**Preplan and Scope:**

The first thing I always do when tackling a new assignment is analyze the requirements of the project, consider the deadline (if there is one), and gauge my own ability to complete that assignment within that timeframe. Depending on if its a long term project, or a short term assignment, my first goal is usually to focus on and determine what is most critical to the requirements of the project, and create a minimal-viable product based on that. After the MVP is complete, then I will start adding on the other features. The assignment was issued late Wednesday afternoon and was asked to be completed "early next" week. So I gave myself a deadline of Monday to complete the assignment. Since I was going to be out of town Saturday, that gave me about 3 half days and 1 full day to complete assignment. As I worked on the project, I used this time estimate to gauge the scope of my project, what frameworks I would use, and how much functionality would be able to be completed.

**Framework:**

For this assignment, I went with just a simple single page app, focusing on JavaScript with AngularJS. There are two main reasons why I didn't decide to go for a more complex stack. Firstly, I believed I would be able achieve most of the functionality I desired without needing to implement a backend or database. And secondly, I considered the audience, which theoretically would be Little League coaches, but in reality is you, Nick Gassmann, my interviewer. Keeping that in mind, since this is merely a prototype and not a full-on application, I wanted the demonstration process to be a painless as possible, without the need to setup an environment or download dependencies. Nor did I necessarily want to set up the app in a server just for a prototype meant to test my front-end capabilties.

As for future expansion of the app, I would definitely use a fuller stack. Key future features highlighted in the project description include saved rosters per game, and saved rosters per season. All this points to the necessity of a user account system integrated with a true database. However, I did not find that absolutely critical for the purposes of this demonstration and so I did not implement it. In addition to that, I have also come up with a couple more features that could be possible: full-on player models that you can save attributes (such as preferences, picture, etc) and reload during the roster creation process. That way you don't have to reinput name or position preferences each time like you would in the current app. In my experience this points to the need for a relational database, so I would choose a framework more focused on that like Django or Ruby on Rails (but consider me biased towards those since I've worked the most with them). And since my current app is just pure javascript, I can easily integrate it into any larger framework.

**Functionality:**

Now here's where it gets interesting. To summarize the goals of this assignment, it was to create a lineup of of players based on some variable conditions: player preferences, AA rules, and optional rules. Originally I had planned to create an algorithm that would consider the conditions and create the lineup pattern itself dynamically.

However, that proved to be quite complex. VERY complex. More complex than I originally intended. I spent a day tinkering and prototyping the algorithm, and I found it to be quite the complex task to achieve, especially considering the limited work time I had. At the moment, it would essentially be brute force algorithm and I found that I would not able to properly design and test that algorithm within this limited timeframe.

So I set about creating the lineup a different way. I'd be more than happy to go over this with you in the in-person interview as I found the entire process quite interesting, but the gist of it is that it takes a preset lineup pattern and populates it with the players. The app doesn't generate a lineup so much as it fills in one already created. Considering the scope of expectations and limited timeframe, I found this to be the best course of action to take. Developing an algorithm that would dynamically create a lineup considering ALL the special conditions would definitely take a couple days to fully develop. That being said, the current preset lineup pattern was designed considering ALL the specials rules are true. Meaning whatever my app currently outputs would be a valid lineup regardless of whether or not the special rules are toggled. As such, the special rule toggles and the player position preferences on the page are superflurous, and are there merely to showcase my front-end capabilties.

And lastly, right now the app creates a lineup with only 9 players, regardless of how many players you actually add. However, I HAVE certainly developed a 100% correct lineup pattern for up to 9-15 players (the min and max allowed in Little League). This would need to take about another day to fully implement and test, but I found it more important to prioritize making the styling look good to present and finishing up this writeup. So, I suppose, take it on good faith that I DO know how to implement it :)

**Mockup:**

As for sticking with the mockup, I more or less stuck to it. However, there are a couple front-end design decisions that I consciously chose to do differently. First, the player preferences are show/hide instead of a dropdown. The reason for this is user-friendly functionality. Most readily available multiselects requires the user to hold down CTRL to do so, and simply put I don't believe everyone knows that can be done. And so having a dropdown without an indicator that they can multiselect will have the user doing single clicks and only highlight one preference in the dropdown, even though they have the capacity to highlight multiple. Sure, there is a jqeury method of doing a dropdown with checkboxes, but I stuck with show/hide because it allows the user to open the preference box for multiple players so they can compare and set multiple at once, whereas you usually cannot open up multiple dropdowns at once.

And secondly, the 2nd main deviation from the mock up is how I chose to display the final lineup. Whereas the mockup has Innings on top and Player on the left, I have Innings on Top and \*Position\* at the left. I thought this was a better way to organize the lineup. I figured displaying it this way is more intuitive, since it has the positions listed in their proper numerical order and would be easier to tell at a glance which player is in which position during the current inning.