Secure Electronic Voting System
Nicholas William Maher Goodrich
Senior Project, CpE-499
Advisor: Cherrice Traver
11/11/14

Table of Contents

List of Tables and Figures

1. Report Summary:

Achieving a degree from Union College in the field of Computer Engineering culminates in a senior project, which from idea inception to final implementation rests on the shoulders of the student. Computer Engineering is the fascinating field that merges both software and hardware engineering in a computer or embedded system. As such, computer engineering senior projects should aim to integrate both hardware and software into their system design.

The Secure Electronic Voting System does exactly that; it solves significant electronic voting issues by a combination of software and hardware. The antiquated nature of our voting methods provides an interesting opportunity to design and create a more efficient and usable system that is devoid of fraud. This report will go into the current voting methods starting with paper ballots; then moving into the partial computerization of voting via punch card ballots, where the vote tally is done electronically; and finally the preferred method of electronic voting, namely the Direct Recording Electronic Voting Machines (DRE). Each of these voting methods or systems are flawed in some fundamental way, economically, environmentally, socially et cetera.

This project in the end will be an anonymous secure electronic voting system that is easy to register for and easy to use. The project is broken down into two main sections the hardware section and the software section. The hardware consists of the voting kiosk where votes will be cast as well as initial authentication and registration of voters. This report will introduce the project in terms of the design and design goals as well as contextualizing the project as a whole.

2. Introduction:

Voting is at the heart of the democratic system. In order for voting to be successful, as many voters as possible must cast their votes. At Union College we vote for a variety of positions including, the Executive Board, the Class Officers and a variety of representatives. Current voting systems are antiquated and highly prone to error. Issues arise with each voting system. Errors that occur in current voting systems vary from physical miscounts to electronic data compromises and losses. For example, Union College's voting system is comprised primarily of paper ballots, which can be incredibly cumbersome and inefficient to tally. In addition to tally errors, the identification process is as simple as the voter providing their id number and casting a ballot. The goal is to create an online fingerprint based voting system that is devoid of fraud which simultaneously increasing voter participation. A system like this would increase voter participation by creating a central voter registry that would essentially be an opt-in system, the pros of a voter registry has been researched and is discussed in the background section of this report. Voter registries provide a method for rapid identification and verification. This system while initially implemented and tested on a small scale (Union College) could eventually grow and scale for larger and more important elections or other such instances where voting is required. While increasing voter use is a primary concern, it is important to ensure malicious or improper use of the voting system does not occur.

Direct Electronic Recording voting machines also known as DRE's are a step in the right direction forward towards a significantly more efficient voting system. DRE's locally store all voter data on the machine, that data is then collected by

election personnel; there is no external link into the system. DRE's finally introduced total computerization of the voting process from start to finish. They are quite good at recording and tallying votes in an effective manner. While DRE's are a good first step there are several issues that must be addressed such as security. There are a number of things that someone with malicious intent could do to a system that does not use remote vote checking, such as determine who an individual voted for. Another possibility is that malware could augment vote totals after they have been recorded but before they are downloaded for tallying. Another issue with the DRE system is that a voter is unsure whether their vote has been recorded or not.

The end result of this project will be an anonymous secure electronic voting system which is easy to register for and easy to use. The project is broken down into two main sections; the hardware section and the software section. The hardware consists of a voting kiosk where votes will be cast. The kiosk also handles the initial authentication and registration of voters. The security is paramount, which is why the voter will initialize their voting via the kiosk and their student ID; therefore, there must be a secure way of pairing people's student ID to their voter ID/fingerprint. After the initial pairing of the fingerprint to the verified identity of the individual, all the user needs in order to vote is their finger. Once a student has paired their ID number to their fingerprint the issue of selling votes is eliminated.

The second section will be hosted on a server with the main purpose of remotely storing information for increased speed for verification and retrieval of election data. The server based system has security procedures built in to prevent

the addition of malicious code or malware. The combination of systems hardware and software will effectively deal with the issues that are inherent in electronic voting while simultaneously increasing voter participation.

3. Background:

3.1. Voter Turnout

The democratic process has been in existence for centuries. At the center of this process is voting, where voter turnout is of the utmost importance due to the need for the election result to truly represent the will of the majority. There are a number of aspects that influence voter turnout. The first aspect which is prevalent in the United States is electoral competitiveness, the states in which the electoral race is more competitive drives more people to vote. For example “76% of voting eligible Minnesotans casted ballots, whereas only 45% of eligible Hawaiians did [1]. There are number of factors that influence voter turnout during presidential election years, including local and state voting laws and coinciding elections for high-profile offices like governor and senator, one important factor is the competitiveness of the presidential election in each state. Competitiveness of an election refers to how competitive the actual election is, for example, a perfectly uncompetitive election would be one with only one candidate; and a very competitive election would be a very close election in the polls. Overall, 66% of eligible voters turned out to the polls in the nation's 12 most competitive states in 2012, but only 57% did in the nation's 39 other states (including the District of Columbia). “ [1]. A system which increases voter turnout would greatly impact the democratic process in a substantially beneficial way. Increased voter turnout would

have massive social as well as political benefits. Even in states that are not as competitive it is important to voice the unheard opinion so when lawmakers and policymakers are deciding what is best for their respective state or organization they have a more complete set of facts and desires of their voting demographic. There are other influences to voter turnout such as age demographics. For example, citizens from the age of 18 to 29 had a turnout percentage of 15 to 20 points lower than those 30 or above [1]. The secure electronic voting system aims to increase that number by making voting simpler, more effective and more appealing to the aforementioned age group without losing the votes from older generations. Tech savvy generations are just now entering the voting populace and will soon encompass the majority. At this time it would make sense to fully eliminate non electronic voting methods. It is clear that students at Union college fall into this category. Professors and faculty also fall into this category since they are in close proximity to this demographic on a regular basis, in addition professors and faculty at Union College or any institution are significantly more exposed to emerging technologies which decreases their trepidation towards such technologies.

One idea to modernize voting is to create a universal voter registration. "Complete and accurate voter rolls are essential to the integrity of the electoral process and the legitimacy of results. Yet, as evidenced by recent elections, voter rolls are littered with duplicate registrants and errors. Nearly a third of eligible American voters are not registered to vote and voter registration drives result in a surge of registrations close to an election that are difficult to process and that create unanticipated demands on polling places. As a result, millions of eligible voters are

effectively shut out of the political process." [2] The current system allows for chronic fraud in absentee balloting. On average Americans worry more about voter fraud than do voters in other countries, as Americans do not have a reliable system of national identification.[3] The voting system in the United States is not a nationally decided upon process, it is determined by local legislature which allows for the party with the most funding and money to make it more difficult for the other party or lower class citizens to vote.

3.2. Voting History

The evolution of voting systems in the United States is a rapid one. Paper ballots while they were first used in Rome in 139 BCE, were not implemented in what became the United States until 1629. Arguably this system is one of the least efficient and effective. This system employs a uniform official ballot of various stock weights on which the names of all candidates are printed. Voters recorded their choices next to the candidate then place their sealed ballot in a ballot box. This clearly requires the voter to attend a voting center such as their Town Hall or other community gathering location. As the technological movement continues the use of paper ballots has decreased significantly. In 1996 only 1.7% of the registered voters in the United States used paper ballots, and were primarily used in small communities and rural areas [4]. These paper ballots are also used for absentee ballots, an issue that would be rendered obsolete with the implementation of ASVS. Clearly another issue with paper ballots is the necessity for manual counting of the ballots. In addition to these problems with paper ballots is the difficulty in processing the votes. "Paper ballots are easy to count if there are only a few offices

on the ballot, with only a few candidates per office, as is the norm in most parliamentary democracies. In our general elections, it is common to find well over 30 candidates on one ballot, divided between 8 to 15 offices, and this was the case even before the advent of ballot initiatives! An accurate hand count for ballots of this complexity is both difficult and time consuming."[5] It is clear that due to the size and complexity of our nation a new system would, by necessity, emerge.

The next system that was utilized after paper ballots was Lever Voting Machines that are still in widespread use despite the fact they have been out of production since 1982 [5]. These machines have an array of issues. They are expensive to move, store, and manufacture. They are difficult to test and therefore are not tested frequently. While they may be difficult to tamper by the public, the technician can easily manipulate them. "In effect, with lever voting machines, you put your trust in the technicians who maintain the machines, and if you want to rig an election, all you need to do is buy the services of enough of these technicians. This is quite feasible for a metropolitan political machine."[5]

The next step in voting systems occurred with the introduction of punch card ballots. This system employs a card or cards and a clipboard-sized device for recording the votes. Voters punch holes in the cards next to their desired candidate or voting choice. After the vote is cast the ballot is fed into a computer-tabulating device at the voting center. There are two common types of punchcard systems the "Votamatic" as well as the "Datavote". Though many jurisdictions are now switching from punchcard systems to more advanced DRE system, many counties are still utilizing this antiquated system as well. Approximately 37.3% of registered voters in

the United States still use a punchcard system.[6] While the counting issue has been fixed with this system via automation, there is still a level of fraud present as well as miscounting. An example of this would be in Concord NH in 1975 where due to an improper sizing of the punchcards a number of votes were not counted. [7] Unfortunately there are still many instances of this issue or similar issues that end up misrepresenting the interests of the voting populace. Significant errors were incurred with this system. For example, after a vote was cast, it essentially became anonymous and the voter had no idea whether or not their vote was being accounted for, or even if the system recognized the correct vote. Roy G. Saltzman in 1975 wrote an in depth paper for the National Bureau of Standards entitled "Effective Use of Technology in Vote Tallying" which identified some of the significant missteps in voting processes.[8]

As computers became cheaper and more reliant, a new system was created based on as much computerization of the process as possible. In the 2006 election it was estimated that over 66 million people would be voting on direct recording electronic systems (DRE) in 34% of the nation's counties.[9] The newest and by far the most popular voting system are DRE's In a DRE system the votes are cast on the machine and then stored directly on the machine's internal memory. Electronic voting systems allow those with disabilities to vote more accurately and privately thereby increasing the user base. Scientists have tested DREs though they are proprietary machines, and manufactures generally require confidentiality agreements with those who will acquire them.[10] Another problem with current DRE's is that the tech savvy youth understands the threat of malicious use and if the

security of a system is known to be strong the voter may be more likely to vote. One of the great things about the implementation of the Anonymous Secure Voting System(ASVS) is that the implementation for the most part will be relatively transparent. Currently there are some security vulnerabilities in modern DREs, these vulnerabilities are a result of the same feature, reliance on a computer for casting and recording of votes on a single local machine. There are several things such as malicious code or malware could do to a system like this that does not use remote vote checking, such as determine who an individual voted for. Another possibility is that malware could augment vote totals after they have been recorded but before they are downloaded for tallying.[10] Another issue with the DRE system is that a voter is unsure whether their vote has been recorded or not. The ASVS system while anonymous will be verifiable to anyone who wanted to check his or her own vote status.

