Resolute

The 360-Degree Guide to

Building a Mobile App with React Native

Whitepaper

Introduction

If you are considering building a mobile application, it's likely that you have heard of React Native. Among the two most utilized cross-platform tools for mobile app development in 2022, there's a great deal of information written about the pros and cons of using React Native, how easy or difficult it is to learn the ropes, and so on.

With every popular technology, however, comes confusion and misunderstanding as to when and how to use the technology. And there's not much information available on how businesses should address the topic of React Native.

Which are the specific use cases where React Native should be considered?

What is the process to follow once you validate that React Native is the best match for your mobile app development project?

This paper provides a 360-degree view of React Native from a business perspective and aims to guide product owners and engineering leads alike through the process of selection, validation, the four-stage process of bringing a mobile app to market, and the continuous app optimization post-launch.

About the Author

Georgi Ignatov

React Team Lead

Resolute Software

With over 14 years of software development, Georgi has been crafting React and React Native-powered apps since 2015. He has led large mobile teams at enterprise organizations and software development teams at e-commerce companies. To date, Georgi has published 12 mobile apps to the app stores, of which five are React Native applications. He's taken the challenge of being an entrepreneur as the CTO and co-founder of a SaaS startup.

Table of contents

01

A Brief History of React Native

Facebook had been struggling with their mobile apps for years, investing heavily in HTML instead of native apps, which led to unstable apps and a clunky user experience. About a decade ago, they focused on improving their mobile experience and solving their problems with accessing native APIs quickly. By leveraging React Native's bridge model, developers could code in JavaScript, which easily communicated with that bridge to achieve native-like performance. React Native-powered mobile apps got hassle-free access to mobile device functionality such as the accelerometer, Bluetooth, location, and photos and solved their performance issues and UX clunkiness.

Christopher Chedeau was a young engineer who joined the Facebook Photos team in 2012. When he saw that the company was trying to focus on a mobile-first strategy, Christopher decided to dig deeper into that challenge. Jordan Walke had previously figured out how to use JavaScript to create a UILabel in iOS. Christopher teamed up with Jordan, Ashwin Bharambe, and Lin He to mature the concept and got it working in less than two days. At the end of those two days, they were able to create local UI components from a JavaScript string that runs directly on the mobile device. That solution could have been the answer to Facebook's mobile application problems.
A year later, they organized an internal hackathon to test the framework in the company. At the end of the hackathon, it was clear that the prototype could be used as a tool for mobile development.

In 2015, after months of development, Facebook released the first version of the React JavaScript configuration. In a tech talk, Christopher Chedeau explained that Facebook was already using React Native in production for its Group and Ads Manager mobile apps.

02

What React Native Is Not

React Native isn't a panacea for all mobile app development problems. You should only use it if it addresses your specific needs and challenges. We'll dig into the specific use cases in a minute.

