Deliverable 0: Literature Review
Team members: Brandi Durham, Justin Willis, Austin Hollis

[1] The Technical Guidelines Development Committee (TGDC) was tasked with creating EAC Voting System Testing and Certification Program Manual and it is distributed by the government to let election voting software companies know what is required by the government. The document tells us that because of the Voting Rights Act of 1965 ballots are required to able to be read in any language. This would not be a functional requirement for our project, but it would be for a government election software. The document says that one requirement in particular is that there is a single coding convention. This will be a functional requirement for our project and will not be a problem because we are using Java and JavaFX. It is a requirement that the company provides a way to verify that the application booted correctly and nothing has been tampered with. This would be security requirement set by the company for the client to use to verify the application downloaded or loaded correctly based on the files signature.

[2] The NCSL, The National Conference of State Legislature, posted an article entitled “Voting System Standards, Testing and Certification”. This article explains the different types of voting machines each state uses, whether it be Direct Recording Electronic (DRE) or Optical scan paper ballot systems. What we will be creating for this project would be considered a DRE. This entails a system that uses touchscreen, dial, or push button to directly record votes into the computer memory. The document informs us that most requirements address security, functionality, privacy, usability, and accessibility. The machine must be secure by not allowing additional votes to be cast after the polls have closed, being immune to tampering, auditable, and unable to be connected to the internet. The functional requirements are that it correctly registers and records all votes cast, allows everyone to vote for whoever they want, offers the voter the chance to review their votes before casting them, allows the voter to correct or change their vote before casting them, notifies the voter if they overvoted or undervoted, allows voters to “write-in” their choice, and accumulates the total ballots cast. The machine ensures privacy of the voter and their choices, usability for ease of use, and accessibility for people with visual, physical or cognitive disabilities. These should all be functional requirements of voting systems, including ours.

[3] The technical report, “Building Reliable Voting Machine Software”, by Ka-Ping Yee is his report on the voting software he built. The technical report covers everything from how an election works, to requirements, to even his own source code. The report mentions the goals of the system such as accuracy with displaying the correct ballot, having it cast and count the votes as intended, offering fairness that each voter is authorized, only allows one vote, secret ballots, and equal choice. The author gives a list for each of these as to how to make sure and achieve the requirements. These will be helpful for our functional requirements to check and make sure our system completes these as well. I found this article very useful as it brought up elements I wouldn’t have thought about such as the requirement of accuracy and that the ballot should have no choices selected for the user when started up. Sometimes, machines have the first choice selected by default, which could lead to someone accidentally voting for someone they did not intend to.

[4] The Software Integrity Blog published an article called Weighing the pros and cons of open sourcing election software. The blog makes us aware that some people believe that open source software is more secure because being open source allows more people to see the code and examine it for bugs or defects. The blog post goes on to explain that trusting the community to fix the bugs or risks is not the best practice because it makes the original developer or tester less likely to be thorough in their efforts to test their software. If election software was open-source, then the security risk falls to

the client to understand and mitigate instead of the company. Allowing software to be open source, allows the community of developers to help mitigate bugs but it also allows hackers to analyse the code. This article makes us understand the significance testing for product dealing with such significant data as election results.

[5] The technical report, “Requirements, Design and Implementation of an E-Voting System”, by Ghassan Qadah is their technical report for a generic e-Voting System, where voters can cast their votes anytime using a number of electronic devices including web and mobile phones from anywhere. The report for the e-voting system goes into detail about the generic requirements regarding privacy, authenticity, integrity/accuracy, security, and democracy. Qadah also covers the system-specific requirements such as multi-user, multi-campaign, accessibility and availability. There is also a section that touches on the system architecture in regards to the databases, management system for the database, the web server, and even an SMS (Short Message Service) server. The main contribution that this report brings is the generic aspect that most every e-voting software should take into account when being made.