One of the main purposes of the ASVS system would be to eliminate electronic voter fraud. To date 46 states have prosecuted or convicted cases of voter fraud along with 24 million voter registrations being invalid. There are over 1.8 million dead voters still eligible on the rolls across the country. More than 2.75 million Americans are registered to vote in more than one state. 12 Indiana counties have more registered voters than residents. [11]

4. Design Requirements:

This project in the end will be an anonymous secure electronic voting system that is easy to register for and easy to use. The project is broken down into two main sections, the hardware section and the software section, each with specific

requirements. Voter anonymity and single votes imply that the kiosk software and the server software must have a way to be interlocked or synchronized to avoid extra votes to be inserted. Voter anonymity implies that a reverse lookup of voter to vote should never be able to occur. Because voter participation is important, the system must be easy to use, meaning it must be very intuitive with little to no learning curve.

4.1. Hardware

The voter will initially register and be verified via the kiosk and their student ID; therefore, there must be a secure way of pairing people's student ID to their voter ID/fingerprint. Once a student has paired their ID number to their fingerprint the issue of selling votes is eliminated. This combination of identifications implies the kiosk must be capable of interpreting ID card swipes as well as fingerprint scans. Therefore, the processing element of the kiosk must be able to interface with multiple peripherals simultaneously. As per the design goals, the kiosk must be easy to use and understand. To ensure usability an easy to use touchscreen will be implemented. This is because a button and dial system is more likely to cause confusion whereas a touchscreen the selection process is significantly clearer. The demographic this voting system is targeting is incredibly familiar with smart phone and tablet use and would therefore seamlessly transition to a touchscreen voting format.

4.2. Software

The software behind the voting system must be secure and be designed such that vote anonymity can be ensured. It must be secure against double voting such

that a single person could not vote in the same election twice. Anonymity must be maintained, meaning votes must not be directly linked to a specific person. For this system to be secure it must be able to detect undesired changes to votes or vote tally's and then correct these errors. In order to do this the software should mimic the Bitcoin blockchain in a centralized manner. The blockchain is essentially a decentralized auto-verifying system on which the digital currency Bitcoin is based. All transactions are encrypted and stored on the blockchain and can be verifed with ease. From a technical standpoint, the Bitcoin ledger can be thought of as a state transition system, where there is a "state" consisting of the ownership status of all existing bitcoins and a "state transition function" that takes a state and a transaction and outputs a new state which is the result. This is how the ASVS will maintain its anonymity of votes while simultaneously eliminating the double voting issue. The ASVS will implement a similar type of authentication and verification system, where both the kiosk and the server are aware of the current state and all those preceding it.

4.3. Criterion Selection

The design criterion was selected with specific goals in mind. Due to the nature of this project and voting in general, security and anonymity is absolutely key. In addition the kiosk needs to be easy to use, simple to maintain and incredibly reliable. This system needed to be scalable to a very high degree, such that a very large number of kiosks could eventually be implemented. This system allows for that to occur with minimal changes to the design. The design had to be very user friendly in order to make voting easier and more efficient. The kiosk system is not a

new idea, though this implementation is, which is why mass acceptance of the system would be relatively easy and painless. In national voting there are a number of legal issues as well as standards and codes that must be adhered to, this is in part why Union College will be a great test bed for the functionality of the project and less about the viability of national implementation. Many things can go wrong in voting, just through natural means let alone through malicious intent which is why the aforementioned design goals were of the utmost importance with others such as code size, manufacturability were chosen to be less important.

The first functional design goal is the ease of use. This is very important since it is one of the major aspects that is driving the increase in voter turnout. The next functional design goal is the ability to register to vote, which utilizes both the fingerprint scanner as well as the ID card reader. Each user of the kiosk once registered to vote must of course be able to cast their vote as well as check that the vote they cast is the vote that was recorded. The server side must have the ability to generate elections. This means it must be able to create a new election and add the candidate choices. The server and kiosk must have the ability to tally the votes as well as compare the vote counts. Both the kiosk and the server must have the whole vote chain stored and must update it as soon as the vote has been verified as valid. The server and the kiosk must check the order of the vote chain, which is being passed from kiosk to server and vice versa continuously. The software for the kiosk must have the following functions seen in table 1.

Function/IO	Purpose
Card Reader	Get last name and first name, get id number, check against registered users, check against Union College id numbers
Fingerprint Scanner	Get N bytes of data and pair to ID number in a 2 row database list
Save Election Data	Locally save the current election data/vote chain
Post to server	Send individual vote to server
Append to vote chain	Insert the individual vote into the vote chain
Vote chain check	Check validity of vote chain from server and local storage

Table 1: Kiosk Software Functions

5. Design Alternatives

This section of the report will provide alternative methods for the completion of the design requirements and overall approach for this project.

5.1 Segmented approach

DRE's are a non-segmented voting system with only local storage. This local storage based system is where many vulnerabilities arise such as double voting issues as well as vote augmentation. This is why a segmented approach to an electronic voting system is necessary, it increases security measures and allows for scalability. DRE's only have local storage of the information, albeit in two different locals (local memory and a local tape), it allows for the alteration of votes or miscalculations to occur. Another reason to introduce a server side to this system is for the implementation of a universal voter registry, which will be maintained not only by the kiosk but by the server side as well. This design choice is clear for the requirements to be fully met and is a step forward from modern electronic voting systems. These advantages outweigh the disadvantages of increased security

measures, which must be taken in a variety of locations. For example, the system must secure the kiosk, the server, and the link between the kiosk and server.

5.2. Kiosk

The next set of alternative design choices arrive from the hardware side of the design. The hardware choices are primarily the choices of peripherals to be used in the final design. Each component has been chosen specifically to satisfy a design goal.

The fingerprint scanner is utilized in order to verify identities against the database and to eliminate vote selling thus the fingerprint scanner must be able to recognize and verify a person reliably. This means that only the person that is supposed to be voting is the one actually voting. Alternative methods could be other biometric verification processes such as facial recognition. While facial recognition is a good way of determining who someone is, it is a relatively new technology and is not as reliable as fingerprint scanning. Iris scanning is another form of biometric verification that was considered for this project. Both facial recognition as well as iris scanning are more secure since it is significantly more difficult to replicate someone's facial features or iris. A major problem with these technologies is that facial features and the iris both change significantly over time, which would force users to re-register in the voting registry. Since registration should only occur once for convenience and security these options are less logical than fingerprint scanners. In addition fingerprint scanners are significantly cheaper than facial recognition software and hardware. Other biometric verification techniques could be implemented supplementary to the fingerprint scanner if increased security was

desired. Finally, while it is possible to scam a fingerprint scanner this prototype allows for other methods to be implemented and the realistic possibility of a Union Student attempting to steal another students fingerprint along with their ID card is relatively low.

The next component of the kiosk is the touchscreen monitor which will be used as main input and output of the device for the voter. The touchscreen monitor is used in order to increase the ease of use of this kiosk-based system; instructions and completion of voting will be highly intuitive and thus decrease the time it takes to vote. Another possible option for this would be a simple monitor with buttons or dials on the side. This method is significantly less intuitive and is more prone to error. Button locations can sometimes be viewed to be in the wrong location due to the angle at which the voter is looking at the kiosk. The button dial system also forces the user to move their eyes off-screen, which could potentially make the user, miss important information being displayed on screen. Since touchscreens have both input and output directly on the screen the user never has to look away as long as they are voting.

The card reader is used in order to initially pair the voter's id number and identity to their fingerprint. There is no real alternative to this card reader, at least for Union College, the school's main form of identification are ID cards and therefore a card reader must be used for the initial pairing of ID to person. The final two pieces, Raspberry PI and storage are the backbone of the system, the Raspberry PI is a microcontroller that will be interacting with the aforementioned peripherals as well as dealing with the server side integration.

The processing unit of the kiosk will be a Raspberry PI microcontroller. Since the primary hardware interface is via USB instead of UART or GPIO the Raspberry PI was the most suitable choice. The Raspberry PI works out of the box with a unix command line and a linux GUI, it also has drivers pre-installed for a USB mouse as well as a USB keyboard which is not only utilized for the implementation but also the ID card reader uses a keyboard emulation system thereby allowing a single card swipe to be easily translated into ASCII. Alternatives include Arduino or a PIC microcontrollers, which are significantly more suited towards direct hardware serial interactions. It is significantly more difficult to design and implement a user interface on an Arduino or a PIC microcontroller since there is neither layer of hardware abstraction nor any operating system. They are also very difficult to incorporate video into thus making this a significant deterrent in selection of microcontrollers.

