CSC301 A1

Checkout Calculator: Web App

James Francis Kanu
kanujame

Research

The basic functionality of a checkout calculator can be narrowed down to three main functions: adding an item to cart, removing an item to cart, and lastly, checking out items. The implementation of these functions includes updating the user interface to reflect whether an item was placed in a cart, as well as the subtotal and total of the purchase. Building on this basis, we can then add additional functionalities such as adding discounts, viewing past purchases, and letting a user create new items.

Creating this application in a manner in which it can be easily developed, maintained, and deployed is a multifaceted problem. The open scope of the project allows for a multitude of approaches to tackle the development. Ideally, the program would utilize a front-end and a back-end, and with these two components, several options are already available.

Front-End

Suggestions from the assignment and a simple search on the internet gave us three initial options for creating our front-end: React, Vue, and Angular. Having no experience in any of these frameworks means that our decision for which to use not only affects our efficacy for building a checkout calculator, but our skillset as software engineers. Some criteria to consider including popularity, similarity to familiar frameworks, learning curve, and support.

Wanting to learn a framework that is most prominent in the industry, viewing Stack Overflow's 2019 Developer Survey, our choices were narrowed down to Angular and React. Although Vue is currently rising in popularity, the popularity of it is relatively smaller; finding resources to help in our developmental issues may be harder to find in using a less-popular framework.

Now, the real comparison starts when we look at Angular and React, the current two most popular front-end frameworks. The relatively small scope of the project would mean we needn't be too concerned with shortcomings of each framework such as data-binding or resolving dependencies, but rather limitations that would most affect a Checkout Calculator web application. Both frameworks are meant for creating one-page applications but React increases efficiency in terms of performance since it only sends back and HTML for a changed element rather than sending a completely new HTML each update. A browser would have less load running the calculator application if built from a React framework as opposed to an Angular one.

React was ultimately chosen not only as a subjective choice based on anecdotal evidence, but a pragmatic one too. Speaking to some students who have some front-end experience, none answered Angular over React as the preferred front-end framework. Although Angular had dominated the industry for most of the past decade, the popularity of React had surpassed it in recent years and is continuing to grow, only increasing our incentive to learn it over its competitor.

Back-End

Pulling from past experiences, Python and Node were the initial considerations. As this is a course meant for learning, new frameworks have to at least be researched to see if it is beneficial to try something new, and in looking at the assignment recommendations, Go would be a framework worth looking at. With that, similar considerations to choosing a front-end framework must be taken into account for choosing the back-end: learning curve, popularity, and community support.

From some research we understood Go to be the new kid on the block: rising in popularity and a 'must-learn' if aspiring for a job at Google. Some benefits to note include automatic garbage collection and its 'quick compilation and execution speed'. However, like our dismissal of Vue, Go is still relatively new, and its still growing community support may not be enough to expediate the learning curve of the language. Although learning a new framework can only be beneficial to one's skillset, given the time constraints of the assignment, it was decided that it was best to stick to what we know.

Python and Node are both extremely popular frameworks with a sheer volume of community support. Finding resources to solve problems with developing on either framework would be relatively easy, and the learning curve is flattened due to prior knowledge. As this is a group assignment, both partner experience history must be examined, and it was decided that Python was the framework in which the most knowledge was shared. Having Flask skimmed over in lecture only cemented the decision to choose Python over Node; if it was something new we'd have to learn, it might as well be the one that was glossed over in class.

CI/CD

Given a lifted requirement for CI/CD, there is currently no implementation for it within the application. However, that isn't to say research wasn't done on potential frameworks. Jenkins appeared frequently in attempts to learn CI/CD, with prominent features including what appears to be large communal support and it being entirely free. However, CircleCI was the framework mentioned in class and most frequently in discussion boards. Assuming others are learning for the first time as well, the practical choice is to choose the one I could ask the most questions about if I had them.

Development

Over the course of development, several changes had been made to the original plan of deployment. In the process of learning React, updating states on the interface given events such as button clicks, was among the first things taught after learning to set up an environment. It became apparent with the knowledge available at the time that several functionalities can bypass sending data to a back-end, such as adding and removing items to cart, incrementing items, and calculating subtotal and total.

Given no prior experience in databases, the solution thought to save transaction history was to place it in readable csv file. This would require the use of a file writer in the back-end as with the React knowledge at the time, it seemed impossible to edit a local file from a React application. With developmental hiccups such as losing a partner and time constraints, the skills to use Flask to connect to a React front-end had not been learned at the time of the deadline; progress made on the python back-end will be ignored entirely in favor of an integrated-back-end and only the most recent transaction will be saved on the client-side.

Although not the ideal implementation or developmental process, the checkout calculator satisfies the three main functionalities set out, while providing ease of access to further improve.

Sources

Bakradze, G. (2017, September 14). How to Choose the Best Front-end Framework. Retrieved October 07, 2020, from https://www.toptal.com/javascript/choosing-best-front-end-framework

Best 14 CI/CD Tools You Must Know: Updated for 2019. (2020, July 22). Retrieved October 07, 2020, from https://www.katalon.com/resources-center/blog/ci-cd-tools/

Javinpaul. (2020, August 23). What is Go or Golang Programming language? Why learn Go in 2020? Retrieved October 07, 2020, from https://medium.com/@javinpaul/what-is-go-or-golang-programming-language-why-learn-go-in-2020-1cbf0afc71db

React vs. Angular: The Complete Comparison. (n.d.). Retrieved October 07, 2020, from https://rubygarage.org/blog/react-vs-angularjs

Redmond, S. (2020, January 23). Choosing a Front-End in 2020. Retrieved October 07, 2020, from https://medium.com/@sredmond/choosing-a-front-end-in-2020-70eb796137b6

Rouse, M. (2020, May 11). What is the Go Programming Language? Retrieved October 07, 2020, from https://searchitoperations.techtarget.com/definition/Go-programming-language

Stack Overflow Developer Survey 2019. (n.d.). Retrieved October 07, 2020, from https://insights.stackoverflow.com/survey/2019

Top 8 Best Backend Frameworks. (n.d.). Retrieved October 07, 2020, from https://www.keycdn.com/blog/best-backend-frameworks