Redesigning of Benefit Enrollment Application User Interface at ADP LLC

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

Areeba Kausar

Fall, 2021

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

Signature _____*Areeba Kausar*_____ Date _Dec/6/2021_
Areeba Kausar

Approved _____*Daniel Graham*_____ Date _Dec/6/2021_
Daniel Graham, Department of Computer Science

Redesigning of Benefit Enrollment Application User Interface at ADP LLC

Abstract

As part of my internship at ADP LLC [1] in Alpharetta, GA, I learned the in-house technology stacks and frameworks created by ADP engineers. I then used those frameworks to update and recreate part of the current application, removing the inefficient and soon-to-be-outdated components. This improved end user's application experience as the newer components offer greater responsiveness, accessibility and a cleaner interface design.

1. INTRODUCTION

Software applications are known to be in a perpetual state of Beta. This refers to the constant improvement and maintenance work required by all finished software products. During the summer of 2021, I worked as a Software Engineering Intern at ADP LLC, where I worked with Global Product and Tech Division to facilitate maintenance and enhancement of existing ADP applications. ADP or Automatic Data Processing is a Fortune 500 Company that provides workforce management tools to 30,000+ businesses around the world. My team worked on the Benefit Enrollment Application offered under the Workforce Now Suite. This application allows corporate clients to manage, edit and track various benefits such as health or life insurances, saving and retirement accounts and custom perk packages that their employees and beneficiaries are eligible to earn. However, given the volatile nature of benefit regulations/packages combined with the large customer base enforces ADP to constantly update and improve its application. My tasks over the summer aligned with ADP's general goal to make its various applications' interfaces more uniform and undeviating. The redesign aimed to provide the existing information in a more concise, accessible and responsive manner on devices of varying sizes.

2. RELATED WORK

Reenskaug et al. [2008] declares that a programmer's mental model must align with a users' approach during human computer interactions. A good programmer facilitates the user's ability to think and act while using a graphical interface by providing accurate real-time feedback about user manipulation effects on program state. My task involved finding and designing newer components that better followed these principals and improved the human computer interaction aspect of application.

3. PROJECT LEARNING AND DESIGN:

The internship project could be divided into three distinct, yet interrelated parts, each with its own set of challenges, as discussed below.

a. Project Learning and Design

(I) The first month of my internship consisted of learning the technologies and tools required for the job. I divided this self-guided training process into three further parts and started by familiarizing myself with the complex application features before learning more about codebase. This step required that I request an application demo from a senior testing engineer on my team and improve my higher-level understanding of the system.

Benefit Enrollment Application followed the general model-view-controller design pattern where the user interface, input and system logic are separated to create a compartmentalized codebase with clean inner interactions [2]. Thus, I started learning this pattern alongside well-known web frameworks that were utilized by my team: namely React, Redux and TypeScript. I supplemented the official documentation and tutorials with independent

resources available online including simple project assignments that taught me those technologies in a hands-on manner.

The final step in Part I was to learn the inhouse libraries and frameworks created within ADP that were required for my tasks. Most notable was a framework called MDF. While the applications written in MDF followed the standardized MVC pattern and included React, Redux and Typescript code, they were subjected to additional MDF constraints and conventions. Since documentation and training guides were the only available resources for MDF framework, I often found myself looking at code repositories of other MDF based ADP applications to fully understand its functionality.

(II) I was also required to learn and adapt the coding practices utilized by my team. The most notable was the scrum methodology [3] which required the team to meet every day, report the completed tasks and receive the next assignment. Scrum forced developers to fully understand their assignments and plan its execution before starting the code process. My assignments could be divided into two categories, and I planned for them accordingly:

First, some of my simpler tasks required that I replace small components, such as tables, menus, and grids on current web pages with newer alternatives. Hence, the planning process involved research on the older as well as newer component options to ensure the newer component replicated the full functionality of older ones. It also required that I study any relevant APIs or methods providing data to the components.

Second, I was given a redesigned user interface for the benefit of enrollment application's central dashboard and asked to code a prototype implementation. The research phase for this task involved reading through various ADP component libraries to look for grids, tables, buttons and

other user interface components that fully replicated the new design yet also assimilated within the existing code base.

(III) The final step was the development and deployment of the updated components. Two crucial elements at this stage were correctness and cleanliness of my code contributions. To ensure correctness, I utilized unit testing and problem decomposition, breaking my tasks into smaller coding problems and solving one at a time. Given the large size of the code repository, there were several ways to implement the code. To ensure cleanliness, I closely observed and mimicked the structuring patterns and code convention used by my manager and mentors.

b. Challenges

The experience of working on a large-scale real-world application with experienced engineers came with its own challenges. As a Computer Science student, I have coded small- to medium-sized projects from scratch or boilerplate code. However, at ADP, I was exposed to a code repository with millions of lines of code that I had to contribute to. I soon learned the importance of prioritizing relevant parts of the application when gaining a higher-level understanding of codebase and balancing architectural understanding with code delivery.

Another challenge of working on a large-scale application with several interrelated subsystems and its own development ecosystem was dealing with downtimes and learning to work around them. As a computer science student, working on an assignment can be as simple as opening the code editor and writing the code. However, at ADP, I had to learn to work around team-wide bugs and issues preventing code deployment and testing.

4. RESULTS

ADP rolls out weekly updates to the suite and all updates that I incorporated into the existing application were merged and released in the next version of the application for use by corporate clients. The coded prototypes that I created for the redesigned dashboard was part of a longer project and my prototype was retained by my team for further expansion and eventual release.

5. CONCLUSIONS

This internship was my first experience of working in software development industry on a large-scale application. This position gave me a clear look into day-to-day life and tasks of a software developer and helped me understand how the professional duties compare to the academics. The technical and social lessons I learned from this position are invaluable for my future career.

6. COURSEWORK EVALUATION

The coursework at UVA prepared me very well for my summer internship. This preparation could be categorized into technical and fundamental skills that I learned from my coursework.

One of the major technical skills that UVA equipped me with was Scrum and Agile methodology. Since CS 3240 - Advanced Software Development Methods requires students to create an entire application in a team environment using these methods to organize the workflow, it was easier for me to pick up the concepts of user stories, sprints and daily stand-up meetings at the internship.

Having gone through rigorous UVA courses like CS 2150 - Programming and Data Representation, I was able to manage my time well and quickly create plans to learn the new languages required for the project. The process of learning the new languages was made easier by the Coding and Object-Oriented programming Fundamentals that I learned in Introductory CS Courses.

One possible area of improvement for the CS Curriculum would be to offer a React-based CS Elective. Since MERN (MongoDB, Express, React, Node.js) Stack is quickly becoming the industry standard for JavaScript based development, students can professionally benefit from learning these skills in a classroom environment.

7. FUTURE WORK

Due to the short length of my internship, the focus was on the timely completion of the deliverable. As a result, there are still many unexplored avenues for future expansion of my code contributions. For example, given more time, I would have delved into the need for component integration with ADP's built-in accessibility tools such as the application-based screen reader. This functionality can be integrated in later versions of the application. Moreover, the components I worked on can be redesigned with future releases of external and inhouse libraries, further improving the user experience.

REFERENCES

[1] ADP. Automatic Data Processing LLC. Retrieved from https://www.adp.com

[2] Reenskaug, Trygve; Coplien, James O. (20 March 2009). "The DCI Architecture: A New Vision of Object-Oriented Programming". Artima Developer.

[3] Schwaber, Ken; Sutherland, Jeff (November 2017), The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game, Agile Alliance