Summary

We decided to build a web application to implement this simple checkout price calculator. We used the javascript web framework React to implement the front-end of our web application. Using React allowed us to quickly implement the simple checkout price calculator due to its component-based ideology in which you are able to reuse JSX components which are then rendered onto the browser by React. This saves valuable time from having to manually change the DOM in vanilla Javascript. We used Express for the backend of our web application. Express is a lightweight back-end web application framework for Node.js which easily integrates with a React front-end since they both are written in Javascript. This allows for continuity between the front-end and back-end of our web application. We decided that using a database for our web application would not be necessary due to the simplistic nature of the application. To expand on this point further, since our web application does not require us to store the items that the user has “checked out”, it would not be necessary to store these items in a database. Likewise, since our web application has no login functionality, it would not make sense for the application to store the invoice of each user on a database. To deploy our web application, we used Heroku since it is free and it simplifies the process of deploying Express-based applications.

Short Report

When considering whether to pursue a mobile or web application for this assignment, we felt that it was best to go with a web application since our partner project for this course requested that we build a web application to satisfy their requirements. As such, building a web application would give us a greater insight into the foibles of web development and would in turn help us develop a better product for our partner project.

When considering the options for the front-end of our web application, we narrowed our choice down to three options, vanilla Javascript, Angular, and React. One advantage of vanilla Javascript is the familiarity that we have with it from our coursework in CSC309. Another advantage of using vanilla Javascript is that it gives you fine-grain control on how you want to manipulate the DOM. On the other hand, using vanilla Javascript becomes very tedious for larger projects due to the repeated code that is required in order to manipulate the DOM for certain objects such as elements that appear in a list or a form. As a result of this, we decided not to use vanilla Javascript for this assignment. When considering Angular, we came across the following advantages. Angular allows users to easily create single-page applications that are feature dense while also being relatively easy to pick up for beginners. One of the shortcomings of Angular is that it is quickly becoming outdated due to the rise of other front-end frameworks and thus is no longer the industry standard. Because of this pitfall, we decided not to use Angular and instead look for a framework that is more heavily used. Eventually, we decided to use the front-end framework React. React allows users to easily create single-page web applications whose components are reusable and simple to create. React has a low learning curve for new users and is the current industry standard for front-end web frameworks. Moreover, our partner project requested that we use React to implement the front-end for their website which only further incentivized us to learn and use React for this assignment. One downside of React is that the initial development of a project is more complex due to the time that is required to configure your React app. To learn React, we found a course on Udemy titled “React - The Complete Guide” taught by Maximilian Schwarzmuller and implemented various techniques and ideas from this course into our project.

When considering the options for the back-end of our web application, we narrowed our choice down to three options, Django, Express, and Gin. Django is a back-end framework that is written in Python. Some advantages of Django include our familiarity with Python from previous courses at UofT, good documentation, and its high scalability. On the other hand, setting up a project in Django is very time-consuming, especially for small-scale projects. Due to the simplicity of our web application, we felt that the overhead of setting up a Django project was not worth the benefit of using it. Gin is a back-end web framework written in GoLang. Some of the advantages of using Gin include the rising popularity of GoLang and its high performance. However, Gin is not without its downfalls. Since we are not familiar with GoLang, we would be required to learn this new language in order to make use of Gin. Moreover, Go-based frameworks generally have poor error handling. From this, it was clear that we should not use Gin primarily due to our lack of familiarity with GoLang. Finally, we considered using Express which is a back-end framework for Node.js. One advantage of using Express is our prior experience using Javascript in CSC309 last semester. Since Express is written in Javascript, our familiarity with Javascript from last semester would lend itself well to using Express in our web application. Likewise, since Express is written in Javascript, it allows for better continuity between our front-end and back-end. Finally, Express is a very popular back-end framework that has a large community behind it. However, Express often performs worse in comparison to other back-end frameworks since Javascript is a single-threaded language. Despite this drawback, we still decided to go with Express due to the continuity it provides with our front-end code.

Although we decided to not use a database for this assignment (since it was optional), we still investigated different options for integrating a database into our web application. We looked into both relational and non-relational databases. For relational databases, we considered PostgreSQL since we had prior experience using it in CSC343 which would allow us to easily query our database and retrieve information. However, setting up an instance of PostgreSQL requires a fair bit of work since it requires a well-thought-out schema that will fit the needs of the application. For non-relational databases, we looked into MongoDB and GraphQL. MongoDB would be a great option as it easily integrates with Express using mongoose and stores data in a JSON format that is simple to work with using Javascript. However, a non-relational database often doesn’t provide the same structure that a relational database offers so there is some tradeoff. We also learned about integrating MongoDB with express in CSC309 so familiarity would also be a benefit. GraphQL would be a completely new database option for us as neither of us has any experience working with it. Therefore, although it is another very common and popular database, we didn't put too much thought into using it due to the large learning curve.

For deployment, we simply used Heroku since it's free and allows for easy redeployment via a simple git push operation in the repository. Since we can create a Heroku instance within the same repository in which the code for the application is hosted, any changes we make to our main branch on GitHub can also be easily reflected within our Heroku branch. As a result, using Heroku as our deployment option allows for our web application deployment to be easily repeatable simply through a git command.