웰위(Well-We) 디자인 시스템: React Native 기반 크로스 플랫폼 확장을 위한 전략적 아키텍처 및 컴포넌트 규격 설계 보고서

디지털 웰니스를 위한 디자인 시스템의 전략적 가치와 지향점

현대 디지털 생태계에서 서비스의 성공은 단순히 기능의 나열이 아닌, 사용자가 느끼는 일관된 경험의 질에 의해 결정된다. 특히 건강과 웰빙을 다루는 '웰위(Well-We)'와 같은 서비스는 사용자의 신뢰와 정서적 안정을 구축하는 것이 필수적이며, 이를 위해 모든 접점에서 시각적, 기능적 정교함이 요구된다. 디자인 시스템은 이러한 일관성을 담보하는 '단일 진실 공급원(Single Source of Truth)'으로서, 디자이너와 엔지니어 간의 협업 효율을 극대화하고 제품의 확장성을 보장하는 중추적인 역할을 수행한다.[1]

React Native 기술 스택을 기반으로 모바일 앱에서 웹까지 확장을 고려하는 웰위 디자인 시스템은 플랫폼 간의 경계를 허무는 '유니버설 아키텍처'를 지향해야 한다. 조사에 따르면 현재 공개된 디자인 시스템 중 웹 이외의 플랫폼을 지원하는 비중은 10% 미만에 불과하며, 대다수가 웹 중심적 사고방식에 머물러 있다.[2] 따라서 웰위는 초기 단계부터 플랫폼 중립적인 기반 위에 각 플랫폼의 고유한 사용자 경험(UX) 패턴을 수용할 수 있는 유연한 설계를 채택해야 한다. 이는 단순히 화면 크기를 조정하는 것을 넘어, 터치 중심의 모바일 환경과 키보드/마우스 중심의 웹 환경 사이의 간극을 메우는 포괄적인 접근을 의미한다.[3]

크로스 플랫폼 디자인 시스템의 기술적 토대와 아키텍처

React Native는 자바스크립트 컴포넌트를 iOS의 UIView나 Android의 View와 같은 네이티브 위젯으로 번역함으로써 웹뷰 방식과는 차별화된 성능과 사용자 감각을 제공한다.[4] 이러한 네이티브 경험을 유지하면서 웹으로 확장하기 위해서는 기술적 구조의 모듈화와 재사용성이 핵심이다.

모노레포 및 코드 공유 전략

효과적인 자원 관리를 위해 모노레포(Monorepo) 아키텍처의 도입이 권장된다. 단일 저장소 내에서 UI 라이브러리, 유틸리티, 도메인 SDK를 분리하여 관리하면 버전 관리의 일관성을 확보하고 리팩토링 효율을 극대화할 수 있다.[5] 이 구조에서 디자인 시스템은 독립적인 패키지로 존재하며, 모바일 앱과 웹 애플리케이션이 공통의 디자인 토큰과 컴포넌트 로직을 소비하게 된다.

플랫폼 통합 스타일링 솔루션 분석

React Native 환경에서의 스타일링은 CSS와는 다른 제약 조건을 가지며, 2025년 현재 성능과 개발 효율성을 모두 잡기 위한 다양한 솔루션이 공존하고 있다.[1]

솔루션	기술적 메커니즘	전략적 이점	관련 출처
NativeWind	Tailwind CSS를 React Native에 컴파일	웹과 모바일 간의 스타일 공유 극대화, 빌드 타임 최적화	[7]
Shopify Restyle	타입 안정성 기반의 테마 시스템	엄격한 디자인 규격 강제, 다크 모드 전환의 용이성	[1]
Tamagui	웹 추출 CSS 및 모바일 런타임 최적화	최상위권 성능 제공, 복잡한 엔터프라이즈 시스템에 적합	[7]
gluestack UI	비스타일(Unstyled) 접근성 컴포넌트	접근성 표준(ARIA) 준수 용이, 스타일링 자유도 높음	[7]

웰위 디자인 시스템은 특히 개발자 경험(DX)과 시각적 정교함이 중요하므로, 타입 안정성이 뛰어난 Restyle이나 웹 플랫폼과의 스타일 일치성이 높은 NativeWind를 핵심 스타일링 엔진으로 고려하는 것이 바람직하다.[7]

디자인 토큰: 시스템의 원자적 구성과 명명 규칙

디자인 토큰은 색상, 간격, 타이포그래피와 같은 추상적인 스타일 값을 명명된 변수로 변환하여 플랫폼 간의 시각적 언어를 통합한다.[1] 토큰은 단순히 값을 저장하는 도구가 아니라, 디자인 의도를 전달하는 시맨틱(Semantic) 매개체여야 한다.

