

Moving to React Native for Scale & Efficiency

How to successfully migrate your app for streamlined development



TABLE OF CONTENTS

Maximizing Efficiency and ROI from Your Tech Investments	2
The React Native Advantage in Cross-Platform Development	3
Evaluating Native vs. Cross-Platform App Development	4
Decision Framework: Native App vs. Cross-Platform App	5
Key Benefits of Single Codebase Migrations	6
Evaluation Checklist: Determining the Suitability of React Native for Your Project	7
React Native Migration Case Study	9
Five Essential React Native Migration Tips & Learnings for Tech and Product Leaders	15
The Next Steps to Success	17

Maximizing Efficiency and ROI from Your Tech Investments

As companies of all sizes seek solutions to improve efficiency and ROI from their tech investments, options like migrating existing products to newer frameworks and more efficient approaches come up frequently. But technology migrations can be large and complicated endeavors. To succeed in these migrations, companies need both a solid business case and a clear, comprehensive execution plan – along with the right resources.

If “technology migrations” just made your heart skip a beat or two, odds are you’ve already seen just how badly things can go in the worst-case scenario.

At Whitespectre, we see a growing movement among companies seeking to consolidate their native applications into

a single, more manageable codebase framework – with React Native being the most popular choice. This strategic shift towards a single, unified codebase comes with massive potential. This single codebase approach is more than just a trend – it's a transformative approach.

By converging iOS and Android development efforts, organizations can substantially reduce double-work in areas that overlap, as well as bring together siloed iOS and Android teams, saving build and maintenance costs by as much as 30%*. This leads to reductions in both maintenance overheads and the costs associated with rolling out new features.

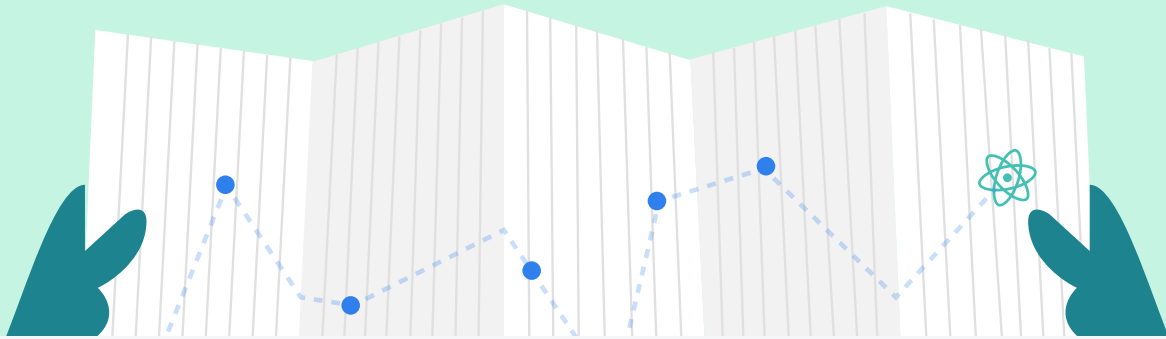
**Merryfield case study, page 9*



Like any good investment opportunity, a replatforming project has a huge upside potential over the long term; the increased pace and efficiency of future development, the improved skillset-resilience within the team and the automatic consistency of behavior between iOS and Android. To shorten the time to start realizing a return, the project to rebuild and rollout must be well planned and executed.”



Nick Tudor
CEO, Whitespectre



The React Native Advantage in Cross-Platform Development

Since its introduction by Facebook (now Meta Platforms) in 2015, React Native has risen in popularity and matured into a leading JavaScript framework to build complex cross-platform applications that deliver a native user experience. Using the same building blocks as native iOS and Android apps, React Native provides a faster, more simplified way to get mobile applications running on multiple platforms, with a streamlined and efficient development process, enabling applications to smoothly operate across multiple platforms.

It also provides robust and reliable tools that allow developers to achieve the same results with far fewer resources as compared to their native counterparts. This accelerates the development timeline with significantly fewer resource-based costs.

In recent years, React Native has grown to support multiple platforms, fostered a sizable, vibrant and engaged developer community, and offers improved performance and better developer tools.



Microsoft

Uber Eats



TESLA

PlayStation®

Evaluating Native vs. Cross-Platform App Development



Native applications have long been known for their higher performance and ability to provide better, highly tailored user experiences specific to their respective iOS and Android platforms. This specialization enables native apps to leverage the full spectrum of device capabilities and technologies, often resulting in better application responsiveness and user engagement.

