GameDex

Your game shelf for the modern day!

Development Team
Arthur de Jesus
Brian Maier
Rabab Singh

Firebase Implementation Analysis

I. Overview

This homework has been the most eye-opening and rewarding homework to date. Most of our group came from having taken CSE 110 where we utilized Firebase in an Android application, but the differences are truly remarkable. The professor constantly pushes the idea of "trade-offs" and that theme was very apparent during our exploration and implementation of this homework. We experimented with VueJS, vanilla Javascript, and of course the Firebase API. We spent countless nights just staring at the magic of the examples posted online and trying to figure out how to sprinkle some of that on our GameDex app. In the end, we realized we were quite often stuck in the weeds of actually implementing our full app, we failed to understand the big picture of exploring and getting the basics right. This clearly shows the Professor's motives were correct and makes his comments in the Slack channel that much more meaningful and relevant. After learning from our mistakes, again and again, we decided to totally strip away our work, cut our losses, and start fresh on a blank white page to demonstrate CRUD and asset management. This is how our assignment came to rest in it's final form. This write up will succinctly detail our progress while highlighting the challenges, failures and time analysis, followed by our lessons learned.

II. Challenges and Failures

To start, none of us are web developers. We all have some background in different languages, and could probably piece together some working GameDex app, but a lot of time spent was actually learning. Firebase for Web is a whole different beast than for Android applications and it took some deep thought to rewire our brains into thinking web focused. After nailing the *idea* of Firebase we took to coding. In hindsight, we should've just taken the demos and ran with it, but that's exactly what we did.

We found the Firebase Github repo with login examples and slapped it in our repo, did some hacking, and viola! We had a working login, but we didn't know what we did. Of course we understood the code, but we hacked it together, and didn't actually learn by doing. Taking a step back to understand the code, we re-tooled it and were able to have a fully functional login, signup, and Google login.

The next logical step for us was CRUD. Where to look other than the CRUD demo posted in the Slack channel. To our dismay, we noticed the TA's used VueJS (a library none of

us were familiar with). Arthur hit the ground running and attempted to rewrite our entire Javascript code and view page to fit the new and *magical* VueJS. Why? Because the TA's used it! It must be the right answer! Wrong. Arthur realized he should have listened to the professor when he mentioned the tradeoff between this amazing library, and the wind-up time. Arthur quickly realized he was way out of his depth with trying to learn VueJS and abandoned all the progress and time spent to implement this. In the end, we realized that the size, performance, learning curve, and functionality wasn't worth the time required to learn how to use VueJS and instead went back to Javascript.

Our final, and most notable, failure was getting too far off base with this assignment. We were so excited to start and actually implement our GameDex that we got stuck in the weeds and were thinking about the assignment all wrong. First, we started to implement Firebase into our existing wireframes. We realized it's a lot harder than we previously thought and spent a significant amount of time trying to get it to work exactly as our final product. Couple that with trying to learn VueJS and Firebase we had a recipe for disaster. Laden with errors and with the due date looming, we scraped the entire idea of using our GameDex wireframes for this assignment and moved to a SPA design as mentioned in the Slack channel (yet another challenge). We raced the clock to redo our weeklong work to create a semi-decent working demo of auth, CRUD, and asset management.

IV. Time Analysis

Overall, we received relatively fast load times (around 300ms for throttled 3g connection), but noticed a significant drop when having to connect to Google's servers. Our intuition for this anomaly is because the developer running the tests (Arthur) has multiple Google Accounts available to choose and it wasn't a seamless login experience.

The major culprits of the long loading time is actually Firebase authentication. Using Chrome developer tools, we saw significant drops in load times when accessing "getAccountInfo" and "verifyPassword." Sometimes the load times would even reach 5 seconds! Unacceptable for a non throttled connection.

V. Lessons Learned

1. Listen to the Professor
    a. We are incredibly lucky to have a professor who is actually teaching us the thinking and logic behind every action we perform, however being at UC San Diego for so long, it's very much ingrained in us to just "plug and chug." We were even warned to think before coding! The main takeaway for listening to the professor is to really, listen to the professor. I think we did a good job at heeding his warning, however. We even failed like he said!

2. Don't jump on the bandwagon for the "latest and greatest"
    a. After seeing the CRUD demo, we jumped on the VueJS train because we saw it accomplishes what we need! And how simple the syntax was too! Again, leading to lesson #1, the professor was right in saying the time it takes to learn

might not be worth the 5 lines of code we are writing. In the end, he was right, again.

3. Give time to learn these concepts
    a. These concepts are hard! Really hard! It's not terribly difficult code to write or implement, but to understand what we're doing and to do it right takes time. We spend a lot of our time doing and not learning. We did end up learning a lot by doing, so not a total loss.

4. Appreciate, but understand, the magic
    a. Some of the Firebase examples were magic. The way it just works is incredible. One of our downfalls was to blindly implement and just appreciate the magic. This, however, wasn't the right approach because it left us confused when we changed something and it broke the whole code. Understanding the magic and then being able to CREATE the magic is the goal.