[6] The technical report, “Software Requirements Specification: E-Elections Software - “The Pericles Project””, was written by Amir Atalla, Mathew Evans, Mike Levy, Cory McKay, and Sitai Sun on their own E-Election software. The report goes into detail on the product functions revolving around using the Pericles package which includes the Election Server, Election Editor, Voting Database, and Voting Client. The report talks briefly about their general constraints which is a good thing to keep in mind while creating any type of software. They then go into the system requirements but focused on the following, in order of importance, with functionality being the most important followed by reliability, maintainability, security, privacy,scalability and finally ending with interfaces. The report then goes into the design constraints and states that the voting client was created in Java but the description language must be written in XML which is good to keep in mind because their election database must have used either MySQL or Postgres to run. The main contribution this article brings is how detailed they were in their testing phase which will give our team a great start when testing our product out.

[7] The article, “The Future of Voting on Follow My Vote”, took a more interesting take on the current voting system. They evaluated a few weaknesses of the current system and found the following: voters with disabilities cannot vote without assistance, communication with remote voters is slow and unreliable, vote tallying is both prone to error and labor-intensive, audits of elections are costly and the process offers little transparency, and voters must implicitly trust election officials and hardware vendors. The article then goes on to some recommended requirements of an end-to-end verifiable voting system or E2E-VIV System. The important thing to take from this article is that it touches on auditing and finding a way to make an auditing capability built into the original system as opposed to an afterthought in order to reduce costs.

[8] “Software Requirements Specification (SRS) Document” appears to be an assignment similar to ours. The writers, Anurag Verma, Ashish Gandhe, Ramkrishan Kumar, Simhadri Harsghavardhan, and Sumesh Dutt Sharma may not have actually created an e-voting software, but they definitely have the documentation for what seems to be good design for such a system. In their documentation they stress that their approach to the system requires two parties. They label these parties the EA (Election Authority) and the voters. This may sound quite rudimentary but I believe the writers explain the bounds of the EA quite well. When discussing the EA, they explain that the primary objective should be “to conduct fair and hassle-free elections”. This will be an underlying goal for our project as we will strive to create a secure system that will also be straight-forward when it comes to the actual voting process. The document also explains key features for the host when accessing the program such as

having the ability to “create/update/delete the election details (posts, candidates, electoral rolls, etc.”. One piece of functionality I think our group can work towards is incorporating host access to certain aspects of what voters will see. For example, we could allow for the host to show the name, picture, and description of candidates. However, this is simply one piece of functionality out of many we may strive to offer.

[9] Another helpful document for this assignment is “Functional Requirements for a Secure Electronic Voting System” by Spyros Ikonomopoulos, Costas Lambrinoudakis, Dimitris Gritzalis, Spyros Kokolakis, and Kostas Vassiliou. As the title alludes to, this is yet another document that explains what should be included when an e-voting system is developed. One portion of the document I enjoyed in particular was when the general principles of voting were discussed. Although these are not functional requirements right away, they can definitely be applied to requirements we can incorporate into our voting system. The first principle is labeled Generality. The writers of this document define this term by saying, “All citizens, unless otherwise stated by adjudication, above a certain age have the right to vote.” From this, we can create a process in our system to ensure a participant is eligible to vote. The next principle listed is Freedom which states that “everyone is free to vote for the party he/she considers more appropriate.” We can use this idea in our system to ensure voters are allowed the opportunity to vote for whomever they so choose. Equality is another principle and the writers describe it by stating that “all votes are considered equal.” One major aspect of this principle for us to employ is that voters only be allowed one chance to vote. Once a vote has been submitted, the user will not be able to alter their entry or resubmit. A nice addition, however, would be to allow voters to view their submission as well as the final results after the end of the election is over. Another principle is Secrecy. The document gives an example of how to abide by this by stating that “none of the actors involved in the voting process should be able to link a ballot to a voter. This somewhat ties to equality again in that every vote must be attached to a particular individual. The final principle is Directness which revolves around the ideas that “no intermediaries are involved in the voting process” and “each and every ballot is directly recorded and counted.” I believe that if we approach our election system with these principles in mind then our final product will be a success.

