**CSE 134B Homework 4**

**Prototyping the Core CRUD and Authentication Features**

By Sherman Cheung and Trent Miller

**Performance Challenges**

One of the lessons we learned in this assignment was avoiding unnecessary calls to the database. Our original implementation of adding a Hero was very inefficient. Instead of simply appending a new card to our already existing list of Hero cards, and then updating the database in the background, we fetched all of the users heroes from the database, waited for the result, and then regenerated all of the HTML templates for the cards. This caused a significant delay in page loading time. We mitigated this by by removing unnecessary database queries and generating only HTML code that has changed.

Because this assignment was mostly focused on CRUD and leveraging Firebase’s toolset, we did not run into any major performance problems. By using Firebase’s built in database, we were able to read and write to the database very quickly. This was especially noticeable when we took user inputted data and updated values in the database. Using the Firebase console, we were able to see the data updated almost instantly. This makes us optimistic for the performance of our app for the next phase of the project.

**Development Challenges**

**Updating DOM based on DB vs user inputs**

One of the development challenges we faced was figuring out when to update content in the DOM when the user updated information. This was a problem that the professor briefly touched on in class. When the user wishes to update their personal stats, we update the information in the database based on a form they fill out. However, we debated whether to update the DOM content based on the form they just submitted, or query the database to update the information based on that. On one hand, updating based on the form would be entirely client-side and result on a faster user experience. However, if the write to the database failed, this would make the client inconsistent with the database. Reading the updated data from the database may be safer for this reason.

In the end, we decided to use the information from the user submitted form rather than query the database again. This is because Firebase performs database queries and updates asynchronously. SInce we needed to both update the database with the latest values and then read these updated values back from the database, we would have to force our read to block until the write completed. This seemed to go against Firebase’s design philosophy so we decided to handle the updates in the client.

**Exploration**

**Library Considerations**

Initially, we considered using a library like jQuery in order to simplify our developer experience and enable us to implement our project more efficiently. However, after doing lots of research on StackOverflow and other sites, many of the highly voted answers were against the idea of using jQuery because of the size of the library. Since our app is not too complicated, loading a large library like jQuery may not be worth it. It is unlikely that we would use the full power of the library. As a result, including such a large library would create a large burden for the user and may result in a poor user experience.

Furthermore, from a developer and learning perspective, neither of us had used JavaScript prior to this project. Therefore, we agreed that it would make sense to figure out how much we could do with just JavaScript’s baseline functionality before leveraging the power of an external library. After implementing the majority of the CRUD functionality, we believe that this was a good decision since vanilla JavaScript enabled us to do everything we wanted.