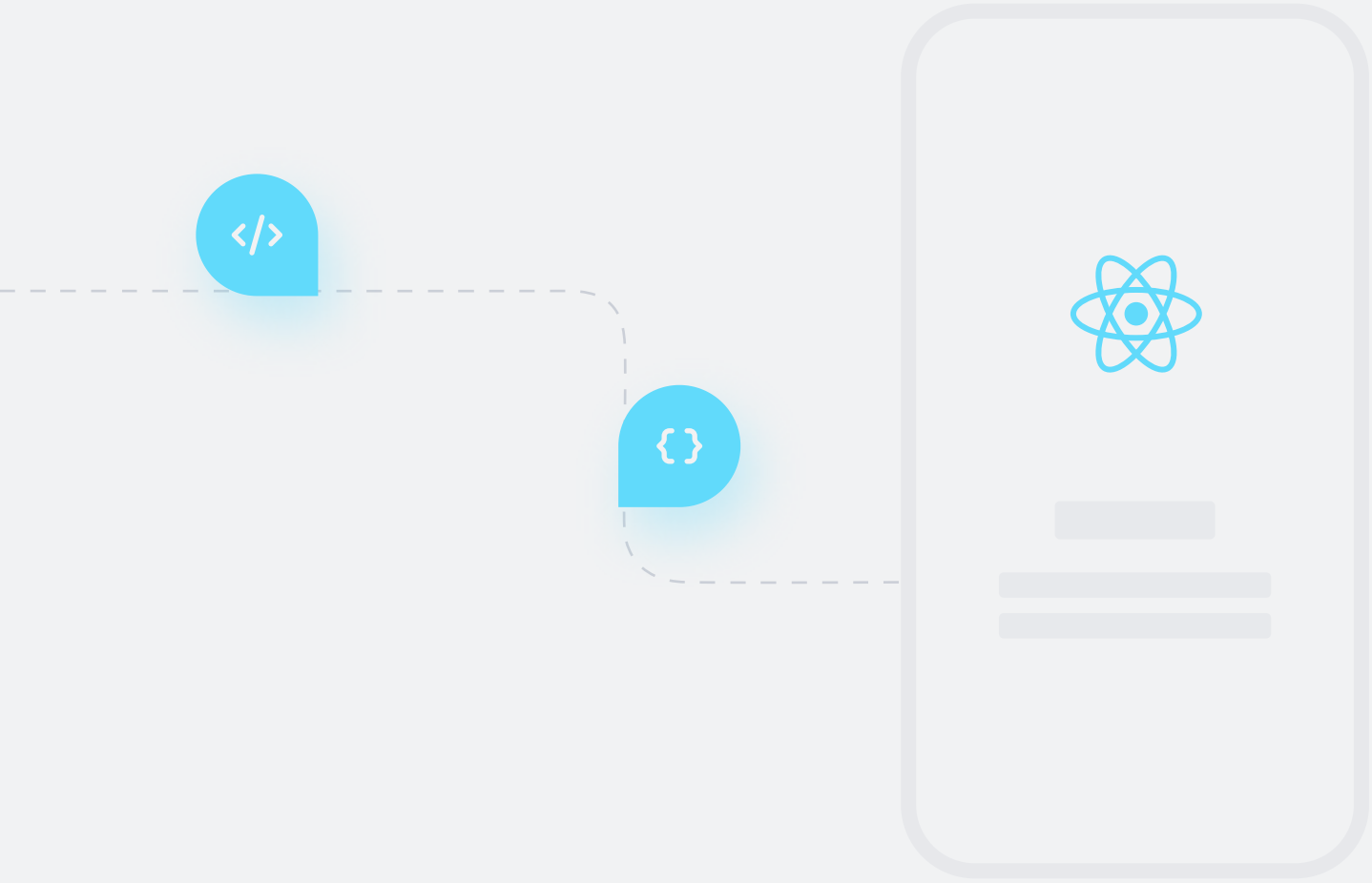
React Native is a hybrid app development framework but that doesn't mean the code you write is 100% cross-platform; the behavior of some components varies depending on the platform. That's why sometimes you need to write native code for Android/iOS. Therefore, the slogan of React Native is "Learn once, write everywhere" and not "Write once, deliver everywhere."

Fortunately, the React Native community today is quite large, and many libraries have been developed, so we don't need to reinvent the wheel. Sometimes, however, you still need to write both native code bases and then write a common wrapper in JavaScript to share with the native platforms.

React Native's codebase is usually 80-90% compatible with Android and iOS, but sometimes you still need to write platform-specific code to make your apps look and feel the same on both platforms. You can achieve 100% by using the correct libraries, which we'll introduce later in this guide.

03

Why Is React Native a Good Choice for Mobile App Development?



Performance

The performance of React Native-powered apps is less efficient than native apps. The reason is the additional JavaScript layer for communicating with native APIs that React Native has, which is the case with any cross-platform development solution.

A compelling reason for using React Native over WebView-based tools is the case where you want to achieve 60 frames per second and offer a native app look and feel simultaneously. The React Native team has done its best to deliver battery-saving UI performance by default, but sometimes that's impossible. An additional advantage of React Native is that while end-users enjoy a close-to-native experience, developers can share business logic with their ReactJS- powered web app, benefiting from faster time to market and more feature-rich development output.

Security

If you need to build an application that requires Type A security (the way online banking apps do), React Native becomes a rather poor choice. In such cases, developers need to give additional consideration to malicious pieces of code that may reside in third-party packages.

Scalability

The scalability of your mobile app depends not only on the framework you use but, more importantly, on the architecture and understanding of how things work under the hood.

By design, React Native is pretty scalable, but sometimes you might run into limitations if you don't understand exactly what you're doing. Here are a few examples:

- No support for parallel threading.
- Abstract layer limitations - An abstract layer was formed above the native platform to generalize the React Native framework's functionalities. While executing the code, even if a single error occurs in this abstraction layer, it might create cascading errors and warnings in the entire application.

