Finding an appropriate bootstrap template was a good starting point for me. This would allow me to focus primarily on the implementation of features and offer me more time to find a fitting API. I wanted a template that didn’t go overboard on the design aspects and would show the information first and foremost. <br><br>  
  
Speaking of the API, I began by looking over how to use the one suggested in the project documentation, openweathermap.org, but found myself not liking it, so I began in search of another API and I landed upon using https://darksky.net/ which is another widely used weather API. <br><br>  
  
I began to look at the documentation and online resources on how to use dark sky and how to begin implementation. I began coding and after a couple hours realized that I would need to streamline the process of making the API calls as dark sky requires, for each call a set of latitude and longitude coordinates corresponding to the city you want shown. So, I created a function with several parameters named “createCall” that would be used in my main HTML file to pass in the required info that I wanted displayed. (this can be found in js/app.js) <br><br>  
  
For the extended forecasts section that was to be created, I put together another JavaScript function for displaying the extended forecast for the cities by displaying hourly and daily data for the city selected. For the design of these I knew I had to figure out a way to nicely display the data as there would be a lot of data coming in and I would need to parse through it. My solution for this was to place the information into HTML tables, giving them a simple readable format. But I also wanted to not make it look like a basic HTML table as I’ve taken design classes in college and have learned about the importance of design so I designed it to look aesthetically pleasing to the eye as well as providing readable data. <br><br>  
  
Some problems I ran into revolved around JQuery. To dynamically show the extended forecast, I used the JQuery function replaceWith() and replacing the div containing the cities and display the selected city's extended forecast, but ultimately decided to implement a back button and use multiple pages to display the extended forecast. Another issue that presented itself was the fact that the API presented dates/times as a Unix timestamp and not a standard date/time number. I’ve never run into Unix time stamps before so I had to find a way to convert to a readable time. After some research, I was able to find some references on how to convert the Unix time into a 24-hour format to have it at least somewhat readable to the end user.