Even so, for businesses with established products running on applications with native-specific features, the commitment to native app development means a significant investment. The need to develop, update, and maintain separate codebases for iOS and Android not only escalates operational costs but also increases the time required to deploy features and fixes across two parallel platforms.

As codebases are completely different between platforms, the development effort is doubled, as are the skills required from the team.

Decision Framework: Native App vs. Cross-Platform App

A structured comparison supports the decision between native and cross-platform approaches, ensuring alignment of each option with your specific business needs, project characteristics, and overarching goal.


 NATIVE APP WHEN:	 CROSS-PLATFORM APP WHEN:
<p>Specialized Native Features are Crucial: Your application requires access to advanced, device-specific functionalities or intends to leverage the latest technology exclusive to a platform.</p> <p>High-Performance 3D Games and Animations: You're developing visually intensive 3D games or applications where animation performance is non-negotiable.</p> <p>Single-Platform Utility or High-Performance Apps: The project is focused on delivering a high-performance application tailored to a specific operating system, or when computational demands exceed what cross-platform technologies can efficiently handle.</p>	<p>Efficient Multi-Platform Release is a Priority: You aim to launch your mobile application across iOS and Android simultaneously, working within constrained timelines or resource limitations.</p> <p>Broad User Base Targeting: It's essential to cater to both iOS and Android users without a preference or priority for one platform over the other.</p> <p>Speed in Development and Maintenance is Key: Your project benefits from quicker development cycles and easier maintenance updates, reducing time to market and ongoing support costs.</p> <p>Uniform Functionality Across Platforms: The app's core functions work universally for a seamless user experience on all devices.</p>

If your ecosystem includes web applications developed with React, opting for React Native allows for the reuse of code components, enhancing development efficiency and consistency across platforms and the web.

Key Benefits of Single Codebase Migrations

Cost Efficiency: Transitioning to a single codebase can significantly reduce the long-term operational costs associated with parallel development paths. This is a pivotal consideration, irrespective of the current phase of active development for your app.

Simplified Implementation and Maintenance: A unified codebase streamlines both the implementation of new features and the maintenance of existing ones. It mitigates the risk of feature discrepancies between platforms, therefore enhancing the efficiency of technical teams and improving the coherence of customer service and marketing efforts.

 *Regardless of how actively you're developing an app, maintaining a single codebase can reduce the ongoing cost of your project. Simplifying implementation and maintenance for a single common technical product will lower likelihood for features to be out of sync between the platforms. This will result in efficiencies not only for the tech team, but for customer service and marketing as well.*



Evaluation Checklist: Determining the Suitability of React Native for Your Project

Migrating to a single codebase with React Native is a strategic decision that should be made with a comprehensive understanding of your product's specific needs, the potential for unification of the user experience, and the scope of work involved. This approach not only enables gains in operational efficiency, but also aligns with a vision for a more integrated and seamless product ecosystem.

1. Percentage of Native-Specific Functionality

- Assess the extent of your app's reliance on native features. Is it 60%, 30%, or 10%? A higher proportion of native-specific functionality could affect the potential benefits of unifying under a single codebase. Your return is directly proportional to how much you can unify the codebase.
- How core is that native functionality to the core value your product delivers? The significance of these features both now and in the foreseeable future plays a crucial role in this evaluation.

- Are there any additional native features in your product's roadmap? Current and planned native functionality is key when considering how much your product would benefit from having a single codebase.

2. Assessing User Experience Consistency

- If there's a variance in the user experience across your iOS and Android apps, determine the feasibility and desirability of unifying these experiences. The ability to offer a consistent user experience post-migration can significantly influence the success of adopting React Native.

Evaluation Checklist:

- Acknowledge that while a unified codebase offers many advantages, certain platform-specific UI and UX expectations may still need to be met. React Native's flexibility in customization should be considered as part of your strategy.
- Is there a timeline you need to meet in order to move to a smaller consolidated team? Identify any critical deadlines that could influence the migration plan. The ability to meet these deadlines while moving towards a more streamlined team structure is essential.

Always keep in mind that iOS and Android users will not expect the exact same UI & UX in specific app interactions and elements, but React Native allows you to customize it when needed.

