Software Development Internship Experiential Learning Takeaways and Observations

CS4991 Capstone Report, 2024

James Nathan Barnette
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
jnb8kry@virginia.edu

ABSTRACT

MessageGears is a midsized tech startup that has struggled to keep up with much larger competitors, placing an emphasis on product development and differentiation to gain a foothold in the enterprise marketing space. I became a part of their software development team to contribute to the product development pipeline, working on many subtasks as part of larger product initiatives. I initially worked on the backend in Java, writing different tests in JUnit, Mockito, and Groovy. As I gained experience I started working on the Angular frontend as well, learning TypeScript, Selenium, HTML, and CSS styling to do so. During my last summer of employment I was on a team responsible for transitioning the frontend from Angular to the more modern React framework, in which I needed to learn Jest, JSX, and system architecture principles as a whole. I contributed roughly 1800 lines of production code shipped in Summer's 2022 product release and gained additional responsibilities in 2023 to help plan the intern program. This was a very rewarding experience which has sparked a lot of personal and professional growth.

1. INTRODUCTION

I started my internship journey as the sole intern at a tech startup with an engineering team that was nine people in its entirety. I finished my last stint four years later with three direct reports on a scrum team with eleven members. Our engineering team now totals over 70 employees. As the company I worked for grew, I grew with it. I gained invaluable insight into how software development projects are segmented into manageable bite sized pieces. How teams are structured as a company increases in size to maximize efficiency. How to communicate effectively with coworkers - both to get work done more efficiently and to make our lives easier. Not to mention any of the hundreds of practical software development skills I learned on the job. Most of these technical learning moments happen in code review. Because I was writing production code, all of it had to be meticulously combed over to make sure I wasn't introducing any new bugs to the product. More experienced developers are very nitpicky with newer employees during this process, each with their own preferences. After spending time submitting code to merge and receiving feedback, I've come to appreciate these nitpicks and explanations. Learning what I could improve in my own code, learning why it could be improved, has proven much more beneficial to my own development than the binary style of grading autograders generally use on assignments in class.

While this technical report will detail the intricacies of the work I completed at MessageGears, there is simply too much ground to cover to discuss every single small

task I completed during my employment. Instead, I will focus on my contributions from this previous summer and briefly discuss the overall technologies and frameworks that makeup the MessageGears development stack. Many different decisions are made in designing the architecture for such a large scale enterprise software and I gained a lot as a developer by understanding why MessageGears uses the technologies that it does.

2. RELATED WORKS

Agile development is an iterative and flexible approach to software development that emphasizes collaboration, adaptability, and customer satisfaction. Scrum is one of the most popular frameworks within the agile methodology. The Scrum process is designed to help teams deliver high-quality software by breaking down complex projects into smaller, more manageable tasks. All of these tasks are compiled into a product backlog, which a product manager combs through to determine an individual sprint's work. A sprint is a time-boxed iteration during which a specific set of tasks or work items are supposed to be completed. They typically last between two to four weeks. Each sprint has its own planning period, daily standups, and a retrospective review period (Sutherland, 2014). The engineering teams I was placed on all used a Scrum workflow, and it was instrumental in breaking up work into manageable tasks for team members to individually complete.

My teams also had issues with scaling out the Angular frontend at times. Parts of the same SPA application were fragmented and it was difficult to reuse code. React is a language meant to solve some of these issues in web development. It is incredibly modular, easy to understand, easy to maintain, and easy to scale (Technologies, 2023). My team lead was also very excited to be able to use Jest for unit testing on the frontend.

3. PROJECT DESIGN

Full stack engineering required me to learn and use many different development frameworks across repositories. When I first started working at MessageGears I was mainly writing backend unit tests in a language called Groovy, an offshoot of Java used for testing. I also converted out of date unit tests from a testing framework called EasyMock to a framework named Mockito. I then graduated to working on small bug fixes and easier to complete tasks that the very limited development team had in their sprint. Initially, these tasks and fixes were completely written in Java and located on the backend. Until they weren't. Fortunately, my team was starved enough for manpower that they gave me more responsibility whenever reasonably possible. I started writing API tests - learning about REST controllers, mocking data store services. I learned how the frontend queries the backend to fetch data it needs to display. Started to write javascript logic on the frontend to save myself from convoluted fixes on the backend. Studied HTML and CSS to touch up and build out my own web pages. By the end of my second internship, pieces of the puzzle that is full stack development had eased into place.

3.1 SYSTEM ARCHITECTURE REVIEW

The heart of MessageGears' codebase is split into 3 repositories bundled together and shipped to clients as a single product. The three repositories are separate buckets for the frontend, backend, and cloud code to be housed. The frontend houses a Single Page Application (SPA) which dynamically loads information from the backend (the database) and displays it on a styled web page. I did not work much on the cloud side of things, but the cloud repository mostly dealt with data transfer and security, making sure everything was properly encrypted and secure.

Some older web pages were also written server-side, in PHP, on the backend. These

pages were notoriously hard to debug and were very slow compared to SPA pages, load times sometimes being over ten seconds for large queries. Because of these issues and in an effort to update the UI of the app as a whole, an initiative to convert old PHP pages began. After mulling over options this process morphed into an initiative to convert the entire MessageGears frontend from Angular to React. React was chosen in part because of the breadth of the capabilities of its components. The MessageGears app is large and spread out, and it was quite difficult to reuse code in some situations. The nature of components in react was supposed to ease these problems.

3.2 REACT CONVERSION PROCESS

This past summer I helped plan the internship program at MessageGears. I coordinated with management to discuss an onboarding process for the group and personally onboarded half of the group for the first week of work this summer. Afterwards, I had three direct intern reports while working on converting our front-end webapp from a more out of date framework, Angular, to the more modernized React framework. The conversion from Angular to React was slated to be a year-long process, whose start date happened to align with the beginning of my last summer as an intern. To do this conversion I had to learn the new React framework, utilize the technical skills I learned from Angular, and share my knowledge of how both connected to the backend to make the transition seamless.

Initially, my team of interns was set to work as its own independent unit, with questions from interns bubbling up through me to our team lead. This carried on for a few weeks until we ran out of intern work in the backlog. We then assimilated into the dev team and acted as normal full fledged software developers working on the new React repository.

Due to application size as well as dependency limitations, we had to create every single component from scratch for the new UI. We split the creation of components into separate buckets, focusing on creating the smallest, most specific building blocks for pages first. We made various buttons, accordion designs, type ahead inputs, and anything else that product requested be in the new pages we were focusing on. The building blocks, called cogs, then went into larger more comprehensive components called gears. These gears were then placed on pages and styled accordingly.

4. RESULTS

Myself and the three interns I worked with on this team ended up converting a total of four pages over a two month period, compared to an additional ten pages completed by the rest of the full time, seven-person strong development team. Part of the reason it took so long to get a page done was because of the “from scratch” nature of our work. It became much easier to build out pages as soon as we had the necessary components completed to do so.

This level of contribution from a team of interns was very unexpected - as much was said in our exit interviews - and yet our project velocity was still behind projections when I left MessageGears. We as a team placed a great deal of emphasis on quality code output and reusability, wanting to ensure the new repository had good bones. The nature of agile development often causes an inability to predict the length of projects, showcased by the fact that even with an added four employees to increase our production output, we as a team still fell short of our goals.

REFERENCES

Sutherland, J. (2014). *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Currency.

Technologies, T. (2023, December 20). *Top 5 benefits of REACT for interactive web development*. Medium. https://medium.com/@marketing_26756/top-5-benefits-of-react-for-interactive-web-development-69875dd8cab5