1. Introduction

This was midterm election year! There has been a lot of discussion and publicity about the electronic voting, which has been adopted in several states\(^1\). There have also been questions on how transparent and secure is this process\(^2\). On the other hand, there are advantages in electronic voting: easier access to voting process, faster counting, economizing on resources, such as paper. (There has also been a great xkcd comic about voting software at the end of this document.)

For this semester project, you will need to build the requirements document, system design, objects, and the subset of the software application of an electronic voting system using detailed documentation, UML, and Java. There will be multiple deliverables described in the deliverables section that you will need to submit by the deadlines included in this section. I will be the client for this project and you will need to meet with me especially when you build the requirements engineering document, the use cases and scenarios.

You will work in groups of minimum two, maximum four students. A good publication on group collaboration that I recommend for you to read is titled: \textit{“All I Really Need to Know about Pair Programming I Learned In Kindergarten”.} Keep in mind that you will submit a team review document in the end of the semester. Project work needs to be completed by all partners equally. Furthermore, you will have a project presentation where I can ask detailed questions about your work and figure out if you have worked equally. Finally, the whole class will have the same project and plagiarism is impossible in a project like this and will be easily discovered. The project will be uniquely built based on your questions and interviews that you will conduct with the client (me ☺).

2. System Objectives

2.1. The e-voting system will be designed for the state of South Carolina. It will allow voters to vote electronically from any SC voting site.

2.2. The e-voting system will abide to the legal, regulatory, behavioral, and sociological aspects of the current voting system. Furthermore, it will add convenience and security to the voting process.

2.3. The system will improve voting in the following ways:
   - 2.3.1. Allow voters to vote from any SC location that has machines with the installed software
   - 2.3.2. Reduce the number of over-voting and eliminate vote tampering

\(^1\) https://ballotpedia.org/Voting_equipment_by_state
2.3.3. Allow voters to check if they are registered prior to voting
2.3.4. Increase voter confidence and improve voting experience
2.4. The system will enhance the voting process in the following ways:
   2.4.1. Allow for unofficial vote tally
   2.4.2. Crosschecking the voter records with casted votes
   2.4.3. Increase reliability, speed compared to humans counting

3. **Functional requirements**
   3.1. Voter Registration
      3.1.1. It must be easy for an individual to register to vote
      3.1.2. An individual must identify themselves, in some way, in order to register
   3.2. Casting a Ballot
      3.2.1. The voter must identify themselves, in some way, in order to vote
      3.2.2. Indicate that a selection has been made or canceled
      3.2.3. House and senate candidates should be included. Imaginary candidates are allowed and welcome (Please, no profanity! You may use cartoon characters or other imaginary characters). You may use ballotpedia (or any other internet resource) if you want to include real candidates: [https://ballotpedia.org/South_Carolina_elections,_2018](https://ballotpedia.org/South_Carolina_elections,_2018)
      3.2.4. Notify the voter when the selection is completed
      3.2.5. Before the ballot is cast, the voter is allowed to review his/her choices and, if he/she desires, to delete or change his/her choices before the ballot is cast
      3.2.6. Prevent the voter from over-voting
   3.3. Tallying the Ballots
      3.3.1. An unofficial in-precinct vote tally will occur once the polls have officially closed
      3.3.2. Votes will also be transferred to a central location to be officially tallied
   3.4. Certifying the Vote
      3.4.1. The number of votes cast should be consistent with number of voters
      3.4.2. Recounts must be possible
   3.5. Database to organize your data (**low requirement, extra credit**): this is an extra credit requirement. You may gain up to 10 points for using a database.

4. **Non-functional requirements**
   4.1. Security
   4.2. Performance
   4.3. Reliability
   4.4. Usability

5. **Deliverables**
   5.1. **Literature review (5 pts)**: Literature review (5 pts): This is the part where you need to “do your homework” on what already exists out there before you discuss with the client. You will need to research different sources (publications, blogs, product specifications) and write a proper literature review with a paragraph summary for each source and proper citations (IEEE style).

Submit your work on Oaks and on Github! My github handle is: **mundruid** for you to share your repo with me.
Length: You will need to cite at least 10 different sources. Describe each source briefly and include what it contributes to the solution of the problem, how it contributes this, and any other information that it may give on the design and implementation of election software.

