;

 public VoterController() {
  databaseHelper = new DatabaseHelper();
  table = new TableView<Voter>();

  TableColumn<Voter, String> usernameColumn = new
        TableColumn<Voter, String>("Username");
  TableColumn<Voter, String> emailColumn = new
        TableColumn<Voter, String>("Email");
  TableColumn<Voter, Button> aliasColumn = new
        TableColumn<Voter, Button>("Alias");
  TableColumn<Voter, HBox> actionColumn = new
        TableColumn<Voter, HBox>("Action");

  usernameColumn.prefWidthProperty().bind(table.
        widthProperty().multiply(0.25));
  emailColumn.prefWidthProperty().bind(table.widthProperty().
        multiply(0.3));
  aliasColumn.prefWidthProperty().bind(table.widthProperty().
        multiply(0.2));
  actionColumn.prefWidthProperty().bind(table.widthProperty()
        .multiply(0.245));

  usernameColumn.setSortable(false);
  emailColumn.setSortable(false);
  aliasColumn.setSortable(false);
  actionColumn.setSortable(false);
  table.getColumns().addAll(Arrays.asList(usernameColumn,
        emailColumn, aliasColumn, actionColumn));
```

```
  table.setFocusTraversable(false);

  usernameColumn.setCellValueFactory(new
          PropertyValueFactory<Voter, String>("username"));
  emailColumn.setCellValueFactory(new PropertyValueFactory<
          Voter, String>("email"));
  aliasColumn.setCellValueFactory(new PropertyValueFactory<
          Voter, Button>("alias"));
  actionColumn.setCellValueFactory(new PropertyValueFactory
          <Voter, HBox>("hBox"));
 }

 public void setSubElectionComboBox(int index) {
  blockName = FXCollections.observableArrayList();
  blockData = databaseHelper.searchBlock("", election.
          getElectionId(), mainApp);

  for (int i = 0; i < blockData.size(); i++) {
   blockName.add(blockData.get(i).getName());
  }

  blockComboBox.setItems(blockName);
  blockComboBox.getSelectionModel().select(index);
 }

 @FXML
 public void setTable() {
  if (blockData.size() > 0) {
   tableData = databaseHelper.searchVoter(searchField.getText
           (),
     blockData.get(blockComboBox.getSelectionModel().
             getSelectedIndex()).getBlockId(), mainApp);
   Platform.runLater(() -> table.refresh());
   PaginationHelper<Voter> paginationHelper = new
           PaginationHelper<Voter>(table, tableData, 4, 1);

   pagination = paginationHelper.getPagination();
   tablePane.getChildren().clear();
   tablePane.getChildren().addAll(pagination);
  } else {
   tablePane.getChildren().addAll(new Label("There are no
           blocks yet."));
  }
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
  if (!mainApp.getHomeController().getElection().getStatus().
          equals("Setup Stage")) {
   bulkAddVotersButton.setDisable(true);
   addVoterButton.setDisable(true);
  }
 }

 public void setElection(Election election) {
  this.election = election;
  titleLabel.setText(election.getName() + " > Voters");
 }

 @FXML
 public void addVoters() {
  mainApp.showActionVoter(null, blockData.get(blockComboBox
          .getSelectionModel().getSelectedIndex()), "add");
 }

 @FXML
 public void bulkAddVoters() {
  mainApp.showBulkAddVoters(blockData.get(blockComboBox.
          getSelectionModel().getSelectedIndex()));
 }
}
```

Block.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.geometry.Pos;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class Block {
 private final IntegerProperty blockId;
 private final IntegerProperty electionId;
 private final SimpleStringProperty name;
 private final SimpleStringProperty description;
 private final Button modifyButton;
 private final Button viewButton;
 private final HBox hBox;
 private MainApp mainApp;

 public Block(int blockId, int electionId, String name, String
         description, MainApp mainApp) {
  this.mainApp = mainApp;
  this.blockId = new SimpleIntegerProperty(blockId);
  this.electionId = new SimpleIntegerProperty(electionId);
  this.name = new SimpleStringProperty(name);
  this.description = new SimpleStringProperty(description);
  this.hBox = new HBox();
  this.modifyButton = new Button("Modify");
  this.viewButton = new Button("View");
  hBox.setSpacing(10);

  viewButton.setOnAction((event) -> {
   view();
  });
  modifyButton.setOnAction((event) -> {
   participation();
  });

  Button editButton = new Button();
  editButton.setOnAction((event) -> {
   edit();
  });
  ImageView imageView = new ImageView(new Image(MainApp
          .class.getResourceAsStream("images/edit.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  editButton.setGraphic(imageView);
  editButton.setGraphic(imageView);
  editButton.getStylesheets().add(MainApp.class.getResource("
          css/fextile.css").toExternalForm());
  editButton.getStyleClass().add("btn-success");
  editButton.getStyleClass().add("btn-icon");

  Button deleteButton = new Button();
  deleteButton.setOnAction((event) -> {
   delete();
  });
  imageView = new ImageView(new Image(MainApp.class.
          getResourceAsStream("images/delete.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  deleteButton.setGraphic(imageView);
  deleteButton.getStylesheets().add(MainApp.class.getResource
          ("css/fextile.css").toExternalForm());
  deleteButton.getStyleClass().add("btn-danger");
  deleteButton.getStyleClass().add("btn-icon");
  if (!mainApp.getHomeController().getElection().getStatus().
          equals("Setup Stage")) {
   deleteButton.setDisable(true);
   editButton.setDisable(true);
  }

  hBox.setAlignment(Pos.CENTER);
  hBox.getChildren().addAll(editButton, deleteButton);
 }

 public HBox getHBox() {
  return hBox;
 }

 public Button getModifyButton() {
  return modifyButton;
 }

 public Button getViewButton() {
  return viewButton;
 }

 public void setBlockId(int blockId) {
  this.blockId.set(blockId);
 }

 public void setElectionId(int electionId) {
  this.electionId.set(electionId);
 }

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

 public void setDescription(String description) {
  this.description.set(description);
 }
```

```
 public int getBlockId() {
  return blockId.get();
 }

 public int getElectionId() {
  return electionId.get();
 }

 public String getName() {
  return name.get();
 }

 public String getDescription() {
  return description.get();
 }

 public void edit() {
  mainApp.showActionBlock(this, null, "edit");
 }

 public void participation() {
  mainApp.showParticipation(this);
 }

 public void delete() {
   DatabaseHelper databaseHelper = new DatabaseHelper();

   if (databaseHelper.hasVoters(getBlockId())) {
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("YuVote");
    alert.initOwner(mainApp.getPrimaryStage());
    alert.setHeaderText("Cannot delete this!");
    alert.setContentText("There are voters in this block.");
    alert.showAndWait();

   } else {
    Alert alert = new Alert(AlertType.CONFIRMATION, "Are
          you sure you want to delete this?", ButtonType.YES,
       ButtonType.NO, ButtonType.CANCEL);
    alert.setTitle("YuVote");
    alert.initOwner(mainApp.getPrimaryStage());
    alert.showAndWait();

    if (alert.getResult() == ButtonType.YES) {
     databaseHelper.deleteBlock(getBlockId());
     mainApp.getCurrentUser().getBlockController().setTable();
    }
   }
 }

 public void view(){
   mainApp.showBallot(this);
  }
}
```

BulletinBoard.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.scene.control.Button;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;

public class BulletinBoard {
 private final IntegerProperty voterId;
 private final IntegerProperty blockId;
 private final SimpleStringProperty alias;
 private final SimpleStringProperty verificationCode;
 private final Button viewButton;
 private MainApp mainApp;

 public BulletinBoard(int voterId, int blockId, String alias,
       String verificationCode, MainApp mainApp) {
  this.mainApp = mainApp;
  this.voterId = new SimpleIntegerProperty(voterId);
  this.blockId = new SimpleIntegerProperty(blockId);
  this.verificationCode = new SimpleStringProperty(
        verificationCode);
  this.alias = new SimpleStringProperty(alias);

  viewButton = new Button("View");
  viewButton.setPrefSize(60, 15);
  viewButton.setOnAction((event) -> {
   view();
  });
 }

 public void setVoterId(int voterId) {
  this.voterId.set(voterId);
 }

 public void setBlockId(int blockId) {
  this.blockId.set(blockId);
 }

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

 public void setAlias(String alias) {
  this.alias.set(alias);
 }

 public int getVoterId() {
  return voterId.get();
 }

 public int getBlockId() {
  return blockId.get();
 }

 public String getVerificationCode() {
  return verificationCode.get();
 }

 public String getAlias() {
  return alias.get();
 }

 public Button getViewButton() {
  return viewButton;
 }

 public void view() {
  mainApp.showHashCode(this);
 }

}
```

Candidate.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.geometry.Pos;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class Candidate {
 private final IntegerProperty candidateId;
 private final IntegerProperty positionId;
 private final IntegerProperty partyId;
 private final SimpleStringProperty firstName;
 private final SimpleStringProperty lastName;
 private final HBox hBox;
 private final SimpleStringProperty partyName;
 private final IntegerProperty tally;

 private MainApp mainApp;
 private SubElection subElection;

 public Candidate(int candidateId, int positionId, int partyId,
       String partyName, String firstName, String lastName,
   int tally, MainApp mainApp) {
  this.mainApp = mainApp;
  this.candidateId = new SimpleIntegerProperty(candidateId);
  this.positionId = new SimpleIntegerProperty(positionId);
  this.partyId = new SimpleIntegerProperty(partyId);
  this.firstName = new SimpleStringProperty(firstName);
  this.lastName = new SimpleStringProperty(lastName);
  this.tally = new SimpleIntegerProperty(tally);
  this.partyName = new SimpleStringProperty(partyName);
  this.hBox = new HBox();
  hBox.setSpacing(10);

  Button editButton = new Button();
  editButton.setOnAction((event) -> {
   edit();
  });
  ImageView imageView = new ImageView(new Image(MainApp
        .class.getResourceAsStream("images/edit.png")));
  imageView.setFitHeight(20);
```

```
 imageView.setFitWidth(20);
 editButton.setGraphic(imageView);
 editButton.setGraphic(imageView);
 editButton.getStylesheets().add(MainApp.class.getResource("
        css/fextile.css").toExternalForm());
 editButton.getStyleClass().add("btn-success");
 editButton.getStyleClass().add("btn-icon");

 Button deleteButton = new Button();
 deleteButton.setOnAction((event) -> {
  delete();
 });
 imageView = new ImageView(new Image(MainApp.class.
        getResourceAsStream("images/delete.png")));
 imageView.setFitHeight(20);
 imageView.setFitWidth(20);
 deleteButton.setGraphic(imageView);
 deleteButton.getStylesheets().add(MainApp.class.getResource
        ("css/fextile.css").toExternalForm());
 deleteButton.getStyleClass().add("btn-danger");
 deleteButton.getStyleClass().add("btn-icon");
 if (!mainApp.getHomeController().getElection().getStatus().
        equals("Setup Stage")) {
  deleteButton.setDisable(true);
  editButton.setDisable(true);
 }

 hBox.setAlignment(Pos.CENTER);
 hBox.getChildren().addAll(editButton, deleteButton);
}

public HBox getHBox() {
 return hBox;
}

public void setSubElection(SubElection subElection) {
 this.subElection = subElection;
}

public void setCandidateId(int candidateId) {
 this.candidateId.set(candidateId);
}

public void setPositionId(int positionId) {
 this.positionId.set(positionId);
}

public void setPartyId(int partyId) {
 this.partyId.set(partyId);
}

public void setPartyName(String partyName) {
 this.partyName.set(partyName);
}

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

public void setTally(int tally) {
 this.tally.set(tally);
}

public void lastName(String lastName) {
 this.firstName.set(lastName);
}

public int getCandidateId() {
 return candidateId.get();
}

public int getPositionId() {
 return positionId.get();
}

public int getPartyId() {
 return partyId.get();
}

public String getFirstName() {
 return firstName.get();
}

public String getLastName() {
 return lastName.get();
}

public int getTally() {
 return tally.get();
}

public String getPartyName() {
 return partyName.get();
}

public void edit() {
 mainApp.showActionCandidate(this, null, subElection, "edit");
}

public void delete() {
 Alert alert = new Alert(AlertType.CONFIRMATION, "Are
        you sure you want to delete this?", ButtonType.YES,
    ButtonType.NO);
 alert.setTitle("YuVote");
 alert.initOwner(mainApp.getPrimaryStage());
 alert.showAndWait();

 if (alert.getResult() == ButtonType.YES) {
  DatabaseHelper databaseHelper = new DatabaseHelper();
  databaseHelper.deleteCandidate(getCandidateId());
  mainApp.getCurrentUser().getCandidateController().setTable
        ();
 }
}
}
```

Election.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.geometry.Pos;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class Election {
 private final IntegerProperty electionId;
 private final IntegerProperty creatorId;
 private final SimpleStringProperty name;
 private final SimpleStringProperty description;
 private final SimpleStringProperty status;
 private final HBox hBox;
 private final Button manageButton;
 private MainApp mainApp;

 public Election(int electionId, int creatorId, String name,
       String description, String status, MainApp mainApp) {
  this.mainApp = mainApp;
  this.electionId = new SimpleIntegerProperty(electionId);
  this.creatorId = new SimpleIntegerProperty(creatorId);
  this.name = new SimpleStringProperty(name);
  this.description = new SimpleStringProperty(description);
  this.status = new SimpleStringProperty(status);
  this.hBox = new HBox();
  hBox.setSpacing(10);

  Button editButton = new Button();
  editButton.setOnAction((event) -> {
   edit();
  });
  ImageView imageView = new ImageView(new Image(MainApp
        .class.getResourceAsStream("images/edit.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  editButton.setGraphic(imageView);
  editButton.setGraphic(imageView);
  editButton.getStylesheets().add(MainApp.class.getResource("
        css/fextile.css").toExternalForm());
  editButton.getStyleClass().add("btn-success");
  editButton.getStyleClass().add("btn-icon");

  Button deleteButton = new Button();
  deleteButton.setOnAction((event) -> {
   delete();
  });
  imageView = new ImageView(new Image(MainApp.class.
        getResourceAsStream("images/delete.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  deleteButton.setGraphic(imageView);
  deleteButton.getStylesheets().add(MainApp.class.getResource
        ("css/fextile.css").toExternalForm());
  deleteButton.getStyleClass().add("btn-danger");
  deleteButton.getStyleClass().add("btn-icon");
  if (!status.equals("Setup Stage")) {
```

```
    deleteButton.setDisable(true);
    editButton.setDisable(true);
   }

   hBox.setAlignment(Pos.CENTER);
   hBox.getChildren().addAll(editButton, deleteButton);
   manageButton = new Button("Manage");

   manageButton.setPrefSize(80, 15);
   manageButton.setAlignment(Pos.CENTER);
   manageButton.setOnAction((event) -> {
    manage();
   });
  }

  public Button getManageButton() {
   return manageButton;
  }

  public HBox getHBox() {
   return hBox;
  }

  public void setElectionId(int electionId) {
   this.electionId.set(electionId);
  }

  public void setCreatorId(int creatorId) {
   this.creatorId.set(creatorId);
  }

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

  public void setDescription(String description) {
   this.name.set(description);
  }

  public void setStatus(String status) {
   this.status.set(status);
  }

  public int getElectionId() {
   return electionId.get();
  }

  public int getCreatorId() {
   return creatorId.get();
  }

  public String getName() {
   return name.get();
  }

  public String getDescription() {
   return description.get();
  }

  public String getStatus() {
   return status.get();
  }

  public void edit() {
   mainApp.showActionElection(this, "edit");
  }

  public void manage() {
   mainApp.showManageElection(this);
  }

  public void delete() {
   Alert alert = new Alert(AlertType.CONFIRMATION, "Are
       you sure you want to delete this?", ButtonType.YES,
     ButtonType.NO);
   alert.setTitle("YuVote");
   alert.initOwner(mainApp.getPrimaryStage());
   alert.showAndWait();

   if (alert.getResult() == ButtonType.YES) {
    DatabaseHelper databaseHelper = new DatabaseHelper();
    databaseHelper.deleteElection(getElectionId());
    mainApp.getCurrentUser().getMainVOController().setTable();
   }
  }
}
```

Party.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.geometry.Pos;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class Party {
 private final IntegerProperty partyId;
 private final IntegerProperty electionId;
 private final SimpleStringProperty name;
 private final HBox hBox;
 private MainApp mainApp;

 public Party(int partyId, int electionId, String name,
       MainApp mainApp) {
  this.mainApp = mainApp;
  this.electionId = new SimpleIntegerProperty(electionId);
  this.partyId = new SimpleIntegerProperty(partyId);
  this.name = new SimpleStringProperty(name);
  this.hBox = new HBox();
  hBox.setSpacing(10);

  Button editButton = new Button();
  editButton.setOnAction((event) -> {
   edit();
  });
  ImageView imageView = new ImageView(new Image(MainApp
       .class.getResourceAsStream("images/edit.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  editButton.setGraphic(imageView);
  editButton.setGraphic(imageView);
  editButton.getStylesheets().add(MainApp.class.getResource("
       css/fextile.css").toExternalForm());
  editButton.getStyleClass().add("btn-success");
  editButton.getStyleClass().add("btn-icon");

  Button deleteButton = new Button();
  deleteButton.setOnAction((event) -> {
   delete();
  });
  imageView = new ImageView(new Image(MainApp.class.
       getResourceAsStream("images/delete.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  deleteButton.setGraphic(imageView);
  deleteButton.getStylesheets().add(MainApp.class.getResource
       ("css/fextile.css").toExternalForm());
  deleteButton.getStyleClass().add("btn-danger");
  deleteButton.getStyleClass().add("btn-icon");
  if (!mainApp.getHomeController().getElection().getStatus().
       equals("Setup Stage")) {
   deleteButton.setDisable(true);
   editButton.setDisable(true);
  }

  hBox.setAlignment(Pos.CENTER);
  hBox.getChildren().addAll(editButton, deleteButton);
 }

 public HBox getHBox() {
  return hBox;
 }

 public void setPartyId(int partyId) {
  this.partyId.set(partyId);
 }

 public void setElectionId(int electionId) {
  this.electionId.set(electionId);
 }

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

 public int getPartyId() {
  return partyId.get();
 }

 public int getElectionId() {
  return electionId.get();
 }

 public String getName() {
```

```
    return name.get();
  }

  public void edit() {
    mainApp.showActionParty(this, null, "edit");
  }

  public void delete() {
    DatabaseHelper databaseHelper = new DatabaseHelper();
    if (databaseHelper.canDeleteParty(getPartyId())) {
      Alert alert = new Alert(AlertType.CONFIRMATION, "Are
          you sure you want to delete this?", ButtonType.YES,
        ButtonType.NO);
      alert.setTitle("YuVote");
      alert.initOwner(mainApp.getPrimaryStage());
      alert.showAndWait();

      if (alert.getResult() == ButtonType.YES) {
        databaseHelper.deleteParty(getPartyId());
        mainApp.getCurrentUser().getPartyController().setTable();
      }
    } else {
      Alert alert = new Alert(AlertType.INFORMATION);
      alert.setTitle("YuVote");
      alert.initOwner(mainApp.getPrimaryStage());
      alert.setHeaderText("Cannot delete this!");
      alert.setContentText("There are candidates in this party");

      alert.showAndWait();
    }
  }
}
```