시맨틱 명명 체계 및 계층 구조

토큰의 명칭은 '무엇인가(What)'가 아닌 '어디에 쓰이는가(Where/Why)'를 나타내야 한다. 예를 들어 red-500 대신 color.surface.error와 같은 명칭을 사용함으로써, 향후 브랜드 색상이 변경되거나 다크 모드가 적용되더라도 컴포넌트 코드를 수정할 필요 없이 토큰 값만 조정하여 시스템 전체를 업데이트할 수 있다.[1]

토큰 계층	역할 및 정의	예시
Global (Base)	시스템에서 사용 가능한 모든 원시 값	global.palette.blue.600
Semantic (Alias)	특정 역할이나 의미가 부여된 참조 값	semantic.color.primary.background
Component	특정 컴포넌트의 세부 속성에 할당된 값	component.button.primary.label

이러한 계층 구조는 Figma의 변수 관리 방식과 일치하며, JSON 기반의 중앙 저장소를 통해 관리되어야 한다.[9] Bolt의 사례에 따르면, 디자이너가 소유한 JSON 저장소에서 엑스포트 스크립트를 통해 Android XML, iOS Assets, React Native Styles로 자동 변환되는 파이프라인을 구축함으로써 대규모 업데이트 시 발생할 수 있는 인적 오류를 원천 차단할 수 있다.[9]

핵심 인터랙션 컴포넌트 규격 및 설계 규칙

컴포넌트는 단순히 시각적 요소의 결합이 아니라, 특정 기능을 수행하는 독립적인 소프트웨어 단위로 설계되어야 한다.[11]

버튼**(Button)** 및 터치 인터랙션

버튼은 사용자의 의지를 행동으로 전환하는 가장 중요한 컴포넌트이다. React Native에서는 기본 Button 컴포넌트의 제약을 넘어 Pressable이나 TouchableOpacity를 활용한 커스텀 버튼 설계가 일반적이다.[11]

- 터치 타겟**(Touch Targets):** 모바일 환경에서 사용자의 물리적 제약을 고려하여 최소 44x44px(Apple 기준) 또는 48x48px(Google 기준) 이상의 영역을 확보해야 한다.[14]
- 상태 정의**:** 기본(Default), 눌림(Pressed), 호버(Hover), 비활성(Disabled), 로딩(Loading) 상태를 포함해야 하며, 각 상태 전환 시 시각적 피드백이 즉각적으로 제공되어야 한다.[15]
- 애니메이션 속도**:** 상태 변화 애니메이션은 150-200ms의 딜레이를 통해 우발적인 트리거를 방지하고, 250-300ms의 트랜지션을 통해 부드러운 연결감을 제공하는 것이 이상적이다.[17]

입력 시스템**(Inputs)** 및 데이터 검증

웰위 서비스의 사용자 데이터 입력은 신뢰감을 주어야 한다. 기본 TextInput을 래핑하여 브랜드 아이덴티티가 투영된 커스텀 입력 컴포넌트를 구축하는 것이 필수적이다.[18]

- 스마트 피드백**:** 사용자가 입력을 시작하면 레이블이 상단으로 이동하거나 색상이 변하는 등 활성 상태를 명확히 표시한다.[18]

- 유효성 검사 로직**:** 실시간 검증(OnChange)은 모바일 환경에서 입력 흐름을 방해하고 성능을 저하시킬 수 있으므로, 포커스를 잃었을 때(OnBlur) 검증하는 방식을 기본으로 채택한다.[19]
- 특수 입력 유형**:** 검색 필드는 입력 초기화(Clear) 버튼과 데바운싱(Debounce) 처리가 내장되어야 하며, 비밀번호 필드는 가시성 토글 기능이 포함되어야 한다.[18]

데이터 전시**:** 카드**(Cards)** 및 리스트**(Lists)**

카드는 정보를 논리적인 단위로 그룹화하는 컨테이너이다.[21] 웰니스 콘텐츠는 정보의 밀도보다 가독성과 스캔 용이성이 중요하므로 여백과 시각적 계층 구조가 강조되어야 한다.

