**Post production notes**

**Changes made to the design**

**Deliverable 1:**

* The Design Document will be constantly updated as we move along through the project. It may be possible that we decide to change the way we implement different aspects of the game. We may also decide to change other parts of the game based on recommendations made by our peers or teaching assistants.
  + We may change our design because we will most likely encounter problems with our initial design and then possibly come up with a better solution to our problems. If it is not a problem we may decide to change different aspects of the game such as the color scheme or the image displayed on each of the buttons.

**Deliverable 2:**

* We included AI versus AI even though we were not supposed to include this functionality this early in the process.
  + This is because we had more time than expected and implemented AI versus AI anyways. We also found it fairly easy to implement this functionality. This allowed for us to have more time to work on other parts of the project because we were able to finish this early.
* We did not have a graphical user interface working but we did have our ASCII display of the board along with the pieces working.
  + This is because we did not have enough time to create and add the graphical user interface. We were experiencing some trouble implementing the GUI as well as trying to display the board game and pieces.

**Deliverable 3:**

* A big change that was made was to our project development was the AI's logic. We changed the thinking process of the AI so that it could make somewhat logical moves when playing against a human. We updated and changed the logic of the AI for all difficulties.
  + This change was made so that it would affect the game-play against the AI because this change would allow the AI to react and block out the opponent’s pieces. This allows for the AI to capture or block out threatening pieces that would allow the opponent to win. This is an important change because it also allows for easier and better implementation of the different difficulty settings.
* We decided to make the change and allow for pieces to have values such as giving them a priority when playing the game. The value and priority of the pieces was based on how far up the board they were. So, the closest piece to the opponent’s side would be taken into account first when the AI made a move.
  + We ended up changing this and gave value to all the pieces because we knew it would be a nice medium for both the player and the AI. We assume that this will balance out any errors that we would come across if we only gave value and high priority to pieces that were the closest to the opponent's side.
* We changed the area accessible to the AI because at first we excluded the first two rows from where the AI starts. This is because those two rows will be the starting position and where the AI would be moving away from. So later on we decided to add back those two rows from where the AI will be starting.
  + We made this change because it will allow for better performance of the AI, we excluded them at first because we assumed that they were not needed. We ended up including the first two rows because it would allow for better evaluation of pieces and it would also allow better implementation of the AI.
* We decided to change the int type of the integers that were being stored for each piece.
  + The values given to each piece were stored in integers but we soon realized that these values were too large. This is why we then changed the type of 'int' to 'long int'. These are the numbers that were saved for each piece which were also the priority of the piece.

**Difficulties**

**Deliverable 1:**

* Design document changed several times as we worked through the project because better ideas and implementations came about while working. We also ran into trouble when we were almost done implementing one part of the project but then stumbled upon a better way to solve it.

**Deliverable 2:**

* We ran into some trouble when setting up the server part of the project because we could not get the server to communicate with the clients. For example, reading the sockets while waiting for a response was a hassle because multiple messages would arrive at the same time.
* When sending the input through a parser we ran into difficulties and so we ended up utilizing a lexer to read the input from the player. There was trouble because we would input the right commands (or what we preferred) and then it gave us errors about how we formatted our input.
* Saving the states of the board game was a bit of a problem. We did this to allow players to temporarily suspend the game and resume game-play after returning.
* We had trouble getting the AI to work correctly at the beginning. This was because we were trying to simultaneously get the AI to respond and move after we completed a move as well as implementing the algorithm. The hardest part was trying to implement the AI to work with the algorithm.

**Deliverable 3:**

* We ran into trouble getting the evaluation function to work. We could not get the correct response or output when playing the game so we needed an efficient way to traverse a tree to help the function to work.
* Utilizing the min-max algorithm and getting it to work was a bit of a hassle. We tried using the algorithm recommended to us in class but we could not get it to work with our code.
* We also ran into trouble when utilizing alpha-beta pruning and increasing the processing speed of the search algorithm. The implementation was either too slow to work with or gave us multiple errors.
* We wanted to not only increase the speed but we wanted to increase the reliability of Alpha-Beta pruning. There were a few mishaps when using the algorithm because the game pieces were only moving diagonally.

**Solutions**

* After the first week, we would constantly have the design document updated instead of waiting until the last minute to input changes. We had everything documented about what we did that day in a shared text file so that every member was aware of what went every single day.
* The problem with communication between the server and the client was mostly solved with multiple trial and error attempts. Our code was right but we had it in the wrong spot so we kept messing with our code structure until we successfully communicated with the server and the clients.
* We utilized a lexer instead of the parser because it allowed for the inputs to be read without any difficulties and made it easier to read input in general.
* We were able to save the states of the game by implementing a class that would save each state of the board-game throughout the whole game.
* We temporarily suspended the AI functionality and algorithm so that we could work on other vital parts of the project. So, we decided to assign the AI to make a random move even if it was not a logically intelligent move.
* The issue with the evaluation function was resolved by making a class for Tree and Node and then breaking it down into multiple helper functions to allow better ease of use.
* The algorithm presented to us in class did not work for us so we ended up looking online for a better solution and algorithm. We found something we could use so we took part of what we found and made it into our own code to use in our project.
* The last two bullets of difficulties were resolved by explicitly having the algorithm look at only the relevant parts of the tree.

**Lessons Learned**

* The idea of SCRUM allowed us to work in a productive strategical way because it allowed us to manage our project development. This is because of the multiple meetings that were held and the face-to-face communication and interaction. We have a close collaboration online but it was best to meet up because it made it easier for us to communicate and pitch in ideas to our initial design.
* The utilization of vectors to save all previous states worked best so we could easily use our "UNDO" command.
* Modifying our search algorithm to focus on relevant parts of a tree allowed for increase in processing speed as well as the reliability of our game.
* Instead of making a class for each cell so that we could manipulate them we found that it was best to make a vector of vectors containing char's. This is because it would be faster and easier for us to work with.

**Individual workload percentages**

**Ryan(32%)**

* Ryan has added a good backbone to the team with his knowledge in seemingly everything. Ryan has shown up to all of the meetings and knows in depth what each program does and how it is supposed to run.

**Nathan(32%)**

* Nathan has added an extremely good spin on different logic that allows the project to seemingly come together. Nathan has shown up to all of the meetings and knows in depth what each program does and how it is supposed to run.

**Jonathan(32%)**

* Jonathan has added a dedication that helps the team come together as a whole. Jonathan has shown up to all of the meetings and knows in depth what each program does and how it is supposed to run.

**Victor (4%)**

* Victor has created Post-Production notes.