Position.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.geometry.Pos;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class Position {
  private final IntegerProperty positionId;
  private final IntegerProperty subElectionId;
  private final SimpleStringProperty name;
  private final IntegerProperty ballotOrder;
  private final IntegerProperty maxVote;
  private final HBox hBox;
  private final Button candidateButton;
  private MainApp mainApp;
  private SubElection subElection;

  public Position(int positionId, int subElectionId, String name
      , int ballotOrder, int maxVote, MainApp mainApp) {
    this.mainApp = mainApp;
    this.positionId = new SimpleIntegerProperty(positionId);
    this.subElectionId = new SimpleIntegerProperty(
        subElectionId);
    this.name = new SimpleStringProperty(name);
    this.ballotOrder = new SimpleIntegerProperty(ballotOrder);
    this.maxVote = new SimpleIntegerProperty(maxVote);
    this.candidateButton = new Button("View");
    this.hBox = new HBox();
    hBox.setSpacing(10);

    candidateButton.setOnAction((event) -> {
      viewCandidate();
    });

    Button editButton = new Button();
    editButton.setOnAction((event) -> {
      edit();
    });
    ImageView imageView = new ImageView(new Image(MainApp
        .class.getResourceAsStream("images/edit.png")));
    imageView.setFitHeight(20);
    imageView.setFitWidth(20);
    editButton.setGraphic(imageView);
    editButton.setGraphic(imageView);
    editButton.getStylesheets().add(MainApp.class.getResource("
        css/fextile.css").toExternalForm());
    editButton.getStyleClass().add("btn-success");
    editButton.getStyleClass().add("btn-icon");

    Button deleteButton = new Button();
    deleteButton.setOnAction((event) -> {
      delete();
    });
    imageView = new ImageView(new Image(MainApp.class.
        getResourceAsStream("images/delete.png")));
    imageView.setFitHeight(20);
    imageView.setFitWidth(20);
    deleteButton.setGraphic(imageView);
    deleteButton.getStylesheets().add(MainApp.class.getResource
        ("css/fextile.css").toExternalForm());
    deleteButton.getStyleClass().add("btn-danger");
    deleteButton.getStyleClass().add("btn-icon");

    if (!mainApp.getHomeController().getElection().getStatus().
        equals("Setup Stage")) {
      deleteButton.setDisable(true);
      editButton.setDisable(true);
    }
    hBox.setAlignment(Pos.CENTER);
    hBox.getChildren().addAll(editButton, deleteButton);
  }

  public void setSubElection(SubElection subElection) {
    this.subElection = subElection;
  }

  public HBox getHBox() {
    return hBox;
  }

  public Button getCandidateButton() {
    return candidateButton;
  }

  public void setPositionId(int positionId) {
    this.positionId.set(positionId);
  }

  public void setSubElectionId(int subElectionId) {
    this.subElectionId.set(subElectionId);
  }

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

  public void setBallotOrder(int ballotOrder) {
    this.ballotOrder.set(ballotOrder);
  }

  public void setMaxVote(int maxVote) {
    this.maxVote.set(maxVote);
  }

  public int getPositionId() {
    return positionId.get();
  }

  public int getSubElectionId() {
    return subElectionId.get();
  }

  public String getName() {
    return name.get();
  }

  public int getBallotOrder() {
    return ballotOrder.get();
  }

  public int getMaxVote() {
    return maxVote.get();
  }

  public void edit() {
    mainApp.showActionPosition(this, subElection, "edit");
  }

  public void viewCandidate() {
    mainApp.showCandidate(this, subElection);
  }

  public void delete() {
    Alert alert = new Alert(AlertType.CONFIRMATION, "Are
        you sure you want to delete this?", ButtonType.YES,
      ButtonType.NO);
    alert.setTitle("YuVote");
    alert.initOwner(mainApp.getPrimaryStage());
    alert.showAndWait();
```

```
   if (alert.getResult() == ButtonType.YES) {
    DatabaseHelper databaseHelper = new DatabaseHelper();
    databaseHelper.deletePosition(this);
    mainApp.getCurrentUser().getPositionController().setTable();
   }
  }
}
```

PrivKey.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import java.math.BigInteger;

import javafx.beans.property.SimpleStringProperty;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;

public class PrivKey {
 private final SimpleStringProperty fileName;
 private final BigInteger privateKey;
 private final Button deleteButton;
 private MainApp mainApp;

 public PrivKey(String fileName, BigInteger privateKey,
        MainApp mainApp) {
  this.mainApp = mainApp;
  this.fileName = new SimpleStringProperty(fileName);
  this.privateKey = privateKey;

  deleteButton = new Button();
  ImageView imageView = new ImageView(new Image(MainApp
        .class.getResourceAsStream("images/delete.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  deleteButton.setGraphic(imageView);
  deleteButton.getStylesheets().add(MainApp.class.getResource
        ("css/fextile.css").toExternalForm());
  deleteButton.getStyleClass().add("btn-danger");
  deleteButton.getStyleClass().add("btn-icon");
  deleteButton.setOnAction((event) -> {
   remove();
  });
 }

 public void setFileName(String fileName) {
  this.fileName.set(fileName);
 }

 public String getFileName() {
  return fileName.get();
 }

 public BigInteger getPrivateKey() {
  return privateKey;
 }

 public Button getDeleteButton() {
  return deleteButton;
 }

 public void remove() {
  Alert alert = new Alert(AlertType.CONFIRMATION, "Are
        you sure you want to delete this?", ButtonType.YES,
     ButtonType.NO);
  alert.setTitle("YuVote");
  alert.initOwner(mainApp.getPrimaryStage());
  alert.showAndWait();

  if (alert.getResult() == ButtonType.YES) {
   mainApp.getCurrentUser().getResultController().deleteKey(
         this);
  }
 }
}
```

Requirements.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.SimpleStringProperty;
import javafx.scene.control.Label;
import javafx.scene.control.Tooltip;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.util.TooltipHelper;

public class Requirements {
 private final SimpleStringProperty requirement;
 private final SimpleStringProperty status;
 private final HBox hBox;

 public Requirements(String requirement, String status) {
  this.requirement = new SimpleStringProperty(requirement);
  this.status = new SimpleStringProperty(status);
  this.hBox = new HBox();

  hBox.setSpacing(10);
  Label label = new Label(requirement);
  Image image = new Image((MainApp.class.getResource("
        images/info.png")).toExternalForm());
  ImageView imageView = new ImageView();
  imageView.setImage(image);
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);

  String stringTooltip = "";
  if (requirement.equals("Sub-elections"))
   stringTooltip = "There should be at least 1 sub-election";
  if (requirement.equals("Blocks"))
   stringTooltip = "There should be at least 1 block";
  if (requirement.equals("Block Participation"))
   stringTooltip = "Each block should participate in at least 1
         sub-election";
  if (requirement.equals("Positions"))
   stringTooltip = "Each sub-election should have at least 1
         position";
  if (requirement.equals("Candidates"))
   stringTooltip = "Each position should have at least 2
         candidates \nand should not exceed the max number of
          votes";
  if (requirement.equals("Voters"))
   stringTooltip = "Each block should have at least 1 voter";

  Tooltip tooltip = new Tooltip(stringTooltip);
  TooltipHelper.hackTooltipStartTiming(tooltip);

  Tooltip.install(hBox, tooltip);
  hBox.getChildren().addAll(label, imageView);
 }

 public HBox getHBox() {
  return hBox;
 }

 public String getRequirement() {
  return requirement.get();
 }

 public void setRequirement(String requirement) {
  this.requirement.set(requirement);
 }

 public String getStatus() {
  return status.get();
 }

 public void setStatus(String status) {
  this.status.set(status);
 }
}
```

Result.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;

public class Result {
 private final SimpleStringProperty candidateName;
 private final IntegerProperty tally;

 public Result(String candidateName, int tally) {
  this.candidateName = new SimpleStringProperty(
        candidateName);
  this.tally = new SimpleIntegerProperty(tally);
 }

 public void setCandidateName(String candidateName) {
  this.candidateName.set(candidateName);
 }

 public void setTally(int tally) {
  this.tally.set(tally);
 }
```

```
 public String getCandidateName() {
  return candidateName.get();
 }

 public int getTally() {
  return tally.get();
 }

}
```

SubElection.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.geometry.Pos;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class SubElection {
 private final IntegerProperty subElectionId;
 private final IntegerProperty electionId;
 private final SimpleStringProperty name;
 private final IntegerProperty subElectionOrder;
 private final HBox hBox;
 private MainApp mainApp;

 public SubElection(int subElectionId, int electionId, String
       name, int subElectionOrder, MainApp mainApp) {
  this.mainApp = mainApp;
  this.subElectionId = new SimpleIntegerProperty(
        subElectionId);
  this.electionId = new SimpleIntegerProperty(electionId);
  this.name = new SimpleStringProperty(name);
  this.subElectionOrder = new SimpleIntegerProperty(
        subElectionOrder);
  this.hBox = new HBox();
  hBox.setSpacing(10);

  Button editButton = new Button();
  editButton.setOnAction((event) -> {
   edit();
  });
  ImageView imageView = new ImageView(new Image(MainApp
        .class.getResourceAsStream("images/edit.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  editButton.setGraphic(imageView);
  editButton.setGraphic(imageView);
  editButton.getStylesheets().add(MainApp.class.getResource("
        css/fextile.css").toExternalForm());
  editButton.getStyleClass().add("btn-success");
  editButton.getStyleClass().add("btn-icon");

  Button deleteButton = new Button();
  deleteButton.setOnAction((event) -> {
   delete();
  });
  imageView = new ImageView(new Image(MainApp.class.
        getResourceAsStream("images/delete.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  deleteButton.setGraphic(imageView);
  deleteButton.getStylesheets().add(MainApp.class.getResource
        ("css/fextile.css").toExternalForm());
  deleteButton.getStyleClass().add("btn-danger");
  deleteButton.getStyleClass().add("btn-icon");
  if (!mainApp.getHomeController().getElection().getStatus().
        equals("Setup Stage")) {
   deleteButton.setDisable(true);
   editButton.setDisable(true);
  }
  hBox.setAlignment(Pos.CENTER);
  hBox.getChildren().addAll(editButton, deleteButton);
 }

 public HBox getHBox() {
  return hBox;
 }

 public void setSubElectionId(int subElectionId) {
  this.subElectionId.set(subElectionId);
 }

 public void setElectionId(int electionId) {
  this.electionId.set(electionId);
 }

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

 public void setSubElectionOrder(int subElectionOrder) {
  this.subElectionOrder.set(subElectionOrder);
 }

 public int getSubElectionId() {
  return subElectionId.get();
 }

 public int getElectionId() {
  return electionId.get();
 }

 public String getName() {
  return name.get();
 }

 public int getSubElectionOrder() {
  return subElectionOrder.get();
 }

 public void edit() {
  mainApp.showActionSubElection(this, null, "edit");
 }

 public void delete() {
  DatabaseHelper databaseHelper = new DatabaseHelper();

  if (databaseHelper.hasPosition(getSubElectionId())) {
   Alert alert = new Alert(AlertType.INFORMATION);
   alert.setTitle("YuVote");
   alert.initOwner(mainApp.getPrimaryStage());
   alert.setHeaderText("Cannot delete this!");
   alert.setContentText("There are positions in this sub-
         election.");
   alert.showAndWait();

  } else if (databaseHelper.subElectionHasBlockParticipation(
        getSubElectionId())) {
   Alert alert = new Alert(AlertType.INFORMATION);
   alert.setTitle("YuVote");
   alert.initOwner(mainApp.getPrimaryStage());
   alert.setHeaderText("Cannot delete this!");
   alert.setContentText("There are participating blocks in this
         sub-election.");
   alert.showAndWait();
  } else {
   Alert alert = new Alert(AlertType.CONFIRMATION, "Are
         you sure you want to delete this?", ButtonType.YES,
     ButtonType.NO);
   alert.setTitle("YuVote");
   alert.initOwner(mainApp.getPrimaryStage());
   alert.showAndWait();

   if (alert.getResult() == ButtonType.YES) {
    databaseHelper.deleteSubElection(getElectionId(),
          getSubElectionId(), getSubElectionOrder());
    mainApp.getCurrentUser().getSubElectionController().
          setTable();
   }
  }
 }
}
```

User.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.geometry.Pos;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.controller.BlockController;
import ph.edu.upm.cas.dpsm.bjyu.controller.CandidateController
      ;
```

```
import ph.edu.upm.cas.dpsm.bjyu.controller.MainSAController;
import ph.edu.upm.cas.dpsm.bjyu.controller.MainVOController;
import ph.edu.upm.cas.dpsm.bjyu.controller.PartyController;
import ph.edu.upm.cas.dpsm.bjyu.controller.PositionController;
import ph.edu.upm.cas.dpsm.bjyu.controller.ResultController;
import ph.edu.upm.cas.dpsm.bjyu.controller.
    SubElectionController;
import ph.edu.upm.cas.dpsm.bjyu.controller.VoterController;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class User {
 private final IntegerProperty userId;
 private final SimpleStringProperty username;
 private final SimpleStringProperty email;
 private final SimpleStringProperty type;
 private final HBox hBox;
 private final byte[] salt;
 private MainApp mainApp;
 private MainSAController mainSAController;
 private MainVOController mainVOController;
 private PartyController partyController;
 private BlockController blockController;
 private PositionController positionController;
 private CandidateController candidateController;
 private SubElectionController subElectionController;
 private VoterController voterController;
 private ResultController resultController;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 public void setMainSAController(MainSAController
     mainSAController) {
  this.mainSAController = mainSAController;
 }

 public MainSAController getMainSAController() {
  return mainSAController;
 }

 public void setMainVOController(MainVOController
     mainVOController) {
  this.mainVOController = mainVOController;
 }

 public MainVOController getMainVOController() {
  return mainVOController;
 }

 public void setPartyController(PartyController partyController)
     {
  this.partyController = partyController;
 }

 public PartyController getPartyController() {
  return partyController;
 }

 public void setBlockController(BlockController blockController)
     {
  this.blockController = blockController;
 }

 public BlockController getBlockController() {
  return blockController;
 }

 public void setPositionController(PositionController
     positionController) {
  this.positionController = positionController;
 }

 public PositionController getPositionController() {
  return positionController;
 }

 public void setSubElectionController(SubElectionController
     subElectionController) {
  this.subElectionController = subElectionController;
 }

 public SubElectionController getSubElectionController() {
  return subElectionController;
 }

 public void setCandidateController(CandidateController
     candidateController) {
  this.candidateController = candidateController;
 }

 public VoterController getVoterController() {
  return voterController;
 }

 public void setVoterController(VoterController voterController)
     {
  this.voterController = voterController;
 }

 public CandidateController getCandidateController() {
  return candidateController;
 }

 public ResultController getResultController() {
  return resultController;
 }

 public void setResultController(ResultController
     resultController) {
  this.resultController = resultController;
 }

 public User(int userId, String username, byte[] salt, String
     email, String type) {
  this.userId = new SimpleIntegerProperty(userId);
  this.username = new SimpleStringProperty(username);
  this.salt = salt;
  this.email = new SimpleStringProperty(email);
  this.type = new SimpleStringProperty(type);
  this.hBox = new HBox();
  hBox.setSpacing(10);

  Button editButton = new Button();
  editButton.setOnAction((event) -> {
   edit();
  });
  ImageView imageView = new ImageView(new Image(MainApp
      .class.getResourceAsStream("images/edit.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  editButton.setGraphic(imageView);
  editButton.setGraphic(imageView);
  editButton.getStylesheets().add(MainApp.class.getResource("
      css/fextile.css").toExternalForm());
  editButton.getStyleClass().add("btn-success");
  editButton.getStyleClass().add("btn-icon");

  Button deleteButton = new Button();
  deleteButton.setOnAction((event) -> {
   delete();
  });
  imageView = new ImageView(new Image(MainApp.class.
      getResourceAsStream("images/delete.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  deleteButton.setGraphic(imageView);
  deleteButton.getStylesheets().add(MainApp.class.getResource
      ("css/fextile.css").toExternalForm());
  deleteButton.getStyleClass().add("btn-danger");
  deleteButton.getStyleClass().add("btn-icon");

  hBox.setAlignment(Pos.CENTER);
  hBox.getChildren().addAll(editButton, deleteButton);

 }

 public HBox getHBox() {
  return hBox;
 }

 public void setUserId(int userId) {
  this.userId.set(userId);
 }

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

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

 public void setType(String type) {
  this.type.set(type);
 }

 public int getUserId() {
  return userId.get();
 }

 public String getUsername() {
  return username.get();
 }
```

```
 public String getEmail() {
  return email.get();
 }

 public String getType() {
  return type.get();
 }

 public byte[] getSalt() {
  return salt;
 }

 public void edit() {
  mainApp.showEditUser(this);
 }

 public void delete() {
  DatabaseHelper databaseHelper = new DatabaseHelper();

   if (databaseHelper.hasElection(getUserId())) {
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("YuVote");
    alert.initOwner(mainApp.getPrimaryStage());
    alert.setHeaderText("Cannot delete this!");
    alert.setContentText("This user manages an election.");
    alert.showAndWait();

   } else {
    Alert alert = new Alert(AlertType.CONFIRMATION, "Are
          you sure you want to delete this?", ButtonType.YES,
       ButtonType.NO);
    alert.setTitle("YuVote");
    alert.initOwner(mainApp.getPrimaryStage());
    alert.showAndWait();

    if (alert.getResult() == ButtonType.YES) {
     databaseHelper.deleteUser(getUserId());
     mainApp.getCurrentUser().getMainSAController().search();
    }
   }
  }
 }
```