카드 구성 요소	설계 규칙	관련 지침
컨테이너(Container)	고도(Elevation)나 외곽선을 통해 표면 분리	[22]
미디어(Media)	이미지나 비디오 배치, 종횡비 유지	[22]
헤더/타이틀	정보의 핵심을 요약, 최대 2줄 이내 권장	[23]
액션 영역	기본 액션은 카드 전체, 보조 액션은 별도 버튼	[22]

대규모 데이터를 표시하는 리스트의 경우, 메모리 효율을 위해 반드시 FlatList나 SectionList를 사용하여 가상화 성능을 확보해야 하며, 데이터가 없는 경우를 대비한 빈 상태(Empty State) 디자인이 반드시 포함되어야 한다.[4]

사용자 감각의 정교화**:** 햅틱 및 마이크로 인터랙션

디자인 시스템의 완성도는 시각을 넘어 촉각적 피드백인 햅틱(Haptic)에서 완성된다. 햅틱은 사용자의 행동에 대한 물리적 메타포를 제공하여 인지 부하를 줄이고 만족도를 높인다.[26]

햅틱 피드백 설계 원칙

햅틱은 '적을수록 좋다'는 원칙 아래 신중하게 사용되어야 한다. 과도한 진동은 사용자의 주의를 분산시키고 피로감을 유발할 수 있다.[26]

- 강도와 중요도의 상관관계**:** 빈번한 작업(스크롤, 항목 이동)은 아주 미세한 틱(Tick)을 사용하고, 중대한 작업(성공적인 저장, 오류 발생)은 더 강하고 뚜렷한 진동을 적용한다.[26]
- 샤프니스**(Sharpness)**와 강도**(Intensity):** Apple의 햅틱 엔진 철학을 차용하여, 동작의

성격에 따라 날카롭거나 둔탁한 감각을 구분하여 설계한다. 예를 들어 체크박스 선택은 날카로운 틱을, 페이지 새로고침 완료는 부드럽지만 묵직한 진동을 사용한다.[28]
- 시청각 동기화**:** 햅틱은 시각적 애니메이션 및 소리와 완벽하게 동기화되어야 한다. 엇박자로 발생하는 피드백은 사용자에게 기기 고장과 같은 부정적인 인상을 줄 수 있다.[26]

플랫폼 적응형 레이아웃 및 반응형 전략

웰위 디자인 시스템은 모바일 앱으로 시작해 웹까지 아우르기 때문에, 다양한 화면 크기에 유연하게 대응하는 반응형 전략이 아키텍처 수준에서 설계되어야 한다.[29]

2025년 기준 권장 중단점 및 레이아웃 규칙

중단점**(Breakpoint)**	대상 장치 너비	주요 레이아웃 전략
Compact (Mobile)	< 600px	단일 열, 하단 탭 바, 엄지손가락 중심 UI
Medium (Tablet)	600px - 900px	다단 레이아웃, 사이드 네비게이션 전환 검토
Expanded (Web)	> 900px	그리드 확장, 카드 가로 배치, 마우스 호버 강화

반응형 디자인 구현 시 고정된 픽셀값보다는 rem, em 또는 clamp()와 같은 유동적인 단위를 사용하는 것이 웹 플랫폼에서의 가독성 확보에 유리하다.[14] React Native Web에서는 useWindowDimensions 훅을 통해 현재 뷰포트의 너비를 감지하고 적절한 스타일 클래스나 컴포넌트를 조건부로 렌더링하는 전략을 취한다.[6]

다크 모드 및 포용적 접근성 설계 가이드라인

접근성은 디자인 시스템의 선택 사항이 아닌 필수 요구 사항이다. 특히 건강 관련 서비스는 시각적 제약이 있는 사용자나 고령층을 포함한 다양한 스펙트럼의 사용자를 고려해야 한다.[24]

다크 모드 설계의 심층적 접근

다크 모드는 단순히 색상을 반전시키는 것이 아니라, 저조도 환경에서의 가독성을 극대화하는 방향으로 설계되어야 한다.[34]

- 대비 및 색상**:** WCAG 2.1 가이드라인에 따라 일반 텍스트는 4.5:1, 큰 텍스트는 3:1 이상의 대비 비율을 확보해야 한다.[34] 순수 검정색(#000000)보다는 짙은 회색(#121212) 배경을 사용하여 눈의 피로를 유발하는 고대비를 완화한다.[34]
- 채도 조절**:** 밝은 배경에서 선명하게 보이던 원색은 어두운 배경에서 눈부심을 유발할 수

