Kinsley Crowdis

Crowdis@mail.etsu.edu

04/15/2024

Blue Cross Blue Shield Tenneesee

CSCI 4905

Spring 2024

During my Internship with Blue Cross Blue Shield Tennessee (BCBST) this semester, I worked with the Consumer Portals team. I created tests for API endpoints, set-up a new environment to be deployed, and created elements for their new webpages. My main goal for this internship rotation was to understand what a developer does daily.

One of my learning objectives was to learn how a team works together and collaborates. I know GitHub is a popular platform for code collaboration and I wanted to know if large companies even use the platform, and if so, how? BCBST does partially use GitHub and my team was in the beginning stages of utilizing it, as there were many meetings about how it should be used, when to create and name branches, and when pull requests should be made. These meetings answered many questions for me and at the end of my internship I began to use the repo as well. I am using the knowledge I gained from these meetings and my experience in my Software Engineering 1 and 2 project where we are using a Git repository to house our code as well. Each “feature”/user story should get its own branch and be named after that user story. Never commit to the main branch. Push/pull frequently especially if you and another team member are working on the same story. When it’s finished creating a pull request and another team member or the team lead can review the code. If the code review is good and nothing needs to be fixed, then it can be pushed into main.

Some of these steps I already understood from lessons in Software Engineering this semester, but also from experience working in Git repositories last semester for Server-Side Web Programming. I would create repositories and ask my team members to create branches and push their code to those branches to avoid merge conflicts. I would then make sure there were none and that everyone was happy with the code. Then let everyone know I was merging it into main so be ready to pull again. This system is very similar to how it worked with my internship team. I

was surprised that some team members were not familiar with a process like this, and some even continued pushing into main instead of their own branch.

My second learning objective was to know what tools businesses use. This objective ties into my first one. My team used GitHub but what else? My team used lots of tools, and I am still learning what else is used and how. There were about three for deployment purposes, one for scrum planning, lots of Microsoft Office tools for planning and retrospectives. For tech support, you needed to have a specific IDE installed and on a virtual machine. When I was creating API tests, I needed to install software so I could view the database but another one to create and run tests, and the software needed to be a specific version otherwise it would not work. It was an overwhelming process especially since most of my downloads were blocked in the beginning. I was going to program from the notepad and console if Visual Studio Code did not download!

My third learning objective for me was to understand scrum and sprints better. My team was split into two groups, Blue team and Orange team. I was a part of the Blue team. Blue team did more backend work (APIs) while Orange did more front-end work (React, CSS, etc). We were also divided into Onshore and Offshore where Onshore was BCBST employees and Offshore were contractors. Every day, except Friday, there were two stand-ups for each team (blue and orange). On Friday, there would be one stand-up for both teams. Stand-ups consisted of the development team, software quality assurance team (SQA), and product owners. One difference I noticed was that in Software Engineering it was mentioned that there would be one person as a product owner, not a committee. However, for my team, there were multiple product owners, around five or six, and more would join depending on the projects we were working on.

Every stand-up our scrum master would ask about the stories you had assigned. Our stories had points that estimated how long it would take to complete. Eight points would mean it would take the whole sprint. One- or two-point stories would take one or two days.

At first my point capacity was 8 points, like everyone else on the team, and my mentor/buddy had created stories for me to complete that were 8 points. However, once I had finished those stories and began to work on backlog items that were not created by my mentor but instead a product owner, my mentor set my capacity down to 4 points. This was because I am only working from 8 AM to 12 PM, half the time that everyone else on the team is working. My sprint capacity was reduced to reflect that, since the stories I am working on are no longer being created with that in mind.

I began working on two stories both with two points, which to our scrum master means it should take me four days to complete both. However, I had to read up on a lot of documentation (was using a new language and framework) and it took a day for me to gain access to a repository so I could begin working. This story should have been at five points for me because of my shorter work schedule. This was an issue I ran into where our scrum master insisted that I have the story completed by the end of the day or the next. This time constraint caused a lot of stress for me and when I told my mentor about how I felt, he told me that I should be more assertive and let our scrum master know that I can't complete a story in four hours, and I am not here all day. He also let me know that by standing my ground I can make a lot of things happen even if I am just a junior developer.

Another similar issue was when I had to deploy a new environment. I had been setting the environment up by creating tickets and requests by following a step-by-step document. However, once I got past those steps the document's steps became vague and I needed to ask

someone for help. I was not sure who to contact and where to go to complete the next steps. I had to ask multiple people and I contacted people on different teams and was notifying them that we were ready for deployment and there was radio silence. I had my email approved by our scrum master before it was sent, and she was expecting them to respond with what other information they needed as well. Our scrum master then told me to email another member of our team who had deployed another environment last sprint. I sent an email to that member explaining everything and providing every bit of context that I could. This email to this day has not received a response.

This environment was supposed to be deployed on Monday and it wasn't until my mentor got back from paid time off on Wednesday that it got deployed, because he knew what information the other teams needed. This situation was incredibly frustrating not only because I felt stranded and everyone I reached out to did not respond to me, but also because this situation is normalized. I talked about this feeling with my mentor and my team lead, and they essentially said, "that's just how it is." They explained that I should've gotten more help from the team but that is how some teams operate. This response is upsetting because I don't think teams should not respond to emails and messages just because it does not have the information they need, especially if they can see *intern* in my signature. If an email comes through and they are unsure, state what is required and ask for more context.

I expected people who had been working in this industry for years to have a better handle on this. It also sucked to be left out to dry by your team but rushed because it must be deployed as soon as possible.

I found that with my course work instead of me taking what I had learned from my classes and using it on the job, it was the opposite. I was taking what I was learning in my

intership and applying it in class. My mentor would often ask me what I knew and I would tell him I was familiar with the term but I had not *learned* about it yet. A lot of what he was telling me sounded like something I would learn in Software Engineering II. But, he helped me "get ahead" as he would ask me what is the next topic(s) I am learning and he would help me get familiar with it.

Career goal wise, this internship solidified for me that I do like the idea of being a back-end developer. I have had to work with CSS recently and I do not enjoy it. My only issue would be having to be in meetings almost all day long. I would have to figure out how to multitask and get work done while in meetings. My mentor said "don't be a software engineer if you don't want to sit in meetings all day." Honestly, I wouldn't mind I would just need to do better with time management.

During my exit interview with my mentor and team lead, they said a lot of really nice things and asked me to keep in touch with them. If I could stick with the Consumer Portals team for the rest of the internship rotations, I absolutely would! The position I worked in during this rotation is a position I can see myself in, and I enjoyed it.