Voter.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.geometry.Pos;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class Voter {
 private final IntegerProperty voterId;
 private final IntegerProperty userId;
 private final IntegerProperty blockId;
 private final SimpleStringProperty username;
 private final SimpleStringProperty alias;
 private final SimpleStringProperty email;
 private final HBox hBox;
 private MainApp mainApp;

 public Voter(int voterId, int userId, int blockId, String
       username, String alias, String email, MainApp mainApp
       ) {
  this.mainApp = mainApp;
  this.voterId = new SimpleIntegerProperty(voterId);
  this.userId = new SimpleIntegerProperty(userId);
  this.blockId = new SimpleIntegerProperty(blockId);
  this.username = new SimpleStringProperty(username);
  this.email = new SimpleStringProperty(email);
  this.alias = new SimpleStringProperty(alias);
  this.hBox = new HBox();
  hBox.setSpacing(10);

  Button editButton = new Button();
  editButton.setOnAction((event) -> {
   edit();
  });
  ImageView imageView = new ImageView(new Image(MainApp
        .class.getResourceAsStream("images/edit.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  editButton.setGraphic(imageView);
  editButton.setGraphic(imageView);
  editButton.getStylesheets().add(MainApp.class.getResource("
        css/fextile.css").toExternalForm());
  editButton.getStyleClass().add("btn-success");
  editButton.getStyleClass().add("btn-icon");

  Button deleteButton = new Button();
  deleteButton.setOnAction((event) -> {
   delete();
  });
  imageView = new ImageView(new Image(MainApp.class.
        getResourceAsStream("images/delete.png")));
  imageView.setFitHeight(20);
  imageView.setFitWidth(20);
  deleteButton.setGraphic(imageView);
  deleteButton.getStylesheets().add(MainApp.class.getResource
        ("css/fextile.css").toExternalForm());
  deleteButton.getStyleClass().add("btn-danger");
  deleteButton.getStyleClass().add("btn-icon");

  if (!mainApp.getHomeController().getElection().getStatus().
        equals("Setup Stage")) {
   deleteButton.setDisable(true);
   editButton.setDisable(true);
  }
  hBox.setAlignment(Pos.CENTER);
  hBox.getChildren().addAll(editButton, deleteButton);
 }

 public HBox getHBox() {
  return hBox;
 }

 public void setVoterId(int voterId) {
  this.voterId.set(voterId);
 }

 public void setUserId(int userId) {
  this.userId.set(userId);
 }

 public void setBlockId(int blockId) {
  this.blockId.set(blockId);
 }

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

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

 public void setAlias(String alias) {
  this.alias.set(alias);
 }

 public int getVoterId() {
  return voterId.get();
 }

 public int getUserId() {
  return userId.get();
 }

 public int getBlockId() {
  return blockId.get();
 }

 public String getUsername() {
  return username.get();
 }

 public String getEmail() {
  return email.get();
 }

 public String getAlias() {
  return alias.get();
 }

 public void edit() {
  mainApp.showActionVoter(this, null, "edit");
 }

 public void delete() {
  Alert alert = new Alert(AlertType.CONFIRMATION, "Are
        you sure you want to delete this?", ButtonType.YES,
     ButtonType.NO);
  alert.setTitle("YuVote");
```

```
    alert.initOwner(mainApp.getPrimaryStage());
    alert.showAndWait();

    if (alert.getResult() == ButtonType.YES) {
     DatabaseHelper databaseHelper = new DatabaseHelper();
     databaseHelper.deleteVoter(getVoterId());
     databaseHelper.deleteUnusedUser();
     mainApp.getCurrentUser().getVoterController().setTable();
    }
   }
}
```

DatabaseHelper.java

```
package ph.edu.upm.cas.dpsm.bjyu.util;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.controller.MainSAController;
import ph.edu.upm.cas.dpsm.bjyu.model.Block;
import ph.edu.upm.cas.dpsm.bjyu.model.BulletinBoard;
import ph.edu.upm.cas.dpsm.bjyu.model.Candidate;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Party;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.SubElection;
import ph.edu.upm.cas.dpsm.bjyu.model.User;
import ph.edu.upm.cas.dpsm.bjyu.model.Voter;

public class DatabaseHelper {
 private Connection conn;

 public DatabaseHelper() {
  Properties prop = new Properties();
  InputStream input = null;
  String keyStore = "", keyStorePassword = "", trustStore =
        "", trustStorePassword = "";
  String url = "jdbc:mysql://";
  String user = "", password = "";
  try {

   input = new FileInputStream("config.properties");
   prop.load(input);
   keyStore = prop.getProperty("javax.net.ssl.keyStore");
   keyStorePassword = prop.getProperty("javax.net.ssl.
         keyStorePassword");
   trustStore = prop.getProperty("javax.net.ssl.trustStore");
   trustStorePassword = prop.getProperty("javax.net.ssl.
         trustStorePassword");
   user = prop.getProperty("db_user");
   password = prop.getProperty("db_password");
   url += prop.getProperty("db_host") + ":" + prop.
         getProperty("db_port");
   url += "/yuVote?verifyServerCertificate=true&useSSL=true
         &requireSSL=true";
  } catch (IOException ex) {
   ex.printStackTrace();
  }

  System.setProperty("javax.net.ssl.keyStore", keyStore);
  System.setProperty("javax.net.ssl.keyStorePassword",
        keyStorePassword);
  System.setProperty("javax.net.ssl.trustStore", trustStore);
  System.setProperty("javax.net.ssl.trustStorePassword",
        trustStorePassword);

  try {
   conn = null;
   Class.forName("com.mysql.jdbc.Driver");
   conn = DriverManager.getConnection(url, user, password);
  } catch (Exception e) {
   System.out.println(e);
  }
 }

 public User usernameMatchesPassword(String username, String
       password) {
  try {
   PasswordHelper passwordHelper = new PasswordHelper();
   String sql = "SELECT * FROM user WHERE username = ?
         ";
   PreparedStatement ps = conn.prepareStatement(sql);
   ps.setString(1, username);
   ResultSet rs = ps.executeQuery();
   if (rs.next()) {
    byte[] salt = rs.getBytes(4);
    if (rs.getString(3).equals(passwordHelper.
          getSecurePassword(password, salt))) {
     User user = new User(rs.getInt(1), rs.getString(2), rs.
           getBytes(4), rs.getString(5),
       rs.getString(6));
     return user;
    }
   } else
    return null;
  } catch (Exception e) {
   System.out.println(e);
  }
  return null;
 }

 public boolean isUsernameTaken(String username) {
  try {
   String sql = "SELECT * FROM user WHERE";
   sql += " username = ? ";
   PreparedStatement ps = conn.prepareStatement(sql);
   ps.setString(1, username);
   ResultSet rs = ps.executeQuery();
   return rs.next();
  } catch (Exception e) {
   return false;
  }
 }

 public boolean isEmailTaken(String email) {
  try {
   String sql = "SELECT * FROM user WHERE";
   sql += " email = ? ";
   PreparedStatement ps = conn.prepareStatement(sql);
   ps.setString(1, email);

   ResultSet rs = ps.executeQuery();

   return rs.next();
  } catch (Exception e) {
   return false;
  }
 }

 public void updateUsernameEmail(String username, String
       email, int userId) {
  try {
   String sql = "UPDATE user SET username = ?, email = ?
         WHERE user_id = ?";
   PreparedStatement ps = conn.prepareStatement(sql);
   ps.setString(1, username);
   ps.setString(2, email);
   ps.setInt(3, userId);
   ps.executeUpdate();
  } catch (Exception e) {
   e.printStackTrace();
  }
 }

 public void updateUsernameEmailPassword(String username,
       String email, byte[] salt, String password, int userId) {
  try {
   String sql = "UPDATE user SET username = ?, email = ?,
         password = ? WHERE user_id = ?;";
   PreparedStatement ps = conn.prepareStatement(sql);
   ps.setString(1, username);
   ps.setString(2, email);
   PasswordHelper passwordHelper = new PasswordHelper();

   String hashedPassword = passwordHelper.getSecurePassword(
         password, salt);
   ps.setString(3, hashedPassword);
   ps.setInt(4, userId);

   ps.executeUpdate();// integer
  } catch (Exception e) {
   e.printStackTrace();
  }

 }

 public ObservableList<User> searchVotingOfficial(String
       keyword, MainApp mainApp,
   MainSAController mainSAController) {
  ObservableList<User> data = FXCollections.
        observableArrayList();
```

```
 try {
  String sql = "SELECT * FROM user WHERE";
  sql += " type = ? AND (username LIKE ? OR email LIKE
       ?)";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, "Voting Official");
  ps.setString(2, "%" + keyword + "%");
  ps.setString(3, "%" + keyword + "%");
  ResultSet rs = ps.executeQuery();

  while (rs.next()) {
   User user = new User(rs.getInt(1), rs.getString(2), rs.
        getBytes(4), rs.getString(5), rs.getString(6));
   user.setMainApp(mainApp);
   user.setMainSAController(mainSAController);
   data.add(user);
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public void addVotingOfficial(String username, String password
      , String email) {
 try {
  String sql = "INSERT INTO user (username, password, salt,
       email, type) VALUES (?, ?, ?, ?, ?)";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, username);
  PasswordHelper passwordHelper = new PasswordHelper();
  byte[] salt = passwordHelper.getSalt();
  String hashedPassword = passwordHelper.getSecurePassword(
       password, salt);
  ps.setString(2, hashedPassword);
  ps.setBytes(3, salt);
  ps.setString(4, email);
  ps.setString(5, "Voting Official");

  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public ObservableList<Election> searchElection(String
      keyword, int creatorId, String status, MainApp mainApp
      ) {
 ObservableList<Election> data = FXCollections.
      observableArrayList();

 try {
  String sql = "SELECT * FROM election WHERE";
  sql += " creator_id = ? AND election_name LIKE ? AND
       status LIKE ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, creatorId);
  ps.setString(2, "%" + keyword + "%");
  if (status.equals("All"))
   status = "";
  ps.setString(3, "%" + status + "%");
  ResultSet rs = ps.executeQuery();

  while (rs.next()) {
   Election election = new Election(rs.getInt(1), rs.getInt(2),
        rs.getString(3), rs.getString(4),
     rs.getString(7), mainApp);

   data.add(election);

  }
 } catch (Exception e) {
  e.printStackTrace();
 }
 return data;
}

public boolean isElectionTaken(String electionName) {
 try {
  String sql = "SELECT * FROM election WHERE
       election_name = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, electionName);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  return false;
 }
}

public void addElection(int creatorId, String electionName,
      String description) {
 try {
  String sql = "INSERT INTO election (creator_id,
       election_name, description, status, end_date, start_date,
        public_key, vote_point)"
    + " VALUES (?, ?, ?, ?, ?, ?, ?, ?);";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, creatorId);
  ps.setString(2, electionName);
  ps.setString(3, description);
  ps.setString(4, "Setup Stage");
  Timestamp date = new java.sql.Timestamp(0);
  ps.setTimestamp(5, date);
  ps.setTimestamp(6, date);

  ps.setString(7, "");
  ps.setString(8, "");
  ps.executeUpdate();// integer
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void updateElection(String electionName, String
      description, int electionId) {
 try {
  String sql = "UPDATE election SET election_name = ?,
       description = ? WHERE election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, electionName);
  ps.setString(2, description);
  ps.setInt(3, electionId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }

}

public ObservableList<Party> searchParty(String keyword, int
      electionId, MainApp mainApp) {
 ObservableList<Party> data = FXCollections.
      observableArrayList();
 try {
  String sql = "SELECT * FROM party WHERE";
  sql += " election_id = ? AND party_name LIKE ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setString(2, "%" + keyword + "%");
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   Party party = new Party(rs.getInt(1), rs.getInt(2), rs.
        getString(3), mainApp);
   data.add(party);
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public boolean isPartyTaken(String partyName, int electionId)
     {
 try {
  String sql = "SELECT * FROM party WHERE";
  sql += " party_name = ? AND election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, partyName);
  ps.setInt(2, electionId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  return false;
 }
}

public void addParty(String partyName, int electionId) {
 try {
  String sql = "INSERT INTO party (election_id, party_name)
       VALUES (?, ?);";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setString(2, partyName);
  ps.executeUpdate();// integer
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void updateParty(int partyId, int electionId, String
      partyName) {
 try {
```

```
  String sql = "UPDATE party SET party_name = ? WHERE
        election_id = ? AND party_id = ?;";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, partyName);
  ps.setInt(2, electionId);
  ps.setInt(3, partyId);

  ps.executeUpdate();// integer
 } catch (Exception e) {
  e.printStackTrace();
 }

}

public ObservableList<Block> searchBlock(String keyword, int
      electionId, MainApp mainApp) {
 ObservableList<Block> data = FXCollections.
       observableArrayList();

 try {
  String sql = "SELECT * FROM block WHERE";
  sql += " election_id = ? AND block_name LIKE ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setString(2, "%" + keyword + "%");
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   Block block = new Block(rs.getInt(1), rs.getInt(2), rs.
         getString(3), rs.getString(4), mainApp);
   data.add(block);
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public boolean isBlockTaken(String blockName, int electionId)
     {
 try {
  String sql = "SELECT * FROM block WHERE";
  sql += " block_name = ? AND election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, blockName);
  ps.setInt(2, electionId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  return false;
 }
}

public void addBlock(String blockName, int electionId, String
      description) {
 try {
  String sql = "INSERT INTO block (election_id, block_name,
        description) VALUES (?, ?, ?);";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setString(2, blockName);
  ps.setString(3, description);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void updateBlock(int blockId, int electionId, String
      blockName, String description) {
 try {
  String sql = "UPDATE block SET block_name = ?,
        description = ? WHERE election_id = ? AND block_id
        = ?;";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, blockName);
  ps.setString(2, description);
  ps.setInt(3, electionId);
  ps.setInt(4, blockId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public ObservableList<Position> searchPosition(String
      keyword, SubElection subElection, MainApp mainApp) {
 ObservableList<Position> data = FXCollections.
       observableArrayList();

 try {
  String sql = "SELECT * FROM position WHERE";
  sql += " sub_election_id = ? AND position_name LIKE ?
        ORDER BY ballot_order";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, subElection.getSubElectionId());
  ps.setString(2, "%" + keyword + "%");
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   Position position = new Position(rs.getInt(1), rs.getInt(2),
         rs.getString(3), rs.getInt(4),
     rs.getInt(5), mainApp);
   position.setSubElection(subElection);
   data.add(position);

  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public boolean isPositionTaken(String positionName, int
      electionId) {
 try {
  String sql = "SELECT * FROM position INNER JOIN
        sub_election ON position.sub_election_id = sub_election.
        sub_election_id"
    + " WHERE position.position_name = ? AND sub_election.
         election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, positionName);
  ps.setInt(2, electionId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  return false;
 }
}

public void addPosition(int subElectionId, String
      positionName, int maxVote) {
 try {
  String sql = "SELECT MAX(ballot_order) FROM position
        WHERE sub_election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, subElectionId);
  ResultSet rs = ps.executeQuery();
  int ballotOrder = 0;
  if (rs.next())
   ballotOrder = rs.getInt(1) + 1;

  sql = "INSERT INTO position (sub_election_id,
        position_name, ballot_order, max_vote) VALUES (?, ?,
        ?, ?);";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, subElectionId);
  ps.setString(2, positionName);
  ps.setInt(3, ballotOrder);
  ps.setInt(4, maxVote);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void updatePosition(int positionId, String
      positionName, int maxVote) {
 try {
  String sql = "UPDATE position SET position_name = ?,
        max_vote = ? WHERE position_id = ?;";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, positionName);
  ps.setInt(2, maxVote);
  ps.setInt(3, positionId);

  ps.executeUpdate();// integer
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public ObservableList<String> searchParticipatingBlock(int
      blockId) {
 ObservableList<String> data = FXCollections.
       observableArrayList();

 try {
  String sql = "SELECT * FROM sub_election INNER JOIN
        block_sub_election ON sub_election.sub_election_id =
        block_sub_election.sub_election_id";
  sql += " WHERE block_sub_election.block_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ResultSet rs = ps.executeQuery();
```

```
  while (rs.next()) {
   data.add(rs.getString(3));

  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public ObservableList<String> searchAllSubElection(int
      electionId) {
 ObservableList<String> data = FXCollections.
       observableArrayList();

 try {
  String sql = "SELECT * FROM sub_election";
  sql += " WHERE election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   data.add(rs.getString(3));

  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public void addParticipation(int blockId, int electionId,
      String subElectionName) {
 try {
  String sql = "SELECT * FROM sub_election";
  sql += " WHERE election_id = ? AND sub_election_name =
       ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setString(2, subElectionName);
  ResultSet rs = ps.executeQuery();
  if (rs.next()) {
   int subElectionId = rs.getInt(1);
   sql = "SELECT * FROM block_sub_election WHERE
        sub_election_id = ? AND block_id = ?";
   ps = conn.prepareStatement(sql);
   ps.setInt(1, subElectionId);
   ps.setInt(2, blockId);
   rs = ps.executeQuery();
   if (!rs.next()) {
    sql = "INSERT INTO block_sub_election (sub_election_id,
         block_id) VALUES (?, ?);";
    ps = conn.prepareStatement(sql);
    ps.setInt(1, subElectionId);
    ps.setInt(2, blockId);
    ps.executeUpdate();// integer
   }
  }
 } catch (Exception e) {
  System.out.println(e);
 }
}

public void deleteParticipation(int blockId, int electionId,
      String subElectionName) {
 try {
  String sql = "SELECT * FROM sub_election";
  sql += " WHERE election_id = ? AND sub_election_name =
       ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setString(2, subElectionName);
  ResultSet rs = ps.executeQuery();
  if (rs.next()) {
   int subElectionId = rs.getInt(1);
   sql = "DELETE FROM block_sub_election WHERE
        sub_election_id = ? AND block_id = ?";
   ps = conn.prepareStatement(sql);
   ps.setInt(1, subElectionId);
   ps.setInt(2, blockId);
   ps.executeUpdate();// integer

  }
 } catch (Exception e) {
  System.out.println(e);
 }
}

public ObservableList<Candidate> searchCandidate(String
      keyword, int positionId, MainApp mainApp,
  SubElection subElection) {
 ObservableList<Candidate> data = FXCollections.
       observableArrayList();

 try {
  String sql = "SELECT * FROM candidate INNER JOIN
       party ON candidate.party_id = party.party_id WHERE
       ";
  sql += " position_id = ? AND (first_name LIKE ? OR
       last_name LIKE ? OR party_name LIKE ?)"
    + " AND first_name != ? ORDER BY candidate_id";