[10] One source for information on voting systems we found was the Brennan Center for Justice at New York University School of Law. They published an article entitled “Voting System Security and Reliability Risks” which should help us be aware of the potential dangers found in a voting system. One section in particular describes how the host of the software can protect the integrity of their databases. The first point they bring up is that manipulation of databases in any form can be avoided if backups are involved. They mention that this can even include paper copies of registration lists. Although we will most likely not encounter data the size of real-world political elections, keeping backups of whatever data we can should still come in handy. Another extremely helpful piece of advice is that the system be “programmed to run frequent, automated scans of registration activity to monitor for and alert election officials to potentially fraudulent or abnormal activity, such as a high volume of traffic or oddly timed traffic.” Again, although I am doubtful we will be exposed to major cases of our program extensively being used, it would not hurt to allow for scans of our system to occur. One final point that is made in the article is that “voter registration databases should not contain any information other than what’s required to register, or specified information relevant to the administration of elections.” This seems like an important point for any type of system one might make. We will definitely heed this piece of advice and hopefully keep the information held within the database to a minimum of what is essential.

[1] Testing & Certification Program Manual, Version 2, U.S. Election Assistance Commission, Silver Spring, MD, USA, 2015, https://www.eac.gov/assets/1/28/Cert%20Manual%207%208%2015%20FINAL.pdf
[2] VOTING SYSTEM STANDARDS, TESTING AND CERTIFICATION, National Conference of State Legislature, Published on August 8, 2018, Accessed on January 19, 2019, [online] Available: http://www.ncsl.org/research/elections-and-campaigns/voting-system-standards-testing-and-certification.aspx
[3] K. Yee, “Building Reliable Voting Machine Software”, University of California at Berkeley, December 19, 2007, Accessed: January 19, 2019, [online] Available: http://digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2007-167.pdf
[4] T. Mackey, Weighing the pros and cons of open source election software, Software Integrity Blog, March 22, 2018, Accessed: January 19, 2019, [online] Available: https://www.synopsys.com/blogs/software-security/pros-cons-open-sourcing-election-software/
[5] Qadah, G. (2019). [online] Pdfs.semanticscholar.org. Available at: https://pdfs.semanticscholar.org/5c68/de8ebe21c7457e23a2bf1b9289326f52c3d5.pdf [Accessed 18 Jan. 2019].
[6] Atalla, A., Evans, M., Levy, M., McKay, C. and Sun, S. (2019). [online] Music.mcgill.ca. Available at: http://www.music.mcgill.ca/~cmckay/software/computer_science/PericlesElections/PericlesRequirementsDoc.pdf [Accessed 19 Jan. 2019].
[7] Follow My Vote. (2019). The Future of Voting - Follow My Vote. [online] Available at: https://followmyvote.com/survey-research/the-future-of-voting/ [Accessed 20 Jan. 2019].
[8] Verma, A., Gandhe, A., Kumar, R., Harshavardhan, S. and Dutt Sharma, S. (2019). Software Requirements Specification (SRS) Document. [online] Dos.iitm.ac.in. Available at: http://dos.iitm.ac.in/OOSD_Material/CaseStudies/CaseStudy2/eVote-srs.pdf [Accessed 22 Jan. 2019].
[9] Ikonomopoulos, S., Lambrinoudakis, C., Gritzalis, D., Kokolakis, S. and Vassiliou, K. (2002). Functional Requirements for a Secure Electronic Voting System. [online] ResearchGate. Available at: https://www.researchgate.net/publication/220722492_Functional_Requirements_for_a_Secure_Electronic_Voting_System [Accessed 22 Jan. 2019].
[10] Brennancenter.org. (2016). Voting System Security and Reliability Risks. [online] Available at: https://www.brennancenter.org/sites/default/files/analysis/Fact_Sheet_Voting_System_Security.pdf [Accessed 22 Jan. 2019].