3. What is the Scope of Work to be Undertaken?

- Clearly define whether the migration involves transitioning from a single platform to React Native or consolidating two distinct apps into one codebase. The scale of this transition will significantly impact your strategy.
- Can you identify an approach where you can make a gradual transition to a single codebase without impacting feature releases? Explore the possibility of a phased approach to migrating to a single codebase. This could mitigate risk and allow for continuous feature development without significant interruptions.

React Native Migration Case Study: Merryfield's Successful Transition to a Unified Codebase

In this section, we share the story of how we unified the codebase of a 2-year-old iOS Native app with its React Native-built Android app counterpart.

PROJECT OUTCOMES

- Exponentially increasing the app's weekly active user count on both platforms.
- A 4.9 stars rating on iOS and 4.7 stars on Android ★
- A 30% cost reduction developing and maintaining two apps in parallel, following a friction-free migration.
- Continued launching of new features on both apps while working to parity and a seamless iOS cut-over.
- Exciting new brand and retail partnerships along with tons of user feedback on the amazing impact Merryfield brings through their platform.

Background: Merryfield approached us in 2019, seeking a tech partner to help them launch the iOS application that would be the cornerstone of their entire business.

Merryfield's vision was to develop a loyalty platform for better-for-you brands. Customers would discover and engage with these brands via an app that would reward them with points and special offers each time they purchased from these brands.

Merryfield knew that to succeed they needed to distinguish themselves from the existing loyalty apps, most of which were cluttered and transactional. Instead, Merryfield wanted a beautiful, easy-to-use app with engaging animations, and clean design – the perfect showcase for getting this new type of brand on board. The app would require significant native functionality to support receipt scanning.

The Evolution of Business Goals and Technology


By early 2021, Merryfield had new decisions to make. Business goals change over time, and over the course of 2.5 years, our client now has a vast selection of brands on a highly-rated app, so expanding the user base was the top priority. The Android audience was an obvious next win.

And naturally, we needed to ensure that the Android app would meet the same high level of standards as the iOS app – complete with an appealing UI, UX, animations, flawless performance, and of course, stability.

Key considerations for us as their tech partner were the selection of a development framework for the Android app that aligned with efficiency, maintenance costs, and time-to-market advantages and the evolved state of React Native, assessing its

ability to deliver the required user experience level:

- Which programming framework should Merryfield go with for the new app – native Android, React Native, or something else?
- How would their choice impact the effort to maintain the two apps in parallel? Specifically – did the introduction of Android give us the opportunity to move the product toward a single codebase and all the benefits that could bring i.e. efficiency, lower maintenance costs and faster time to market?
- Had React Native's capabilities evolved enough over the past 2.5 years, to give us confidence that a React Native app would offer the level of UX we needed?



For Merryfield to grow, we absolutely needed to launch Android to cover the market. The natural, organic channel is fragmented. These brands need a way to directly reach their consumers and tell their story, so offering that central solution is even more critical.”



Jennie Aleckson

Co-Founder, Head of Digital Product, Merryfield



Key Roles

The Whitespectre managed delivery team, instrumental in bringing this vision to life, was thoughtfully composed to cover all critical aspects of app development:

UI/UX Designer

Focused on ensuring the app's visual and interaction design met high usability standards and provided a seamless experience across both platforms.

React Native Tech Lead

Provided technical direction, ensuring the application's architecture was robust, scalable, and leveraged React Native's strengths to the fullest.

React Native Engineers

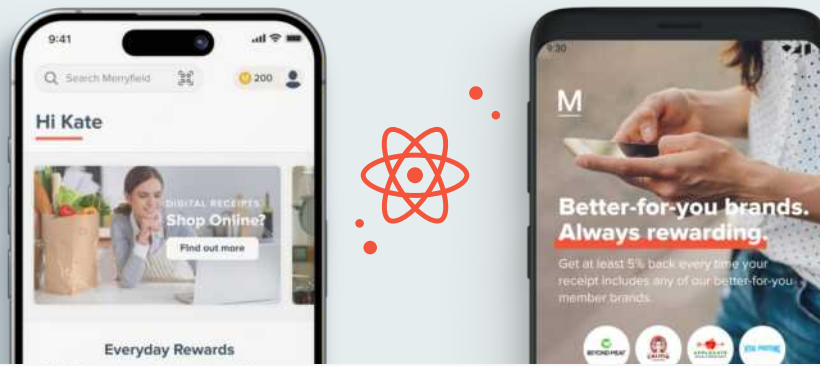
Tasked with executing the development plan, these engineers transformed designs and specifications into a functional, high-quality app.

