Affirm Coding Challenge Write-Up

Michael Quint

1. In total I spent about 5-6 hours coding the credit card submission form. I know that I went beyond the expected time limit but I was having fun building the component and challenging myself. I learned a lot throughout the exercise and there were many times where I faced unexpected hurdles and had to refactor my code to optimize for different use cases. One of the main reasons I love react is that there is so much documentation and support available online that it is simple to find specific information for a unique problem I needed to solve. Before I began to actually write code I took about 15-30s minutes to plan out my approach. The wireframe gave me a good idea of the individual components I would create, and I read through the instructions a couple of times to get a solid grasp on the various component validation requirements.
2. Even if a form is functional, if it is not easily accessible to users, they will not want to interact with your component and it may cause difficulties for ADA users. I will begin by discussing accessibility compliance. Although I had limited time, I ensured by testing with the voiceover functionality on iOS that my component was accessible to visually impaired users by reading out the field names accurately and by including alt text for the logo images. The form is fully accessible through keyboard navigation as well.

For form validation I decided to show a red line of text under each field when an error occurs. Having the error show immediately on input change is useful to allow the user to know what went wrong and on what field. I wrote specific messages for different validation failure cases on each field. If I had more time I would have changed some of the validations to only set off on blur, that way the user wouldn’t be startled with an error message before he/she was finished typing the information. I implemented spacing on the credit card input so you could see the 4 groups of 4 digits for Visa and 4-6-5 grouping of numbers for AmEx cards to help you more accurately enter your number. Finally, I restricted all of the non-name fields to entering digits only and I set maximum digit length features so the user could not enter an input that is longer than valid. These usability features should help guide the user to a more intuitive and pleasurable experience by limiting chance of errors and guiding them towards successful completion of the form.

1. When you hit submit, the form would make an API call submitting all of the data packaged into a JavaScript object. The user’s unique ID, and other usage-based parameters might get passed in along with the form data.

{

‘account\_id’: ‘2414564’,

‘purchase\_item’: ‘9734’,

‘credit\_card\_data’: {

‘name’: ‘Michael Quint’,

‘number’: ‘4123123412341234’,

‘cvv2’: ‘123’,

‘expiration’: ‘01/2020’

}

}

Server-side validations might check the credit card information and find the data inputted to be invalid or that the card simply does not exist. In this case I would have my service call formatted as a promise and if it is rejected, to return the errors to the form. I would then use these errors to generate visual warnings to the user. I might clear the invalid field(s) and have the user try again and tell them what went wrong so they can fix the problem.

1. These types of form inputs should be concise, easily understood, and functional. The forms should also be responsive, which is something I did not have time to implement, but would in a future enhancement. People access applications on mobile devices and web browsers of all sizes, so it is important that the form be appealing and accessible across all form factors. It is also useful to let the user know at all times what their completion status of the form is. I achieved this by disabling the submit form until the form is fully valid. The form is valid when all fields have been filled and there are no errors. A future enhancement here would be to have errors appear on empty fields if a user skipped one and tried to hit submit. Setting focus onto the skipped field with an accessible error message is especially helpful to ADA users. Finally, the form is ordered in a way that reads like a credit card which should help users fill out the form with less hassle.

Overall, creating this react app was a wonderful experience and I learned a lot. I have test coverage for every component and validation function as well as snapshot testing of the entire form container. Throughout building the form, I was continually iterating over my code and making several improvements along the way for both functionality and efficiency improvements. I hope you enjoy testing my credit card input form and I appreciate the opportunity.