Brightlayer UI Mobile (Hybrid) Frameworks

Brightlayer UI supports hybrid mobile application development using React Native. Some of the key considerations when using React Native are:

- Renders native UI elements (not a WebView)
- Can re-use application logic code (React)
- Requires writing new code for UI (learning curve)
- Does not use CSS for styling (learning curve)
- Provides access to hardware functionality through various plugins / libraries
- Has a large community of support
- Learn more on the React Native Website

Building your application

Follow our React Native guide for information on developing using this framework. This guide will walk you through the process of developing your application and ultimately creating a final binary to distribute (either to your testers or your final customers).

Deploying your application

Once you have built your application into a distributable binary, there are two options for distributing it.

Visual Studio App Center (formerly HockeyApp)

Visual Studio App Center gives you a way to test your app with users before you are ready for final distribution. This service allows you to upload app binaries and create lists of people with whom to share them. These people will be notified via email when new versions of the app are available, and they will be prompted to download and install them directly onto their devices. You will need to request access to Visual Studio App Center from IT.

App Store / Play Store

For the final distribution of your application, you will need to utilize App Store Connect and / or Google Play Console.

What about other mobile frameworks?

There is a wide variety of different technologies on the market for developing mobile apps, including native (Objective C / Swift, Kotlin / Java) and hybrid frameworks (Flutter, Ionic, Cordova, Xamarin, React Native, etc.).

We recommend and support development using React Native for these key reasons:

- Maintaining a single code base that can deploy to iOS or Android saves time over building two separate native apps.
- React Native gives a truer native experience than other "web wrapper" frameworks, which tend to present a clunkier user experience.
- React Native has a well-supported open-source implementation of Material Design.
- Business logic code can be shared with existing web applications built using React.
- Supporting one framework allows us to concentrate our resources on building more features rather than having a smaller feature set across multiple technologies.
- Encouraging teams to use the same technology stack maximizes the potential for code sharing between different teams.

There may be circumstances where a different technology makes more sense for your project. For example, if you have an existing web app built using Angular, you may not want to start from scratch and learn React in order to use React Native. In this case, you may be better off using a technology like Ionic to wrap your existing application in a wrapper for mobile.

When selecting a technology for your project, you will need to weigh the pros and cons to determine which approach is best.