5.3 Server

The server side of the project is a centOS virtual server. A virtual server runs its own copy of an operating system, in this case centOS, allowing the user to have root access only to that instance of the server. For all intents and purposes it functions similar to a physical server though are easier to create, maintain and configure. This was chosen as per recommendation of Tom Yankulas, the Union College network administrator. The virtual server is running centOS6 which is the LTS distribution of centOS. There is a newer version, namely centOS7 which while newer does have some issues including a variety of deprecated functions such as ifconfig. The frontend of the system is running HTML with a PHP backbone with

phpMyAdmin databases that utilize mySQL for queries. These were chosen, as all three are the standard for web application such as this. In addition to the front/middle end of the server there will be the backend that runs on both the server and the kiosk. This will be written in Python. There are alternatives for this such as java, C++ etc. but python has good built in libraries for encryption (PGP), as well as can be interfaced with the php/html front/middle end.

6. Proposed Design

The proposed design is the initial design of the entire system. The design will be presented by initially showing a top-level design of the system followed by the kiosk design, and finally the server design.

6.1.Top level

The top-level design contains both the server side and the kiosk side. The overall diagram can be seen below in figure 1. The interaction between server and kiosk is via Internet connection. The kiosk is connected to the internet via Ethernet and the server is hosted on union college's campus.

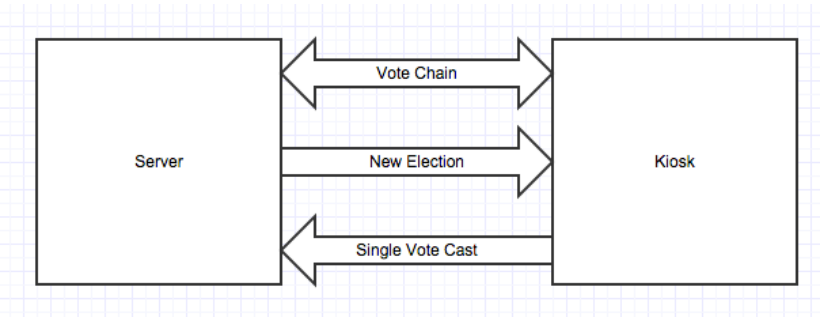


Figure 1: Top Level Diagram

The server and the kiosk communicate in three specific ways the first of which being that the server dictates which elections and candidates will be available

on the kiosk. Initially the types of elections will only be general elections i.e. elections that all of the student body will vote in. The Kiosk will post each vote to the server initially before adding it to the vote chain. The design of the vote chain can be seen below in figure 2.

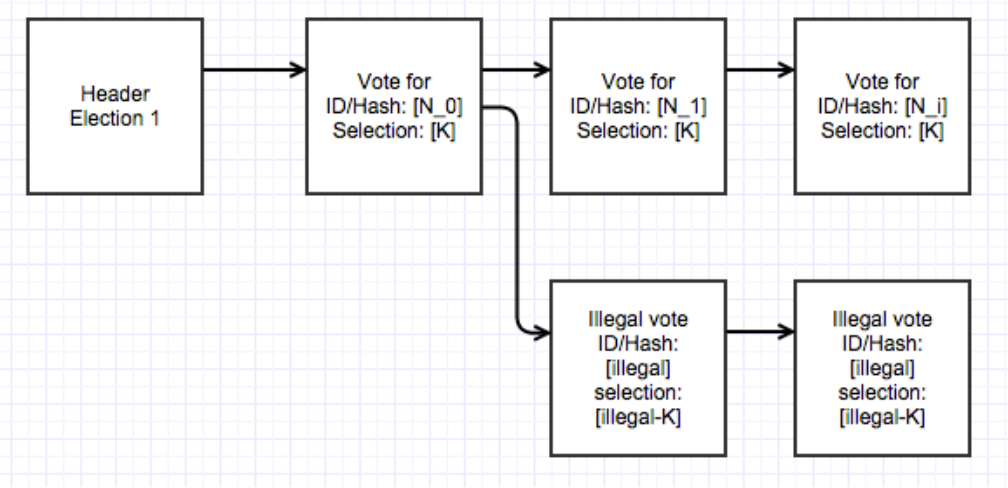


Figure 2: Vote Chain

Each election has a specific header block or identification block. Each election can be found by querying all election chains for the specific election header. Once a vote is cast the vote as specified previously is added to the internal chain on the server as well as the internal chain on the kiosk. The chain is then passed from kiosk to server at a specified rate. During each pass the kiosk and the server both check the validity of the vote chain. Each vote block has a parent and a child node, which are set when the vote is cast. When a vote is cast a new block is added to the chain containing the ID/Hash value of the voter as well as their candidate selection. An illegal vote chain, which can be seen in figure 3, would attempt to insert an illegal vote block or chain of blocks into the vote chain or at the end. In the event of the illegal insertion in the middle of the chain, the child of the inserted block would be

changed. Since the Kiosk and the Server both have the correct child for that vote block it would re-connect the correct parent to the correct child, essentially deleting the illegal vote blocks. In the event an illegal vote block or chain is added to the end of the chain the kiosk and the server both have the correct count of blocks as well as the final or most recently added block. If an illegal block were added at the end, on the next validity pass the final block would be re-updated to the correct block.

6.3. Kiosk-Top Level

The design of the hardware involves a number of peripherals and Input/output devices connected to the central microcontroller. The top-level diagram for the hardware can be seen below in figure 3.

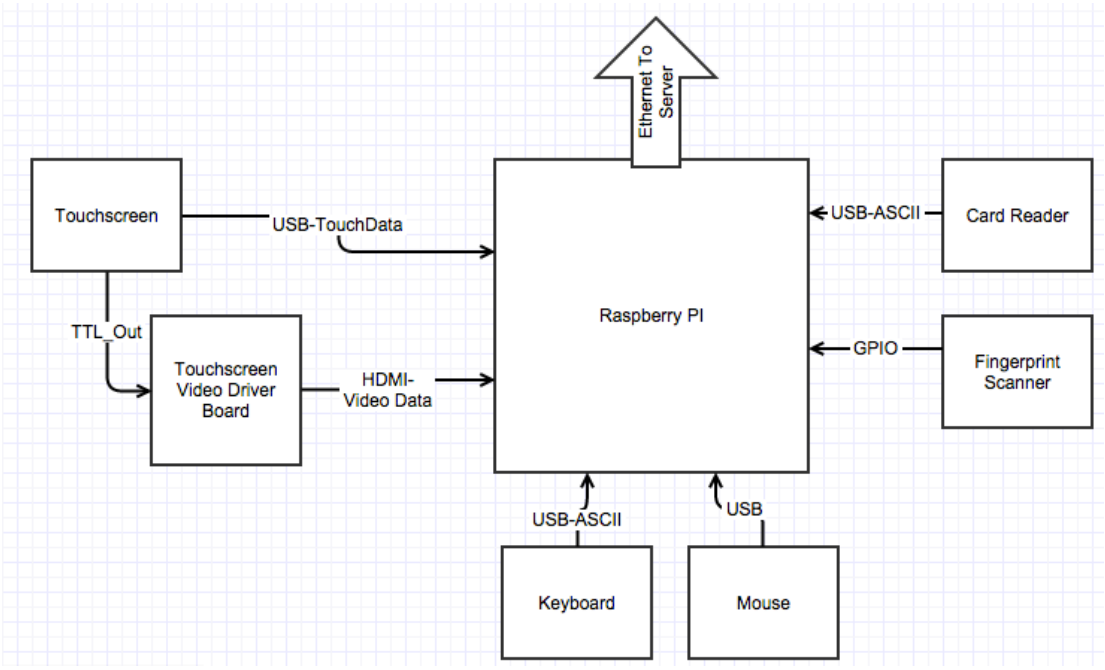


Figure 3: Hardware Top Level

The microcontroller is a Raspberry PI B+, which has four USB ports as well as an Ethernet port. The Ethernet port is how the raspberry pi will connect to the

server using standard protocols. The Raspberry PI is running a linux distribution called Raspbian, which has certain drivers already built in, such as keyboard and mouse.

The first two IO devices are the keyboard and mouse, which are for the kiosk technician and for initial implementation. The card reader is also a USB device which uses keyboard emulation to send data to the Raspberry PI, this means it encodes the information in the same way that a keyboard does and therefore the standard keyboard driver will work for the card reader as well. The only thing that needs to be done when accepting card reader data is to ensure the correct USB bus is selected when receiving from the card reader.

The next IO peripheral is the fingerprint scanner, which connects to the Raspberry PI using the general-purpose IO ports (GPIO). These ports are pins on the Raspberry PI that are slightly more difficult to deal with since there is already a software layer on the microcontroller (raspbian linux distro).

The touchscreen connects to the Raspberry PI in two different ways. The first is the video. The screen has a Transistor–transistor logic (TTL) connection to the video driver board, which then connects to the Raspberry PI via a HDMI cord. The Raspberry PI by default displays to a HDMI screen and thus there are default drivers that come with the linux OS. The second way the touchscreen connects to the Raspberry PI is via a four pin TTL connection on the screen, which has a connector to a male USB which is then connected to the Raspberry PI via the third USB port. The touch data is encoded in such a way that the kernal for the linux OS must be updated to emulate mouse click commands.

6.4. Kiosk-user

A state machine describing the actions of the kiosk and the user interface can be seen below in figure 4.

