2015 Intern Review

Manager Review

Jared Katzman
Performance Review: 01/01/2015 - 12/31/2015

Reviewer: John Nienart **Role:** Manager

Status: Completed **Manager:** John Nienart

Overall Performance Summary

Employee Summary:

1. Write a server-side library that interacts with various Payments systems to create/manage sandboxes and simulations. (exceeded goal) -- After the finalization of design and main user requirements, I had a lot of momentum coding. I faced very few blocks, received code reviews in a timely manner, and finished with time to spare.

2. Integrate Sandbox into Payments Platform UI with full functionality to create sandboxes and run simulations. (did not meet goal - yet) -- I have recently reached external blocks that are prohibiting me from making progress on some of the most integral parts of the system. However, majority (~90%) of the code is written and functional for a viable product. If these blocks can be resolved quickly and code reviews done efficiently, I think we can make great strides toward accomplishing this goal.

Overall, this project has taught me some invaluable skills in creating multi-tier systems that are modular and extensible for code to come.

Manager Summary:

I think everyone who was aware of it agrees that you had a very successful internship, in which you managed to get a significant chunk of the large project scope completed. I'll note that you arrived here less than a month after I started managing the team, and without an actual manager before me, no one had given any thought to intern projects. As a result, we had to scramble a bit to figure out things for each of you to do. So the timelines and milestone dates were quickly put together, and were definitely aggressive, perhaps too much so. Despite that, you rose to the challenge and completed more than I think anyone actually expected.

Looking at some metrics, you sent out 15 code reviews over the course of your internship (not counting boot camp), of which 13 had been submitted as of 8/13. First, this is a lot of commits for an intern, period. Second, it was just over three weeks from your start date when you posted three CRs in two days. Although they were small and simple (just config and model changes), they showed you had ramped up quickly. And the next week you posted three more CRs, which included substantial amounts of code, including a substantial amount of test code, about which more below.

The nine CRs that followed demonstrate solid, uncontroversial (this is a good thing -- it means you didn't have a lot of CR churn) work, some with no comments at all beyond a ship-it. Where you did get feedback, I see that you promptly and reasonably incorporated it. What all this shows me is that you took our advice from the outset and worked on producing mostly small, manageably-sized chunks of work that reviewers could easily digest. This is a surprisingly difficult skill for many people

to master, and for some it takes years.

With only one day remaining in your internship, you had three open CRs remaining, so its unlikely that all your code will make it to mainline before you leave. Push it to a branch. A challenge with these last CRs is that there are very few developers in the platform who are familiar with the technologies (Ruby on Rails and Alpaca.js), and there were five other interns this summer had projects on the same UI platform, leading to reviewers not having enough time for everyone, and getting exhausted. No one is going to fault you for this.

One thing that makes your progress so impressive is the multi-faceted complexity of your project. In case you were't keeping score, the project included, as your mentor laid it out: "writing server side library for Sandbox in Java, User Interface for running Simulations in Ruby rails/Javascript/alpaca, defining model for Sandbox request [and response] using ion sdl, database as dynamodb (with design and implementation) to persist the Sandbox data. Apart from this, project included operational tasks to create a brazil package, version set, creating AWS account, setting up and using Odin material sets etc." I don't believe I've seen such a complex intern project in the last several years. And although you note elsewhere that you were familiar with some of the technologies in use, there are a great many in that list that are specific to Amazon. A reviewer wrote: "Given the short period of time for the internship, it was thrilling to see how well he learned various technology [stacks]."

It's unfortunate that the project, despite its complexity, was such that you didn't have opportunity to interact with many people besides your mentor, and later in the project, a couple other developers. However, the feedback they provided on you is strongly positive. Let me end this section with two quotes.

One reviewer noted that you are an "extremely smart and result oriented individual", who is "quick in understanding and brainstorming various ideas."

Another wrote the following: "I was impressed with Jared's ability to research and determine best practices for testing in both Java and Ruby on Rails. I see many interns completely neglecting tests altogether. Either they don't create them or the ones they do create involve too many components and cover a very limited number of test cases. Jared looked at the existing tests in the codebase and created very suitable tests that were aimed at the specific features he implemented. He successfully tested all branches and success and failure cases. Because of this, I feel that our team is confident that he has built working and maintainable features."

Identify Strengths

Employee Summary:

Customer Obsession - I have experience with user-centered design, so customer obsession comes easily for me. Even though our customer at times was ambiguous (for example, having to design a sandbox for types of simulations that don't conceptually exist yet), in design meetings I always made sure to ask questions and think through the user experience. When finalizing the UI design, I looked for user feedback to make sure the sandbox was intuitive and easy to use.

Invent & Simplify - While I sometimes questioned why we used Alpaca.js to generate forms for the UI - because of its fragility and seemingly simplistic complexity - after spending weeks studying and experimenting, I now appreciate its power and ability to provide enormous support for dynamic and extensible systems. At certain times during development, I felt as if it would have been better to scrap the framework because of a necessary feature that seemed too difficult or impossible to implement. Luckily, I did not give up. Now we have a system that can easily handle changing requirements with as few changes as possible (which has personally saved me from having to rewrite hundreds of lines of configurations to just having to rewrite two) and that accomplishes all of the features we need.

