Assignment 1 – Checkout Price Calculator – Mobile

We build an Android shopping app with a simple checkout out calculator and functionalities that improve user experience. We assume that there are preloaded items in which users can add a subset of those to the checkout.

Three technologies: Android Studio, Xcode, React Native

Many stacks support both frontend and backend. Android Studio is open-source software, and it updates often. Java is the official language for the backend of Android Studio, whereas XML is a markup language for the frontend. Java has a big ecosystem of well-tested libraries and frameworks for any use case. However, it requires a lot of code to be written when developing Android applications. XML in Android Studio is easy and efficient to use when designing the layout. React Native is a cross-platform framework that enables web developers to create mobile applications using JavaScript knowledge. It can share code across iOS, Android, and Web, so developers can build native mobile apps with no need to learn Android specific Java or iOS's Swift. It also has a significant amount of reusable code when coding for iOS and Android and component libraries and UI toolkits that help save time and build the applications faster. Since it is UI focused, it makes apps load quickly and have a smoother feel, whereas the emulator in Android Studio and simulator in Xcode starts very slow. However, React Native can fall behind Android Studio and Xcode when it comes to accessibility of native features. Swift is a relatively new programming language used by Xcode for building iOS and Mac applications. Its language features enable an optimizing compiler to produce swift code. Therefore, the coding process with Swift appears to be faster than Java. However, Android Studio has background compilation, which will quickly highlight issues, while Xcode needs an explicit build stage. In terms of popularity, Swift is not as popular as the other two languages are.

Three CI/CD choices: Circle CI, GitHub Action, Travis CI

All three tools allow tasks to be run concurrently on the same machine, support common tools like Slack notifications and various VCS platforms, and have community support. GitHub Actions and Circle CI has a continuous delivery pipeline, that is, automated builds, tests and deployments are orchestrated as one release workflow. However, Travis CI specifically built around GitHub pull requests. Besides the official documentation and software, GitHub Actions has various premade workflows, while Circle CI and Travis CI don’t support plugins.

Three database choices: SQLite, Realm, Room

SQLite gears toward moving away from server-client architecture and storing all the app information directly on a mobile device. The code written on Realm is more laconic compared to SQLite. Realm gained wide popularity, especially among newbies in Android development,

because of the complexity of SQL. Realm for Android development is fast. In some cases, it worked even faster than pure SQLite, which was proved by the research published on the Realm's official website. However, SQLite coupled with Room is a better option for a complex and large-scale project. Room is a library that was designed by Google to smooth out all the weaknesses of SQLite. It provides an abstraction layer over SQLite to solve several problems with this database. With the help of Room, SQLite can catch up and overtake Realm in some cases.

We build our mobile app with Java, Android Studio, and SQLite because we have previous experience with these technologies. Also, Android comes in with built-in SQLite database implementation, which is supposed to be used for local data storage. So it is easier for us to build the database with SQLite since it is embedded in Android by default, and we already have some prior knowledge of SQL query. We are more likely to accurately estimate the time and effort needed to put into the work. We don't need to spend more time familiarizing technology before building the app. Therefore, product development is predictable. Xcode, React Native, Room, and Realm are new technologies to us, so more time and effort are required before proceeding to the building process. Although React Native offers a way to develop mobile apps for iOS and Android, native apps provide better performance when optimized. Finally, for CI/CD, GitHub Actions is preferred because it has a wide variety of premade workloads. However, it is no longer available, so we switch to our second choice, Circle CI. We choose Circle CI over Travis CI because Circle CI has a continuous delivery pipeline, whereas Travis CI is built around pull requests.