  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, positionId);
  ps.setString(2, "%" + keyword + "%");
  ps.setString(3, "%" + keyword + "%");
  ps.setString(4, "%" + keyword + "%");
  ps.setString(5, "ABSTAIN");
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   Candidate candidate = new Candidate(rs.getInt(1), rs.getInt
        (2), rs.getInt(3), rs.getString(9),
     rs.getString(4), rs.getString(5), rs.getInt(6), mainApp)
        ;
   candidate.setSubElection(subElection);
   data.add(candidate);

  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public boolean isNameTaken(int positionId, String firstName,
      String lastName) {
 try {
  String sql = "SELECT * FROM candidate WHERE";
  sql += " position_id = ? AND first_name = ? AND
       last_name = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, positionId);
  ps.setString(2, firstName);
  ps.setString(3, lastName);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  return false;
 }
}

public void addCandidate(int positionId, int partyId, String
      firstName, String lastName) {
 try {
  String sql = "INSERT INTO candidate (position_id, party_id
       , first_name, last_name, tally) VALUES (?, ?, ?, ?, ?);";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, positionId);
  ps.setInt(2, partyId);
  ps.setString(3, firstName);
  ps.setString(4, lastName);
  ps.setInt(5, 0);
  ps.executeUpdate();// integer
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public ObservableList<SubElection> searchSubElection(String
      keyword, int electionId, MainApp mainApp) {
 ObservableList<SubElection> data = FXCollections.
       observableArrayList();
 try {
  String sql = "SELECT * FROM sub_election WHERE";
  sql += " election_id = ? AND sub_election_name LIKE ?
       ORDER BY sub_election_order";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setString(2, "%" + keyword + "%");
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   SubElection subElection = new SubElection(rs.getInt(1), rs.
        getInt(2), rs.getString(3), rs.getInt(4),
     mainApp);
   data.add(subElection);
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public boolean isSubElectionTaken(String subElectionName, int
      electionId) {
```

```
 try {
  String sql = "SELECT * FROM sub_election WHERE";
  sql += " sub_election_name = ? AND election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, subElectionName);
  ps.setInt(2, electionId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  return false;
 }
}

public void addSubElection(String subElectionName, int
      electionId) {
 try {
  String sql = "SELECT MAX(sub_election_order) FROM
       sub_election WHERE election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ResultSet rs = ps.executeQuery();
  int subElectionOrder = 0;
  if (rs.next())
   subElectionOrder = rs.getInt(1) + 1;

  sql = "INSERT INTO sub_election (election_id,
       sub_election_name, sub_election_order) VALUES (?, ?,
       ?);";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setString(2, subElectionName);
  ps.setInt(3, subElectionOrder);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void updateSubElection(int subElectionId, String
      subElectionName) {
 try {
  String sql = "UPDATE sub_election SET sub_election_name
       = ? WHERE sub_election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, subElectionName);
  ps.setInt(2, subElectionId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public int getElectionId(int positionId) {
 try {
  String sql = "SELECT * FROM sub_election INNER JOIN
       position ON "
    + "sub_election.sub_election_id = position.sub_election_id
         WHERE";
  sql += " position_id = ?";

  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, positionId);

  ResultSet rs = ps.executeQuery();
  rs.next();
  return rs.getInt(2);
 } catch (Exception e) {
  e.printStackTrace();
  return 0;
 }
}

public void updateCandidate(int candidateId, int partyId,
      String firstName, String lastName) {
 try {
  String sql = "UPDATE candidate SET party_id = ?,
       first_name = ?, last_name = ? WHERE candidate_id =
       ?;";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, partyId);
  ps.setString(2, firstName);
  ps.setString(3, lastName);
  ps.setInt(4, candidateId);

  ps.executeUpdate();// integer
 } catch (Exception e) {
  e.printStackTrace();
 }

}

public ObservableList<Voter> searchVoter(String keyword, int
      blockId, MainApp mainApp) {
 ObservableList<Voter> data = FXCollections.
      observableArrayList();
 try {
  String sql = "SELECT * FROM voter INNER JOIN user on
       voter.user_id=user.user_id "
    + "WHERE voter.block_id = ? AND (user.username LIKE
       ? OR user.email LIKE ?)";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ps.setString(2, "%" + keyword + "%");
  ps.setString(3, "%" + keyword + "%");
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   Voter voter = new Voter(rs.getInt(1), rs.getInt(2), rs.
       getInt(3), rs.getString(10), rs.getString(4),
     rs.getString(13), mainApp);
   data.add(voter);

  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public void addVoterAndAccount(String username, String email
      , int blockId) {
 try {
  String sql = "INSERT INTO user (username, email, type,
       password, salt) VALUES (?, ?, ?, ?, ?);";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, username);
  ps.setString(2, email);
  ps.setString(3, "Voter");
  ps.setString(4, "");
  ps.setString(5, "");
  ps.executeUpdate();// integer

  sql = "SELECT * FROM user WHERE username = ?";
  ps = conn.prepareStatement(sql);
  ps.setString(1, username);
  ResultSet rs = ps.executeQuery();
  rs.next();
  int userId = rs.getInt(1);

  sql = "INSERT INTO voter (user_id, block_id, voter_salt,
       voter_password, alias, voted, verification_code)
       VALUES (?, ?, ?, ?, ?, ?, ?)";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, userId);
  ps.setInt(2, blockId);
  ps.setString(3, "");
  ps.setString(4, "");
  ps.setString(5, "");
  ps.setInt(6, 0);
  ps.setString(7, "");
  ps.executeUpdate();

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

public void updatePositionOrder(int positionId, int
      ballotOrder) {
 try {
  String sql = "UPDATE position SET ballot_order = ?
       WHERE position_id = ?;";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, ballotOrder);
  ps.setInt(2, positionId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public boolean accountExistOnDifferentElection(String
      username, String email, int electionId) {
 try {
  String sql = "SELECT * FROM voter INNER JOIN user ON
       voter.user_id = user.user_id "
    + "INNER JOIN block ON block.block_id = voter.block_id
       INNER JOIN election "
    + "ON election.election_id = block.election_id"
    + " WHERE user.username = ? AND user.email = ? AND
       block.election_id != ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps = conn.prepareStatement(sql);
  ps.setString(1, username);
  ps.setString(2, email);
```

```
  ps.setInt(3, electionId);

  ResultSet rs = ps.executeQuery();
  return rs.next();

 } catch (Exception e) {
  e.printStackTrace();
 }
 return false;
}

public boolean accountExistOnThisElection(String username,
       String email, int electionId) {
 try {
  String sql = "SELECT * FROM voter INNER JOIN user ON
         voter.user_id = user.user_id "
    + "INNER JOIN block ON block.block_id = voter.block_id
         INNER JOIN election "
    + "ON election.election_id = block.election_id"
    + " WHERE user.username = ? AND user.email = ? AND
         block.election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);

  ps = conn.prepareStatement(sql);
  ps.setString(1, username);
  ps.setString(2, email);
  ps.setInt(3, electionId);

  ResultSet rs = ps.executeQuery();
  return rs.next();

 } catch (Exception e) {
  e.printStackTrace();
 }
 return false;
}

public void addVoter(String username, int blockId) {
 try {
  String sql = "SELECT * FROM user WHERE username =
       ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, username);
  ResultSet rs = ps.executeQuery();
  rs.next();
  int userId = rs.getInt(1);

  sql = "INSERT INTO voter (user_id, block_id, voter_salt,
       voter_password, alias, voted, verification_code)
       VALUES (?, ?, ?, ?, ?, ?, ?)";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, userId);
  ps.setInt(2, blockId);
  ps.setString(3, "");
  ps.setString(4, "");
  ps.setString(5, "");
  ps.setInt(6, 0);
  ps.setString(7, "");
  ps.executeUpdate();

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

public void deleteCandidate(int candidateId) {
 try {
  String sql = "DELETE FROM candidate WHERE
       candidate_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, candidateId);
  ps.executeUpdate();// integer
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public boolean canDeleteParty(int partyId) {
 try {
  String sql = "SELECT * FROM candidate WHERE party_id
       = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, partyId);
  ResultSet rs = ps.executeQuery();
  return !rs.next();
 } catch (Exception e) {
  System.out.println(e);
 }
 return false;
}

public void deleteParty(int partyId) {
 try {
  String sql = "DELETE FROM party WHERE party_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, partyId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void deleteVoter(int voterId) {
 try {
  String sql = "DELETE FROM voter WHERE voter_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, voterId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void deletePosition(Position position) {
 try {
  String sql = "DELETE FROM position WHERE position_id
       = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, position.getPositionId());
  ps.executeUpdate();

  sql = "DELETE FROM candidate WHERE position_id = ?";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, position.getPositionId());
  ps.executeUpdate();

  sql = "UPDATE position SET ballot_order = ballot_order -
       1 WHERE sub_election_id = ? AND ballot_order > ?";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, position.getSubElectionId());
  ps.setInt(2, position.getBallotOrder());
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public boolean hasVoters(int blockId) {
 try {
  String sql = "SELECT * FROM voter WHERE block_id = ?
       ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  System.out.println(e);
 }
 return true;
}
/*
 * public boolean hasSpecificPosition(int blockId) { try {
       String sql =
 * "SELECT * FROM position WHERE res_block_id = ? ";
       PreparedStatement ps =
 * conn.prepareStatement(sql); ps.setInt(1, blockId);
       ResultSet rs =
 * ps.executeQuery(); return rs.next(); } catch (Exception e)
       {
 * System.out.println(e); } return true; }
 */

public void deleteBlock(int blockId) {
 try {
  String sql = "DELETE FROM block WHERE block_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ps.executeUpdate();

  sql = "DELETE FROM block_sub_election WHERE block_id
       = ?";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public boolean hasPosition(int subElectionId) {
 try {
  String sql = "SELECT * FROM position WHERE
       sub_election_id = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
```

```
  ps.setInt(1, subElectionId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  System.out.println(e);
 }
 return true;
}

public boolean subElectionHasBlockParticipation(int
      subElectionId) {
 try {
  String sql = "SELECT * FROM block_sub_election WHERE
        sub_election_id = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, subElectionId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  System.out.println(e);
 }
 return true;
}

public void deleteSubElection(int electionId, int
      subElectionId, int subElectionOrder) {
 try {
  String sql = "DELETE FROM sub_election WHERE
        sub_election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, subElectionId);
  ps.executeUpdate();

  sql = "UPDATE sub_election SET sub_election_order =
        sub_election_order - 1 WHERE election_id = ? AND
        sub_election_order > ?";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setInt(2, subElectionOrder);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void deleteElection(int electionId) {
 List<Integer> subElectionIdList = new ArrayList<Integer>();
 subElectionIdList = getSubElectionId(electionId);
 List<Integer> positionIdList = new ArrayList<Integer>();
 for (int i = 0; i < subElectionIdList.size(); i++) {
  positionIdList.addAll(getPositionId(subElectionIdList.get(i))
        );
 }
 List<Integer> blockIdList = new ArrayList<Integer>();
 blockIdList = getBlockId(electionId);

 try {
  String sql = "DELETE FROM block WHERE election_id =
        ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.executeUpdate();

  for (int i = 0; i < subElectionIdList.size(); i++) {
   sql = "DELETE FROM block_sub_election WHERE
         sub_election_id = ?";
   ps = conn.prepareStatement(sql);
   ps.setInt(1, subElectionIdList.get(i));
   ps.executeUpdate();
  }

  for (int i = 0; i < positionIdList.size(); i++) {
   sql = "DELETE FROM position WHERE position_id = ?";
   ps = conn.prepareStatement(sql);
   ps.setInt(1, positionIdList.get(i));
   ps.executeUpdate();

   sql = "DELETE FROM candidate WHERE position_id =
         ?";
   ps = conn.prepareStatement(sql);
   ps.setInt(1, positionIdList.get(i));
   ps.executeUpdate();
  }

  for (int i = 0; i < blockIdList.size(); i++) {
   sql = "DELETE FROM voter WHERE block_id = ?";
   ps = conn.prepareStatement(sql);
   ps.setInt(1, blockIdList.get(i));
   ps.executeUpdate();
  }

  sql = "DELETE FROM election WHERE election_id = ?";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.executeUpdate();

  sql = "DELETE FROM party WHERE election_id = ?";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.executeUpdate();

  sql = "DELETE FROM sub_election WHERE election_id =
        ?";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.executeUpdate();

  sql = "DELETE FROM election WHERE election_id = ?";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.executeUpdate();

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

}

public void deleteUnusedUser() {
 try {
  String sql = "SELECT * FROM user WHERE type = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, "Voter");
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   sql = "SELECT * FROM voter WHERE user_id = ?";
   ps = conn.prepareStatement(sql);
   ps.setInt(1, rs.getInt(1));
   ResultSet rs1 = ps.executeQuery();
   if (!rs1.next()) {
    sql = "DELETE FROM user WHERE user_id = ?";
    ps = conn.prepareStatement(sql);
    ps.setInt(1, rs.getInt(1));
    ps.executeUpdate();
   }
  }
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public List<Integer> getSubElectionId(int electionId) {
 List<Integer> subElectionIdList = new ArrayList<Integer>();
 try {
  String sql = "SELECT * FROM sub_election WHERE
        election_id = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   subElectionIdList.add(rs.getInt(1));
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return subElectionIdList;
}

public List<Integer> getPositionId(int subElectionId) {
 List<Integer> positionIdList = new ArrayList<Integer>();
 try {
  String sql = "SELECT * FROM position WHERE
        sub_election_id = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, subElectionId);
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   positionIdList.add(rs.getInt(1));
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return positionIdList;
}

public List<Integer> getBlockId(int electionId) {
 List<Integer> blockIdList = new ArrayList<Integer>();
 try {
  String sql = "SELECT * FROM block WHERE election_id =
        ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ResultSet rs = ps.executeQuery();
```

```
  while (rs.next()) {
   blockIdList.add(rs.getInt(1));
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return blockIdList;
}

public boolean hasElection(int userId) {
 try {
  String sql = "SELECT * FROM election WHERE creator_id
       = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, userId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  System.out.println(e);
 }
 return true;
}

public void deleteUser(int userId) {
 try {
  String sql = "DELETE FROM user WHERE user_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, userId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public boolean hasSubElection(int electionId) {
 try {
  String sql = "SELECT * FROM sub_election WHERE
       election_id = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  System.out.println(e);
 }
 return true;
}

public boolean hasBlock(int electionId) {
 try {
  String sql = "SELECT * FROM block WHERE election_id =
       ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  System.out.println(e);
 }
 return true;
}

public boolean blockHasBlockParticipation(int blockId) {
 try {
  String sql = "SELECT * FROM block_sub_election WHERE
       block_id = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  System.out.println(e);
 }
 return true;
}

public boolean hasValidCandidates(int positionId) {
 try {
  String sql = "SELECT * FROM position WHERE
       position_id = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, positionId);
  ResultSet rs = ps.executeQuery();
  rs.next();
  int maxCandidate = rs.getInt(5);

  sql = "SELECT * FROM candidate WHERE position_id = ?
       ";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, positionId);
  rs = ps.executeQuery();
  rs.last();
  int size = rs.getRow();
  if (size > 1 && maxCandidate <= size)
   return true;
  else
   return false;
 } catch (Exception e) {
  System.out.println(e);
 }
 return false;
}

public void updateElectionStatus(int electionId, String status)
     {
 try {
  String sql = "UPDATE election SET status = ? WHERE
       election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, status);
  ps.setInt(2, electionId);
  ps.executeUpdate();
  ps.close();

  if (status.equals("Voting Stage")) {
   sql = "UPDATE election SET start_date = ? WHERE
       election_id = ?";
   ps = conn.prepareStatement(sql);
   ps.setTimestamp(1, new java.sql.Timestamp(new java.util.
       Date().getTime()));
   ps.setInt(2, electionId);
   ps.executeUpdate();

  } else if (status.equals("Tallying Stage")) {
   sql = "UPDATE election SET end_date = ? WHERE
       election_id = ?";
   ps.close();
   ps = conn.prepareStatement(sql);

   ps.setTimestamp(1, new java.sql.Timestamp(new java.util.
       Date().getTime()));
   ps.setInt(2, electionId);
   ps.executeUpdate();

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

public List<Integer> getVoterId(int blockId) {
 List<Integer> voterIdList = new ArrayList<Integer>();
 try {
  String sql = "SELECT * FROM voter WHERE block_id = ?
       ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   voterIdList.add(rs.getInt(1));
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return voterIdList;
}

public void addAlias(int voterId, String alias) {
 try {
  String sql = "UPDATE voter SET alias = ? WHERE
       voter_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, alias);
  ps.setInt(2, voterId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void addVoterPassword(int voterId, byte[] salt, String
     password) {
 try {
  String sql = "UPDATE voter SET voter_password = ?,
       voter_salt = ? WHERE voter_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, password);
  ps.setBytes(2, salt);
  ps.setInt(3, voterId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}
```

```
public Voter getVoter(int voterId, MainApp mainApp) {
 try {
  String sql = "SELECT * FROM voter INNER JOIN user ON
       voter.user_id = user.user_id WHERE voter.voter_id = ?
       ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, voterId);
  ResultSet rs = ps.executeQuery();
  rs.next();
  return new Voter(rs.getInt(1), rs.getInt(2), rs.getInt(3), rs
      .getString(10), rs.getString(4),
    rs.getString(13), mainApp);
 } catch (Exception e) {
  System.out.println(e);
 }
 return null;
}

public void addPublicKey(int electionId, byte[] publicKey) {
 try {
  String sql = "UPDATE election SET public_key = ?
      WHERE election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setBytes(1, publicKey);
  ps.setInt(2, electionId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public boolean checkBlockParticipation(int blockId, int
     subElectionId) {
 try {
  String sql = "SELECT * FROM block_sub_election WHERE
      block_id = ? AND sub_election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ps.setInt(2, subElectionId);
  ResultSet rs = ps.executeQuery();
  return rs.next();
 } catch (Exception e) {
  System.out.println(e);
 }
 return false;
}

public void addVotePoint(int electionId, byte[] votePoint) {
 try {
  String sql = "UPDATE election SET vote_point = ?
      WHERE election_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setBytes(1, votePoint);
  ps.setInt(2, electionId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void updateSubElectionOrder(int subElectionId, int
     subElectionOrder) {
 try {
  String sql = "UPDATE sub_election SET sub_election_order
      = ? WHERE sub_election_id = ?;";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, subElectionOrder);
  ps.setInt(2, subElectionId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public List<Position> getAllPosition(int subElectionId,
     MainApp mainApp) {
 List<Position> positionList = new ArrayList<Position>();
 try {
  String sql = "SELECT * FROM position WHERE
      sub_election_id = ? ORDER BY ballot_order";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, subElectionId);
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   Position position = new Position(rs.getInt(1), rs.getInt(2),
       rs.getString(3), rs.getInt(4),
     rs.getInt(5), mainApp);
   positionList.add(position);
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return positionList;
}

public List<Candidate> getCandidate(int positionId, MainApp
     mainApp) {
 List<Candidate> candidateList = new ArrayList<Candidate
     >();
 try {
  String sql = "SELECT * FROM candidate INNER JOIN
      party ON candidate.party_id = party.party_id WHERE
      candidate.position_id = ? ORDER BY candidate.
      candidate_id";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, positionId);

