Haocheng Hu, Valerie Roussel
A1 Group 15
CSC301: Introduction to Software Engineering
5 Feb, 2021

Assignment 1 Report

Here are the options we weighed for each part of the assignment, and the reasons we picked each one:

Front-end:
Potential Options: React, Angular, JS
Option 1: **React**

a) **Ease of Development:** Very easy, we can write HTML-like JSX code into the JS file so creating and deleting elements requires only HTML writing knowledge, not JS DOM manipulation. We learned how to use React in CSC309.
b) **Maturity of tech:** 7 years since original release, maintained by Facebook
c) **Domains covered by tech:** Front-end work.
d) **Popularity of tech:** One of the most popular front-end solutions.
e) **Performance of solutions:** React is slightly slower than DOM manipulation in traditional JS. The difference in speed is negligible.

Option 2: **Angular**

a) **Ease of Development:** More difficult, we don't have experience with this framework. We would have to devote some time to learning how to use the system.
b) **Maturity of tech:** 4 years since original release, maintained by Google.
c) **Domains covered by tech:** Web and desktop app frontend.
d) **Popularity of tech:** Less popular than React but still decently popular.
e) **Performance of solutions:** High performance.

Option 3: **JavaScript** (HTML DOM manipulation)

a) **Ease of Development:** Very easy, we also learned about this in CSC309. Additionally, using JS with no framework is less setup work.
b) **Maturity of tech:** 25 years since original release. Grew to be the most popular web scripting language during the 2000s. Countless frameworks and libraries to add extra functionality.
c) **Domains covered by tech:** Can use for both front and back-end. Since our app is so simple, we can consolidate both into a single .js file.
d) **Popularity of tech:** JavaScript is obviously one of the most popular languages. It's used by most websites to handle client-side operations.
e) **Performance of solutions:** Most JS frameworks contain the extra step of converting their code into more standard JS code that is then run by the browser. This means that vanilla JS is the fastest frontend solution. Still, the difference is pretty insignificant.

Selection: JavaScript (HTML DOM manipulation). We can make the main website in HTML/CSS, and there will be very little front-end work to do beyond that. The only elements that need to be changed from their initial state are the items in the cart and the totals at the bottom. We could use a framework like React to handle these, but it's not really worth the extra setup. For an app of this low complexity there's no real need to use anything beyond the standard HTML DOM manipulation functions.

Back-end:

Potential Options: Python, JS, node.js

Option 1: **Python**

a) **Ease of Development:** Moderately easy, we've used Python in other courses like CSC148 and CSC207. The challenge would come with integrating the Python back-end with the rest of the project, neither of us have ever used Python for web dev back-end
b) **Maturity of tech:** 30 years since original release. Many different third party libraries that adapt the language to perform many different functions.
c) **Domains covered by tech:** Is commonly used for back-end, machine learning, scientific computing, language and image processing.
d) **Popularity of tech:** Python is one of the fastest growing programming languages. Its simplicity is renowned, which is why many learn it as their first language.
e) **Performance of solutions:** High performance.

Option 2: **JavaScript**

a) **Ease of Development:** Very easy, we learned how to use JS extensively in CSC309.
b) **Maturity of tech:** 25 years since original release. Grew to be the most popular web scripting language during the 2000s. Countless frameworks and libraries to add extra functionality.
c) **Domains covered by tech:** Can use for both front and back-end. Since our app is so simple, we can consolidate both into a single .js file.
d) **Popularity of tech:** JavaScript is obviously one of the most popular languages. It's used by most websites to handle client-side operations.
e) **Performance of solutions:** High performance. If used for front and back-end, eliminates time spent interfacing between languages.

Option 3: **Node.js**

f) **Ease of Development:** Somewhat easy, we learned how to use node.js in CSC309.
g) **Maturity of tech:** 11 years since release.
h) **Domains covered by tech:** Client and server-side operations. Event-driven design, database integration.
i) **Popularity of tech:** Highly popular solution for back-end database integration.
j) **Performance of solutions:** High performance.

Selection: JavaScript. By using JS for both the front and back-end, we are reducing the number of technologies in our stack. We decided not to use Node.js because it doesn't provide

any functionality that is necessary for our simple web app. Our goal with this selection is to minimize the development time spent on the project.

CI/CD:

There are many tools available for CI/CD, and some of the more common ones are Jenkins, Travis, CircleCI, and GitHub actions.

We considered Travis, CircleCI, and GitHub Actions (GHA). For the complexity of the software we were building, all these tools offered mostly similar functionality. They all need to define a workflow in a YAML file, but have their own respective syntaxes and quirks to follow.