5.2. D1 - Inception (10 pts): For this deliverable, you will need to submit the following inception documents for your software such as: Vision, Use-Case enumeration with 2 use cases "fully dressed", use case UML diagram, supplemental specs, and a glossary.

Code: You will need to start working on the design of your User Interface (UI). You may use powerpoint or draw.io. You need to have a good design, multiple screens, clean buttons etc. You may start coding the basic view using Java FX and screen builder. Code submission in this case is not required but encouraged.

Demo: short (min 7 mins, max 15 mins) presentation of vision, use cases, GUI prototype, and any other items you would like to add. Demo day: Sept, 20 during class

5.3. D2 - Elaboration (10 pts): Submit all use cases fully dressed and domain model. Include a short description of the domain diagram and justification of its components.

Code: Code submission will be part of your grade this time. Submit an initial version of your View (UI) using Java FX. This code will be graded.

Documentation and Revisions: Collect all your documents: vision, supplemental specs, glossary, use case diagram, domain model diagram, and all use cases fully dressed, in one document. At this point you may start revising your previous deliverables, such as use case diagram etc.

Naming and tools: You will need to use GitHub for version control. You will need to add me to your repo (my github handle is mundruid). Include your documentation in the repository. Use Eclipse or IntelliJ to create project named: “TeamMateLastNames-MidtermElectionApp”. Use package structure com.csci360.electionapp.

5.4. D3 - Elaboration (10 pts): Submit all the system sequence diagrams and operation contracts required for your election software. You may add these diagrams to your SAD (Software Architecture Document) and submit a single document. Add any revisions you may think are necessary based on my past feedback.

Code: Submit an improved version of your View (UI) using Java FX. Start coding the controller and model parts of the project. You may submit blueprints of classes. You will need to have at least 10% of your classes fully implemented.

Demo: short (min 7 mins, max 15 mins) presentation of progress with UI, SSDs, SDs, and any other diagrams you think are worth demonstrating. Demo day: Oct. 23 during class.
5.5. **D4 - Objects (10 pts):** Identify objects needed to implement your operations. Submit a class diagram for the voting system. Suggest object responsibilities and collaborations. Submit sequence diagrams that include labels that indicate all the object collaborations based on GRASP. Include a short justification of your design choices.

**Code:** Submit any revisions on View (UI) using Java FX. Second version of classes should include additional attributes and operations. Populate classes; at least 30% of classes should be complete.

5.6. **D5 – Objects (10 pts):** Complete the object design, revise sequence diagrams. Note: it is important to apply the GRASP principles, especially GoF Adapter, Factory, Facade, Singleton, Strategy, observer and GRASP design principles such as Information Expert, Creator, Controller, Polymorphism, etc.

**Code:** Submit any revisions on View (UI) using Java FX. Populate classes; at least 60% of classes should be complete. Include Junit tests for *all classes*. Think about improving the way you store data (file vs database) and include this in your object design. Address important non-functional requirement of security especially when you store sensitive data.

5.7. **D6 - Java code and Testing (60 pts):** Finalize your code and submit all the test cases that you created using Junit or any other tool. Submit any other test cases that you would like me to run.

What to submit:
1. README file that gives detailed instructions on how to run your code.
3. A zip folder with your code. That is just to make sure that I have a backup with your latest version of the code.
5. The powerpoint slides of your presentation.

5.8. **Powerpoint final presentation & Demo (10 pts):** You will need to present your work in class. This will be a 15 min presentation with 5 mins left for Q&A. The presentation should have professional slides that will present briefly your requirements, analysis, system design, objects, and a demo your prototype.

**Tips:**
Collaboration tool: [https://www.teamwork.com/](https://www.teamwork.com/)
Voting Software

Asking Aircraft Designers about airplane safety:

Nothing is ever foolproof, but modern airliners are incredibly resilient. Flying is the safest way to travel.

Asking Building Engineers about elevator safety:

Elevators are protected by multiple tried-and-tested failsafe mechanisms. They're nearly incapable of falling.

Asking Software Engineers about computerized voting:

That's terrifying.

Wait, really? Don't trust voting software and don't listen to anyone who tells you it's safe. Why?

I don't quite know how to put this, but our entire field is bad at what we do, and if you rely on us, everyone will die.

They say they've fixed it with something called "blockchain." AAAAA!!! Whatever they sold you, don't touch it. Bury it in the desert. Wear gloves.