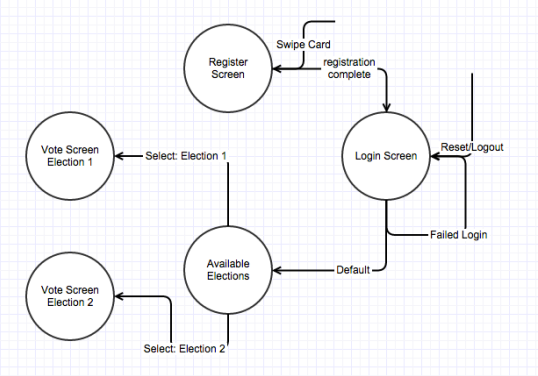


Figure 4: Kiosk State Machine

The user begins at the login screen, which prompts the user to either swipe, their card in order to register, which would take them to the registration screen. This screen is where the ID card and fingerprint pairing takes place. Once the id card has been paired to the fingerprint the kiosk stores this information and sends it to the server. The kiosk checks to make sure neither the fingerprint nor the id card has been previously registered, if they have been they are returned to the login screen. The login screen then upon receiving a valid fingerprint scan defaults to a screen that displays the current elections being held. In figure 5 they are election 1 and election 2. The user then can select which election they would like to vote in, which brings them to the voting screen. This is where the actual vote is cast. Once selected the system defaults back to the available elections page. This available elections page still contains all possible elections but if the user selects an election they have already voted in, it simply displays what their vote was. This allows the voter to

check whether or not their cast vote was actually the vote that was tallied. The user cannot change their vote once it has been submitted. Basic templates for the kiosk pages can be seen in appendix Kiosk Screen Design.

6.5. Kiosk-Backend Functions

The kiosk, in addition to the user interface will have a back end that deals with the election data. The kiosk will maintain a database of ID-Fingerprint hash values, which can be queried to determine whether or not a user has already registered their ID card or an individual has already registered their fingerprint. The Kiosk will be able to create and append new vote blocks as per the description in vote chain; in addition the kiosk will be able to verify all the current election vote chains.

6.6. Server

The server is where elections can be created and posted to the kiosk. An administrator may log in, view and export election data as well as create and post new elections. This frontend will be designed in HTML with data populated by PHP. The databases for all election header blocks are also stored in these php databases along with their final vote tally. The basic screen layouts for the server side can be seen in the Appendix under Server Screenshots. The backend is written in python which is the actual vote chain data. This is also where the vote chain will be counted in addition to on the kiosk. The vote chain is simply iterated through adding a value to each candidate's total tally. The output of the vote tally is the winner with number of votes and the losers with their number of votes. This information is then uploaded into the MySQL database. This abstraction and redundancy significantly

increases security by not allowing the server frontend to actually tally or alter the vote chain. This is in the event of a recount the vote chain is ensured to be unaltered.

7. Final Design and Implementation

This section of the report will document the final design and its implementation. It will begin with an overview of the top level and the modifications from the original design. Following this section will be the kiosk design and implementation and finally the server.

7.1 Top Level

The top level of the project varied slightly from the original design. The new diagram can be seen below in figure 5.

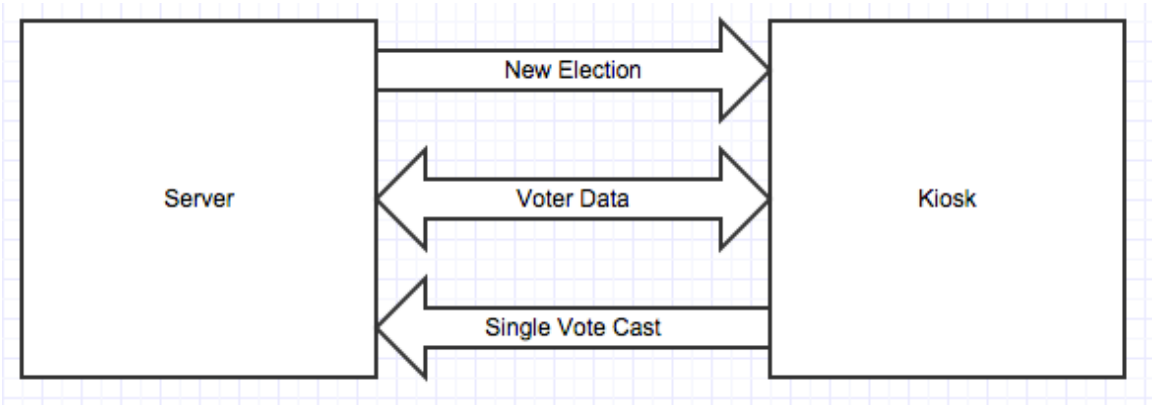


Figure 5: Top Level Design

The entire system works in stages beginning with the server. The sever initiates an election as seen from the top arrow in figure 5. The server posts an election to the kiosk, how this is done will be further discussed in section 7.3. The second multi-directional arrow is voter data being relayed from the server to the kiosk and visa versa. Voter data consists of attributes specific to a voter, excluding a vote. Voter data is primarily a voter's ID number from their Union College ID card as

well as their serialized fingerprint data, more about this will be in section 7.2. Finally, the kiosk posts each vote to the server incrementing the candidates tally by a single vote.

7.2 Kiosk Design and Implementation

This section will detail the design and implementation of the kiosk. Initially the overview of the kiosk's hardware will be discussed and how it interfaces with the main processing unit. Next each page of the user interface utilizes different hardware aspects which will be discussed starting with root UI , continuing with the welcome page, the registration page, the verification/login page, the elections page and finally the candidates and vote page.

7.2.1 Kiosk Hardware

The Kiosk performs a multitude of functions utilizing a variety of hardware. The section will describe the hardware and how it interfaces with the rest of the system. Starting with the main processing piece of the project, the Raspberry PI, then continuing to the card reader, then the touchscreen and finally the fingerprint scanner.

The first and most important piece of the hardware is the Raspberry PI. The Raspberry PI is the microprocessor that runs the kiosk user interface. In addition the Raspberry PI deals with all external interactions to the kiosk. The Raspberry PI works relatively well out of the box. A micro SD card is inserted for the main device memory which is where the boot data and the main storage is located in separate partitions. An image of the Raspberry PI can be seen below in figure 6.

Figure 6: Raspberry PI

The above photo shows the raspberry pi along with all possible input and output locations. On the bottom right of the Raspberry Pi is where the four USB ports are located. To the left of the USB ports is the Ethernet port, which allows for Internet access. Next going along the outside of the Raspberry PI is the HDMI, which for this project connected to a video driver board for the touchscreen, which will be discussed later in this section. The Raspberry PI uses a Linux distribution called Raspbian which is based off a common Debian distribution.

The next and simplest piece of hardware is the USB Card reader, which can be seen below in figure 8. The card reader reads the magnetic strip off a students Union College ID card which contains some encrypted data and the unencrypted ID number. The card reader seen in figure 7 is the same MagTek card reader used in the final implementation. The Card reader connects to the Raspberry PI via a USB port and interacts rather simply. The card reader simply mimics a USB keyboard to provide input to the system.

Figure 7: Card Reader

Because the card reader mimics a keyboard security protocols were put in place ensuring non-valid id numbers were not allowed. This was a simple check on length though more intricate validation techniques could easily be implemented. Due to the nature of the Linux distribution on the Raspberry PI the terminal window always has default focus over the python UI script that is running. Therefore in order to use a keyboard emulating card reader the focus of all keyboard and mouse IO was forced to the UI.

The next piece of hardware is the fingerprint scanner. The fingerprint scanner connects directly to the Raspberry PI via a USB connection. The final choice for the fingerprint scanner is a Microsoft fingerprint scanner, which can be seen below in figure 8.

Figure 8: Fingerprint Scanner

In order for the fingerprint scanner to work on the raspberry PI certain drivers were needed. The driver that was used is called libfprint. This library is written in C and depends on libusb for USB communication and glib. The driver Supports imaging and downloading of live fingerprint scans from the device, as well as image processing for fingerprint verification. Finally this driver supports enrollment and verification which is essentially enrolling a print from a known user, and then later comparing a live scan to the enrolled print [12]. The Microsoft fingerprint scanner is similar enough to another device that the driver supports called the Digital Persona U.are.U 4000/4000B/4500.

This driver is written in C and the secure electronic voting system is written primarily in python. Therefore in order to utilize this driver a python wrapper had to be generated. In order to do this code was found on github[13] that would auto-generate this module using a python file and a swig file. “Swig is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages. SWIG is used with different types of target

languages including common scripting languages such as Javascript, Perl, PHP, Python, Tcl and Ruby."[14]. Swig finds the libfprint install location in the boot partition of the flash drive, and using the python file, the .i (swig) file and the c file from libfprint and generates a usable python module.

The final hardware piece implemented was the touchscreen. The touchscreen can be seen below in figure 9.

Figure 9: Touchscreen monitor and Driver Boards

As seen in figure 9, there are multiple pieces to the touchscreen. The touchscreen connects to the video driver board via TTL connection. The Driver board connects to the Raspberry PI via HDMI. In order to use the touch component there is a 3 prong flat wire output from the touchscreen which is fed into a USB converter. This takes voltages from the thin film transistor layer on top of the screen and converts them into a usable USB format. The Raspberry PI does not by default allow for touchscreen connections to be made. Therefore a rebuild of the Raspbian kernel had to be done. In order to do this the USB device must be located via the command:

pi@raspberrypi ~ $ lsusb

This produces a list of all USB connected devices including:

Bus 001 Device 005: ID 0eef:0001 D-WAV Scientific Co., Ltd eGalax TouchScreen

Next the kernel must be built according to these specifications. A full tutorial on how to rebuild the kernel for this specific monitor and touchscreen can be found in the works cited [15]. In order to calibrate the touchscreen a module called xinput_calibrator was used.