  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   Candidate candidate = new Candidate(rs.getInt(1), rs.getInt
       (2), rs.getInt(3), rs.getString(9),
     rs.getString(4), rs.getString(5), rs.getInt(6), mainApp)
         ;
   candidateList.add(candidate);
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return candidateList;
}

public void addBallot(int blockId, String subElectionName, int
     positionId, int ballotOrder, String positionName,
  int maxVote, int candidateId, String partyName, String
      candidateName) {
 try {
  String sql = "INSERT INTO ballot (block_id,
      sub_election_name,position_id, ballot_order,
      position_name, max_vote, candidate_id, party_name,
      candidate_name) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ps.setString(2, subElectionName);
  ps.setInt(3, positionId);
  ps.setInt(4, ballotOrder);
  ps.setString(5, positionName);
  ps.setInt(6, maxVote);
  ps.setInt(7, candidateId);
  ps.setString(8, partyName);
  ps.setString(9, candidateName);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public List<SubElection> getSubElectionOfBlock(int blockId,
     MainApp mainApp) {
 List<SubElection> subElectionList = new ArrayList<
     SubElection>();
 try {
  String sql = "SELECT * FROM block_sub_election INNER
      JOIN sub_election ON sub_election.sub_election_id =
      block_sub_election.sub_election_id WHERE block_id = ?
       ORDER BY sub_election.sub_election_order";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   SubElection subElection = new SubElection(rs.getInt(2), rs.
       getInt(4), rs.getString(5), rs.getInt(6),
     mainApp);
   subElectionList.add(subElection);
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return subElectionList;
}

public void addLookupTable(int electionId, int multiple, byte[]
     product) {
 try {
  String sql = "INSERT INTO lookup_table (election_id,
      multiple, product) VALUES (?, ?, ?)";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setInt(2, multiple);
  ps.setBytes(3, product);

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

```
 }
}

public List<Integer> getCandidateId(int positionId) {
 List<Integer> candidateId = new ArrayList<Integer>();
 try {
  String sql = "SELECT * FROM candidate WHERE
       position_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, positionId);

  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   candidateId.add(rs.getInt(1));
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return candidateId;
}

public ResultSet getCandidateVotes(int candidateId) {
 try {
  String sql = "SELECT * FROM vote WHERE candidate_id
       = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, candidateId);

  ResultSet rs = ps.executeQuery();
  return rs;
 } catch (Exception e) {
  System.out.println(e);
 }
 return null;
}

public int getMultiple(int electionId, byte[] product) {
 try {
  String sql = "SELECT * FROM lookup_table WHERE
       election_id = ? AND product = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setBytes(2, product);
  ResultSet rs = ps.executeQuery();
  if (rs.next())
   return rs.getInt(3);
  else
   return -1;
 } catch (Exception e) {
  System.out.println(e);
 }
 return -1;
}

public void setTally(int candidateId, int tally) {
 try {
  String sql = "UPDATE candidate SET tally = ? WHERE
       candidate_id = ?;";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, tally);
  ps.setInt(2, candidateId);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public ObservableList<BulletinBoard> searchBulletinBoard(
      String keyword, int electionId, MainApp mainApp) {
 ObservableList<BulletinBoard> data = FXCollections.
       observableArrayList();
 try {
  String sql = "SELECT * FROM voter INNER JOIN block
       ON voter.block_id = block.block_id"
    + " WHERE block.election_id = ? AND voted = ? AND (
         voter.alias LIKE ? OR voter.verification_code LIKE
         ?)";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setInt(2, 1);
  ps.setString(3, "%" + keyword + "%");
  ps.setString(4, "%" + keyword + "%");
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   BulletinBoard bulletinBoard = new BulletinBoard(rs.getInt
        (1), rs.getInt(3), rs.getString(4),
     rs.getString(8), mainApp);
   data.add(bulletinBoard);
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return data;
}

public ResultSet getBallot(int blockId) {
 try {
  String sql = "SELECT * FROM ballot WHERE block_id = ?
       ORDER BY ballot_order";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ResultSet rs = ps.executeQuery();
  return rs;
 } catch (Exception e) {
  System.out.println(e);
 }
 return null;
}

public ResultSet getVote(int voterId, int candidateId) {
 try {
  String sql = "SELECT * FROM vote WHERE voter_id = ?
       AND candidate_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, voterId);
  ps.setInt(2, candidateId);
  ResultSet rs = ps.executeQuery();
  rs.next();
  return rs;

 } catch (Exception e) {
  System.out.println(e);
 }
 return null;
}

public List<Voter> getVoted(int blockId, MainApp mainApp)
     {
 List<Voter> voterList = new ArrayList<Voter>();
 try {
  String sql = "SELECT * FROM voter WHERE block_id = ?
       AND voted = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ps.setInt(2, 1);
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   Voter voter = new Voter(rs.getInt(1), rs.getInt(2), rs.
        getInt(3), "", rs.getString(4), "", mainApp);
   voterList.add(voter);
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return voterList;
}

public void addHashCode(int electionId, String alias, String
     subElectionName, String positionName,
  String hashCode) {
 try {
  String sql = "INSERT INTO hash_code (election_id, alias,
       sub_election_name, position_name, hash_code) VALUES
       (?, ?, ?, ?, ?)";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ps.setString(2, alias);
  ps.setString(3, subElectionName);
  ps.setString(4, positionName);
  ps.setString(5, hashCode);
  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public void setVerificationCode(int voterId, String
     verificationCode) {
 try {
  String sql = "UPDATE voter SET verification_code = ?
       WHERE voter_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, verificationCode);
  ps.setInt(2, voterId);

  ps.executeUpdate();
 } catch (Exception e) {
  e.printStackTrace();
 }
}

public ResultSet getHashCode(int electionId, String alias) {
 try {
  String sql = "SELECT * FROM hash_code WHERE
```

```
            election_id = ? AND alias = ?";
    PreparedStatement ps = conn.prepareStatement(sql);
    ps.setInt(1, electionId);
    ps.setString(2, alias);
    ResultSet rs = ps.executeQuery();
    return rs;
   } catch (Exception e) {
    System.out.println(e);
   }
   return null;
  }
}
```

EmailHelper.java

```
package ph.edu.upm.cas.dpsm.bjyu.util;

import java.util.Properties;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class EmailHelper {
 Message message;

 public EmailHelper() {
  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");
  props.put("mail.smtp.ssl.trust", "smtp.gmail.com");

  Session session = Session.getInstance(props, new javax.mail.
        Authenticator() {
   protected PasswordAuthentication getPasswordAuthentication
         () {
    return new PasswordAuthentication("yuvote96@gmail.com",
          "yuvoteforsp17");
   }
  });
  message = new MimeMessage(session);

 }

 public boolean sendMail(String email, String subject, String
       body) {
  try {
   message.setFrom(new InternetAddress("from-email@gmail.
         com"));
   message.setRecipients(Message.RecipientType.TO,
         InternetAddress.parse(email));
   message.setSubject(subject);
   message.setText(body);
   Transport.send(message);
   return true;

  } catch (MessagingException e) {
   return false;
  }
 }
}
```

PaginationHelper.java

```
package ph.edu.upm.cas.dpsm.bjyu.util;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Node;
import javafx.scene.control.Pagination;
import javafx.scene.control.TableRow;
import javafx.scene.control.TableView;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.VBox;
import javafx.util.Callback;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;

public class PaginationHelper<S> {

 private int rowsPerPage;
 private int itemsPerPage;
 private TableView<S> table;
 private ObservableList<S> tableData;

 public PaginationHelper() {
 }

 public PaginationHelper(TableView<S> table, ObservableList<
       S> tableData, int rowsPerPage, int itemsPerPage) {
  this.table = table;
  this.tableData = tableData;
  this.rowsPerPage = rowsPerPage;
  this.itemsPerPage = itemsPerPage;
  table.setFocusTraversable(false);

  table.getStylesheets().add(MainApp.class.getResource("css/
        fextile.css").toExternalForm());
  table.setId("table-view");

  table.setRowFactory(new Callback<TableView<S>, TableRow
        <S>>() {
   @Override
   public TableRow<S> call(TableView<S> tableView) {
    TableRow<S> row = new TableRow<S>() {
     @Override
     protected void updateItem(S s, boolean empty) {
      if (getIndex() % 2 == 1)
       this.setId("odd-row");
      else
       this.setId("even-row");
      this.setStyle("-fx-cell-size: 40px");

     }
    };

    return row;
   }
  });
 }

 public Pagination getPagination() {
  int numOfPages = 1;
  if (tableData.size() % rowsPerPage == 0) {
   numOfPages = tableData.size() / rowsPerPage;
  } else if (tableData.size() > rowsPerPage) {
   numOfPages = tableData.size() / rowsPerPage + 1;
  }
  if (numOfPages == 0)
   numOfPages++;

  Pagination pagination = new Pagination((numOfPages), 0);
  pagination.setPageFactory(new Callback<Integer, Node>() {
   @Override
   public Node call(Integer pageIndex) {
    if (pageIndex > tableData.size() / rowsPerPage + 1) {
     return null;
    } else {
     return createPage(pageIndex);
    }
   }
  });

  AnchorPane.setTopAnchor(pagination, 0.0);
  AnchorPane.setRightAnchor(pagination, 0.0);
  AnchorPane.setBottomAnchor(pagination, 0.0);
  AnchorPane.setLeftAnchor(pagination, 0.0);
  return pagination;
 }

 public VBox createPage(int pageIndex) {
  int lastIndex = 0;
  int displace = tableData.size() % rowsPerPage;
  if (displace > 0) {
   lastIndex = tableData.size() / rowsPerPage;
  } else {
   lastIndex = tableData.size() / rowsPerPage - 1;
   displace = rowsPerPage;
  }

  VBox box = new VBox(5);
  int page = pageIndex * itemsPerPage;

  for (int i = page; i < page + itemsPerPage; i++) {
   if (tableData.size() == 0) {
    table.setItems(FXCollections.observableArrayList());
   } else if (lastIndex == pageIndex) {
    table.setItems(FXCollections.observableArrayList(
      tableData.subList(pageIndex * rowsPerPage, pageIndex *
            rowsPerPage + displace)));
   } else {
    table.setItems(FXCollections.observableArrayList(
      tableData.subList(pageIndex * rowsPerPage, pageIndex *
            rowsPerPage + rowsPerPage)));
   }
   box.getChildren().add(table);
  }
  return box;
 }
}
```

PasswordHelper.java

```
package ph.edu.upm.cas.dpsm.bjyu.util;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.SecureRandom;

//This class is credited to http://howtodoinjava.com/security/
      how-to-generate-secure-password-hash-md5-sha-
      pbkdf2-bcrypt-examples/
public class PasswordHelper {
 private SecureRandom random = new SecureRandom();

 public String generatePassword() {
  return new BigInteger(130, random).toString(32).substring(0,
       8);

 }

 public String getSecurePassword(String passwordToHash, byte[]
      salt) {
  String generatedPassword = null;
  try {
   MessageDigest md = MessageDigest.getInstance("SHA");
   md.update(salt);
   byte[] bytes = md.digest(passwordToHash.getBytes());
   StringBuilder sb = new StringBuilder();
   for (int i = 0; i < bytes.length; i++) {
    sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).
         substring(1));
   }
   generatedPassword = sb.toString();
  } catch (NoSuchAlgorithmException e) {
   e.printStackTrace();
  }
  return generatedPassword;
 }

 public byte[] getSalt() throws NoSuchAlgorithmException,
      NoSuchProviderException {
  SecureRandom sr = SecureRandom.getInstance("SHA1PRNG",
       "SUN");
  byte[] salt = new byte[16];
  sr.nextBytes(salt);
  return salt;
 }
}
```

TooltipHelper.java

```
package ph.edu.upm.cas.dpsm.bjyu.util;

import java.lang.reflect.Field;

import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.scene.control.Tooltip;
import javafx.util.Duration;

public class TooltipHelper {
 public static void hackTooltipStartTiming(Tooltip tooltip) {
  try {
   Field fieldBehavior = tooltip.getClass().getDeclaredField("
        BEHAVIOR");
   fieldBehavior.setAccessible(true);
   Object objBehavior = fieldBehavior.get(tooltip);

   Field fieldTimer = objBehavior.getClass().getDeclaredField("
        activationTimer");
   fieldTimer.setAccessible(true);
   Timeline objTimer = (Timeline) fieldTimer.get(objBehavior);

   objTimer.getKeyFrames().clear();
   objTimer.getKeyFrames().add(new KeyFrame(new Duration
        (250)));
  } catch (Exception e) {
   e.printStackTrace();
  }
 }
}
```

B.2 Voter Application

HomeController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.io.IOException;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Insets;
import javafx.scene.control.Label;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.BorderPane;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Voter;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class HomeController {
 private MainApp mainApp;
 private Voter voter;
 private Election election;

 @FXML
 Label usernameLabel;
 @FXML
 private AnchorPane middlePane;
 @FXML
 private AnchorPane leftPane;
 @FXML
 private BorderPane borderPane;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
  usernameLabel.requestFocus();
 }

 public void setVoter(Voter voter) {
  this.voter = voter;
  DatabaseHelper databaseHelper = new DatabaseHelper();
  this.election = databaseHelper.getElection(voter.getBlockId()
       );
  usernameLabel.setText("Welcome, " + voter.getUsername());
 }

 public void showVoterMainView() {
  try {
   FXMLLoader loader = new FXMLLoader();
   loader.setLocation(MainApp.class.getResource("view/
        NavBarVoter.fxml"));
   AnchorPane navBarVoterFXML = (AnchorPane) loader.load()
        ;

   borderPane.setLeft(leftPane);
   borderPane.setBottom(null);
   borderPane.setCenter(null);
   leftPane.setPadding(new Insets(0, 0, 0, 0));
   leftPane.setPrefHeight(middlePane.getHeight());
   leftPane.getChildren().setAll(navBarVoterFXML);
   AnchorPane.setTopAnchor(leftPane.getChildren().get(0), 0.0);
   AnchorPane.setRightAnchor(leftPane.getChildren().get(0),
        0.0);
   AnchorPane.setLeftAnchor(leftPane.getChildren().get(0), 0.0);
   AnchorPane.setBottomAnchor(leftPane.getChildren().get(0),
        0.0);

   NavBarVoterController navBarVoterController = loader.
        getController();
   navBarVoterController.setHomeController(this);
   navBarVoterController.setButtonAlignment();
   showVoterHome();
  } catch (IOException ioe) {
   ioe.printStackTrace();
  }
 }

 public void showVoterHome() {
  try {
   FXMLLoader loader = new FXMLLoader();
   loader = new FXMLLoader();
   loader.setLocation(MainApp.class.getResource("view/
        HomeVoter.fxml"));
   AnchorPane homeVoterFXML = (AnchorPane) loader.load();

   borderPane.setCenter(null);
   borderPane.setCenter(middlePane);
   middlePane.getChildren().setAll(homeVoterFXML);
   AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
        0.0);
   AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
         0.0);
   AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
        0.0);
   AnchorPane.setBottomAnchor(middlePane.getChildren().get
        (0), 0.0);

   HomeVoterController homeVoterController = loader.
        getController();
```

```
    homeVoterController.setVoter(voter);
   } catch (IOException ioe) {
   }
  }

  public void showVote() {
   try {
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(MainApp.class.getResource("view/Vote.
            fxml"));
    AnchorPane voteFXML = (AnchorPane) loader.load();

    borderPane.setCenter(null);
    borderPane.setCenter(middlePane);
    middlePane.getChildren().setAll(voteFXML);
    AnchorPane.setTopAnchor(middlePane.getChildren().get(0),
            0.0);
    AnchorPane.setRightAnchor(middlePane.getChildren().get(0),
             0.0);
    AnchorPane.setLeftAnchor(middlePane.getChildren().get(0),
            0.0);
    AnchorPane.setBottomAnchor(middlePane.getChildren().get
            (0), 0.0);

    VoteController voteController = loader.getController();
    voteController.setMainApp(mainApp);
    voteController.setVoter(voter);
    voteController.setElection(election);
    voteController.setup();
   } catch (IOException ioe) {
    ioe.printStackTrace();
   }
  }

  @FXML
  public void logout() {
   mainApp.showLogout();
  }
}
```

HomeVoterController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Label;
import ph.edu.upm.cas.dpsm.bjyu.model.Voter;

public class HomeVoterController {
 @FXML
 private Label aliasLabel;

 public void setVoter(Voter voter) {
  aliasLabel.setText("Your alias is " + voter.getAlias());
 }

}
```

LoginController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Voter;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;

public class LoginController {
 private MainApp mainApp;
 DatabaseHelper databaseHelper;

 @FXML
 private Label errorLabel;
 @FXML
 private TextField usernameField;
 @FXML
 private PasswordField passwordField;

 public void setup() {
  databaseHelper = new DatabaseHelper();
  errorLabel.requestFocus();
 }

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 @FXML
 public void login() {
  errorLabel.setText("");
  if (usernameField.getText().isEmpty())
   errorLabel.setText("The username is empty.");
  else if (passwordField.getText().isEmpty())
   errorLabel.setText("The password is empty.");
  else {
   Voter voter = databaseHelper.usernameMatchesPassword(
           usernameField.getText(), passwordField.getText());
   if (voter != null) {
    mainApp.setCurrentVoter(voter);
    mainApp.showHome();
   } else
    errorLabel.setText("Username or password is incorrect.");
  }
 }
}
```

LogoutController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.stage.Stage;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;

public class LogoutController {
 private MainApp mainApp;

 @FXML
 Button noButton;
 @FXML
 Button yesButton;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

 @FXML
 public void no() {
  Stage stage = (Stage) noButton.getScene().getWindow();
  stage.close();
 }

 @FXML
 public void yes() {
  no();
  mainApp.showLogin();
 }

}
```