있으므로, 다크 모드 전용 토큰에서는 채도를 낮춘(Desaturated) 파스텔 톤이나 뮤트 톤을 사용한다.[34]
- 고도 표현: 그림자(Shadow)는 어두운 배경에서 시각적 깊이감을 표현하기 어려우므로, 배경색 위에 더 밝은 오버레이를 얹어 고도를 표현하는 방식을 채택한다.[34]

접근성 기술 규격 **(A11y)**

- 시각적 보조: 모든 이미지 컴포넌트에는 accessibilityLabel을 필수적으로 포함하여 스크린 리더(VoiceOver, TalkBack)가 내용을 설명할 수 있도록 한다.[33]
- 동작 제어: 사용자가 시스템 설정에서 '동작 줄이기(Reduce Motion)'를 활성화한 경우, 앱 내의 모든 비필수 애니메이션을 중단하거나 즉각적인 상태 전환으로 대체해야 한다.[37]
- 포커스 관리: 모달이나 팝업이 열릴 때 포커스를 내부의 첫 번째 요소로 강제 이동시켜 사용자가 맥락을 잃지 않게 관리한다.[33]

차세대 디자인 시스템: **AI** 코딩 및 문서화 전략

최근 디자인 시스템의 역할은 인간 개발자를 위한 가이드를 넘어, 생성형 AI(Claude, Cursor 등)가 정확한 코드를 작성할 수 있도록 돕는 사양(Specification) 데이터로서의 가치가 커지고 있다.[38]

웰위 디자인 시스템은 AI 친화적인 시스템으로 거듭나기 위해 다음과 같은 사양 체계를 갖추어야 한다.

1. 구조화된 마크다운 사양: 각 컴포넌트의 의도, Props 인터페이스, 상태 관리 로직을 정형화된 마크다운 형식으로 기술한다. PLAIN(Product Language for AI Notation)과 같은 규격은 AI가 시스템의 제약 조건을 이해하고 프로덕션 급 코드를 생성하는 데 도움을 준다.[39]
2. **LLM** 컨텍스트 파일: 프로젝트 루트에 llms.txt와 같은 파일을 배치하여 AI가 시스템의 전체적인 규칙, 색상 팔레트, 컴포넌트 사용법을 빠르게 학습할 수 있는 환경을 제공한다.[41]
3. 컴포넌트 인벤토리 자동화: 코드로 구현된 컴포넌트의 API를 자동으로 추출하여 문서화하고, 이를 다시 설계 도구인 Figma와 동기화하는 '디자인-코드 일치 자동화' 파이프라인을 구축한다.[42]

결론: 웰위 디자인 시스템의 단계적 구축 및 실행 로드맵

웰위 디자인 시스템은 단순한 라이브러리 구축을 넘어, 제품 개발 문화의 변화를 이끄는 핵심 자산이 되어야 한다. 연구를 통해 도출된 핵심 성공 요인은 다음과 같다.

첫째, 플랫폼 중립적인 디자인 토큰을 설계의 중심에 두어야 한다. 이는 모바일과 웹이라는 서로 다른 기술 환경을 잇는 유일한 공통 언어이며, 향후 서비스 확장이나 브랜드 변화에 대응하는 가장 강력한 무기가 될 것이다.[9]

둘째, 성능과 접근성을 컴포넌트 설계의 최우선 순위로 설정해야 한다. 특히 건강 관련 서비스에서 로딩 지연이나 조작의 어려움은 사용자 이탈의 결정적 요인이 되므로, React

Native의 성능 최적화 기법과 WCAG 접근성 표준을 아키텍처 수준에서 강제해야 한다.[3]

셋째, 도구 간의 긴밀한 통합이다. Figma 설계 단계부터 React Native 구현 코드, 그리고 AI 코딩 어시스턴트가 소비하는 문서 사양에 이르기까지 데이터가 단절 없이 흐르는 에코시스템을 구축함으로써 개발 효율을 극대화할 수 있다.[1]

이러한 전략적 기반 위에 구축된 웰위 디자인 시스템은 모바일 앱의 정교한 감각과 웹의 광범위한 접근성을 동시에 확보하며, 사용자에게 정서적 안정과 건강한 가치를 전달하는 신뢰받는 서비스로 성장하는 토대가 될 것이다. 디자인 시스템은 완성이 없는 살아있는 유기체와 같으므로, 지속적인 사용자 테스트와 피드백 수용을 통해 시스템을 고도화해 나가는 과정이 필수적이다.[24] 이를 통해 웰위는 기술적 우수성과 사용자 중심 철학이 결합된 독보적인 디지털 웰니스 플랫폼으로 자리매김할 수 있을 것이다.