7.2.2 Kiosk Software

The software of the Kiosk is based around the user interface, which in turn executes backend functions. The overarching system diagram can be seen below in figure 10. The kiosk is programmed in python utilizing a large number of dependencies which can be seen in the Required Dependencies section of the Appendix. To program the user interface a python UI module named Tkinter was utilized. This deals with the display of pages, buttons and functions that pages must execute. Each page of the UI does something different and will be discussed later in this section.

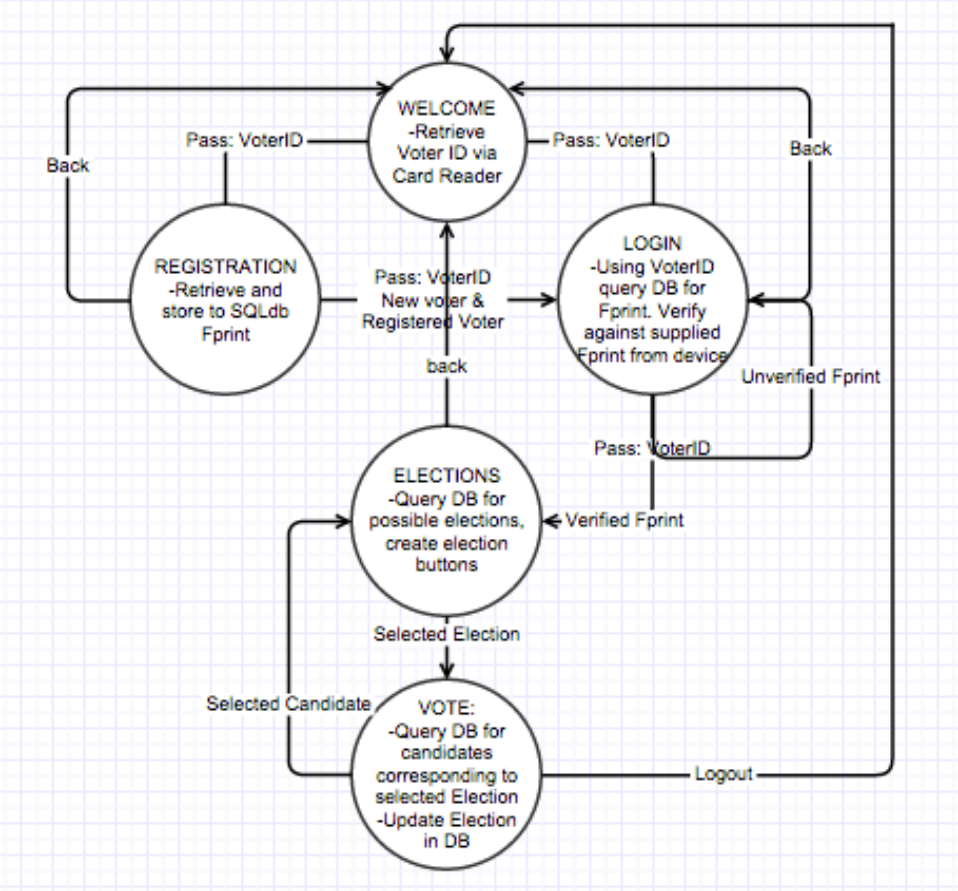


Figure 10: Kiosk Flow Diagram

This diagram shows the users trajectory through the different pages on the UI and what functions each page serves. The Welcome, Registration, Login, Election and Vote pages are instantiated once and immediately upon use of the main function. The main contains a mainview, which can be seen below in figure 11.

```
class MainView(tk.Frame):
    def __init__(self,*args,**kwargs):
        tk.Frame.__init__(self,*args,**kwargs)
        buttonframe = tk.Frame(self, background="white")
        container = tk.Frame(self, background="white")

        im = Image.open("Background.png")
        bg_img=ImageTk.PhotoImage(im)

        bgl=tk.Label(self,image=bg_img)
        self.image= bg_img
        bgl.place(in_=buttonframe,x=950,y=10)
```

Figure 11: MainView Init

The mainview is instantiated using a tkinter frame, which is essentially the base container for all labels and buttons in the user interface. There are two other containers that are created, the container, which will hold all pages and labels, and the buttonframe which sits on top of the container frame. This buttonframe is where all the buttons for each page are displayed. Also in the initialization of the mainview each page is instantiated, an example of which can be seen in figure 12.

```
self.idN = idNum()
voteP = votePage(buttonframe)
voteP.place(in_=container, x=0, y=0, relwidth=1, relheight=1)

electionP = electionPage(buttonframe,voteP,container,self.idN)
electionP.place(in_=container, x=0, y=0, relwidth=1, relheight=1)


self.loginP = loginPage(buttonframe,electionP,self.idN)
self.loginP.place(in_=container, x=0, y=0, relwidth=1, relheight=1)

self.registerP = registerPage(buttonframe,self.loginP,self.idN)
self.registerP.place(in_=container, x=0, y=0, relwidth=1, relheight=1)

self.welcomeP = welcomePage(buttonframe,self.registerP,self.loginP,self.idN)
self.welcomeP.place(in_=container, x=0, y=0, relwidth=1, relheight=1)
electionP.welcomeP = self.welcomeP

self.welcomeP.show()
```

Figure 12: Sample page instantiations

The order of instantiations is important because each page depends upon the previous page. Therefore the object must be passed to the parent page for the parents initialization. For example, in figure13 the welcome page is the parent of the register page and the login page and thus must be passed into the instantiation. The final piece of the initialization is to show the initial page, which is the welcome page. In order to run the module we must run the root tkinter module as seen in figure 13.

```
if __name__=="__main__":
    root =  tk.Tk()
    main = MainView(root)
    main.pack(side="top", fill="both", expand=True)
    root.wm_geometry("1280x720")
    root.bind('<KeyPress>', main.welcomeP.onKeyPress)
    root.overrideredirect(1)
    root.focus_force()
    root.mainloop()
```

Figure 13: Running the module

This is where the module is actually run. Tkinter creates a continuous loop where all pages are running until the application is closed. Within this run loop certain things must occur. For example in figure 13 the root.bind command binds all key presses, those from the keyboard emulating card reader to execute the onKeyPress function within the welcome page. In addition the root.ovverridedirect function forces the application to take the full screen, not only over any other applications but over the operating system itself. This ensures the OS cannot be accessed when the application is running. The application is now can be run which as mentioned previously shows the welcome screen.

The welcome screen serves two main functions. The first function is to retrieve the ID card number from the user. At any time on the welcome page the user can scan their ID card in the cardreader and the welcomepage will parse the supplied characters to only retrieve the ID number. This can be seen below in figure 14.

```
def onKeyPress(self,event):
    self.id_num.append(event.char)

def getRealID(self):
    id_num_real = ""
    num = []
    idlen = len(self.id_num)
    num = self.id_num[idlen-12:idlen-4]
    for n in num:
        id_num_real+=n
    print(id_num_real)
    self.IdN.setID(id_num_real)
```

Figure 14: ID Retrieval and Setting

The onKeyPress function is called anytime a key is pressed on in the welcome page section. This takes all the read characters and puts them in a temporary array. The getRealID function takes the temporary array parses out the ID number, puts it into a string, and sets the id number in the id number python module to be the supplied string. The id number module is used so that the id number supplied can be used not only in the registration process but also in the verification process. The welcome page also displays two navigation buttons allowing the user to register if they have not done so yet, or login.

The main purpose of the registration page is to retrieve the fingerprint from the fingerprint scanner, serialize the data, and post it to the MySQL database on the server. The finger registration function can be seen below in figure 15.

```
def registerFinger(self):
    device = pyfprint.discover_devices()[0]
    device.open()
    enrollOutput= device.enroll_finger()
fp=enrollOutput[0]
    data=fp.get_data()
print(data)
device.close()
    self.databaseAdd(data)
self.shownum.pack_forget()
    self.fingerScan.pack_forget()
    self.nextpage.showAll()
```

Figure 15: Register Finger

The register finger function finds the device and takes the first available one, in all cases it is the Microsoft fingerprint scanner. Then it must open the device and uses the pyfprint wrapper to enroll the finger. The enroll output contains the image and the fingerprint object then the function retrieves the serialized fingerprint data from the fingerprint object. Once this is complete the fingerpint data along with the id number as the primary key is sent to the database. Finally since the registration process is complete the fingerscan button must be removed from the page and the login page is brought up.

The login page is the other point in which the fingerpint scanner is utilized. The login page consists of a single button, which the user selects once they are ready to scan their finger. Upon selecting the login page the database retrieves the fingerprint data corresponding to the supplied ID card number as seen below in figure 16.

```
cursor = db.cursor()
sql = "SELECT * FROM  `VotingSystem`.`Voter` WHERE id=%s ;"

try:
    cursor.execute(sql,(idnumber))
    results=cursor.fetchall()
fpDataList=results[0]
fpData=fpDataList[1]
return(fpData)
except:
    db.rollback()
db.close()
```

Figure 16: Database Retrieve

This is part of the function that retrieves the serialized data from the server. It fetches the data in string form corresponding to the supplied id number. After this data is retrieved the system can now verify the fingerprint. This can be seen below in figure 17.

```
def login(self,nextPage):
isVerified=self.verify()
if(isVerified==True):
    self.nextP(nextPage)
else:
    print("not verified")

def verify(self):
    device = pyfprint.discover_devices()[0]
    device.open()
dataRetrieved=self.databaseRetrieve()
fpDB=pyfprint.Fprint(serial_data=dataRetrieved)
verified= device.verify_finger(fpDB)
verified=verified[0]
device.close()
return verified
```

Figure 17: Verification and Login

This process opens the fingerprint scanner from the list of discovered devices. Then it gets the data from the code shown in figure 17. This data is used to instantiate a new fingerprint object which is then passed into the driver command verify_finger which takes a supplied fingerprint from the device and compares it with the passed fingerprint and returns true if there is a close enough match. The resulting value is then passed into the login function which then depending on

whether or not there was a match will remove the login screen and show the elections page screen.

