Summer 2024 Technology Internship: Internal Data Discrepancies in Customer Servicing Platform

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Alex Catahan

Spring, 2024

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

Briana Morrison, Department of Computer Science

Summer 2024 Technology Internship: Internal Data Discrepancies in Customer Servicing Platform

CS4991 Capstone Report, 2023

Alex Catahan
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
rac4sq@virginia.edu

ABSTRACT

The company I worked for, a prominent North American banking company, has struggled with a critical issue in its newly developed customer agent servicing platform —an issue involving internal data discrepancies that affects around 20,000 customers. To address the challenges stemming from this issue, my team and I utilized Vue.js and Node.js to develop the front-end and back-end of a system that detects an internal data discrepancy in a customer's account and promptly notifies the customer service agent using a modal. My team utilized an agile methodology to streamline and coordinate our workflow, and we used different javascript libraries to develop and test our code. I also practiced cross-team communication skills, and gained the ability to work in an enormous codebase. This project is expected to reduce customer agent incident tickets by 5-10 weekly, and decrease agent handling time by an average of 20%. This detection system will be extended to other components of the customer agent servicing platform, and will be maintained as the platform continues to develop.

1. INTRODUCTION

Customer service can be a frustrating experience for both customers and employees, especially if the account attempting to be serviced is causing the system to malfunction. This is exactly what happened to customers of my company with accounts with multiple Enterprise Servicing Customer Identification numbers (ESCIDS). This problem stems from data discrepancies within internal systems, and can be caused by a variety of reasons but most often is a result of user error when inputting a personal identifiable number such as a social security number or birthday.

During my internship, my team performed analysis of this issue and determined that there were around 100 daily occurrences. The prevalence of this issue caused many customer service agent incident tickets to be submitted, taking up developer time to diagnose this issue because the existing system provided extremely vague feedback. Customer experiences were also negative when this issue occurred, as the customer service agents were of no help.

2. RELATED WORKS

Many articles have covered the importance of having good customer service, which was the motivation behind my summer project. In fact, Lemons, et. al. (2016) conducted a recent study which showed "improving the customer experience received the most number one rankings when executives were asked about their top priorities." As customer service is a key feature in the overall customer experience, studies and sentiments such as this served as the motivation to resolve a critical issue in the customer servicing platform, which led to the creation of my summer project.

Data discrepancies, such as the one affecting my company, is not a new issue for corporations. Bozeman (2019) asserts that across corporations, "approximately 20% of the average database is dirty." He also discusses potential causes for this issue, one of which is relevant for my work—"Inaccurate Customer Data"—and suggests "companies to invest in data enrichment initiatives," which aligns with the goal given to my summer intern team.

3. PROJECT DESIGN

The following sections describe the design of my project, including: overview of the system's architecture; challenges faced when onboarding for development; coding of a solution; and key components of the process.

3.1 Overview of System's Architecture

The customer service platform my team and I worked on utilized a micro-frontend architecture. This meant the platform was designed so the front-end components of the web application were broken up into smaller pieces, typically by functionality, allowing parallel development on different functions, as well as independent testing and deployment. This approach ultimately allowed for more flexibility and agility in development in comparison to a monolithic architecture.

Effectively, this meant there are many GitHub repositories that house the code for every separate front-end element. For example, front-end elements that dealt with "payments" would be in a different GitHub repository than elements that dealt with "customer profiles," allowing both sections to be developed independently while simultaneously ensuring complications would be isolated to a single repository. Understating this architecture and how the code relates to what is actually displayed on the web application was critical for successful development.

3.2 Onboarding Challenges

In order to add a feature to the customer servicing platform, I first had to set up the development environment on my laptop. This lengthy process involved gaining company access to needed entitlements, using many terminal commands to download needed software, and debugging when the environment failed to work. Once my development environment was set up properly, my team and I had to understand how the platform's codebase was structured. This involved quickly clicking through dozens of files while analyzing the code and attempting to trace the front-end features to lines of code.

Once I gained a sufficient understanding, I began developing; however, I had to adhere to the current standards of coding, as well as adopt the codebases' technology stack. For the whole team, this meant learning new javascript frameworks, Node.js and Vue.js.

3.3 Coding Solution

My team's solution for the data-discrepancy notification system was split into four parts: orchestrating back-end infrastructure; integrating the user-interface (UI); building the UI; then testing the system.

