**Frontend Technology:**

***Angular***

**Pros:**

* One of the most popular web frameworks currently
* Feature rich with numerous libraries, templates, and testing utilities at a developer’s disposal Two-way Data binding to transfer changes in the Model to the View
* Ideal for large scale applications
* Used by many renowned companies and has frequent updates every six months
* Supports native, hybrid and web app development

**Cons:**

* Steep Learning Curve as a familiarity in Typescript is needed as well as time to integrate with the large volumes of libraries
* Slow performance for heavy applications due to the large amount of features
* Mixed reviews from developers who have used this framework

***React***

**Pros:**

* The most popular web framework currently and for the near future
* Virtual Dom allows for more efficiency for updating the view of a webapp
* We already have some experience in React
* Frequent updates by the Facebook team
* Supports cross-platform development

**Cons:**

* Due to the dynamic nature of react, large scale applications may not be ideal for react
* Required to learn JSX and other Quirks of the framework

***Vue***

**Pros:**

* Virtual Dom allows for more efficiency for changing UI states
* Straightforward Learning Curve
* Quick deployment of application with high performance, easy development
* Newer framework that is gaining in popularity
* Most highly rated framework on Github (Stars)

**Cons:**

* No prior experience
* Not ideal for more heavy weight applications due to lack of libraries
* Small community of developers

**Backend Technology:**

***Node.js***

**Pros:**

* Has recently become a very popular web backend
* JavaScript Full stack development allows:
  + Easy transition for frontend developers to work on backend
  + Code sharing and reuse
  + Developer productivity, efficiency
  + Superior speed and performance compared to other backend technologies
* Prior experience in Node.js backend
* Npm, the default Node.js package manager provides close to a million available libraries

**Cons:**

* Slow processing when it comes to heavy computation due to single-threaded nature of JS
* Open source nature causes some tools in npm to lack quality control

***Python***

**Pros:**

* Numerous frameworks and libraries to help with web development
* Easy learning curve: syntax is pseudo code like
* Versatile as it can be used for webapps, desktop apps, ML, DevOps and more
* Django, a python framework works well with many well-known databases
* Extremely popular, used in almost every major company

**Cons:**

* Lack of microprocessor support resulting in code limitations
* Not good for memory intensive tasks due to its dynamically typed nature
* Less experienced developers in its community than others

***PHP***

**Pros:**

* Massive community with lots of frameworks, documentation, and other resources
* Works efficiently with HTML, servers, and databases
* Faster webpages compared to other technologies
* Low barrier to entry means a relatively easy learning curve

**Cons:**

* Declining popularity
* Lack of libraries that support modern needs such as ML
* Security flaws when used by novice programmers
* No prior experience

**CI/CD:**

***GitHub Actions***

**Pros:**

* Tons of premade workflows available in marketplace
* Easy to set up for beginners and is integrated into Github
* Free and open source
* New features added constantly

**Cons:**

* Lack of command line configuration options
* Deployment Limit

***CircleCI***

**Pros:**

* Integration with GitHub and is also Cross Platform
* Relatively cheap compared to other CI/CD options with free option as well
* Lightweight, easily readable YAML configuration
* Fast Builds

**Cons:**

* By default, not compatible with some languages
* Poor documentation and steep learning curve for highly configured integration tests

***TravisCI***

**Pros:**

* Build matrix allows for easy build jobs
* Integrates well with Github and other tools such as Slack and VS Code
* Good documentation, clean interface with decent customization

**Cons:**

* Slower builds compared to other CI tools
* Lack of free tier for private apps

**Database:**

***MongoDB***

**Pros:**

* NoSQL, Easier/quicker to code without impacting performance with lightweight app
* Easy to set up for beginners
* Free and open source.
* Easy to scale and fast access to data
* Document query language supported and is simpler than SQL queries

**Cons:**

* Limit in size of each document in database
* Uses high memory for data storage
* Not Secure by Default

***PostgreSQL***

**Pros:**

* Parallel processing allows for fast Queries
* Highly Scalable
* Predefined functions and available interfaces
* Supports JSON

**Cons:**

* Inefficient Storage
* SQL database is excessive for our application
* Confusing documentation

***MySQL***

**Pros:**

* Extremely popular database for webapps
* Lots of functionality for no fee
* Variety of user interfaces

**Cons:**

* Overkill for a simple checkout app
* Slow updates and new content
* Limitations with tables and complex schemas

**Our Choice: React, Express, Nodejs, MongoDB**

We chose then MERN Stack as the tool to build our app. The main reason was that we were familiar with most of the technologies in this stack and each component being well known in and of itself allowed for good documentation for all the technologies.

For our frontend we chose react because it is suitable for lightweight applications and is the most popular web framework right now. Vue was heavily considered because of its rising popularity and similarity with react. Another option was to go vanilla JavaScript, but we felt becoming more familiar with react was more useful as there is a high chance, we may use it for the project in this course.

We chose to use the Node framework Express for our backend mostly because of its extensive libraries and known compatibility with react. We also wanted a technology that would still be widely used in the future and due to PHP’s older libraries and declining popularity, we did not think it was the right choice. Finally, what made us choose Node.js over Python was because the research we did showed that integrating a Node backend with a React frontend was a more seamless and efficient option.

For our database, we chose MongoDB because we felt NoSQL was the obvious option for this assignment. It is a lot faster to implement than SQL databases and suits our basic purposes for storing checkout items. We chose to run our database on MongoDB Atlas as it is easy to setup and the data is stored on the cloud.

Finally, had our classroom not exceeded the maximum github actions allowed, we would have implemented a CI/CD workflow on GitHub actions as it is easy to setup, has premade workflow’s on the marketplace and is integrated to GitHub.

**Deployment and CI/CD Attempts**

We spent a day attempting to deploy our application onto Heroku but were unsuccessful. We were able to successfully build our app on Heroku, but the deployed website always showed “Cannot GET /”. We know this is an error in serving our index.html file by our Express Server but were not able to troubleshoot it.

For CI/CD, we attempted to setup a CircleCI workflow after we learned that GitHub actions was unavailable but it turns our that for some reason when we connected our CircleCI workflow to our GitHub repo, it didn’t work as no credits were available. This resulted in our CI/CD workflow to be set up but not functional.

Since our CI/CD is not functional, our tests for the calculations and frontend UI are carried out by unit tests which are run on “npm test”.