The election page and vote page are similar in function. The election page retrieves the elections that the voter may vote in and then generates buttons for those elections. An election button is then selected which directs the user to the vote page. The vote page then pulls the candidates from the corresponding election and displays the corresponding candidate buttons. Once the candidate is selected the vote tally for that candidate is updated in the server and the election page is then re-displayed. If there are no more elections to be displayed the application is redirected to the welcome screen.

7.3 Server Design and Implementation

The server consists of two main aspects, the front end and the back end. The front-end entry point is a simple login page which can be seen below in figure 18.

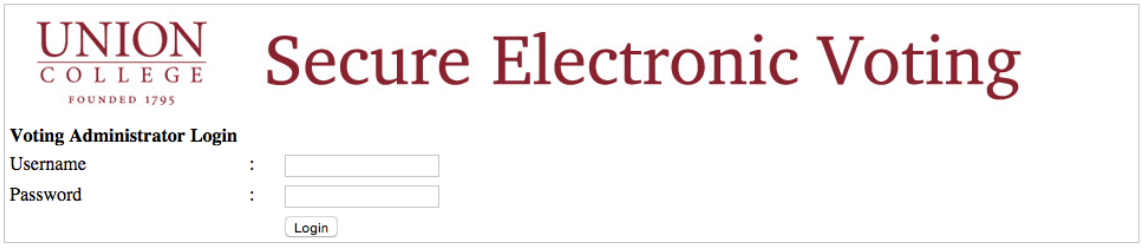


Figure 18: Server Login

This home page is located at goodricn-vm.union.edu. This page takes the supplied username and password checks it against the MySQL database backend using PHP. If a correct username and password are supplied the home page is displayed. The home page consists of two links one to the election creator and one to the election results manager. The election creator is a simple form that allows the voting

administrator to create a new election with candidates. The election results page simply displays all candidates for all elections. A sample of this can be seen below in figure 19.

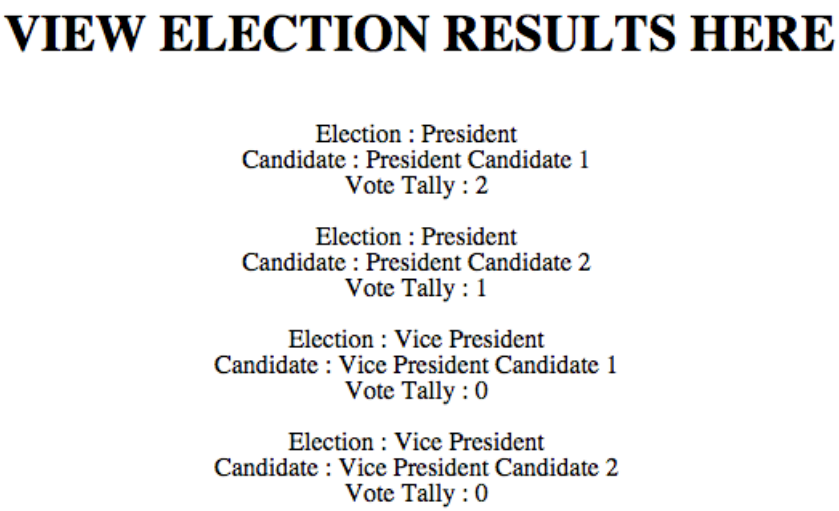


Figure 19: Election Results Display

The election results are retrieved from the backend MySQL database.

The backend MySql Database can either be accessed from a user interface called PHP Myadmin located at goodricn-vm.union.edu/phpmyadmin. The MySQL database consists of two main tables. The first is the voter database which contains a list of all voter's ID numbers as a primary key paired with the serialized fingerprint data and finally the possible elections the voter is allowed to vote in. The Second main table is the elections table which consists of the election and candidate as the primary key and the vote tally for that specific candidate.

8. Performance Estimates and Results

The performance estimates consist of a selection of parameters as follows: ease of use, speed, level of security and anonymity. Each criterion is specific to this project.

The first two criterion, ease of use and speed are directly linked. Ease of use is determined by the difficulty in the users understanding the voting process. Results varied depending upon the engagement of the user. If the on screen instructions were not thoroughly read the user sometimes found themselves attempting to log in with the incorrect id number therefore fetching the incorrect fingerprint to verify. The server side is incredibly easy to use as there are only two options on the front end and one is static. Speed of the system is not an issue except on startup, it can take anywhere form 1-3 seconds to boot. The only other bottleneck for the speed of the system is in the registration and verification process. The fingerprint scanner that was used in this project is not an industrial grade fingerprint scanner and therefore can take up to 15 seconds to register a finger. For verification the process can take up to 30 seconds and approximately 50% of the time the user must re-scan their finger for verification purposes. It was found that these numbers varied greatly depending upon the person who was registering or verifying their finger, as it turns out some people do not have easily identifiable ridges on their finger making it difficult for the fingerprint scanner to get usable data. Unfortunately, the vendor does not specify the registration or verification time, these metrics could be found through other user reviews, but it makes scanner selection more difficult.

The next performance metric is level of security. This is a multi-faceted problem, the security of the server, kiosk and kiosk-database connection. The server is relatively secure as there is a login page that protects against SQL injection. The database itself has standard MySQL security in place but encrypting the data within the server could increase security here. The kiosk itself is relatively secure as it is in an enclosure and the only inputs to the system are the touchscreen, fingerprint scanner and card reader. The only possible security fault would be the card reader that emulates a keyboard but there are checks in place to ensure validity of ID numbers. Finally the kiosk to database connection is not nearly secure enough. The MySQL database is a pseudo secure connection but could use an open SSL connection to connect thereby encrypting the connection between kiosk and server.

The next performance criterion is anonymity. The implemented system is pseudo-anonymous. There is no way to effectively see who an individual has voted for but there is a link between Union College ID numbers and the stored fingerprint. To fix this the verification process could be altered. Instead of supplying the fingerprint scanner with a fingerprint to match, the system could read in a fingerprint and verify it against all stored fingerprints, without having to identify them by ID number. Because ID numbers are stored with the corresponding fingerprint, if the database was corrupted a malicious user could then reproduce the fingerprints of each person. The link to the ID number poses an anonymity threat. Due to the limiting factors of the device driver the current implementation was required.

9. Production Schedule

The project was effectively completed in eight weeks. A flow diagram of work and time frame is outlined below in figure 20.

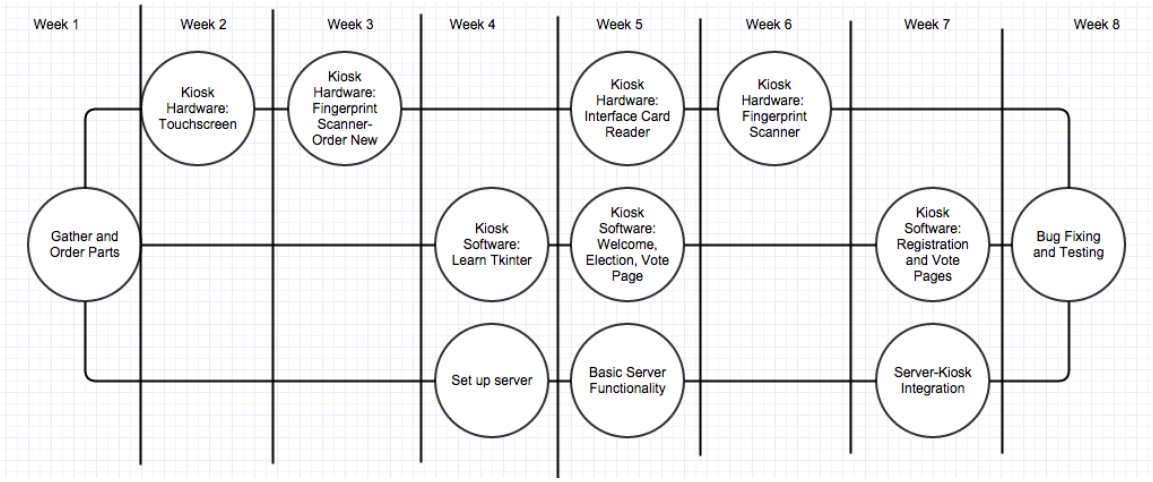


Figure 20: Work Flow

The project was completed by simultaneously working on the Kiosk Hardware, Software and the Server. There were two main things that slowed down the production of this prototype. In week 3 a new fingerprint scanner had to be ordered as the implementation changed from using a GPIO based scanner to a USB based scanner. In addition, the card reader was purchased from an unreliable vendor and took a long time to arrive. Working on different parts of the project simultaneously was a good strategy as they all came together in week 7 for integration.

10. Cost Analysis

The initial cost comparisons made in this project can be seen below in figure 21.