NavBarVoterController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.geometry.Pos;
import javafx.scene.control.Button;
import javafx.scene.input.MouseEvent;

public class NavBarVoterController {
 private HomeController homeController;

 @FXML
 Button homeButton;
 @FXML
 Button voteButton;

 public void setHomeController(HomeController homeController)
         {
  this.homeController = homeController;
 }

 public void setButtonAlignment() {
  homeButton.setAlignment(Pos.CENTER_LEFT);
  voteButton.setAlignment(Pos.CENTER_LEFT);

  addHoverColor(homeButton);
  addHoverColor(voteButton);

  homeButton.setStyle("-fx-background-color: #76d276");
  homeButton.setUserData(true);
  voteButton.setUserData(false);
 }

 @FXML
 public void home() {
  homeController.showVoterHome();
  homeButton.setStyle("-fx-background-color: #76d276");
  voteButton.setStyle("-fx-background-color: #d3d3d3");
  homeButton.setUserData(true);
  voteButton.setUserData(false);
```

```
 }

 @FXML
 public void vote() {
  homeController.showVote();
  voteButton.setStyle("-fx-background-color: #76d276");
  homeButton.setStyle("-fx-background-color: #d3d3d3");
  voteButton.setUserData(true);
  homeButton.setUserData(false);
 }

 public void addHoverColor(Button button) {
  button.addEventHandler(MouseEvent.MOUSE_ENTERED,
          new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent e) {
     button.setStyle("-fx-background-color: #76d276");
    }
   });

   button.addEventHandler(MouseEvent.MOUSE_EXITED, new
          EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent e) {
     if (!(Boolean) button.getUserData())
      button.setStyle("-fx-background-color: #d3d3d3");
    }
   });
 }
}
```

VoteController.java

```
package ph.edu.upm.cas.dpsm.bjyu.controller;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.ResultSet;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.DatatypeConverter;

import org.bouncycastle.asn1.x9.X9ECParameters;
import org.bouncycastle.crypto.ec.CustomNamedCurves;
import org.bouncycastle.crypto.ec.ECElGamalEncryptor;
import org.bouncycastle.crypto.ec.ECPair;
import org.bouncycastle.crypto.params.ECDomainParameters;
import org.bouncycastle.crypto.params.ECPublicKeyParameters;
import org.bouncycastle.math.ec.ECPoint;

import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.CheckBox;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.ToggleGroup;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import ph.edu.upm.cas.dpsm.bjyu.MainApp;
import ph.edu.upm.cas.dpsm.bjyu.model.Candidate;
import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Position;
import ph.edu.upm.cas.dpsm.bjyu.model.Voter;
import ph.edu.upm.cas.dpsm.bjyu.util.DatabaseHelper;
import ph.edu.upm.cas.dpsm.bjyu.util.EmailHelper;

public class VoteController {
 private MainApp mainApp;
 private Voter voter;
 private Election election;
 private List<Position> positionList;
 private List<List<Candidate>> candidateList;
 private List<List<CheckBox>> allCheckBox;
 private List<List<RadioButton>> allRadioButton;
 private List<Label> errorLabelListRadio;
 private List<Label> errorLabelListCheckBox;
 private List<Label> hashLabelListRadio;
 private List<Label> hashLabelListCheckBox;
 private List<Integer> positionIndexRadio;
 private List<Integer> positionIndexCheckBox;
 private List<List<ECPair>> allNullVoteCheckBox;
 private List<List<ECPair>> allVoteCheckBox;
 private List<List<ECPair>> allNullVoteRadio;
 private List<List<ECPair>> allVoteRadio;
 private List<Integer> maxVoteCheckbox;
 private List<String> subElectionName;

 private DatabaseHelper databaseHelper;
 @FXML
 Label electionLabel;
 @FXML
 ScrollPane scrollPane;
 @FXML
 Button submitButton;

 public void setMainApp(MainApp mainApp) {
  this.mainApp = mainApp;
 }

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

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

 public void setup() {
  databaseHelper = new DatabaseHelper();
  electionLabel.setText(election.getName());
  if (election.getStatus().equals("Voting Stage")) {

   ResultSet ballot = databaseHelper.getBallot(voter.getBlockId
          ());
   List<Candidate> positionCandidateList = new ArrayList<
          Candidate>();
   subElectionName = new ArrayList<String>();
   positionList = new ArrayList<Position>();
   candidateList = new ArrayList<List<Candidate>>();
   hashLabelListRadio = new ArrayList<Label>();
   hashLabelListCheckBox = new ArrayList<Label>();
   try {
    Position currentPosition = new Position(0, "", 0);
    while (ballot.next()) {
     if (currentPosition.getPositionId() != ballot.getInt(4)) {
      if (positionCandidateList.size() > 0) {
       candidateList.add(positionCandidateList);
       positionCandidateList = new ArrayList<Candidate>();
      }
      currentPosition = new Position(ballot.getInt(4), ballot.
             getString(5), ballot.getInt(6));
      subElectionName.add(ballot.getString(3));
      positionList.add(currentPosition);
     }
     positionCandidateList.add(new Candidate(ballot.getInt(7),
            ballot.getInt(4), ballot.getString(8),
       ballot.getString(9)));
    }
    candidateList.add(positionCandidateList);
   } catch (Exception e) {
    e.printStackTrace();
   }

   X9ECParameters parameterSpec = CustomNamedCurves.
          getByName("curve25519");
   ECDomainParameters domainParams = new
          ECDomainParameters(parameterSpec.getCurve(),
          parameterSpec.getG(),
     parameterSpec.getN(), parameterSpec.getH(),
            parameterSpec.getSeed());

   ECPoint publicKeyPoint = parameterSpec.getCurve().
          decodePoint(election.getPublicKey());
   ECPublicKeyParameters publicKey = new
          ECPublicKeyParameters(publicKeyPoint,
          domainParams);

   ECPoint votePoint = parameterSpec.getCurve().decodePoint(
          election.getVotePoint());

   ECPoint nullPoint = parameterSpec.getCurve().getInfinity();
   ECElGamalEncryptor ec = new ECElGamalEncryptor();
   ec.init(publicKey);

   VBox vbox = new VBox();
   allCheckBox = new ArrayList<List<CheckBox>>();
   allRadioButton = new ArrayList<List<RadioButton>>();
   errorLabelListRadio = new ArrayList<Label>();
   errorLabelListCheckBox = new ArrayList<Label>();
   positionIndexCheckBox = new ArrayList<Integer>();
   positionIndexRadio = new ArrayList<Integer>();
   allNullVoteCheckBox = new ArrayList<List<ECPair>>();
```

```
allVoteCheckBox = new ArrayList<List<ECPair>>();
allNullVoteRadio = new ArrayList<List<ECPair>>();
allVoteRadio = new ArrayList<List<ECPair>>();
maxVoteCheckbox = new ArrayList<Integer>();
String currentSubElection = "";
for (int i = 0; i < positionList.size(); i++) {
 if (!currentSubElection.equals(subElectionName.get(i))) {
  currentSubElection = subElectionName.get(i);
  Label subElectionLabel = new Label(subElectionName.get(i
      ));
  subElectionLabel.setFont(new Font("Arial", 18));
  vbox.getChildren().addAll(subElectionLabel);

 }
 String restriction = getRestriction(positionList.get(i).
     getMaxVote());
 Label positionLabel = new Label(positionList.get(i).
     getName() + restriction + ":");
 Label errorLabel = new Label();
 errorLabel.setTextFill(Color.RED);

 vbox.getChildren().addAll(errorLabel, positionLabel);

 if (positionList.get(i).getMaxVote() > 1) {
  errorLabelListCheckBox.add(errorLabel);
  List<CheckBox> checkBoxList = new ArrayList<
      CheckBox>();
  List<ECPair> nullVoteList = new ArrayList<ECPair>();
  List<ECPair> voteList = new ArrayList<ECPair>();
  maxVoteCheckbox.add(positionList.get(i).getMaxVote());
  positionIndexCheckBox.add(i);
  int checkboxIndex = allCheckBox.size();
  for (int j = 0; j < candidateList.get(i).size() - 1; j++)
      {
   CheckBox checkBox = new CheckBox(candidateList.get(i).
       get(j).getName()
     + getParty(candidateList.get(i).get(j).getPartyName())
         );
   checkBox.selectedProperty().addListener(new
       ChangeListener<Boolean>() {
    @Override
    public void changed(ObservableValue<? extends Boolean
        > observable, Boolean oldValue,
      Boolean newValue) {
     updateCheckbox(checkBoxList, checkboxIndex);
    }
   });

   checkBoxList.add(checkBox);
   nullVoteList.add(ec.encrypt(nullPoint));
   voteList.add(ec.encrypt(votePoint));
   vbox.getChildren().addAll(checkBox);
  }
  CheckBox checkBox = new CheckBox(
    candidateList.get(i).get(candidateList.get(i).size() -
        1).getName());
  checkBoxList.add(checkBox);
  nullVoteList.add(ec.encrypt(nullPoint));
  voteList.add(ec.encrypt(votePoint));
  checkBox.selectedProperty().addListener(new
      ChangeListener<Boolean>() {
   @Override
   public void changed(ObservableValue<? extends Boolean>
       observable, Boolean oldValue,
     Boolean newValue) {
    for (int i = 0; i < checkBoxList.size() - 1; i++) {
     checkBoxList.get(i).setSelected(false);
     checkBoxList.get(i).setDisable(newValue);
    }
    updateCheckbox(checkBoxList, checkboxIndex);
   }
  });
  allNullVoteCheckBox.add(nullVoteList);
  allVoteCheckBox.add(voteList);
  allCheckBox.add(checkBoxList);
  Label hashLabel = new Label("Hash Code: ");
  Label separatorLabel = new Label(" ");
  separatorLabel.setFont(new Font("Arial", 5));
  hashLabelListCheckBox.add(hashLabel);
  vbox.getChildren().addAll(checkBox, hashLabel,
      separatorLabel);

 } else {
  errorLabelListRadio.add(errorLabel);
  positionIndexRadio.add(i);

  ToggleGroup toggleGroup = new ToggleGroup();
  List<RadioButton> radioButtonList = new ArrayList<
      RadioButton>();
  List<ECPair> nullVoteList = new ArrayList<ECPair>();
  List<ECPair> voteList = new ArrayList<ECPair>();
  int radioButtonIndex = allRadioButton.size();
  for (int j = 0; j < candidateList.get(i).size() - 1; j++)
      {
   RadioButton radioButton = new RadioButton(
       candidateList.get(i).get(j).getName()
     + getParty(candidateList.get(i).get(j).getPartyName())
         );
   radioButton.selectedProperty().addListener(new
       ChangeListener<Boolean>() {
    @Override
    public void changed(ObservableValue<? extends Boolean
        > observable, Boolean oldValue,
      Boolean newValue) {
     updateRadioButton(radioButtonList, radioButtonIndex);
    }
   });
   radioButton.setToggleGroup(toggleGroup);
   radioButtonList.add(radioButton);
   nullVoteList.add(ec.encrypt(nullPoint));
   voteList.add(ec.encrypt(votePoint));
   vbox.getChildren().addAll(radioButton);
  }
  RadioButton radioButton = new RadioButton(
    candidateList.get(i).get(candidateList.get(i).size() -
        1).getName());
  radioButton.setToggleGroup(toggleGroup);
  radioButton.selectedProperty().addListener(new
      ChangeListener<Boolean>() {
   @Override
   public void changed(ObservableValue<? extends Boolean>
       observable, Boolean oldValue,
     Boolean newValue) {
    updateRadioButton(radioButtonList, radioButtonIndex);
   }
  });
  radioButtonList.add(radioButton);
  nullVoteList.add(ec.encrypt(nullPoint));
  voteList.add(ec.encrypt(votePoint));

  allNullVoteRadio.add(nullVoteList);
  allVoteRadio.add(voteList);
  allRadioButton.add(radioButtonList);
  Label hashLabel = new Label("Hash Code: ");
  Label separatorLabel = new Label(" ");
  separatorLabel.setFont(new Font("Arial", 5));
  hashLabelListRadio.add(hashLabel);
  vbox.getChildren().addAll(radioButton, hashLabel,
      separatorLabel);
 }
 }
 vbox.setSpacing(3);

 scrollPane.setContent(vbox);
} else {
 Label label = new Label("The election is finished!");
 scrollPane.setContent(label);
 submitButton.setDisable(true);
}
}

@FXML
public void submit() {
 String status = databaseHelper.getElectionStatus(election.
     getElectionId());
 if (status.equals("Voting Stage")) {
  boolean hasError = false;
  for (int i = 0; i < allCheckBox.size(); i++) {
   int numChosen = 0;
   for (int j = 0; j < allCheckBox.get(i).size(); j++) {
    if (allCheckBox.get(i).get(j).selectedProperty().getValue()
        )
     numChosen++;
   }
   int max = maxVoteCheckbox.get(i);
   if (numChosen > max || numChosen < 1) {
    if (numChosen == 0)
     errorLabelListCheckBox.get(i).setText("Must choose at
         least one");
    else
     errorLabelListCheckBox.get(i).setText("Must choose a
         maximum of " + max + " candidate/s");
    hasError = true;
   } else
    errorLabelListCheckBox.get(i).setText("");
  }
  for (int i = 0; i < allRadioButton.size(); i++) {
   int numChosen = 0;
   for (int j = 0; j < allRadioButton.get(i).size(); j++) {
    if (allRadioButton.get(i).get(j).selectedProperty().
        getValue())
     numChosen++;
   }
```

```
 if (numChosen < 1) {
  if (numChosen == 0)
   errorLabelListRadio.get(i).setText("Must choose at least
         one");
  hasError = true;
 } else
  errorLabelListRadio.get(i).setText("");
}
Timestamp date = new java.sql.Timestamp(new java.util.
      Date().getTime());

if (!hasError) {
 File directory = new File(System.getProperty("user.home")
       + "/Desktop");

 BigInteger verificationCode = new BigInteger("0");

 String hashCodeDirectory = directory.getAbsolutePath() +
      "\\hash_code" + voter.getVoterId() + ".txt";
 try {
  PrintWriter writer = new PrintWriter(hashCodeDirectory,
       "UTF-8");
  writer.println("Voting receipt for " + election.getName()
       + ":");
  int currentRadioIndex = 0;
  int currentCheckBoxIndex = 0;
  String currentSubElection = "";
  for (int i = 0; i < positionList.size(); i++) {
   BigInteger sum = new BigInteger("0");
   ECPoint point1, point2;
   if (!currentSubElection.equals(subElectionName.get(i))) {
    currentSubElection = subElectionName.get(i);
    writer.println();
    writer.println(currentSubElection);
   }

   if (positionIndexRadio.size() > currentRadioIndex) {
    if (positionIndexRadio.get(currentRadioIndex) == i) {
     for (int j = 0; j < allRadioButton.get(
           currentRadioIndex).size(); j++) {
      if (allRadioButton.get(currentRadioIndex).get(j).
            selectedProperty().getValue()) {
       point1 = allVoteRadio.get(currentRadioIndex).get(j).
             getX();
       point2 = allVoteRadio.get(currentRadioIndex).get(j).
             getY();
      } else {
       point1 = allNullVoteRadio.get(currentRadioIndex).get(
             j).getX();
       point2 = allNullVoteRadio.get(currentRadioIndex).get(
             j).getY();

      }
      sum = sum.add(new BigInteger(point1.getEncoded(false
            )));
      sum = sum.add(new BigInteger(point2.getEncoded(false
            )));
     }
     verificationCode = verificationCode.add(sum);
     writer.println(positionList.get(i).getName() + " | " +
           getMD5(sum.toString()));
     currentRadioIndex++;
    }
   }
   if (positionIndexCheckBox.size() > currentCheckBoxIndex
         ) {
    if (positionIndexCheckBox.get(currentCheckBoxIndex)
          == i) {
     for (int j = 0; j < allCheckBox.get(
           currentCheckBoxIndex).size(); j++) {
      if (allCheckBox.get(currentCheckBoxIndex).get(j).
            selectedProperty().getValue()) {
       point1 = allVoteCheckBox.get(currentCheckBoxIndex).
             get(j).getX();
       point2 = allVoteCheckBox.get(currentCheckBoxIndex).
             get(j).getY();
      } else {
       point1 = allNullVoteCheckBox.get(
             currentCheckBoxIndex).get(j).getX();
       point2 = allNullVoteCheckBox.get(
             currentCheckBoxIndex).get(j).getY();

      }
      sum = sum.add(new BigInteger(point1.getEncoded(false
            )));
      sum = sum.add(new BigInteger(point2.getEncoded(false
            )));
     }
     verificationCode = verificationCode.add(sum);
     writer.println(positionList.get(i).getName() + " | " +
           getMD5(sum.toString()));
     currentCheckBoxIndex++;
    }
   }
   writer.println();
   writer.println("Verification Code: " + getMD5(
         verificationCode.toString()));
   writer.println();
   writer.println("Your alias is: " + voter.getAlias());
   writer.println();
   writer.println("Timestamp: " + date);
   writer.close();
  } catch (IOException e) {
   e.printStackTrace();
  }

  databaseHelper.deletePreviousVote(voter.getVoterId());
  databaseHelper.setVoted(voter.getVoterId());
  for (int i = 0; i < allCheckBox.size(); i++) {
   for (int j = 0; j < allCheckBox.get(i).size(); j++) {
    ECPair encrypted;
    if (allCheckBox.get(i).get(j).selectedProperty().getValue
          ()) {
     encrypted = allVoteCheckBox.get(i).get(j);
    } else {
     encrypted = allNullVoteCheckBox.get(i).get(j);
    }
    databaseHelper.addVote(voter.getVoterId(),
      candidateList.get(positionIndexCheckBox.get(i)).get(j).
            getCandidateId(), encrypted,
      date);
   }
  }
  for (int i = 0; i < allRadioButton.size(); i++) {
   for (int j = 0; j < allRadioButton.get(i).size(); j++) {
    ECPair encrypted;
    if (allRadioButton.get(i).get(j).selectedProperty().
          getValue()) {
     encrypted = allVoteRadio.get(i).get(j);
    } else {
     encrypted = allNullVoteRadio.get(i).get(j);
    }
    databaseHelper.addVote(voter.getVoterId(),
      candidateList.get(positionIndexRadio.get(i)).get(j).
            getCandidateId(), encrypted, date);
   }
  }
  EmailHelper emailHelper= new EmailHelper();
  String body = "Thank you for voting. Your voting receipt is
        attached below.";

  emailHelper.sendMail(voter.getEmail(), "Voting Receipt",
       body, hashCodeDirectory);
  Alert doneAlert = new Alert(AlertType.INFORMATION);
  doneAlert.setTitle("YuVote");
  doneAlert.initOwner(mainApp.getPrimaryStage());

  doneAlert.setHeaderText(null);
  doneAlert.setContentText("Your vote was submitted!");
  doneAlert.showAndWait();

  mainApp.showHome();