We first orchestrated the back-end infrastructure. This meant adding an Application Programming Interface (API) route that queried a back-end data source, telling us whether an account had a data-discrepancy. We utilized an existing API endpoint that returned an error code when queried with a customer ID of a data discrepant account. While an existing route already used this endpoint, the parameters of the query varied slightly with the needs of our project, so we simply used the existing code as a template and created our new route.

Once the back-end infrastructure was created we began integrating it with the UI. This meant coding the necessary infrastructure for our newly-created API route to be called when the landing page of a customer's account is loaded on the servicing website. This step involved creating new VueX stores and dispatching actions that invoked the new API within the "created" lifecycle hook of the landing page's Vue file, efficiently updating the state of the current customer upon page load.

Once our back-end was created and integrated with this UI, we could finally build the UI. This means creating the actual features displayed on the website that an agent would see. While this may seem simple, the nature of the problem we were solving is inherently sensitive, as it dealt with personally identifiable information for customers, such as date of birth or social security numbers. Providing the steps that enabled agents to remedy the issue was a complicated legal issue, and the developers do not decide what to display. My team and I had to retrieve the information to display from another database populated with messages approved by the legal team.

Once the message to present to the agents was successfully retrieved, we created a simple modal using an internal library of front-end elements, and connected it to our UI integration.

When development was completed extensive testing was necessary to validate the functionality of our added system. We performed integration testing, where we manually went in our development environments with different accounts to check for a desired outcome. We also completed performance testing, where we used a software called JMeter to ensure our back-end could handle large loads without crashing. Throughout the process we also performed unit testing as a sanity check while actually coding.

3.4 Key Components of Process

During our process there were many smaller steps of equal importance as the main programming steps. First, my team and I conducted an analysis to understand the data-discrepancy problem more deeply, and to gauge how much of the customer servicing platform was affected. Throughout our developmental process, we had to perform unit testing for every line of code we added, which helps autonomously ensure that the code provides the correct functionality. This meant learning how to use unit test frameworks in order for our added and modified files to reach 100% code coverage.

Because the customer servicing platform used a micro-frontend architecture, each container of functionality for the platform was "owned" by a different team

within my company. Since our project's reach was well beyond the container owned by my team, our project fostered a lot of cross-team communication, which is critical for coding in industry. Many steps of our coding process had to be approved by other teams, and attendance to code reviews held by the team whose codebase we were contributing to were required to have our code be accepted into their codebase.

4. ANTICIPATED RESULTS

My team and I were able to complete our project quickly and efficiently, and by the end of the summer our code was pushed to the actual production customer servicing platform. However, there was an allow-list with employee ID's that only allowed associates overseeing our project to test it in production. While our contributions were not broadly released during my time there, my manager and product owner would continue our work for a full release of our system. Once broadly released, the new notification system was expected to reduce agent handling time by an average of 20%, as well as reduce agent incident tickets by 5-10 weekly. It was also expected for other teams to adopt our code and implement a system similar to ours in their own containers.

5. CONCLUSION

The customer servicing platform used at my company was developed in 2018. From the platform's infancy, developers knew about this data discrepancy issue; however, the issue was overlooked, possibly due to underestimating its scale. Through analysis, my team determined there were ~20,000 customers affected by this issue, and we created the pipeline to resolve this issue for agents. The ultimate goal of our project was to increase customer satisfaction and experience when encountering this issue. We not only accomplished that, but also reduced agent handling time and eliminated developer time handling a simple data-discrepancy issue.

6. FUTURE WORK

When I ended my summer internship, my project was completed and released to my manager and our product owner to be thoroughly tested in the production environment before general release. When my manager and product owner approve of our changes, it will be released to a test-team of customer service agents in order to conduct live testing with our added features. If all goes well, our added system goes into general release, where any agent will see our features if they assist a customer with a data-discrepancy issue.

REFERENCES

Bozeman, Ryan. "Common Customer Data Issues and How to Fix Them." *RevOps Agency*, impulsecreative.com/blog/common-customer-data-issues-and-how-to-fix-them. Accessed 12 Nov. 2023.

Lemon, Katherine N., and Peter C. Verhoef. "Understanding customer experience throughout the customer journey." *Journal of Marketing*, vol. 80, no. 6, 2016, pp. 69–96, https://doi.org/10.1509/jm.15.0420.