The defining moment for us was when we considered ease of integration. While the setup for both travis and circle were not difficult, GHA was by far the easiest to set up because of it's integration with our repo hosting tool, which was Github. Travis and Circle also required additional permissions which we did not have because the ownership of the repo on Github that we were developing in did not belong to us. As a result, the decision to use GHA was not a surprising one, given its ease of integration with the environment we were developing on. That being said, it is also one of the up and coming CI/CD tools, and is widely regarded to have similar functionality to the competition.

When we did some research on this topic, because GHA was so new, it seems like in its early days it did not have certain functionality like re-running successful tasks, etc. But since it's release, it has been consistently fixing these issues and adding even more standard issues like a command line interface, etc. As GHA is provided by Github, we can be confident that it will be well maintained, and support for it will not be difficult to come by.

Overall, setting up GHA was rather simple, and while learning the workflow syntax took some time, it is not more time than it would have taken to learn the syntax of other CI/CD tools.

Final choice: GHA

- Ease of Development: similar to rivals
- maturity of tech: relatively new, but actively maintained
- domains covered by tech: covers all you would need a CI/CD tool for
- popularity of tech: widely used, even by the official git repo for git
- performance of solutions built with tech: runs on a docker container, which is fast enough for all intents and purposes
- Easy of integration: much easier than other choices

Testing:

There were many ways to test our project, but our focus was on getting the development workflow working as smoothly as possible. The current tests we have that run on the project include python tests that verify the validity of the HTML source, and also a command line test

that ensures the syntax and style of the javascript source. These tests are being run through commands given to github actions.

Final choice: python tests and js linter

- Ease of Development: python unit tests are easy to add to and there is not much to consider in terms of development for the js linter
- maturity of tech: both are relatively mature
- domains covered by tech: python has quite a lot of great libraries which enable testing of many different types
- popularity of tech: python and js are still rather popular, in fact python is the most popular programming language in the world by some measures
- performance of solutions built with tech: both are fast enough for our needs

Deployment:

The workflow for deployment in our app is as follows:

1. A push is made to the master branch by a contributor to the project.
2. Github actions starts running the tests that we have defined, which include checking that the HTML for the website is well formed, and that the javascript source is free of any errors and follows proper style.
3. The repo that we are working on is our development environment.
4. Once our CI tests pass, the changes incorporated in the push are then duplicated and pushed to our production environment, which is hosted in another separate private github repository.
5. The website is then rebuilt (automatically) and the changes are then reflected on the website.
6. Depending on the time it takes to run the full suite of tests and for the website to rebuild, the full workflow takes about 5 minutes from when a push is made to when the changes are reflected on the live site.

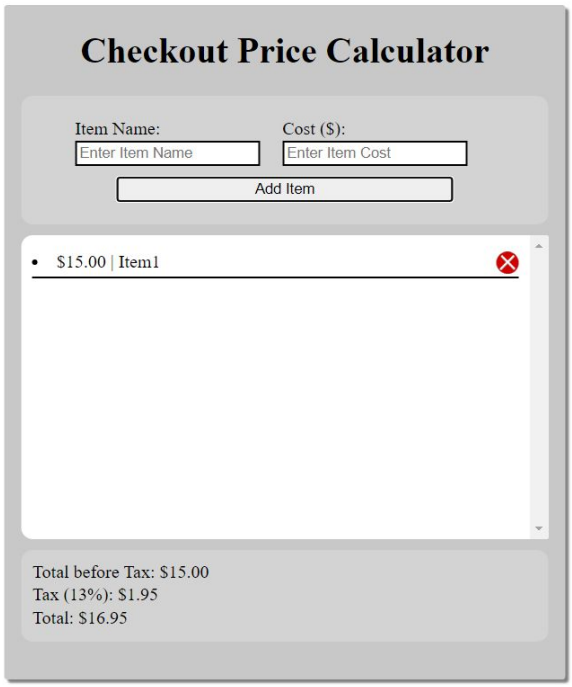
Instructions to run:

The instructions to access our application are laid out in the readme which is located inside our repo: https://github.com/csc301-winter-2021/assignment-1-15-valerieroussel-hhc97 The application has been deployed at: https://hhc97.github.io/csc301_a1/ which runs from a private repo that we are using for our production environment.

Any pushes made to this repo (the development environment) will automatically go through a set of tests before being automatically deployed to our production environment if those tests pass.

You can test our workflow by making a non-breaking change (for example changing some text in the index.html file) and verifying that after a while, the changes do indeed show up live on the web app. If you make a breaking change, you can also verify that the production environment is not affected.

The application itself is a very simple one where you can add items to a list and the total price including tax will be computed. Items can also be removed from the list. A screenshot of the application is attached below.



To add an item to the cart, enter the items name and price into the text fields and click the “Add item” button. It is possible to add discounts using this same method: simply add a negative number in the cost field to represent the amount discounted.

To remove an item from the cart, simply click the “X” icon next to the item listing.

The price totals before and after tax are listed at the bottom. They are updated each time an item is added to or removed from the cart.