QA Engineer

Ensured the app met all quality standards, conducting thorough testing to identify and resolve any issues before release.

Product Manager

Acted as the project's leader, aligning business goals with development efforts and ensuring the project stayed on track to meet its objectives.



The Process

The React Native Android app development and iOS app migration project was broken down into 4 key phases:

Phase 1: Building the Android app on React Native

The initial phase focused on developing the Android version of the app, ensuring that the core functionalities and user experience were on par with the iOS version, despite the platform differences.

Phase 2: Continue Releasing to Get to Full Feature Parity

Once the Android app was live, the team worked to ensure that any new features or updates were simultaneously developed for both platforms, maintaining consistency and feature parity.

Phase 3: Migrating the iOS app to React Native

The migration of the iOS app to React Native was a significant undertaking. It involved re-architecting the app to not only match the existing functionality and performance but also to leverage the benefits of a unified codebase.

Phase 4: Transitioning to Simultaneous Development Under a Single Codebase

The final phase marked the transition to a single codebase for both platforms, significantly streamlining the development process, making maintenance more manageable, and facilitating quicker updates and feature rollouts.

From Roadmap to Reality: The Process Desigmet to Deliver

We kicked off the project with an 8-week inception phase. This deep foundational work we put in at the beginning – to align on business goals, key success metrics, and a high-level approach – was key to ensure success of Merryfield’s React Native Android app development and iOS app migration.

The important conversations we had during the inception phase with the Merryfield team enabled us to:

- Maximize the efficiency and effectiveness of our tech team, and the process for all stakeholders.
- Significantly reduce the chance of wasted work, unnecessary burn, and delay.

Inception phase deliverables included:

- Mapping out screens and user flows and the shared nomenclature between projects.
- Identifying Android UI requirements and the working process going forward.
- Breaking down epics into stories and creating estimates.
- Aligning on the roadmap and building an execution plan.
- Tech investigations for Microblink/
native Android components.

We began building the Android app on React Native by meticulously replicating the existing iOS native app, only deviating where appropriate per Google's Material Design guidelines. Once complete and in users' hands, we commenced the 'consolidation' phase where we adapted the experience of the app in React Native for iOS users, replicating the existing UX and UI exactly. Both apps work, look and feel the exact same way.

Once consolidated, we drove forward efficiently by developing new features on the single codebase.

 Microblink/nati

Outcomes

The strategic decision to migrate Merryfield's app to a unified React Native codebase had profound implications:

Unified User Experience

Users on both iOS and Android platforms now enjoy a seamless and consistent experience, strengthening brand loyalty and user satisfaction.

Reduced Development and Maintenance Costs

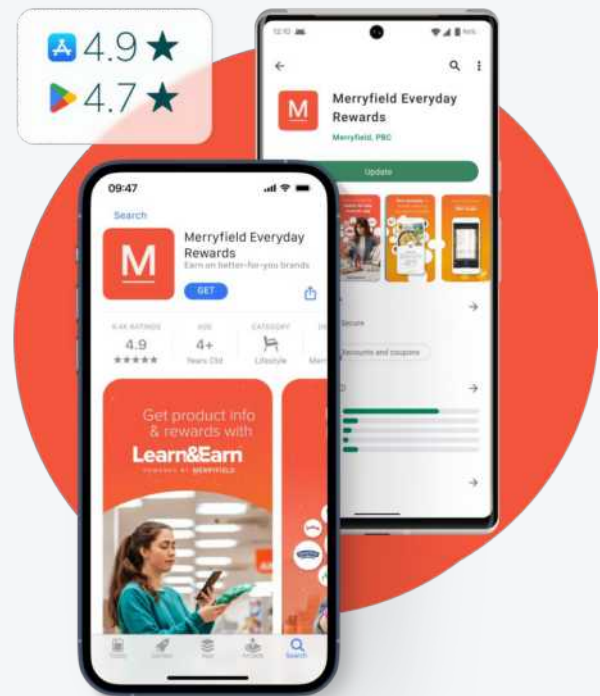
The single codebase approach has significantly cut down development time and costs, as updates and features now require less time to implement across platforms.

Agility in Feature Development and Market Responsiveness

With a more efficient development process, Merryfield can now respond more swiftly to market trends and user feedback, maintaining a competitive edge in the loyalty platform market.

Expanded User Base

By extending the app's availability to Android, alongside improving the overall app quality, Merryfield successfully broadened its reach and attracted a larger segment of users interested in healthy lifestyle choices, significantly increasing its user base, and consequently making it even more attractive to potential brand partners on the B2B side.



