Dynamic Web Dashboards Using React and Single-SPA

Software Development Practices

COSC 4354

Profs. Venkat Subramanian and Chang Yun

Team:

Debugger Kings

Client Name: Halliburton Company

Submitted By

Vanessa Huynh

Majid Al-Darraji

Mason Zhang

Daniel Viray

Blake Kushwaha

Rafael Galvan

We chose, as our development project, the proposal from Halliburton to create a dynamic web dashboard that combined responsiveness and ease of development with extensive flexibility and portability. The Single Page Application or SPA, is basically a framework for bringing together multiple JavaScript microfrontends in a frontend application. Using Single-SPA's unique import map and lazy loading functionality, we were able to embed in a root React dashboard app six individual widgets, including an interactive status bar widget, from 4 different frameworks, all which can be loaded and unloaded without refreshing the page.

This approach carries well naturally over to Docker, where each widget and its corresponding .NET backend can be run in their own container and deployed at will. In addition, Single-Spa allowed us to develop each microfrontend largely independently, as the root application doesn't inherently "care" about what framework it's being asked to load so long as it's JavaScript. The choice of a React root as the dashboard stemmed from React's ability to manage component lifecycles, as well as its performance advantages for applications with involved state. For our individual widgets, each of which correspond to different elements of an oil rig's operations, we chose as our frameworks two written in React, two including the status bar written in Svelte, and one each in Angular and Vue. Our project is also Typescript-based due to its advantages. Our overall goals in the project were to capture relevant aspects of an Oil Rig, from technical to administrative data, and display it in a containerized and distributable environment, showing a case in point how different aggregated frontends and backend sources can be seamlessly delivered to a single application viewing.

During the evolution of this project, we all learned many things both technically, about ourselves, and about the profession of software development in general, and we each had our own individual experiences as well as sentiments we all shared. Each of us were grateful to have the ability to familiarize ourselves with most of the major frameworks used in web development today, as well as tools such as Docker, Jenkins, npm, and even Git. During this project, we were all called upon to use our relative strengths and weaknesses to the benefit of the project. Stronger programmers such as Majid and Rafael helped less-experienced developers such as Daniel and Blake with helpful tips on the major challenges of web development, such as project initiation and backend integration, allowing them to challenge themselves productively and make rapid improvement. We all were faced with the mental challenge of managing this project alongside

time-intensive courses such as RTS, Virtual Reality, and Numerical Methods. For many team members, this was the first full stack app we had worked on, and it taught us how to take a full stack project from the very beginning, from how to define a good project structure to the integration of microfrontends and backends, and putting it all together with single-spa. Given that each of us more or less had pre-defined responsibilities pertaining to our widgets or certain backends, we all had our own individual challenges with our own widgets, as well as integration and design challenges for the larger project.

However, strong and efficient team leadership from our experienced programmers and our team lead Vanessa allowed us to eventually surmount the biggest technical hurdles. Key technical accomplishments were getting widgets to render and switching dashboards in a SPA style (without page refresh), unloading parcels through window events, and managing widget's load state and layout through URL parameters. Another crucial addition to the project was the implementation of PNPM for efficient package management, ensuring streamlined development workflows. We also accomplished configuring Prettier and ESLint to enforce code consistency and best practices, as well as a combined test suite in Jest for our entire project. We crafted the root app in React and meticulously designed the dashboard using Tailwind CSS and Preline, educating ourselves in the finer nuances of CSS and UI dependencies. Throughout the process, our coordination and productivity was enhanced by regular meetings 3 times a week in an Agile-methodology fashion. New features and TODOs were discussed with Bipin and Edgar from Halliburton and added on a weekly basis on the team Trello page.

Despite clinical organization and efforts to document and carefully vet all contributions to main, particular issues did come up during the long development process. In the words of Vanessa, “A lot of issues that cropped up had to do with setting up testing with Jest. It seemed like every app we integrated into the root had some problem involving Jest that required extra work to set up automated testing, like missing/outdated dependencies or incorrect setup of config files. We had some problems with the Jenkins server in the beginning as well, since there were mismatched Jest versions. We were able to iron out all of these problems, though.” Such hiccups are normal in projects involving essential frameworks such as Jest and ESLint, and throughout the semester we were extremely grateful to have the opportunity to experience and troubleshoot these issues together in a forgiving academic environment, rather than in a high pressure

real-world situation where we might be working individually. Another one of the frustrating but extremely normal issues we ran into was two of our less experienced programmers, Daniel and Blake, having difficulties continuously integrating with the mother app which our aces seemed to add a new feature to or refactor every week. Again this shows how this project allowed us to learn and grow ourselves at an even faster rate than we would working independently. Blake in particular was new to the Git workflow which caused him to be reticent of contributing in the beginning, however by the end of the semester he was about to fluently merge main into his PR, deal with conflicts, and create successful contributions on time. In Daniel's words, "Another of the major issues I ran into was unit testing. As I had little experience actually writing tests, I had to dive right in and learn on the go. It had taken me a significant amount of time, but I'm glad that I had taken the time and effort to do them." Even our skilled programmers such as Majid had issues implementing lazy loading multiple widgets at a time because of React state re-rendering. Angular was also a particularly finicky widget as well, with Mason reporting that "using npm workspace for angular requires double downloading for the packages," and overall was notoriously difficult to get working in both local development and Docker, a challenge we also all faced and were glad to tackle before entering a professional environment.

Overall, our experience and personal reward from the project is best summed up by Rafael: "Through these endeavors, I've not only expanded my technical skill set but also fostered a deeper appreciation for teamwork and perseverance in the face of challenges." Almost all of our team members are graduating this semester or in the fall, and getting to work on a real production-quality software for a discerning and cutting-edge company such as Halliburton will no doubt be one of the experiences we look back on as spearheading our development as computer scientists and contributing members of society.