The React Native development team constantly works to improve the developer experience and the tools at your disposal.

React Native Use Cases

When To Use React Native

E-commerce, news, and business apps

Walmart is an excellent example of a complex e-commerce application, and let's not forget that Salesforce leverages React Native for their mobile business apps.

A few reasons why the technology is suitable for this type of app:

Excellent performance for e-commerce apps as they are usually simple. You need to render a catalog with products and not much else, which **does not require expensive computations**. Plus, ReactJS can be very SEO friendly with SSR like NextJS. Additionally, you can share your mobile and web business logic between the web and native mobile applications. The latter is very handy for news applications like Bloomberg, which uses React Native under the hood.

Shared codebase between iOS and Android.

You can leverage **real-time push notifications** with a hosting service like Firebase.

Community-built packages for payments that work on both platforms.

Social networks and chat applications

Instagram, Facebook, Pinterest, Discord, and Skype use React Native extensively.

Here's why the technology is well-suited for social networks and chat apps:

The activity feed components in React Native enable the creation of a **dynamic news feed** and improve user engagement, retention and conversion rates for social media apps.

With React Native, you can create **real-time chat apps** using two elements - a server-side library and a client-side library - that instantly connect users with friends and acquaintances.

Firebase **authentication** can be used to create a login function with different login options.

React Native-powered social media apps enable users to like a photo and respond to it with an emoji. Also, single-page apps **facilitate the actions of liking** or reacting to a post or photo. And we all know how valuable React can be for creating SPAs.

End-users can **upload photos** without hassle.

Developers can use the Firebase database to integrate **push notifications** into the app. A social media app needs a real-time notifications feature to enable end-users to receive messages and interact with their friends' posts.

React Native Use Cases

When React Native Is Not the Best Choice

Games

React Native is unsuitable for real-time games that need complex interactions. However, you can build idle clicker games with React Native or chess. In short, simple games.

Utilities

React Native is not a good fit for apps that constantly leverage native features and platform APIs. It's easier to build those using native platforms. Some examples are battery monitors, brightness controllers, and antivirus software.

Make and receive calls

If you are building apps that need to leverage the mobile device's call functionality, you should not choose React Native either unless you feel like writing a lot of native code. It is hard to handle native calling interfaces in React Native. You can use react-native-callkeep for that but achieving a fully native look and feel for applications requiring calling is still a challenge. So, if your application requires a native calling functionality, our advice would be to write your own packages or just go straight for native mobile development.

Background processing apps

These are resource-intensive apps that feature a lot of background processes. React Native does not handle background tasks well, and there is no library yet to make the process easy and smooth. There are some libraries with limited functionality, for example:

- Featuring only a periodic worker (no support for queue)
- It only runs once every 15 minutes. There is no way to speed this up.
- Generally, not feature-rich.

Banking software

React Native is less suitable for banking apps unless you want to be extremely careful with the third-party packages you install.

In the world of npm, you can easily publish malicious source code residing within specific packages, and with React Native, we often rely on third-party packages. When building banking software, you must rely on secure-source packages that are not community-based. In theory, it's not impossible, but you must be very careful when installing packages within your project, or you can choose to develop your own packages for React Native.

04

The 4-stage Process To Bringing Your Mobile App To Market Using React Native Product Definition

This section will go through the step-by-step process of developing and launching a successful mobile app. The team of mobile app development experts at Resolute uses a proven four-step method for building mobile apps with React Native. Let's have a look at it.

Say you have an excellent idea for a new application or service, and all stakeholders are on board – indeed, this is the first step. Now you're probably thinking about how to build it and who could help you. Great. But wait a minute, what exactly are you building? Will your app manage to attract customers when it is published? Does it fit your business objectives? Is there a clear need that you will be addressing with it? These are the questions you should ask yourself before you get everything rolling.

Step 1: App Definition

Does every mobile app need to have a definition? Yes. Because if you don't have one, you might get there if you're lucky, but in most cases, you'll either fail or drastically increase your costs.

Once you write your mobile app's business case and discuss your product-market fit with your stakeholders, it's time to define the minimum viable product. This phase is called scoping or concept development and focuses on refining the product strategy.

In this phase it is important to define the below details:

Business case

Business research starts with evaluating the total addressable market (how big is the market?), then you should focus on segmenting and targeting (building your ideal client profile and primary buyer persona, your app's value proposition, positioning, GTM strategy, etc.) Last but not least, you should define what success means and how you measure it.