 }
 } else {
  Alert alert = new Alert(AlertType.INFORMATION);
  alert.initOwner(mainApp.getPrimaryStage());
  alert.setTitle("YuVote");
  alert.setHeaderText(null);
  alert.setContentText("Election is over, you cannot vote
        anymore");
  alert.showAndWait();
  mainApp.showHome();
 }
}

public String getRestriction(int max) {
 if (max != 1)
  return "(Max: " + max + ")";
 else
  return "";
}

public String getParty(String partyName) {
 if (partyName.equals("Independent"))
  return "";
 else
  return " (" + partyName + ")";
}

public String getMD5(String str) {
```

```
 try {
  MessageDigest md = MessageDigest.getInstance("MD5");
  md.update(str.getBytes());
  byte[] digest = md.digest();
  return (DatatypeConverter.printHexBinary(digest).
        toUpperCase());
 } catch (NoSuchAlgorithmException nsae) {
  return null;
 }
}

public void updateCheckbox(List<CheckBox> checkBoxList,
      int checkboxIndex) {
 boolean hasChosen = false;
 for (int i = 0; i < checkBoxList.size(); i++) {
  hasChosen = hasChosen || checkBoxList.get(i).
        selectedProperty().getValue();
 }
 if (hasChosen) {
  BigInteger sum = new BigInteger("0");
  ECPoint point1, point2;
  for (int i = 0; i < checkBoxList.size(); i++) {
   if (checkBoxList.get(i).selectedProperty().getValue()) {
    point1 = allVoteCheckBox.get(checkboxIndex).get(i).getX();
    point2 = allVoteCheckBox.get(checkboxIndex).get(i).getY();
   } else {
    point1 = allNullVoteCheckBox.get(checkboxIndex).get(i).
          getX();
    point2 = allNullVoteCheckBox.get(checkboxIndex).get(i).
          getY();

   }
   sum = sum.add(new BigInteger(point1.getEncoded(false)));
   sum = sum.add(new BigInteger(point2.getEncoded(false)));
  }
  hashLabelListCheckBox.get(checkboxIndex).setText("Hash
        Code: " + getMD5(sum.toString()));
 } else {
  hashLabelListCheckBox.get(checkboxIndex).setText("Hash
        Code: ");
 }
}

public void updateRadioButton(List<RadioButton>
      radioButtonList, int radioButtonIndex) {
 BigInteger sum = new BigInteger("0");
 ECPoint point1, point2;
 for (int i = 0; i < radioButtonList.size(); i++) {
  if (radioButtonList.get(i).selectedProperty().getValue()) {
   point1 = allVoteRadio.get(radioButtonIndex).get(i).getX();
   point2 = allVoteRadio.get(radioButtonIndex).get(i).getY();
  } else {
   point1 = allNullVoteRadio.get(radioButtonIndex).get(i).getX
         ();
   point2 = allNullVoteRadio.get(radioButtonIndex).get(i).getY
         ();

  }
  sum = sum.add(new BigInteger(point1.getEncoded(false)));
  sum = sum.add(new BigInteger(point2.getEncoded(false)));
 }
 hashLabelListRadio.get(radioButtonIndex).setText("Hash
       Code: " + getMD5(sum.toString()));
}
}
```

Candidate.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;

public class Candidate {
 private final IntegerProperty candidateId;
 private final IntegerProperty positionId;
 private final SimpleStringProperty name;
 private final SimpleStringProperty partyName;

 public Candidate(int candidateId, int positionId, String
       partyName, String name) {
  this.candidateId = new SimpleIntegerProperty(candidateId);
  this.positionId = new SimpleIntegerProperty(positionId);
  this.name = new SimpleStringProperty(name);
  this.partyName = new SimpleStringProperty(partyName);
 }

 public void setCandidateId(int candidateId) {
  this.candidateId.set(candidateId);
 }

 public void setPositionId(int positionId) {
  this.positionId.set(positionId);
 }

 public void setPartyName(String partyName) {
  this.partyName.set(partyName);
 }

 public void setFirstName(String name) {
  this.name.set(name);
 }

 public int getCandidateId() {
  return candidateId.get();
 }

 public int getPositionId() {
  return positionId.get();
 }

 public String getName() {
  return name.get();
 }

 public String getPartyName() {
  return partyName.get();
 }
}
```

Election.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;

public class Election {
 private final IntegerProperty electionId;
 private final IntegerProperty creatorId;
 private final SimpleStringProperty name;
 private final SimpleStringProperty description;
 private final SimpleStringProperty status;
 private final byte[] publicKey;
 private final byte[] votePoint;

 public Election(int electionId, int creatorId, String name,
       String description, String status, byte[] publicKey,
   byte[] votePoint) {
  this.electionId = new SimpleIntegerProperty(electionId);
  this.creatorId = new SimpleIntegerProperty(creatorId);
  this.name = new SimpleStringProperty(name);
  this.description = new SimpleStringProperty(description);
  this.status = new SimpleStringProperty(status);
  this.publicKey = publicKey;
  this.votePoint = votePoint;
 }

 public void setElectionId(int electionId) {
  this.electionId.set(electionId);
 }

 public void setCreatorId(int creatorId) {
  this.creatorId.set(creatorId);
 }

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

 public void setDescription(String description) {
  this.name.set(description);
 }

 public void setStatus(String status) {
  this.status.set(status);
 }

 public int getElectionId() {
  return electionId.get();
 }

 public int getCreatorId() {
  return creatorId.get();
 }

 public String getName() {
  return name.get();
 }

 public String getDescription() {
  return description.get();
```

```
 }

 public String getStatus() {
  return status.get();
 }

 public byte[] getPublicKey() {
  return publicKey;
 }

 public byte[] getVotePoint() {
  return votePoint;
 }
}
```

Position.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;

public class Position {
 private final IntegerProperty positionId;
 private final SimpleStringProperty name;
 private final IntegerProperty maxVote;

 public Position(int positionId, String name, int maxVote) {
  this.positionId = new SimpleIntegerProperty(positionId);
  this.name = new SimpleStringProperty(name);
  this.maxVote = new SimpleIntegerProperty(maxVote);
 }

 public void setPositionId(int positionId) {
  this.positionId.set(positionId);
 }

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

 public void setMaxVote(int maxVote) {
  this.maxVote.set(maxVote);
 }

 public int getPositionId() {
  return positionId.get();
 }

 public String getName() {
  return name.get();
 }

 public int getMaxVote() {
  return maxVote.get();
 }
}
```

Voter.java

```
package ph.edu.upm.cas.dpsm.bjyu.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;

public class Voter {
 private final IntegerProperty voterId;
 private final IntegerProperty userId;
 private final IntegerProperty blockId;
 private final SimpleStringProperty username;
 private final SimpleStringProperty voterPassword;
 private final SimpleStringProperty alias;
 private final SimpleStringProperty email;
 private final byte[] voterSalt;

 public Voter(int voterId, int userId, int blockId, String
       username, byte[] voterSalt, String voterPassword,
   String alias, String email) {
  this.voterId = new SimpleIntegerProperty(voterId);
  this.userId = new SimpleIntegerProperty(userId);
  this.blockId = new SimpleIntegerProperty(blockId);
  this.username = new SimpleStringProperty(username);
  this.voterSalt = voterSalt;
  this.voterPassword = new SimpleStringProperty(
        voterPassword);
  this.email = new SimpleStringProperty(email);
  this.alias = new SimpleStringProperty(alias);
 }

 public void setVoterId(int voterId) {
  this.voterId.set(voterId);
 }

 public void setUserId(int userId) {
  this.userId.set(userId);
 }

 public void setBlockId(int blockId) {
  this.blockId.set(blockId);
 }

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

 public void setVoterPassword(String voterPassword) {
  this.voterPassword.set(voterPassword);
 }

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

 public void setAlias(String alias) {
  this.alias.set(alias);
 }

 public int getVoterId() {
  return voterId.get();
 }

 public int getUserId() {
  return userId.get();
 }

 public int getBlockId() {
  return blockId.get();
 }

 public String getUsername() {
  return username.get();
 }

 public String getVoterPassword() {
  return voterPassword.get();
 }

 public String getEmail() {
  return email.get();
 }

 public String getAlias() {
  return alias.get();
 }

 public byte[] getVoterSalt() {
  return voterSalt;
 }
}
```

DatabaseHelper.java

```
package ph.edu.upm.cas.dpsm.bjyu.util;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

import org.bouncycastle.crypto.ec.ECPair;

import ph.edu.upm.cas.dpsm.bjyu.model.Election;
import ph.edu.upm.cas.dpsm.bjyu.model.Voter;

public class DatabaseHelper {
 private Connection conn;

 public DatabaseHelper() {
  Properties prop = new Properties();
  InputStream input = null;
  String keyStore = "", keyStorePassword = "", trustStore =
        "", trustStorePassword = "";
  String url = "jdbc:mysql://";
  String user = "", password = "";
  try {
```

```
  input = new FileInputStream("config.properties");
  prop.load(input);
  keyStore = prop.getProperty("javax.net.ssl.keyStore");
  keyStorePassword = prop.getProperty("javax.net.ssl.
        keyStorePassword");
  trustStore = prop.getProperty("javax.net.ssl.trustStore");
  trustStorePassword = prop.getProperty("javax.net.ssl.
        trustStorePassword");
  user = prop.getProperty("db_user");
  password = prop.getProperty("db_password");
  url += prop.getProperty("db_host") + ":" + prop.
        getProperty("db_port");
  url += "/yuVote?verifyServerCertificate=true&useSSL=true
        &requireSSL=true";
 } catch (IOException ex) {
  ex.printStackTrace();
 }
 System.setProperty("javax.net.ssl.keyStore", keyStore);
 System.setProperty("javax.net.ssl.keyStorePassword",
        keyStorePassword);
 System.setProperty("javax.net.ssl.trustStore", trustStore);
 System.setProperty("javax.net.ssl.trustStorePassword",
        trustStorePassword);

 try {
  conn = null;
  Class.forName("com.mysql.jdbc.Driver");
  conn = DriverManager.getConnection(url, user, password);
 } catch (Exception e) {
  System.out.println(e);
 }
}

public Voter usernameMatchesPassword(String username,
      String password) {
 try {
  PasswordHelper passwordHelper = new PasswordHelper();
  String sql = "SELECT * FROM voter INNER JOIN user ON
        user.user_id = voter.user_id WHERE user.username =
        ? AND user.type = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setString(1, username);
  ps.setString(2, "Voter");
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   byte[] salt = rs.getBytes(6);
   if (rs.getString(5).equals(passwordHelper.
        getSecurePassword(password, salt))) {
    Voter voter = new Voter(rs.getInt(1), rs.getInt(2), rs.
        getInt(3), rs.getString(10), rs.getBytes(6),
      rs.getString(5), rs.getString(4), rs.getString(13));
    return voter;
   }
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return null;
}

public Election getElection(int blockId) {
 try {
  String sql = "SELECT * FROM block WHERE block_id = ?
        ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ResultSet rs = ps.executeQuery();
  rs.next();
  int electionId = rs.getInt(2);

  sql = "SELECT * FROM election WHERE election_id = ? ";
  ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  rs = ps.executeQuery();
  rs.next();
  Election election = new Election(rs.getInt(1), rs.getInt(2),
        rs.getString(3), rs.getString(4),
    rs.getString(7), rs.getBytes(8), rs.getBytes(9));
  return election;
 } catch (Exception e) {
 }

 return null;
}

public List<Integer> getSubElectionId(int blockId) {
 List<Integer> subElectionIdList = new ArrayList<Integer>();
 try {
  String sql = "SELECT * FROM block_sub_election WHERE
        block_id = ? ";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ResultSet rs = ps.executeQuery();
  while (rs.next()) {
   subElectionIdList.add(rs.getInt(2));
  }
 } catch (Exception e) {
  System.out.println(e);
 }
 return subElectionIdList;
}

public void addVote(int voterId, int candidateId, ECPair
      eCPair, Timestamp date) {
 try {
  String sql = "INSERT INTO vote (voter_id, candidate_id,
        time, encrypted_point1, encrypted_point2) VALUES (?,
        ?, ?, ?, ?)";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, voterId);
  ps.setInt(2, candidateId);
  ps.setTimestamp(3, date);
  ps.setBytes(4, eCPair.getX().getEncoded(false));
  ps.setBytes(5, eCPair.getY().getEncoded(false));
  ps.executeUpdate();
 } catch (Exception e) {
  System.out.println(e);
 }
}

public void deletePreviousVote(int voterId) {
 try {
  String sql = "DELETE FROM vote WHERE voter_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, voterId);
  ps.executeUpdate();// integer
 } catch (Exception e) {
  System.out.println(e);
 }
}

public ResultSet getBallot(int blockId) {
 try {
  String sql = "SELECT * FROM ballot WHERE block_id = ?
        ORDER BY ballot_order";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, blockId);
  ResultSet rs = ps.executeQuery();
  return rs;
 } catch (Exception e) {
  System.out.println(e);
 }
 return null;
}

public String getElectionStatus(int electionId) {
 try {
  String sql = "SELECT * FROM election WHERE election_id
        = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, electionId);
  ResultSet rs = ps.executeQuery();
  rs.next();
  return rs.getString(7);
 } catch (Exception e) {
  System.out.println(e);
 }
 return null;
}

public void setVoted(int voterId) {
 try {
  String sql = "UPDATE voter SET voted = ? WHERE
        voter_id = ?";
  PreparedStatement ps = conn.prepareStatement(sql);
  ps.setInt(1, 1);
  ps.setInt(2, voterId);

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

EmailHelper.java

```
package ph.edu.upm.cas.dpsm.bjyu.util;

import java.io.File;
import java.util.Properties;

import javax.activation.DataHandler;
import javax.activation.DataSource;
```

```
import javax.activation.FileDataSource;
import javax.mail.BodyPart;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

public class EmailHelper {
 Message message;

 public EmailHelper() {
  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");
  props.put("mail.smtp.ssl.trust", "smtp.gmail.com");

  Session session = Session.getInstance(props, new javax.mail.
        Authenticator() {
   protected PasswordAuthentication getPasswordAuthentication
         () {
    return new PasswordAuthentication("yuvote96@gmail.com",
          "yuvoteforsp17");
   }
  });
  message = new MimeMessage(session);

 }

 public boolean sendMail(String email, String subject, String
       body, String filename) {
  try {
   message.setFrom(new InternetAddress("from-email@gmail.
         com"));
   message.setRecipients(Message.RecipientType.TO,
         InternetAddress.parse(email));
   message.setSubject(subject);

   BodyPart messageBodyPart = new MimeBodyPart();
   // Now set the actual message
   messageBodyPart.setText(body);

   // Create a multipar message
   Multipart multipart = new MimeMultipart();

   // Set text message part
   multipart.addBodyPart(messageBodyPart);

   // Part two is attachment
   messageBodyPart = new MimeBodyPart();
   DataSource source = new FileDataSource(filename);
   messageBodyPart.setDataHandler(new DataHandler(source));
   messageBodyPart.setFileName(new File(filename).getName())
         ;
   multipart.addBodyPart(messageBodyPart);

   // Send the complete message parts
   message.setContent(multipart);

   Transport.send(message);
   return true;

  } catch (MessagingException e) {
   return false;
  }
 }
}
```

PasswordHelper.java

```
package ph.edu.upm.cas.dpsm.bjyu.util;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.SecureRandom;

