Web Application Report
Yi Chen Zhao

For the web application, we considered three main frontend technologies: Angular, Vue and React. Angular, by far, is the oldest framework and offers a lot of templates and testing utilities, but compared to React or Vue, it has the largest size, which was not suitable for our small-but-lightweight application. Vue is more lightweight than React and much easier to learn, but because it is an open-source project, it is used much less in industry and it is not supported by large companies, which means slower framework evolution. Furthermore, Angular uses real DOM, while React uses virtual DOM and is much more lightweight and adaptive to different web browsers. We wanted our web application to be easily accessed by all different users with different web browsers, so the abstraction in React's virtual DOM is extremely helpful for our uses. Vue has an advantage of the 'ease of integration', allowing developers to quickly insert into a web application to add interactivity, but perhaps is not as strong as Angular or React when building it from ground up. Considering that React is often used on high-traffic websites, our checkout application would also be getting large amounts of traffic in a real world setting, which was what influenced our decision to use React. It is also worthwhile to mention that React has been ranked as the most used Javascript technology in the report "State of Javascript", and over 71% of individuals who used React would use it again to build their projects. Although React is moving away from OOP, which was a much larger learning curve for me, its features were easy to learn due to the simplicity of the entire framework.

For backend technologies, we considered PHP, Go, and finally Nodejs, which we utilized in our application. PHP is historically the industry standard for backend frameworks, because of its ability to easily interact with MySQL/MariaDB/PostgreSQL and ability to work independently of what the databases are built on. Although PHP provides stable performance, it can only process up to half of the total amount of requests Nodejs can, and much less than the total amount of requests Go can. Since we do not work with a database based on SQL in our web app, PHP does more than we need and less for what we needed for our application, which was speed and large requests. As previously mentioned, in a real-world setting, the checkout app would be receiving many requests and heavy traffic, so we preferred a backend technology that can handle large amounts of requests in a short amount of time. One of the benefits of Nodejs is that it is built upon Javascript, which overlaps with React, and also boasts performance improvement. However, Go outperforms Nodejs in many of the factors, but is more suited for *larger* scale products and is a relatively new software that has much less support (in terms of tutorials and tools). There is an honorable mention for Nodejs' NPM, which has the largest amount of third-party packages that are extremely useful while developing, which Go and PHP do not have and has helped tremendously with linking the backend to the frontend and testing.

For storage of data, I opted for storing it in a local file over a database, such as SQL or MongoDB due to wanting to keep the infrastructure of the app simple. For the types of files, I considered XML, CSV and JSON. The large negative to XML is that it is very large in file size, as well as taking up large network bandwidth when transferring small amounts of data. In my case, the web application just sends the username, order id, and content of cart to the backend,

which is not a large amount of information. Similarly, with CSV, although it is much smaller in size compared to JSON, it loses a lot of the advanced formatting such as hierarchical and relational data, and lacks in terms of scalability to expand or minimize the database. JSON, on the other hand, although the drawback is that it's relatively new, and there is less APIs that exist to convert JSON into more used data structures, such as XML, but its small size and easy-to-write to format made it attractive to use for the web application.

The three frameworks we have considered for CI/CD include CircleCI, TravisCI, and Github Actions. From research, it showed that Actions has an advantage to TravisCI for being already *integrated* into Github, where our repo is hosted, compared to TravisCI, which needs to be linked separately to the repo. Although the drawback for Actions is that it lacks in the free amount of build minutes (compared to unlimited for TravisCI) and that Github Actions was a much newer framework that had less support and resources that could help with deployment to a server (Heroku, in our case), it was much more simple to set up directly in our repo on Github itself. In a similar fashion, CircleCI is much more powerful than Github Actions because it not only allows the user to allocate different CPU calculations to different jobs, automatically splitting autotests to be run simultaneously, but also having built-in support for Javascript and custom integration with other API's. Although CircleCI was an attractive option, it added one extra layer of complexity on top of setting up deployment on Heroku (which required a lot of set up), as well as requiring the use of third-party software outside of Github and Heroku as well. Furthermore, Github Actions was chosen mostly for it's ease-of-use with Heroku and setting up directly on the Github website.

Therefore, due to the mentioned reasons, we decided to go with **React** for **frontend** and **Nodejs** for **backend**, as well as saving data in a **JSON** file format, and **CI/CD** with **Github Actions**, and deployed to **Heroku**.

Final Product:

(Please **refresh a few times** because the product data needs to be taken from backend (also deployed on Heroku, and needs to "wake up" before the frontend can access it)
https://a1-nookazon-frontend.herokuapp.com/
https://a1-nookazon-backend.herokuapp.com/

Sources:

https://hackernoon.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847
https://yalantis.com/blog/golang-vs-nodejs-comparison/
https://www.infoworld.com/article/3166109/nodejs-vs-php-an-epic-battle-for-developer-mindshare.html
https://applerepairstation.com/csv-vs-xml-vs-json-which-is-the-best-response-data-format/#:~:text=JSON%20is%20the%20best%20of,bit%20less%20support%20than%20XML.
https://knapsackpro.com/ci_comparisons/circle-ci/vs/github-actions
https://knapsackpro.com/ci_comparisons/github-actions/vs/travis-ci