Skills Needed by Software Engineering Interns

Technical Report
Presented to the Faculty of the
School of Engineering and Applied Science
University of Virginia

By

Rachel Lee

May 11, 2022

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: _____

Approved: _____ Date _____
Rosanne Vrugtman, Department of Computer Science

Skills Needed by Software Engineering Interns

CS4991 Capstone, 2021

Rachel Lee
Computer Science
The University of Virginia

School of Engineering and Applied Science
Charlottesville, Virginia USA
rl3bcw@virginia.edu

ABSTRACT

Fortitude Technologies LLC, a small government contracting company that works in the intelligence branch, needed an improved internal platform to match open positions to current employees or applicants based on wanted skill sets. The other interns and I were assigned to develop this internal platform since we did not have security clearance. The platform utilized React, AWS services, MySQL, and various types of best software development practices. There seems to be a disconnect in what students think working as a software developer looks like in the industry. Being a software developer is so much more than coding; you are expected to review others' code, test your code thoroughly, and most importantly, plan your implementation without guidance. Teaching more industry practices like Agile and full stack development and Continuous Integration as well as coursework that requires students to freely think about their implementation without strict guidelines, would greatly improve students' confidence when going into the industry.

1 INTRODUCTION

Due to COVID-19, security clearances are very hard to obtain, especially for interns. To solve this problem, Fortitude decided to create a project which was to develop an internal platform to quickly obtain information about open government positions, analyze the most wanted skillsets and predict how to get ahead of the competition. Fortitude's goal with this project was to automate the email process in a serverless application.

2 RELATED WORK

Near the end of my internship, the team expressed their desire to put all the interns in for security clearances, but due to COVID-19 and the high-demand for security clearances, the government had shut-off the ability to request for a security clearance.

While the use of AWS services, MySQL, and Python were laid out for us interns in this project by full-time employees, we were given the flexibility to choose between either React or Vue.js for our front-end application. We chose React for three reasons: curiosity, popularity, and project scale.

The other interns and I discussed in the beginning of our internship that we all had some experience in React but would like to have more extensive hands-on experience to become more comfortable. Another aspect that we considered was React's popularity in the industry, it was developed by Meta (formerly known as Facebook) and has become one of the most popular JavaScript frameworks to develop scalable applications [1, 2]; it was also helpful that a full-time employee on the Fortitude team had extensive experience in React and would be able to assist us with any issues that we were experiencing. While doing more research, we also found that in-terms of project scale, React performs better for projects that are larger in nature and with higher complexity [1]. We knew that our project would be developed well past our internship timeline and would only increase in complexity in-terms of all the different features that the team wanted to implement with this product. With all these factors in-mind, we decided that React was the best technology framework for us to develop this project with.

3 PROJECT DESIGN

The internal platform utilized the whole stack; we used React JS for the frontend, AWS Lambda service (Python), Simple Email and Simple Queue Service, S3 bucket, API Gateway, and Serverless Application Model for the backend, and the Aurora RDS MySQL instance as our database. Other tools that my team used were Gitlab (Version Control), Continuous Integration - the practice of automating the integration of code changes from multiple developers into a single software project, Agile development - a type of software development that emphasizes iterative development, and Junit/Integration testing.

To deploy all the resources listed above, we used the AWS Serverless Application Model. This resource allows resources to be easily listed without having to create every resource in an AWS console.

The platform workflow aimed to automate the email process. Once an email was received with a title of 'Open Positions,' that would trigger the AWS Simple Email Service to notify the Lambdas that information needed to be processed and inserted into the database. Our platform had 8 processing and inserting lambdas: Process/Insert Metadata, Process/Insert Positions, Process/Insert Skills, and Process/Insert Skills Analysis. The lambdas send

information to each other through an API Gateway. The API Gateway serves as a passage for lambdas to exchange important information through. The process lambdas always make POST requests to an API Gateway URL for the insert lambdas to receive through GET requests.

The insert lambdas insert into a relational database. We chose a relational database because each table had a unique email ID that was shared throughout all the database tables for organization reasons. The tables consisted of a metadata, positions, skills, and skills analysis table.

The Process Metadata Lambda extracts information about the email itself and the position attachment files and stores that information in an S3 bucket. The metadata parsed consists of the sender, time sent, email subject, etc., but most importantly, the email UUID and S3 bucket ID. The email UUID keeps track of a specific email between the lambdas and throughout the database; it is randomly generated within the Process Metadata Lambda. The S3 bucket ID is a medium in which other lambdas can access information. The lambda also parses the attachment files and assigns each position file with a number. After all the metadata has been parsed and the position attachments have been stored into an S3 bucket, it will send the metadata information as POST data to the API Gateway URL. After successfully posting data, the Insert Metadata Lambda inserts the metadata into the metadata table within the database. After, the Process Positions Lambda will be triggered with the email UUID and S3 bucket that stores the attachments.

The Process Positions Lambda extracts position specific information from the S3 bucket. After the position information is parsed, it sends all the position numbers and titles as POST data to the Gateway URL for the Insert Positions Lambda to insert into the positions table in the database. After that is finished, the Process Skills Lambda is triggered with the position number, title, email UUID, and S3 bucket ID.