Validating the tech stack

Validate that React Native is applicable to your use case and can be leveraged for your product; if not, look for alternatives or how to adapt your product vision so you can take advantage of the React Native flexibility.

1. For example, research if there are packages in the community for the Native APIs you want to use, and ensure they are well-maintained.

2. Validate that it is possible to use a particular native API through React Native. If that's the case, feel free to move forward.

Success metrics

It's important to determine the success metrics early to evaluate and measure progress after launching the app. What are the KPIs you would want to monitor closely? These can be basic metrics like average order value or something more specific like tailored goals that are relevant to your business.

Building a backlog

Finally, you must put together a backlog of app specifications and requirements for your minimal viable product in order to build a roadmap and get rough estimates for the app's development.

Step 2: Designing an Engaging Digital Experience

At this stage, your app starts to take shape. Up until now, you've only imagined in it . By the end of this phase, you'll have a good idea of how the app will look and feel to your customers.

Here's what an app design process usually looks like:

Start by creating wireframes for your mobile app screens

You may already have a few wireframes sketched, but this stage is the perfect opportunity to look at how each screen of your app will appear. Our advice is to use low-fidelity layouts, keeping in mind the ideal user experience - the color variants, images, and text will generally come later. At this stage, you should focus on the user journeys that help customers complete their tasks effectively and enjoyably. Using the wireframes, you can build a clickable prototype connecting all the dots of the UX.

Time to apply the design to your newly built wireframe prototype

At this point, a UI designer starts adding variety and substance. For some, things start to get exciting as the application begins to mature right before their eyes. It's time to get creative - color and shapes; even animations can help you create a visually pleasing app.

A quick tip: Invest in your app design
In the long run, the more you invest in the design stage of your app, the more the process of bringing the app to life becomes easier.

Additional Tips to Facilitate the Development Process

1. Aim to build a clickable prototype as it helps you visualize the customer journey and intended interactions in the app from start to finish (this is possible with tools like Figma, Sketch, Adobe XD, and InVision).

2. Well-implemented design systems can provide many benefits to the design team:

   - ✓ Design (and development) work can be created and reproduced quickly and at scale.
   - ✓ Design systems free up design resources to focus on larger, more complex problems.
   - ✓ They create a common language within the design team and between cross-functional teams.
   - ✓ They create visual consistency across products, channels, and (potentially) siloed departments.
   - ✓ They can serve as excellent learning material for junior developers and designers.

3. Focus on documenting everything that can help the development team deliver better quality faster.

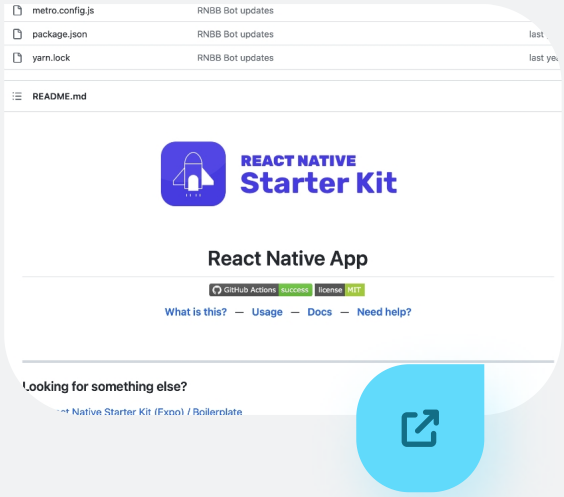
Step 3: Building and Deploying Your Mobile Application

Now that you have a comprehensive product spec for your app and a well-designed user experience, it's time to start developing the application. React Native offers many options to get started with development, for example, Expo, a very friendly platform for beginners in React Native development. But if you want fast performance and control over your app, trust a boilerplate that doesn't use Expo because Expo is an extensive framework built around React Native and includes a lot of libraries that you don't usually need. A simple "Hello World" app built in Expo weighs about 25 MB and has only one screen.

Below, we'll introduce you to some React Native boilerplates and show you how you can build your app in minutes.

React Native Starter Kit

A very minimalistic and simple mobile starter kit that comes complete with all the standard tools you need for developing your application. Plus, it includes Native Base for the UI components.