//This class is credited to http://howtodoinjava.com/security/
      how-to-generate-secure-password-hash-md5-sha-
      pbkdf2-bcrypt-examples/
public class PasswordHelper {
 private SecureRandom random = new SecureRandom();

 public String generatePassword() {
  return new BigInteger(130, random).toString(32).substring(0,
        8);
 }

 public String getSecurePassword(String passwordToHash, byte[]
       salt) {
  String generatedPassword = null;
  try {
   MessageDigest md = MessageDigest.getInstance("SHA");
   md.update(salt);
   byte[] bytes = md.digest(passwordToHash.getBytes());
   StringBuilder sb = new StringBuilder();
   for (int i = 0; i < bytes.length; i++) {
    sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).
          substring(1));
   }
   generatedPassword = sb.toString();
  } catch (NoSuchAlgorithmException e) {
   e.printStackTrace();
  }
  return generatedPassword;
 }

 public byte[] getSalt() throws NoSuchAlgorithmException,
       NoSuchProviderException {
  SecureRandom sr = SecureRandom.getInstance("SHA1PRNG",
        "SUN");
  byte[] salt = new byte[16];
  sr.nextBytes(salt);
  return salt;
 }
}
```

B.3 Web Application

MainController.php

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed')
      ;
header('Access-Control-Allow-Origin: *');

class MainController extends CI_Controller {

 public function __construct() {// a function to load the model
   parent::__construct();
   $this->load->model('main_model');//loads this model
   $this->load->helper('url');
  }

 public function index(){
  $data['election'] = $this->main_model->get_elections();
  $sql = $this->main_model->get_elections();// get the infos
        of the user in the database
  $this->load->view('mainView', $data);
 }

 public function changeBulletinBoard(){
  $voter_JSON = array();
  $election_name = $this->input->post('electionName');
  $block_list = $this->main_model->get_blocks(
        $election_name);
  $i = 0;
  foreach( $block_list ->result() as $block) {
   $voter_list = $this->main_model->get_voted($block->
         block_id);
   foreach( $voter_list ->result() as $voter){
    $voter_JSON[$i] = array("alias" => $voter->alias, "
          verification_code" => $voter->verification_code);
    $i++;
   }
  }
  echo json_encode($voter_JSON);
 }

 public function changeElection(){
  $candidate_JSON = array();
  $i = 0;
  $election_name = $this->input->post('electionName');
  $sub_election_list = $this->main_model->get_sub_elections(
        $election_name);
  foreach( $sub_election_list ->result() as $sub_election) {
   $position_list = $this->main_model->get_positions(
         $sub_election->sub_election_id);
   foreach( $position_list ->result() as $position) {
    $candidate_list = $this->main_model->get_candidates(
          $position->position_id);
    foreach( $candidate_list ->result() as $candidate){
     $candidate_JSON[$i] = array("sub_election" =>
           $sub_election->sub_election_name, "position" =>
           $position->position_name,
```

```
      "candidate" => $candidate->first_name." ".$candidate
            ->last_name, "party" => $candidate->party_name
            , "tally" => $candidate->tally);
     $i++;
    }
   }

 }
 echo json_encode($candidate_JSON);
}

public function download(){
 $this->load->helper('pdf_helper');

 $candidate_JSON = array();
 $i = 0;
 $election_name = $this->input->post('electionName');
  $sub_election_list = $this->main_model->get_sub_elections(
       $election_name);
 foreach( $sub_election_list ->result() as $sub_election) {
   $position_list = $this->main_model->get_positions(
        $sub_election->sub_election_id);
  foreach( $position_list ->result() as $position) {
    $candidate_list = $this->main_model->get_candidates(
         $position->position_id);
   foreach( $candidate_list ->result() as $candidate){
    $candidate_JSON[$i] = array("sub_election" =>
          $sub_election->sub_election_name, "position" =>
          $position->position_name,
      "candidate" => $candidate->first_name." ".$candidate
            ->last_name, "party" => $candidate->party_name
            , "tally" => $candidate->tally);
     $i++;
    }
  }

 }
 $current_sub_election = "";
 $current_position = "";
 $html = "<h2>".$election_name."</h2>";
 for ($i = 0; $i < count($candidate_JSON); $i++) {
  if($current_sub_election != $candidate_JSON[$i]["
       sub_election"]){
   $current_sub_election = $candidate_JSON[$i]["sub_election
        "];
   $html .= "<h3>" . $current_sub_election . "</h3>";
  }
  if($current_position != $candidate_JSON[$i]["position"]){
   $current_position = $candidate_JSON[$i]["position"];
   $html .= $current_position . "<br>";
   $html .= "<table border=\"0.5\" width=\"50%\">";
  }
  if($candidate_JSON[$i]["party"] != "none")
   $html .= "<tr><td>" . $candidate_JSON[$i]["candidate"] .
        " (" . $candidate_JSON[$i]["party"] . ")</td><td>" .
         $candidate_JSON[$i]["tally"] . "</td></tr>";
  else
   $html .= "<tr><td>" . $candidate_JSON[$i]["candidate"] .
        "</td><td>" . $candidate_JSON[$i]["tally"] . "</td
        ></tr>";
  if($i + 1 < count($candidate_JSON)){
   if($current_position != $candidate_JSON[$i + 1]["position
        "]){
    $html .= "</table>";
   }
  }
 }
 $html .= "</table>";

 tcpdf();

 $pdf = new TCPDF('P', PDF_UNIT, PDF_PAGE_FORMAT,
       true, 'UTF-8', false);
 $pdf->SetCreator(PDF_CREATOR);
 $title = "Election Results";
 $pdf->SetTitle($title);
 $pdf->setPrintHeader(false);
 $pdf->setPrintFooter(false);
 $pdf->SetDefaultMonospacedFont('helvetica');
 $pdf->SetMargins(PDF_MARGIN_LEFT,
       PDF_MARGIN_TOP, PDF_MARGIN_RIGHT);
 $pdf->SetAutoPageBreak(TRUE, PDF_MARGIN_BOTTOM);
 $pdf->SetFont('helvetica', '', 9);
 $pdf->setFontSubsetting(false);
 $pdf->AddPage();
 $pdf->writeHTML($html, true, false, false, false, '') ;
 $filename = $election_name. " Results.pdf";
 $pdf->Output($filename,'D');
}

public function hashCode(){
 $hash_code_JSON = array();
 $i = 0;
 $election_name = $this->input->post('electionName');
 $alias = $this->input->post('alias');
 $hash_code_list = $this->main_model->get_hash_codes(
       $election_name, $alias);
 foreach($hash_code_list->result() as $hash_code) {
  $hash_code_JSON[$i] = array("sub_election" => $hash_code
        ->sub_election_name, "position" => $hash_code->
        position_name,
     "hash_code" => $hash_code->hash_code);
    $i++;
 }
 echo json_encode($hash_code_JSON);
}

public function verify(){
 $message = "";
 $verifyReceipt = true;
 if(count($_FILES) > 0){
  $election_name = $this->input->post('electionName');
  $file_content = file_get_contents ($_FILES['file']['tmp_name
       ']);
  $filename = $_FILES['file']['name'];
  $ext = pathinfo($filename, PATHINFO_EXTENSION);
  if(in_array($ext,array("txt"))) {
   $result = "";
   $line = explode("\n", $file_content);
   $position = array("");
   $hash_code = array("");
   $alias;
   $verification_code;

   for($i = 0; $i < count($line); $i++){
    if( strpos( $line[$i], '|' ) !== false ) {
     $positionLine = explode(" | ", $line[$i]);
     array_push($position , $positionLine[0]);
     array_push($hash_code , $positionLine[1]);
    }
    if( strpos( $line[$i], "Verification Code: " ) !== false )
         {
     $verification = explode("Verification Code: ", $line[$i]);
     $verification_code = $verification[1];

    }
    if( strpos( $line[$i], "Your alias is: " ) !== false ) {
     $alias = strtok($line[$i], "Your alias is: ");
    }
   }
   for($i = 1; $i < count($position); $i++){
    //echo $this->main_model->check_hash_code(
         $election_name, $position[$i], $hash_code[$i], $alias);

    $verifyReceipt = $verifyReceipt && $this->main_model
         ->check_hash_code($election_name, $position[$i],
         $hash_code[$i], $alias);

   }
   $verifyReceipt = $verifyReceipt && $this->main_model->
        check_verification_code($verification_code,$alias);
   if($verifyReceipt)
    echo "Your receipt matches our database.";
   else
    echo "Your receipt does not match our database.";
  }
  else
   echo "The file should be a text file.";

 }
 else
  echo "No file chosen.";

 }
}

?>
```

main_model.php

```
<?php
 class Main_model extends CI_Model{
  public function __construct(){
   $this->load->database();
  }

  public function get_elections(){
   $sql = "SELECT * FROM election WHERE status = '
        Finished'";
   $sql = $this->db->query($sql) or die(mysql_error());
   return $sql;
  }

  public function get_sub_elections($election_name){//get the
       account info of the given username
```

```
  $sql = "SELECT * FROM sub_election INNER JOIN
         election on election.election_id = sub_election.
         election_id
    WHERE election.election_name = '".$election_name."'
          ORDER BY sub_election_order";
  $sql = $this->db->query($sql) or die(mysql_error());
  return $sql;
 }

 public function get_positions($sub_election_id){
  $sql = "SELECT * FROM position WHERE sub_election_id
         = '".$sub_election_id."' ORDER BY ballot_order";
  $sql = $this->db->query($sql) or die(mysql_error());
  return $sql;
 }

 public function get_candidates($position_id){
  $sql = "SELECT * FROM candidate INNER JOIN party ON
         candidate.party_id = party.party_id
   WHERE position_id = '".$position_id."' ORDER BY
         candidate_id";
  $sql = $this->db->query($sql) or die(mysql_error());
  return $sql;
 }

 public function get_blocks($election_name){
  $sql = "SELECT * FROM block INNER JOIN election on
         election.election_id = block.election_id
    WHERE election.election_name = '".$election_name."'";
  $sql = $this->db->query($sql) or die(mysql_error());
  return $sql;
 }

 public function get_voted($block_id){
  $sql = "SELECT * FROM voter WHERE block_id = '".
         $block_id."' AND voted = '1'";
  $sql = $this->db->query($sql) or die(mysql_error());
  return $sql;
 }

 public function get_hash_codes($election_name, $alias){
  $sql = "SELECT * FROM hash_code INNER JOIN election
         ON election.election_id = hash_code.election_id
    WHERE election_name = '".$election_name."' AND alias =
          '".$alias."'";
  $sql = $this->db->query($sql) or die(mysql_error());
  return $sql;
 }

 public function check_hash_code($election_name, $position,
        $hash_code, $alias){
  $sql = "SELECT * FROM hash_code INNER JOIN election
         ON election.election_id = hash_code.election_id
    WHERE election.election_name = ? AND hash_code.alias =
          ? AND hash_code.position_name = ? AND hash_code
         .hash_code = ?";

  $arg = array(trim($election_name),trim($alias),trim($position
         ),trim($hash_code));
  $sql = $this->db->query($sql,$arg) or die(mysql_error());
  if($sql->row_array() > 0)
   return true;
  else
   return false;
 }

 public function check_verification_code($verification_code,
        $alias){
  $sql = "SELECT * FROM voter WHERE verification_code =
          ? AND alias = ?";
  $arg = array(trim($verification_code),trim($alias));
  $sql = $this->db->query($sql,$arg) or die(mysql_error());
  if($sql->row_array() > 0)
   return true;
  else
   return false;

 }
}
?>
```

mainView.php

```
<?php
 defined('BASEPATH') OR exit('No direct script access allowed
        ');
 $this->load->helper('url');
 ?>

<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="utf-8">

 <link rel="stylesheet" href="<?php echo base_url(); ?>/assets
       /bootstrap/css/edited-bootstrap.min.css">
 <link rel="stylesheet" href="<?php echo base_url(); ?>/assets
       /css/jquery.dataTables.min.css">
 <link rel="icon" href="<?=base_url()?>/assets/img/YuVote
       Logo.png" type="image/gif">

 <script src="<?php echo base_url(); ?>/assets/js/jquery
       -3.2.1.min.js"></script>
 <script src="<?php echo base_url(); ?>/assets/bootstrap/js/
       bootstrap.min.js"></script>
 <script src="<?php echo base_url(); ?>/assets/js/jquery.
       dataTables.min.js"></script>

 <style type="text/css">
  body {
   overflow-x:hidden;
  }
 </style>
 <title>YuVote</title>
</head>
<body>
<nav id = "navbar" class="navbar navbar-border navbar-fixed
      -top bg-success" >
<div id="container" >
 <div class="page-header" style="padding-left: 15px;border-
       bottom: none">
  <img src="<?=base_url()?>/assets/img/YuVote Text.png"
        class="img-responsive" style="width:250px;height:50px
        ;">
 </div>
 <div class="row" style="padding-left: 15px;">
  <label for="electionSelect  selcls" class="col-sm-2 control-
        label">Election Name:</label>

  <div class="col-sm-3">
   <select class="form-control" id="electionSelect" data-style
         ="btn-success">
    <option value = "none"> </option>
    <?php
     if($election->num_rows() > 0){
      foreach($election->result() as $single_election) {
       echo "<option value = '".$single_election->
             election_name."'>".$single_election->
             election_name."</option>";
      }
     }

    ?>
   </select>
  </div>
 </div>
 <br>
 <div class="row" style="padding-left: 15px;">
  <label for="typeSelect" class="col-sm-2 control-label">
        Type:</label>
  <div class="col-sm-3">
   <select class="form-control" id="typeSelect">
    <option>Election Result</option>
    <option>Bulletin Board</option>
   </select>
  </div>
 </div>
 <br>
</div>
</nav>
<div id = "body" style="padding-left: 15px;padding-right: 15
      px;">
</div>
<div id = "buttonDiv" style="padding-left: 15px;">
</div>


<div class="modal fade" id="hashCodeModal" role="dialog">

  </div>

</body>
</html>

<script type="text/javascript">
$(document).ready(function(){
 var padding = $("#navbar").height() ;
 $("#body").css({ "padding-top": padding});
 changeBody();
 $(function() {
  $('#electionSelect').on('change', function(event) {
   changeBody();
  });
 });
```

```
$(function() {
  $('#typeSelect').on('change', function(event) {
    changeBody();
  });
});
});


function changeBody(){
     var electionName = $('#electionSelect').val();
 var type = $('#typeSelect').val();
 jQuery('#body').html('');
 jQuery('#buttonDiv').html('');
 if(electionName != "none"){
  if(type == "Bulletin Board"){
   $.ajax({
    type: 'POST',
    data: {"electionName": electionName, "type": type},
    url: '<?php echo site_url('maincontroller/
          changeBulletinBoard') ?>',
    dataType: 'json',
    success: function(data){
     html = "<table id='bulletinBoard' class='display'
           cellspacing='0' style='padding:10px;'>";
     html += " <thead>";
     html += "  <tr>";
     html += "   <th><center>Alias</center></th>";
     html += "   <th><center>Verification Code</center></
           th>";
     html += "   <th><center>HashCode</center></th>";
     html += "  </tr>";
     html += " </thead>";
     html += " <tbody>";
     for (i = 0; i < data.length; i++) {
      html += " <tr>";
      html += "  <td><center>" + data[i].alias + "</center
           ></td>";
      html += "  <td><center>" + data[i].verification_code +
           "</center></td>";
      html += "       <td> ";
      html += "   <center><button type='submit' class='btn
           btn-success hash' id='" + data[i].alias + "'>" + "
           View" + "</button></center></td>";
      html += "  </td>";
      html += " </tr>";
     }
     html += " </tbody> ";
     html += " </table> ";
     html += " </table> ";
     html += "<br><h3>Verify your votes by submitting the
           voting receipt: </h3>";
     html += "<input type='file' id='fileName' size='20' accept
           ='.txt'/>";
     html += "<br><br>";
     html += "<input type='submit' value='Verify' class='btn
           btn-success' id= 'verify'/>";

     html += "</form>";


     $("#body").append(html);
     $('#bulletinBoard').DataTable({

     });
     $('.hash').click(function() {
      var alias = $(this).attr('id');
      $.ajax({
       type: 'POST',
       data: {"electionName": electionName, "alias": alias},
       url: '<?php echo site_url('maincontroller/hashCode')
            ?>',
       dataType: 'json',
       success: function(data){
        jQuery('#hashCodeModal').html('');
        var html = "<div class='modal-dialog'>";
        html += "<div class='modal-content'>";
        html += " <div class='modal-header bg-success'>";
        html += "  <button type='button' class='close' data-
             dismiss='modal'>&times;</button>";
        html += "  <h4 class='modal-title'>Hash Code for "
             + alias + "</h4>";
        html += " </div>";
        html += "<div class='modal-body'>";
        var currentSubElection = "";
        for(i = 0; i < data.length; i++){
         if(currentSubElection != data[i].sub_election){
          currentSubElection = data[i].sub_election;
          html += "<p>" + currentSubElection + "</p>";
         }
         html += "<p>"+ data[i].position + " | " + data[i].
              hash_code +"</p>";
        }
        html += "</div>";
        html += "<div class='modal-footer'>";
        html += " <button type='button' class='btn btn-
             default' data-dismiss='modal'>Close</button>";
        html += " </div>";
        html += "</div>";
        html += "</div>";
        $("#hashCodeModal").append(html);
        $('#hashCodeModal').modal('show');
       }
      });
     });
     $('#verify').on('click', function() {
      var file_data = $('#fileName').prop('files')[0];
      var form_data = new FormData();
      form_data.append('file', file_data);
      form_data.append('electionName', electionName);
      $.ajax({
       url: '<?php echo site_url('maincontroller/verify') ?>',
            // point to server-side PHP script
       dataType: 'text',  // what to expect back from the PHP
            script, if anything
       cache: false,
       contentType: false,
       processData: false,
       data: form_data,
       type: 'post',
       success: function(response){
        jQuery('#hashCodeModal').html('');
        var html = "<div class='modal-dialog'>";
        html += "<div class='modal-content'>";
        html += " <div class='modal-header bg-success'>";
        html += "  <button type='button' class='close' data-
             dismiss='modal'>&times;</button>";
        html += "  <h4 class='modal-title'>Verify Receipt</
             h4>";
        html += " </div>";
        html += "<div class='modal-body'>";
        html += response;
        html += "</div>";
        html += "<div class='modal-footer'>";
        html += " <button type='button' class='btn btn-
             default' data-dismiss='modal'>Close</button>";
        html += " </div>";
        html += "</div>";
        html += "</div>";
        $("#hashCodeModal").append(html);
        $('#hashCodeModal').modal('show');
       }
       });
      });
     }
 });
}
else{
 $.ajax({
  type: 'POST',
  data: {"electionName": electionName, "type": type},
  url: '<?php echo site_url('maincontroller/changeElection')
       ?>',
  dataType: 'json',
  success: function(data){
   var currentSubElection = "";
   var currentPosition = "";
   var html = "";
   for (i = 0; i < data.length; i++) {
    if(currentSubElection != data[i].sub_election){
     currentSubElection = data[i].sub_election;
     html += "<h3>" + currentSubElection + "</h3>";
    }
    if(currentPosition != data[i].position){
     currentPosition = data[i].position;
     html += currentPosition + "<br>";
     html += "<table class='table table-bordered' style='
          width: 50%;'><tbody>";
    }
    if(data[i].party != "none")
     html += "<tr><td>" + data[i].candidate + " ("+ data[i
          ].party+ ")</td><td>" + data[i].tally + "</td
          ></tr>";
    else
     html += "<tr><td>" + data[i].candidate + "</td><td
          >" + data[i].tally + "</td></tr>";
    if(i + 1 < data.length){
     if(currentPosition != data[i + 1].position){
      html += "</tbody></table>";
     }
    }
   }
   $("#body").append(html);
   var link = "<form action='<?php echo site_url('
        maincontroller/download') ?>' method='POST'>";
```

```
 link += "<input type='hidden' name='electionName'
        value='" + electionName + "'>";
 link += "<button type='submit' class='btn btn-success
        '>Download Results</button>";
 link += "</form>";
 $("#buttonDiv").append(link);
}
                                                        });
                                                      }
                                                    }
                                                    };

                                                    </script>
```

XI. Acknowledgement

I would like to thank the following people for being part of my college life:

1. God: Thank You for always helping me and guiding me. I also want to thank You for always being there for me and for Your unconditional love.

2. Sir Richard Bryann Chua: Thank you for being my adviser and helping me a lot in my SP. I thank you for also giving me time to consult.

3. Family: Thank you for always being there for me. I also want to thank my parents for giving me so much love and care.

4. UPM: Thank you letting me feel what it means to be an Iskolar ng Bayan.

5. Ate Eden: Thank for being nice to us and helping us in a lot of things.

6. Professors: Thank you for teaching me and letting me learn a lot.

7. Friends: To Ironnel, Andrei, Alfred, Jethro, Jeffrey, Chico, Kier, and Jed, thank you for being my closest friends. Without you, my life in college would be very lonely.

8. Others who helped me in my SP: To Alfred, Joey, Kier, and Chico, thank you for helping me in my SP.

9. Blockmates: Thank you for making my stay in UPM a memorable one.