The Process Skills Lambda extracts the skills wanted by each of the positions from the same S3 bucket that the Positions Lambda used. Once the skills have been parsed, it will send that as POST data to the Insert Skills Lambda to then insert into the skills table in the database. Then, the Process Skills Analysis lambda is triggered with the skills and position data and email UUID.

The Process Skills Analysis lambda weighs skill sets and outputs the top 5 skills, regions, and positions. Once that is done, it will trigger the Insert Skills Analysis Lambda to update the positions table with analytical information each job position to then reference from the skills analysis table which contains a numbered list of the skills (which is also inserted into at this step). Finally, a Simple Notification Topic will send an email to the Fortitude Recruiter to notify them that new positions are ready to be seen. All this information can be seen on a front-facing React Web Application.

The web application user interface was created using the React framework. A user, a Fortitude employee, would sign in using their Fortitude Gmail account and would be able to see a main page greeting them. Two other pages within the web page would be an emails page with all the information about the email, the position information, and required skills. Each position includes the original attachment as well. All information displayed is from the database.

Testing was also an integral part of our design in that the Fortitude team wanted to utilize Continuous Integration (CI) to ensure that our platform was running smoothly per deployment. I set out to implement both the front-end and back-end tests.

I didn't focus too much on the front end, our web app was very simple, so I wrote Snapshot tests for the UI. A snapshot test in React basically uses a 'snapshot,' or an HTML section of code, to compare against what is rendered in the UI. For example, if I expected 'Welcome to Virtual Recruiter, Rachel!" I would have to make a separate snapshot file of that portion of HTML code to compare against what is rendered when the test is being performed. I wrote a few snapshot tests per viewing page within our web application.

For the back end, I mainly wrote Unit tests that mocked AWS services. Using the moto library, I was able to mock resources and test the workflow of each of the lambdas implemented. The API Gateway also had to be mocked by creating a separate file which would automatically return a status code of 200 indicating a correct gateway return. The different scenarios tested for each lambda were with a valid email and an invalid email, the invalid email obviously returning an invalid gateway URL which I embedded within the separate gateway file for testing. For each deployment into Gitlab, the CI would run the tests that I wrote to ensure our code quality was up to our standards.

Agile development was the development process in which we operated. Our schedule was split amongst weekly sprints, with a full-time employee writing weekly tasks within a Gitlab boards for the other interns and me to organize into what we could finish. At the end of each week, we were expected to have deployed a 'mini' platform with a new feature.

4 RESULTS

Today, this platform is actively being updated by Fortitude employees and being used by the Fortitude recruiter.

A full deployable platform was ready to be in use with a fully functioning front-end and improved serverless back-end logic. Testing was incorporated with front-end coverage improving from 0% to 50% and back-end coverage improving from 55% to 83%. Continuous integration was successfully deployed, ensuring faster code quality checks per deploy to Gitlab.

5 CONCLUSIONS

My project at Fortitude benefited the company, but also benefitted my skills and abilities as a software engineer. I participated in research to back up technology choices and discussed those choices with full-time software engineers. I developed a product full-stack and was able to fully understand how each portion of the product worked together. I was also able to practice code safety precautions with in-depth code testing for the front and back-end. I came out as a stronger software engineer with sharper technical skills that are in high demand in the industry.

6 FUTURE WORK

Small improvements to this project include more extensive testing to ensure that code quality is up to par with the team's expectations. High level improvements to this project include some form of business intelligence to predict future skillsets wanted by the government instead of manually thinking of what future skillsets could be wanted. This can be accomplished by analyzing the current data being processed in the platform and processing that data within a machine learning algorithm to predict future wanted skillsets. This was discussed during my internship to compete against other companies in order to fill open government positions quicker and more efficiently. All in all, this internal platform can be elevated with cutting edge machine learning algorithms to make Fortitude Technologies the most competitive government contracting company.

7 UVA COMPUTER SCIENCE PROGRAM EVALUATION

The courses that would have helped me at Fortitude Technologies were Cloud Computing (CS 4740) and Web Programming languages (CS 4640) when working with AWS Cloud Services and front-end development – but I had not yet taken those courses until the Fall 2021 semester. Other than that, Databases (CS 4750), Software Development Essentials (CS 2501 Pilot), and Software Testing (CS 3250) were courses that prepared me at Fortitude.

While it was great to have those courses before my internship, I felt unprepared when it came to developing a product full stack; that is, developing a product from front-end to back-end to server. I didn't understand how all those pieces would fit together. I also felt unprepared when understanding good workplace habits such as Agile development. Our team participated in weekly sprints and daily standups to achieve iterative development—having a small deployable project every week as well as automating code at the very start of the project (Continuous Integration) to make sure our code was clean and testable.

While working at Fortitude, one of the hardest challenges that I faced was getting familiar with all the technologies, frameworks, and software development practices utilized in this project. However, by getting hands-on experience, I was able to fully gain experience in all the aspects of the project.

Other software development interns should be equipped and ready to understand the whole tech stack (even if they are not working on a full stack project) and should be knowledgeable of the best kinds of software development practices. Students would benefit from being taught real life knowledge and practices of the software industry so that students going into their first software development internship will have an easier time onboarding into a team.