Five Essential React Native Migration Tips & Learnings for Tech and Product Leaders

1. Inception time pays off

Don't underestimate the power of a thorough inception phase for a migration project. It's the groundwork that will ensure your team takes steady steps to success. Starting your migration project with a well-thought-out inception phase is crucial for laying a solid foundation, aligning your team's understanding of the project's goals, scope, and constraints. It's an investment that prepares your team for a confident approach to the migration, ensuring that every step taken is in the right direction.

2. Do your homework to understand the must-keep native features on your app

Gather all the background needed on your app in terms of functionality and UI elements. Every bit of information matters when adjusting for iOS and/or Android. Understanding the intricacies of your current app, especially the native features essential to its functionality and user experience, is critical. Audit your app's functionality, UI elements, and user interactions. This deep dive will equip you to make informed decisions about what needs to be preserved, enhanced, or possibly reimaged in the migration process.

3. Assemble a savvy team on cross-platform app behavior

React Native engineers are a must, and having native knowledge is a plus. Your UI/UX team needs to be experts on Material Design and Human Interface guidelines and where the crossover points exist. The composition of your team can significantly impact the outcome of your migration project, and this cross-functional expertise is pivotal for tackling platform-specific challenges and ensuring a cohesive and seamless app experience.



4. Maintain the two platform lenses all the way

Your team needs to keep in mind that your app must work for both platforms, and adjust each and every requirement you define based on that perspective. Keeping the dual nature of your project in mind is key throughout the migration process. Every decision, from feature development to UI adjustments, should be evaluated through the lens of both iOS and Android platforms. This holistic approach ensures that the end product performs flawlessly and resonates well with users on both platforms, maintaining a balance between platform-specific expectations and a unified app identity.

5. Master your roadmapping skills

The thinking around the app's feature parity, the app's consolidation, and setting up a realistic timeline are the foundations of successful execution and delivery. This strategic foresight enables you to set achievable milestones, manage expectations, and guide your team through the complexities of the migration with clarity and purpose.



The Next Steps to Success

In 2024, React Native is poised for greater heights with Fabric and TurboModules – new, faster, and more reliable architecture enhancements already on the horizon. Supported by increased investments from Meta and an ever-expanding global developer community, React Native's role as a leading cross-platform framework is further solidified. This evolution ensures enhanced cross-platform stability and streamlined development, enabling a truly unified user experience.

Both startups and established enterprises managing legacy applications will benefit from the faster customization and delivery that these upcoming improvements enable. Products stand to gain from faster feature development and iteration cycles, more efficient testing protocols, and significantly reduced time-to-market.

Recap: The Foundation for a Successful Migration

Product Alignment:

Assess whether React Native aligns with your product's requirements and future roadmap. The framework's evolving capabilities make it an increasingly compelling choice for a wide range of applications, beyond the simplest content apps to more complex platforms.

Assembling the Right Team:

The success of your migration largely depends on the expertise and insight of your team. Ensure you have access to skilled React Native developers, experienced UI/UX designers familiar with cross-platform considerations, and a project management team capable of navigating the complexities of the migration process.

The ongoing evolution of React Native offers businesses the opportunity to streamline app development, enhance user experience, and accelerate market response. Transitioning to a unified React Native codebase can unlock efficiency and innovation, positioning your product for success. If you have the right fit, strategy, and team, the benefits are too good to miss. Whitespectre can support you along the way.

Strategic Migration Plan:

Develop a clear, well-thought-out migration strategy. This plan should consider your current app's architecture, the specific needs of your user base, and how to leverage React Native's strengths to meet those needs effectively.

About Whitespectre

Whitespectre is a full-service agile development company and a long-term partner of choice for companies of all sizes – from high-growth startups to \$1B enterprise players. Founded in 2013, we've helped our client partners within the US and Europe deliver innovative software that rapidly scales businesses, accelerates growth, and delights millions of users.

Whitespectre's core offering is managed delivery teams. Our engineers, designers, and product managers operate as agile partners for our clients, bringing deep

expertise while covering software delivery needs end-to-end. As a company, we specialize in large-scale platform development as well as creating best-in-class native mobile and web applications, for both consumer-centric and B2B businesses. As strategic partners for our clients, we provide executive tech advisory and additional consulting services across product management, UI/UX, and technical architecture design.

Learn more at whitespectre.com