Card Reader	Website	Voltage Required	Message Format	Interface	Price		
MagTek	http://www.	5v from usb	ASCII	USB	$57.00		
MagTek 2	http://www.	5v from usb	USB-Keyboard Emulation	USB	$48.00		
Card Device	http://www.	24VDC	USB-Keyboard Emulation	USB/Serial	$135.00		
Fingerprint Scanner	**Website**	**Voltage**	**Interface**	**Baud Rate**	**Storage Capacity**	**Price**	
Adafruit	https://www	3.6-6V	TTL Serial	57600	162	$50.00	
Sparkfun	https://www	3.3-6V	UART	9600	200	$50.00	
Touchscreen	**Website**	**Size**	**Voltage**	**Touch Inputs**	**Video Inputs**	**Backlight**	**Price**
ByByte	http://www.	7"	12V	4 wire resisitve touch, USB	VGA, DVI, HDMI, Composite	LED	$170
sainSmart	http://www.	7"	12V	USB	HDMI, 2AV, VGA	LCD	$50
TonyTech	http://www.	7"	12V	USB, UART	HDMI, VGA, AV	LCD	$70
adafruit	https://www	2.8"	12V	USB	HDMI, VGA	LCD	$35
adafruit 2	https://www	5"	12V	USB	HDMI, VGA, NTSC, PAL	LCD	$65

Figure 21: Initial Peripheral Comparisons

The selected touchscreen was in the mid range of possible touchscreen options at a total cost of $55. The Card reader magtek2 was also a relatively cheap option. The initial fingerprint scanner was a price of $50 but the Microsoft USB fingerprint scanner that was implemented was $85. Finally the enclosure for the final kiosk was 3d printed at a high cost of $120 making it by far the most expensive component in the project. The final cost of the project can be seen below in Table 2.

Item	Cost
USB Fingerprint Scanner	$85
GPIO Fingerprint Scanner(NOT USED)	$50
USB Card Reader-Keyboard Emulator	$55
Raspberry PI	$50
3D printed Enclosure	$120
TOTAL	$360

Table 2: Total Cost

While the total was larger than initially projected, it is still a reasonable amount for prototyping a system such as this.

11. Users Manual

The system that has been the product of this senior project is relatively easy to operate and maintain. Operation begins by plugging in three cords from the kiosk. The first is the power cord to the processing unit. The second is the power cord to the touchscreen display. The final cord is the Ethernet cable which must be plugged in to a Union College Ethernet port. The current system will not function properly outside of Union College due to the inability to access the MySQL database off-Campus. To initiate the kiosk the power button on the touchscreen must be turned on. Then the display environment on the processing unit must be started. To do this simply type using the kiosk technician's keyboard: startx. After the display environment has been started, navigate /EVS/SrProg/GUI/ then the main script can be started either in the terminal window by typing python main.py or by double clicking the main.py file in the file browser. Once the system has been started there is no need for maintenance or changes to be made to the kiosk unless there is a system crash.

The server is easily operated simply by going to the address in the browser of your choice: goodricn-vm.union.edu and supplying the login credentials. The default login is: admin, and the default password is: password. This brings you to the main site where elections can be made and election results reviewed. The MySQL database must be maintained properly. The system technician can access the MySQL database either from the url: goodricn-vm.union.edu/phpmyadmin with the login credentials: root, and password: l33tsuperman. Similarly the MySQL database can be accessed via any terminal that has ssh ability. Ssh into goodricn-vm.union.edu and then open MySQL by typing: mysql –u root –p and supplying the

MySQL password. Web server files are located in goodricnn-vm.union.edu's file system at the point /var/www/html. If a user would like to reregister a finger their information must be deleted from the database in order for them to reregister.

12. Conclusion

The Secure Electronic Voting System is an important step towards efficient voter identification and election data gathering. Current methods such as paper ballots or Direct Electronic Recording devices have two main flaws. The first is the potential for fraud and the second is the inefficiency of voter identification and data acquisition.

Fraud is effectively eliminated by creating a ID number to fingerprint registry where a user can only register their ID number once with a single fingerprint. In addition to reducing fraud the ID number to fingerprint registry also makes voter identification significantly more efficient and secure. The problem of data acquisition has been dealt with in this system by introducing a main server that contains all voter information as well as election information. This means that determining the winner of an election is quick and easy due to all election data being at one central access point.

The final performance of this system is good but it is not without its problems. Total time at the kiosk can be anywhere from 30 seconds to 3 minutes. This is primarily due to the length of time it takes for registration and verification to occur. This problem is generate by the latency of the fingerprint scanner, which takes time to initially register the voter, and takes substantial time to verify the voter. In addition voter verification sometimes has to be run multiple times due to

the fingerprint scanner getting a bad read on the voters finger. Another potential performance issue is the security of the system. The kiosk itself is relatively secure, as is the server, but more measures for the kiosk to server link could be implemented.

If this project was to be optimized for real production and implementation into real voting some future work would be required. A secure socket layer connection could be introduced between the MySQL database and the kiosk to ensure the connection is encrypted. Another way to secure the kiosk to server connection would be to implement the original vote chain idea. This would significantly increase the security of the connection while simultaneously allowing for mass scalability. For scalability a puppeteer system could be implemented where the puppeteer is the server and the puppet is each kiosk. More research on the specifics of the fingerprint scanner should also be done in order to reduce the registration and verification time.

The final result was a relatively secure, pseudo-anonymous electronic voting system. The main problems inherent in voting and specifically electronic voting have been dealt with in an interesting and creative way.

Works Cited

[1]
'What Affects Voter Turnout Rates', *FairVote*. [Online]. Available: http://www.fairvote.org/research-and-analysis/voter-turnout/what-affects-voter-turnout-rates/. [Accessed: 22-Nov-2014].

[2]
'Universal Voter Registration', *FairVote*. [Online]. Available: http://www.fairvote.org/reforms/universal-voter-registration/. [Accessed: 22-Nov-2014].

[3]
H. Cooper, 'Voter Fraud is Real: Why the Voting Rights Act Should Be Used to Fight Election Fraud', *National Policy Analysis*. [Online]. Available: http://www.nationalcenter.org/NPA636.html. [Accessed: 22-Nov-2014].

[4]
'Paper Ballots', *Federal Election Commision*. [Online]. Available: http://www.fec.gov/pages/paper.htm. [Accessed: 22-Nov-2014].

[5]
D. Jones, 'Problems with Voting Systems and the Applicable Standards'. 22-May-2001.

[6]
'Punchcards', *Federal Election Comission*. [Online]. Available: http://www.fec.gov/pages/punchrd.htm. [Accessed: 22-Nov-2014].

[7]
C. Arnst, 'IBM Punchards Blamed For Primary Miscount', *ComputerWorld*, 21-Jun-1976.

[8]
R. Saltman, 'Effective Use of Computing Technology in Vote-Tallying', Mar. 1975.

[9]
S. Everett, K. Greene, M. Byrne, D. Wallach, K. Derr, D. Sandler, and T. Torous, 'Electronic voting machines versus traditional methods',*Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, 2008.

[10]
'Voter Fraud Instances 2012 National Election'. [Online]. Available: http://giac2002.org/voter_fraud_instances_2012.htm. [Accessed: 22-Nov-2014].

[11]
E. Fischer and K. Coleman, 'The Direct Recording Electronic Voting Machine (DRE) Controversy: FAQs and Misperceptions', Dec. 2005.

[12] Freedesktop.org, 'libfprint', 2013. [Online]. Available: http://www.freedesktop.org/wiki/Software/fprint/libfprint/.

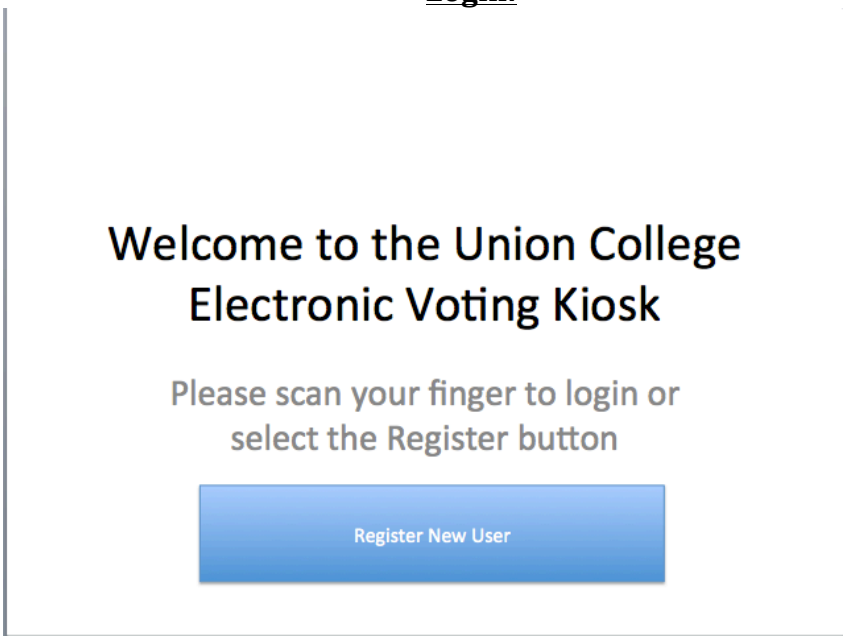
[13]L. Sandström, 'luksan/pyfprint', *GitHub*, 2008. [Online]. Available: https://github.com/luksan/pyfprint/tree/master/pyfprint. [Accessed: 12- Mar- 2015].

[14] Swig.org, 'Simplified Wrapper and Interface Generator', 2015. [Online]. Available: http://swig.org/. [Accessed: 12- Mar- 2015].

[15]A. Istodorescu, 'Engineering(DIY): Adding 7inch display with touchscreen to Raspberry PI', *Engineering-diy.blogspot.ca*, 2013. [Online]. Available: http://engineering-diy.blogspot.ca/2013/01/adding-7inch-display-with-touchscreen.html. [Accessed: 12- Mar- 2015].

