Dev Enviroment Proposal

September 29th, 2017

Team Name:
Gnosis Solutions

Team Members:
Kalen Wood-Wardlow, Christopher Simcox, Thomas Back, Kristoffer Schindele

Project Sponsor:
Dr. Leverington

Faculty Mentor:
Dr. Leverington

Overview:

The purpose of this document, dev environment proposal, is to suggest the best plan of action and give insight on why based on our technical expertise why this is recommended.

React JS Native:

Overview:

React JS Native allows us to build cross platform, efficient, mobile applications using JavaScript. Visuals are handled by web languages. Testing can be done through a digital QR code reader which eliminates testing via hardline connections.

Pros:

- JavaScript is the main language for functionality which everyone in the group has had experience with in one way or another.
- Allows for "code once deploy everywhere" which means our product will be publishable on multiple platforms with no additional modification.
- Trusted by Facebook and other fortune 500 companies.
- Development environment is very lightweight and easy to setup.
- Has prebuilt libraries that support advanced features that we may be interested in for future development of this project:
  - AWS Integration: https://aws.amazon.com/blogs/mobile/announcing-the-aws-sdk-for-react-native/
  - Firebase Integration: https://github.com/invertase/react-native-firebase
- Has "Hot reloading" in which there is no time wasted in compile time when testing no matter how large the reload is.
- UI framework is included inside package.

Cons:

- Only 50% of our group has had direct experience developing in this instance.
- Must learn their own JSX (XML like) structure in order to efficiently develop UI.
- Must learn React methodologies.

Final Recommendation:

This solution would provide faster and easier testing as well as easy integration with more complex features and would also reduce the strain on efforts to test and deploy the solution to different environments.

Android Studio:

Overview:

Android Studio is a standalone IDE and compiler that allows for mobile app development for the Android platform. This platform runs on Java for functionality and XML files for the graphical interface. Testing is done strictly on the Android devices and through emulation on Intel architecture devices only and through hardline connected runs. Must be compiled each time you want to test no matter how small or big the change.

Pros:

- Everyone in the group has had experience developing in this environment.
- Everyone has had experience with Java at one point or another.
- Firebase and google login is supported directly by the IDE instead of an extension library or plugin.

Cons:

- Slower testing time because of Java environment.
- Can only test on standalone devices or on emulation that restricts processor to Intel architecture only.
- Compile time is significant and takes up to 2-3 minutes to compile to just test.
- Can only deploy code on Android platform.
- Support libraries are often bulky and cause poor performance issues.
- All implementations of firebase and google login will have to be done through Google’s Android API. Much harder to establish external database connections

Final Recommendation:

Android Studio is good if we are strictly developing for the Android platform. The cost of trying to compile code to test even a small feature is significant and can dramatically reduce development time. Connections to non Google services are more complicated and can cause performance issues.

Apache Cordova:

Overview:

Apache Cordova is much like React JS Native in that we are using web languages to develop the structure and logic. The biggest difference with this tool is that there are less libraries supported for advanced and useful features.

Pros:

- Has an easier to learn UI integration through other frameworks.

Cons:

- Fast testing but cannot test by QR code.
- Must compile to test code.
- Must include other frameworks to develop UI efficiently.

Final Recommendation:

While this solution is similar to React Native it also has drawbacks that we must include other frameworks for the UI and that logic is not native to this solution which may cause a strain in development time in regards to the front end solution.

Conclusion:

Overall the cost of development would be significantly reduced by using React JS Native in not only the fact that we could deploy to all mobile platforms with the same code but also with the advantages of all packages available to us through this platform. Android Studio is bulky and limited in it’s development power and doesn’t give much leeway on how the solution will be implemented and also would increase strain on testing time. Apache Cordova is a good choice but has disadvantages of having to include external visual frameworks to solve the problem of UI organization that React JS Native has already within the platform as well as Android Studio has. We recommend React Native for lowest cost.