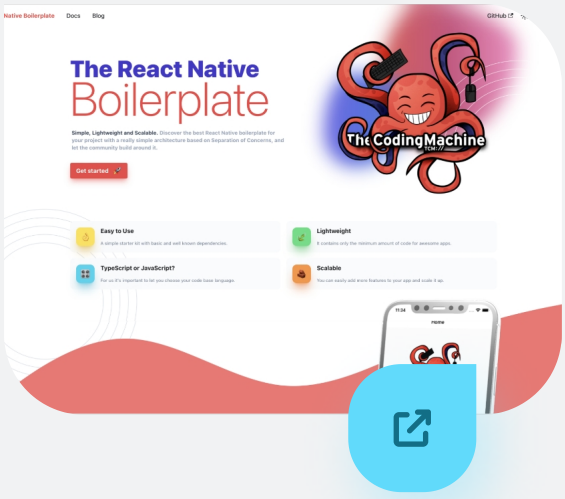


Ignite

Comes with a lot of batteries included. Based on Expo, it packs many good practices, but if it doesn't fit your needs, don't use it. It ships with Expo, which might be a bit heavy, but if you are building your first React Native app, start with Ignite.

React Native Boilerplate

Just like the starter kit, it is lightweight and comes with the essentials only. This boilerplate will be a good choice if you need to start quickly. The downside is that it doesn't include a design system, so you either have to build your own or use an existing one from the community.



Building Your Own Boilerplate

Below, we'll introduce you to some React Native boilerplates and show you how you can build your app in minutes.

Wix React Native UI kit →

A great UI library used by Wix in their own React Native application. It's quite mature.

React Native Navigation Hooks →

A third-party library based on the Wix navigation library. It provides hooks to work with the navigation of hook-based components in React.

React Redux →

A library for state management; there are alternatives like MobX, but we prefer Redux as it's easier to set up and straightforward to work with.

React Native Vector Icons →

A very interesting library that packs over 20,000 icons.

React Native Navigation →

Don't confuse it with React Navigation. It provides ways to organize the content layout on your mobile screens in a logical and performant manner. We find this stack a lot easier to work with. Again, it is powered by Wix, who use it in their application. It is very flexible and easy to work with and maybe a bit hard to set up, but Wix provide comprehensive documentation that you can leverage.

React Native SVG →

It allows you to import SVG as React components into your code. It is helpful for icons or vector graphics that you want to have in your app.

Flipper →

A debugger powered by Facebook.

Jest →

A test runner and a testing framework for JavaScript/TypeScript.

Axios →

A promise-based HTTP client for JavaScript

A tutorial on how to add custom fonts in React Native →

React Native has a large developer community. If you need something, before you reinvent the wheel, do thorough research and check if the community hasn't already developed the feature you're looking for.

Once you pick the technology stack, it's time for your development team to get their hands dirty.

Publishing Your Mobile App

Once you are done developing your mobile app, you can send it to Google Play Store and Apple app store for publishing. There's a review process that takes time, sometimes up to a week, so you should take this into account when scheduling your app release cycles.

At the time of writing (11/21/2022), Google Play Store ask for a $25 one-time publishing fee, while publishing on the Apple Store will cost you $99/year.

Congrats! Your mobile app has been successfully published on the app stores.

Final words

Are you developing a mobile application with a snappy, responsive UI that works on any device?

Considering a modernization project that demands an enterprise development approach with modern technology in mind?

Regardless of the task, deciding on the technology stack, validating your choice, and adjusting the technology to fit your custom requirements can be challenging. More importantly, a lot of work needs to be done before rushing into development – building the business case, validating the tech stack, designing your prototype, and developing and deploying your app while continuously enhancing its performance can be overwhelming.

At Resolute Software, we specialize in solving these challenges. We have a problem-focused approach -

We aim to understand your business dynamics first and create a solution that works best for you.

We are experts in UI, combining UX/UI design and pixel-perfect app development. This way, The Resolute team offers full-scale React development while mitigating risks from dubious-quality community-sourced components. Our rich skill set guarantees a quick go to market, safeguarding your project from expensive, time-consuming complications.

Resolute

sales@resolutesoftware.com

Let's talk about your technology requirements.

Get in touch

USA

MA 01701, Framingham,
945 Concord St,

+1-617 386-9697

Deloitte. Company to watch | Tech Fast 50

Business Professional Services

Clutch ★★★★★

People First Company Award
2020 | 2021 | 2022