Rajvi Rana, Yi Chen Zhao

October 6, 2020

CSC301: Assignment 1 - Full Stack Mobile Application Details

**Frontend:**

The framework we decided to enlist in developing the mobile application was React Native with the Expo CLI. This is because React Native is a framework that allows for multi-platform development, all by using one programming language, JavaScript, which is something that we are already familiar with. The fact that it is capable of developing native applications of multiple platforms (iOS, Android), also means a faster development time. Using React Native, one can develop both an Android and iOS application with just one program. However, a con to developing in React Native is that its navigation component is quite complex. For a first time user, learning react-navigation can be tedious, confusing, and could also take up a large portion of your time that could be used for actual development. Another con to React Native is that not all features native to iOS or Android will be readily available. When developing native features for React Native so they can be cross-platform, native developers are still needed. The issue is that there is a shortage of native developers to develop these features, making many features exclusive to iOS or Android inaccessible by React Native developers. However, this is slowly changing as more developers are contributing to the open-source libraries that React Native runs on, and so over time, more native features will be readily available. One strong contender to React Native is Flutter. Developed by Google, Flutter runs on Google’s own programming language, Dart, and provides developers with a framework to develop both multi-platform mobile applications and web applications. Like React Native, the fact that one can use a single framework to develop multi-platform applications significantly decreases the development time. However, there are also some issues with this framework as well. First of all, Flutter is an immature framework compared to React Native as it is a more recent release and still unstable. Flutter is also not able to provide too many features as many of its libraries are still in the pre-alpha stage of development and have not been released yet. Likewise, the programming language Flutter requires, Dart, is also immature. Compared to other languages for mobile development such as Swift, or Kotlin, Dart is still in the early development phase and many features are not readily available. In addition to it being a very new release, it is also not a good choice for developers who are already proficient in JavaScript, as they will have to take the time to learn a completely new language. In addition to React Native and Flutter, Ionic is also a framework used for mobile application development. Ionic is a framework built for those who have expertise as web developers, providing them with a smoother transition to mobile development. However, the issue with ionic is that many features are only available with the premium version (with payment), and because it is built upon the “web browser” it cannot easily access native functionalities.

**Backend:**

Along with React Native, we chose to use Node.js with Express.js to develop the backend of our mobile application. Among full-stack mobile developers, these frameworks are a strong and popular choice. Ruby on Rails is a strong contender to Node.js. Ruby on Rails provides high-speed development because of its Convention over Configuration approach. This means that developers spend less time on coding the actual product rather than configuring the environment. In addition, the framework is easy to read with its English-like syntax. However, a downside is that it has low performance and slow runtime speed. On the other hand, Node.js with Express.js has high-speed performance as it’s based on one of the fastest engines, Google V8. In addition, it handles requests simultaneously, and most importantly, it is completely written in JavaScript, which is beneficial for developers such as us who are already proficient in the language. Django is another popular framework for backend development. Upon research, Django in fact outperforms Node.js when it comes to cost-effective efficiency, and performance. However, the downsides are that Django does not offer full-stack development and its framework is quite complicated as a developer has to follow a specific path for solving problems. Node.js on the other hand offers full-stack development as both a mobile application’s front and backend can run on JavaScript. Node.js is also less complicated as developers have the freedom to solve problems in various manners. Hence, for a first time full-stack developer whose proficient in JavaScript, Node.js is simply easier to learn and understand quickly.

**CI/CD:**

Github Actions was the preferred method of setting up CI/CD. This was due to the fact that our project repository was already set up on Github, making Github Actions convenient to use. Pros of using Github Actions include multiple file workflow support and the fact that it is already integrated with Github’s Checks API. Alternatives to Github Actions for continuous integration are Circle CI, and Travis CI. However, for our current development needs, it has been noted that Github Actions provides the same functionalities needed as Circle CI and Travis CI. Specific functionalities specifically needed for this project is specific language support in JavaScript, and pipeline building. Both of these functionalities are readily provided in Github Actions. Due to the fact that our project has been set up in GitHub, it was simply more convenient to set up GitHub Workflows along with it. Due to the fact that the Github Actions limit has been reached, unit testing is run using Jest. The biggest pro to using Jest is that it works with minimal setup or configuration, making testing for new developers quick and easy. However a con with Jest is that it is a much newer form of testing compared to older competitors like Mocha. Due to its young age, Jest has fewer libraries available for its users to enlist in. However this. was not a major concern for this project as the testing needed was simple.

**Databases:**

The database chosen for the mobile application was MongoDB Atlass with Mongoose for object modeling. In the mobile application, after leaving the checkout screen, the user is prompted to sign up. After registration, the user’s data and the items they bought are posted to the database (accessing this database is listed in the instructions above). One pro of using MongoDB Atlas to export data is that it is extremely simple to set up. This is useful for a first-time developer as it provides them with an introduction to working with databases without too much hassle and time being taken up. Another alternative to consider for the mobile application was instead exporting the user data to a JSON file. However, this can lack in security, and as someone who wishes to become a professional developer in the future, becoming familiar with working databases is a must. In addition to exporting to a JSON file, another alternative to MongoDB Atlas is DynamoDB. While DynamoDB is a reliable and secure database alternative, one con to it is that an application with DynamoDB can only be deployed on AWS. On the contrary, MongoDB can be deployed anywhere, which is more suitable for this project.

**Conclusion:**

In conclusion, this mobile application is developed using some of the most current and popular frameworks and utilities. Written in JavaScript all throughout, the frontend is developed on React Native with the Expo CLI for simple and quick UI development; while the backend is developed using Node.js with Express.js for quick and ease-of-access for a client-side server. For learning experience in handling databases, MongoDB Atlas proved to be a tool that was quick and easy for new developers to set up, and finally, GitHub Actions has proved to be a reliable and convenient resource for setting up CI/CD for this application.