Have Backbone; Disagree and Commit - In all of my discussions from design requirements to specific code conventions if I disagree or do not understand the purpose, I question, question, question. This has led to very productive conversations, in which afterward I feel more confident about our conclusions because of the scrutiny we put them through.

Bias for Action - I had some familiarity with majority of the technologies (e.g. Java & Ruby on Rails) I worked on this summer and felt prepared to dive straight into the code base without the slow ramp-up time of tutorials and simple coding exercises. I felt comfortable and confident enough with the language, and with the help from Saurabh and the code-reviews I faced very few issues. For the Java portion of my project, this decision led to more productivity and saved days worth of time.

Manager Summary:

I agree with you about all the principles you list above. I wan in particular to call out your Customer Obsession, and offer a couple examples of it in action. First, when a user searches by ID, you don't throw an error if the ID is invalid, but instead list all the user's sandboxes. It seems like a simple thing, but it's rare that a developer would make such a customer-centric choice (at least not without a UX designer prodding him to do so). Another example is how you made the sandbox URLs shareable, allowing multiple users to collaborate; I don't believe this was part of the project description, but it's a useful and logical extension that exists, simply because you thought about your customers and what they might want to do.

In addition to the leadership principles above, your reviewers mentioned several more as your strengths. Here I'll call out two, and quote your reviewers to provide details.

Earn trust of others -- You allude to this as Have Backbone above, and that's not wrong either. But your reviewer wrote: "[Jared] was always open to ... feedback and understand why something was necessary. He would never settle for something without understanding the implications and benefits, which is [a] very good quality. This can be seen in his code reviews." It's that seeking out of understanding that lets your teammates feel that they can trust you to make the right choices even when they aren't available to provide a check on your work. That's a rare quality, and not always seen, even in some fairly senior and experienced engineers.

Deliver results -- I'm aware that you listed this under areas for improvement, but consider this reviewer's words: "[Jared] completed his project in the timely fashion. Even with blockers multiple times, he was able to multi task between his UI code/ server side code and operational tasks. He was always clear on what to do next even if he gets blocked on one part of the project, which enabled him to work efficiently." While you felt stuck, you kept looking for a way forward, kept doing *something*, and therefore kept making progress. That's good stuff.

Identify Areas for Improvement

Employee Summary:

Deliver Results - Unfortunately, in the process of my project's development I reached a number of blocks that stinted my productivity and ability to deliver timely results. Many of these issues were outside of my control and realm of expertise (e.g. dependent packages' failing builds and molasses-like build times). Nonetheless, I believe that I could have motivated people to fix the problems faster so that I could regain my productivity and momentum.

Communicate Clearly - This is a skill I personally always strive to improve. My project went through a number of requirement changes that forced me to have to discard portions of my code. While I understand that is all a part of the software development process, I believe that with clearer communication on requirements, expectations, and action items I would be more productive and

efficient with my time and resources.

Dive Deep - Finding a balance between Dive Deep and Bias for Action proved to be more difficult than expected. As I mentioned in my strengths - I think the calculated decision to forgo beginner training on technologies I was already familiar with provided numerous benefits in the beginning. Once my project shifted focus, from Java to Ruby on Rails, I believe my desire to deliver results hindered my ability to see the big picture. I spent too much time trying to debug code (specifically Ruby syntax and Rspec), that if I had spent a little more time on the basics I would have had the necessary understanding to better accomplish my tasks. I fortunately realized this earlier rather than later and was able to seek the help that I needed while shifting my focus on accessible goals and work.

Manager Summary:

Let me start by noting that neither of your reviewers could think of a single Leadership Principle to list here. That's fairly impressive. But let me address the points you made above...

On Deliver Results, I noted above how others considered this one of your strengths, and also how you made a great deal of progress on an elaborate and challenging project. However, while yes, motivating others to fix problems is sometimes the way to get unblocked, whenever you're affected by a issue that's "outside your control", it's worth considering *very* carefully whether it actually is. Are there really no tools that could have sped up the Ruby build cycle? Is there nothing you could have done to fix the broken dependency builds? I'm not saying there is, in either case (I don't know!). But if the obvious route forward doesn't work, it's worthwhile looking for a more roundabout one.

Regarding Communicate Clearly, I simply don't agree with your assessment. This is usually a gimme when reviewing a strong intern, because a reviewer can always mention reticence about speaking up in meetings. I can't do that, however, because you were consistently engaged and articulate, both in asking questions and offering your own ideas. These positives were noted above under Have Backbone and Earn Trust of Others. As for not having complete clarity about requirements, etc., I'd actually tag that with Dive Deep, which for me is first and foremost about understanding WHAT is to be done, even before getting to HOW.

And that brings us to your last item. The balance between Dive Deep and Bias for Action is indeed a difficult and subtle issue, and many developers fail to notice the inherent conflict between the two principles, so it's to your credit that you do so. I think it's especially challenging to step back and look for that critical understanding when you've been on a roll, and the progress feels really good, and you want to keep driving forward, and you don't realize until you've been stuck for a while that you're trying to drive through a wall. I think the ability to recognize this and correct for it may be the most valuable thing you learned here this summer. While you don't want to be the guy who goes down the hall with a question ten times a day, recognizing the need to stop and think, or to go out and gather answers, is a tremendously important skill.

To close this out, let me quote a reviewer one more time: "It would be really difficult to list areas of improvement for Jared, as he surpasses expectations at multiple level."