Dennis Cao
Deryl Lam
Jeffrey Trang

CSE 134 Homework #4

As noted from the homework write up, our overall goal for this assignment was to create and provide CRUD functionalities between our website and a database as well as providing login authentication for both standard email and Google while having consistent coding organization and format.

As the instructions pointed out, we used Firebase built in methods to implement our login authentication page for both standard and Google emails. We created the registration in a way that allows users to initially choose to register with a Google account or a standard email of their choice. The functionality was pretty straightforward to implement for both standard emails and Google but we noticed and thought about how to accommodate users that were already logged into their Google accounts. For these scenarios we have it bypass the registration/login page and moves straight to the homepage to avoid repeated logins for Google users. Regardless of login, all new users will have a starting watched list and can proceed to add/remove from there.

We started this assignment for our CRUD functions and designs that were based off our initial prototypes and format that we set forth in our previous assignments. This actually hindered our initial process because as we were trying to code JavaScript functionality, the process was being constrained by our existing pages on how it could look and act. This proved to be a little time consuming as we often got frustrated because we could not get the exact functionality we had in mind. Rather than beat ourselves in the head over it, we ended up pivoting around off our original ideas and refactored some of our code in order to accompany our JavaScript to get the functionality that we sought. Once we were comfortable changing up our original HTML code, using Vanilla JavaScript was not a problem at all as it allowed us to be pretty flexible on how we wanted to structure our functions giving us feelings of more control as opposed to be stuck with a predefined framework.

At one point in time during our create action, we used arrays to hold our players in a watched list because it was easier to index them and keep track of our data as we initially thought. However, as we moved on to other areas of CRUD, we realized that this original design not allow for easy access and involved adding and removing entire arrays at a time in order to update user watched list for the updating portion of CRUD that proved to be detrimental. This was definitely not as optimized as we could be as well as having complications with other parts of the CRUD actions. We started our create action without thinking of other parts of CRUD causing us to start over and restructuring our database. Our final design allowed for single deletions and single adds that proved to be faster and better fitting for our needs. We decided against having multiple deletions and pushed toward a single deletion format that proved to be become our final design and strapping our original intentions.

We structured our database so we could easily handle our assets. For example, we have each user that has one watchlist and each player they watch have a corresponding player key to them which we assigned so we won't have any problems with firebase assigning them random keys. We also made the watchlist an object with players so it's easier for us to loop through the players and pull out the player stats and attributes we want. This is essential so we can easily list and order the players in any fashion we want. We did not include images yet but it's the same process as what we have right now. All we would have to do is give the player a new "Image" attribute and give it an image but we were concentrating on getting the database working as intended which we achieved.

Early stages of the assignment, we had scattered Script tags everywhere and not having much coding structure. We ended up combining most of script tags into two main files that share functionality and were used consistently. We had to refactor portions of our code to accommodate our changes which added more time, but we can see retrospectively that it saved huge amounts of time overall because it was indeed easier to keep track of our files and recognize similar functionalities of code that we were then able to reuse rather than start from scratch and not having to reduplicate code over multiple files.