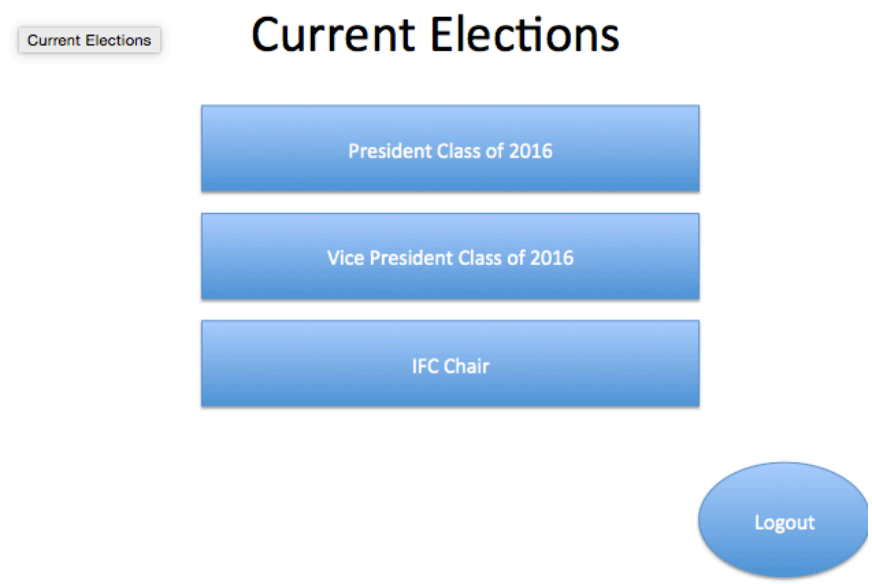
Kiosk Screen Design

Login:

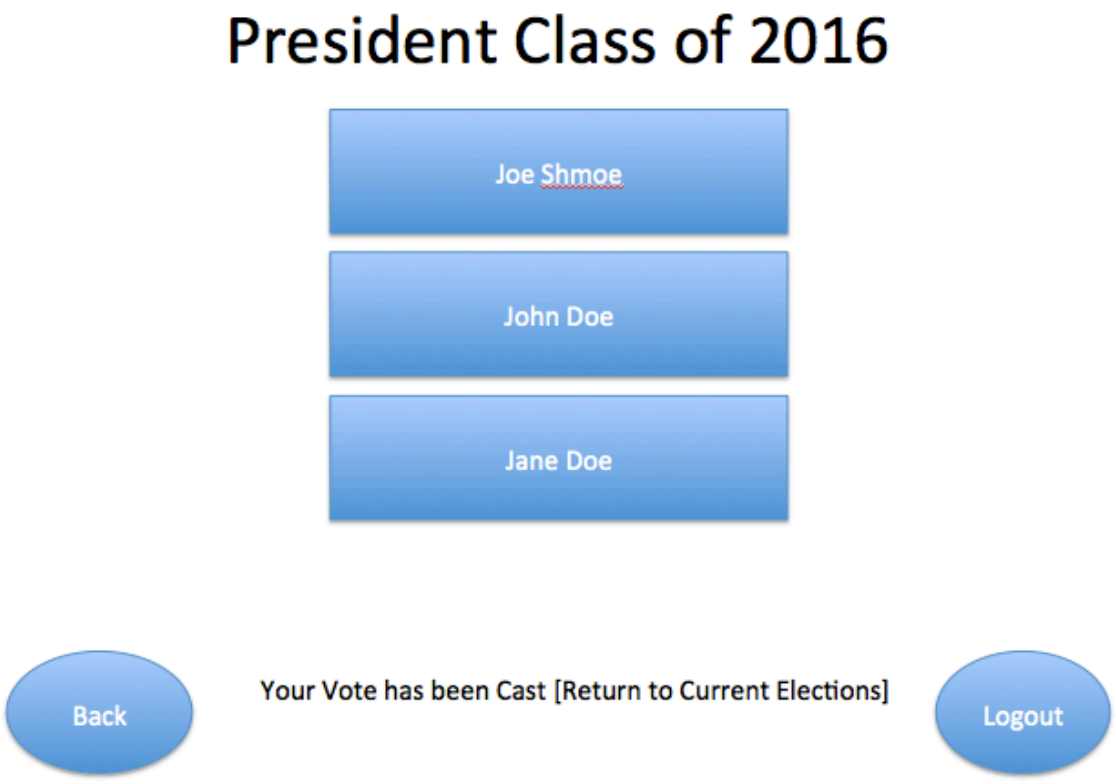


Registration:

Current Elections:

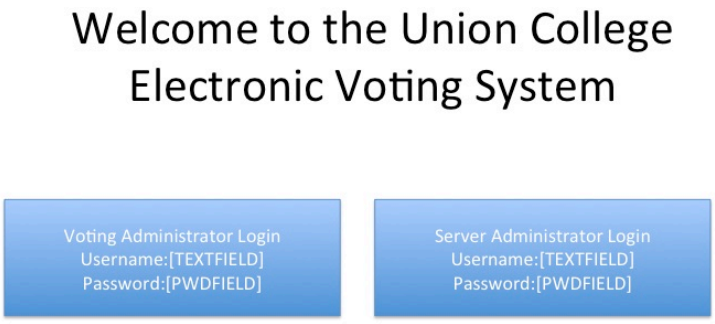


Voting Screen:



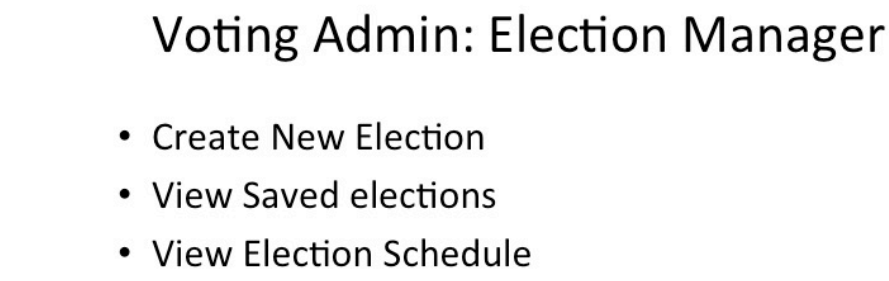
Server Screen Shots

Login:

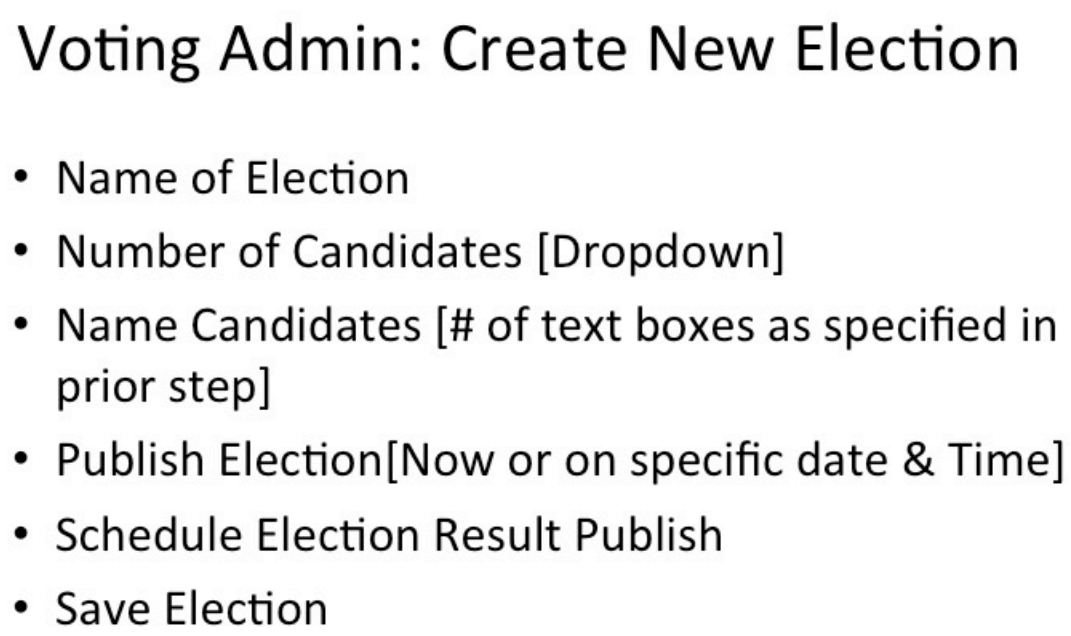


Home:

Election Manager:



Create New Election:



Saved Elections:

View Saved elections

- List of saved elections with [Publish (date/ time) and view]
- Election 1: Publish [date/time] | View(edit)
- …
- …
- …
- Election n: Publish [date/time] | View

View specific election:

View Election 1

- Election 1 [Change Name]
- Candidates [Alter candidate names]
- Add/Remove Candidates
- Publish [Date/Time]

Election Schedule:

Election Schedule

- Calendar with past and scheduled elections

Review Election data:

Review Election Data

- By date [past election calendar]
- By type

- (Once selected taken to election data page)

Election Data(specific):

Election 1 Data

- Date of election
- Candidate 1: x
- Candidate 2: y
- …
- Candidate n: z

- Election won by Candidate _____
  - Publish via email

Server Administrator:

Server Admin

- Manage Kiosks and Kiosk Status
- PHP myadmin
- View registered voter list

Required Dependencies

- libx11-dev
- libxext-dev
- libxi-dev
- x11proto-input-dev
- libncurses5
- libncurses5-dev
- qt4-dev-tools
- python language
- mysql database
- libusb-dev
- libtool
- libssl-dev
- automake
- python (2.7)
- python-dev
- ibxv-dev
- swig
- python-configobj
- git
- libgtk2.0-dev
- python-mysqldb
- libmagick++-dev
- Python-image
- Tkinter