Big Bank Technology: My Experience Interning at a Big Bank

CS4991 Capstone Report, 2023

Sam Harless
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
sdh7ksu@virginia.edu

ABSTRACT
My team at Wells Fargo used a time-consuming manual process in order to turn jobs on/off ice in Autosys when maintenance needed to be done on our servers. To streamline this proccess, I created a function to send API requests to Autosys, then created a web page to access this function. To make the function, I utilized C# and .NET to first draw server information and job names from a database, and then sent API requests to Autosys based on the information retrieved. I then added a page to my team's web application so that they and other teams could access this function and turn the jobs on/off ice for maintenance. I created the front end of this website using Angular, which was able to communicate with the function built into the back end of the application. I successfuly completed this project, and now my team and others can turn jobs on/off ice for different servers with ease. The only future work needed is to add the option to select individual jobs for a specific server to turn on/off ice.

1. INTRODUCTION

Wells Fargo has over 230,000 employees, serving over 70 million customers worldwide. In terms of technology, much of the banks focus is on data. There are many teams whose main responsibility is a certain database. This work involves pipelining data into the database, cleaning and processing the data in the database, and finally pipelining the data out of the database. Since they had so many database teams, our team existed, which was in charge of the servers that hosted the plethora of databases.

One of the pain points was turning all of a server's jobs on ice, which basically pauses a job, whenever maintenance needed to be done on a server. Sometimes this prompt would come from within the team, but we also dealt with a constant stream of requests from other teams for the jobs on their database's server to be turned on ice so that they could do their own maintenance. My manager wanted to solve this pain point, so she assigned me a project to use the AutoSYS API to turn jobs on ice. The AutoSYS API allowed a user to send an API request containing credentials, a server/job, and action to take.

2. RELATED WORKS

One of the main concepts that inspired this project was self-service. Self-service technologies have become increasingly pivotal in the landscape of enterprise technology teams. These systems empower team members to perform tasks that would traditionally require manual intervention, thereby accelerating workflows and reducing operational bottlenecks. A study by the Harvard Business Review found that "81% of all customers attempt to take care of matters themselves before reaching out to a live representative" (Dixon, 2020). Though this statistic is about customers, it shows the same

sentiment, that people will try to solve their own problems before reaching out for help if they have that option.

In the context of enterprise technology, the significance of user-friendly websites extends beyond customer engagement to internal operational efficiency. As one industry expert notes: “If corporate software is easy-to-use, it helps employees perform their routine tasks. Consequently, their productivity is higher” (Miller, 2021). For teams within an organization, a well-designed internal website or portal is not just a luxury but a necessity for streamlined workflows and effective information dissemination. A user-friendly interface minimizes the learning curve and enables employees to quickly access the tools and information they need. Therefore, investing in user-friendly website design for internal enterprise applications is not just about aesthetics but also about optimizing performance and resource utilization.

3. PROJECT DESIGN

I followed an iterative design process when making my solution. I first worked on the functionality to ice one job on one server by sending an API request to the AutoSYS API. This functionality was built using a C#/.NET function. Once I had this functionality working, I extended the solution to loop over all of the jobs on a server, turning them all on/off ice. Each server required a different endpoint and credentials, so I next worked on pulling the server information (including a list of jobs on the server) from my teams internal application database using SQL. Once I was able to pull from the database, I could choose a server as a parameter, which would pull the endpoint, needed credentials, and a list of jobs. Then I could send API requests for each of the jobs using the endpoint and credentials.

Now that I had the functionality of the AutoSYS API working, I needed to create an easy way for my team members and others to access the functionality. My team already had a web application with a good deal of functionality packed in it, including a database and login features. Therefore, all I needed to do was add a page to the application, and it would have access to the database and login features. Before starting to code the user interface of the application, I designed templates showing what the user interface should look like. I used Axure, following the proper user interface guidelines; then presented these three options to my manager, got feedback, and started coding the web page.

I used Angular/TypeScript to develop the front end of the web page. This web page first checked to make sure the user was logged in, and if not sent them to a login page. If the user was logged in, the web page would query the database, pulling a list of the servers the user had access to put on/off ice. Then, a dropdown menu would be populated with the retrieved list, so that a user could choose a server. Once the user had chosen a server, they could click the submit button, which triggered the API functionality described above, passing in the selected server as a parameter. Finally, the API function returned a success/error message, which I displayed on the webpage, so that the user could see the results of their action.

4. RESULTS

Immediatly after I finished, my team started using the solution instead of manually turning the jobs on/off ice, which saved a lot of time. A monotonous task that ussually took 2-3 minutes could now be done with the push of a button. When I left, my team was beginning to tell other teams about the functionality, which would mean our team no longer had to deal with these requests. This saved an estimated 20 minutes per day because other teams could now do self-service instead of needing help.

5. CONCLUSION

The project undertaken during my internship at Wells Fargo represents a significant leap in operational efficiency and user autonomy. By leveraging the capabilities of the AutoSYS API, C#, .NET, and Angular, I was able to transform a tedious, manual process of managing server maintenance tasks into a streamlined, automated system. This development not only saved valuable time for our team—reducing a 2-3 minute task to a mere button push—but also empowered other teams with the ability to manage their server maintenance autonomously. The introduction of this tool marks a notable advancement in internal technological capabilities, aligning with the broader trend in the industry towards self-service technologies.

The successful implementation of this project underscores the importance of user-friendly, self-service technologies within enterprise environments. As organizations continue to evolve, the ability to quickly adapt and provide efficient, automated solutions to operational challenges becomes paramount. My experience at Wells Fargo has highlighted the impactful role that thoughtful software development can play in enhancing productivity and operational efficiency. This project not only served as a valuable learning experience but also contributed meaningfully to the bank's ongoing efforts to optimize its technological infrastructure. The potential for further enhancements, such as the ability to select individual jobs for specific servers, offers an exciting avenue for future work, promising even greater efficiency and customization in server management processes.

6. FUTURE WORK

The server job management tool developed during my internship at Wells Fargo offers numerous opportunities for further enhancement and expansion. The immediate next step is to enable the selection of individual jobs for specific servers. This feature will allow users to have more granular control over the jobs they wish to turn on/off ice, tailoring the process to their specific needs and reducing the risk of affecting unintended jobs during maintenance. Some other possible areas for future work include integration with other servers and systems, customization, and mobile application development.

REFERENCES

Dixon, M. (2020, September 15). Kick-Ass Customer Service. https://hbr.org/2017/01/kick-ass-customer-service

Miller, A. (2021, July 28). *Enterprise UX: The value of usability for enterprise software*. UX Magazine. https://uxmag.com/articles/enterprise-ux-the-value-of-usability-for-enterprise-software