참고 자료

1. React Native for Design Systems - Ropstam Solutions Inc., 1월 15, 2026에 액세스, https://www.ropstam.com/react-native-for-design-systems/
2. An Introduction to Multi-Platform Design Systems - dbanks design, 1월 15, 2026에 액세스, https://dbanks.design/blog/multi-platform/
3. Best Practices for Cross-Platform UX Design - AorBorC Technologies, 1월 15, 2026에 액세스, https://www.aorborc.com/best-practices-for-cross-platform-ux-design/
4. React Native Explained: My Ultimate Guide for 2025 | by Luisa Becker, 1월 15, 2026에 액세스, https://medium.com/@luisabecker/react-native-explained-my-ultimate-guide-for-2025-8670e9bca6a9
5. Scaling React Native: Advanced Architecture Patterns for Enterprise ..., 1월 15, 2026에 액세스, https://medium.com/@ripenapps-technologies/scaling-react-native-advanced-architecture-patterns-for-enterprise-apps-1453649bc544
6. React Native For Web: The Guide To Cross-Platform Development, 1월 15, 2026에 액세스, https://www.logicrays.com/blog/react-native-for-web/
7. The 10 best React Native UI libraries of 2026 - LogRocket Blog, 1월 15, 2026에 액세스, https://blog.logrocket.com/best-react-native-ui-component-libraries/
8. Use Restyle to Create a Design System in React Native (2025), 1월 15, 2026에 액세스, https://javascript.plainenglish.io/use-restyle-to-create-a-design-system-in-react-native-2025-243d8e9a7b34
9. Building a cross-platform design system - Bolt, 1월 15, 2026에 액세스, https://bolt.eu/en/blog/building-a-cross-platform-design-system/
10. Streamlining Your Design System: A Guide to Tokens and Naming ..., 1월 15, 2026에 액세스, https://medium.com/@wicar/streamlining-your-design-system-a-guide-to-tokens-and-naming-conventions-3e4553aa8821
11. A Practical Guide to React Native UI Components - RapidNative, 1월 15, 2026에

액세스, https://www.rapidnative.com/blogs/react-native-ui-components
12. Understanding Components in React Native: A Comprehensive Guide, 1월 15, 2026에 액세스, https://medium.com/@anujgupta5686/understanding-components-in-react-native-a-comprehensive-guide-e2a55a840ca4
13. Handling Touches - React Native, 1월 15, 2026에 액세스, https://reactnative.dev/docs/handling-touches
14. 9 Responsive Design Best Practices for 2025 - NextNative, 1월 15, 2026에 액세스, https://nextnative.dev/blog/responsive-design-best-practices
15. Component Spec: the design system component delivery | by Matan ..., 1월 15, 2026에 액세스, https://uxdesign.cc/component-spec-the-design-system-component-delivery-5f88db6ccf7e
16. Pressable // React Native for Web, 1월 15, 2026에 액세스, https://necolas.github.io/react-native-web/docs/pressable/
17. Button States That Improve UX and Accessibility - Slider Revolution, 1월 15, 2026에 액세스, https://www.sliderrevolution.com/design/button-states/
18. Build Inputs Users Love: Custom Input Components in React Native ..., 1월 15, 2026에 액세스, https://medium.com/@didemsahin1789/build-inputs-users-love-custom-input-components-in-react-native-7677b83940d4
19. React form design patterns: Best practices for complex, form-heavy UIs, 1월 15, 2026에 액세스, https://www.dronahq.com/react-form-ui-tips/
20. Best Practices for Input or Form Validations in React Native for a ..., 1월 15, 2026에 액세스, https://www.reddit.com/r/reactnative/comments/17ltppg/best_practices_for_input_or_form_validations_in/
21. Cards – Material Design 3, 1월 15, 2026에 액세스, https://m3.material.io/components/cards/guidelines
22. Cards - Material Design, 1월 15, 2026에 액세스, https://m2.material.io/components/cards
23. Card design in the web card system - SAP, 1월 15, 2026에 액세스, https://www.sap.com/design-system/fiori-design-web/v1-136/ui-elements/web-card-system/usage
24. Design Principles and Best Practices in React Native App ..., 1월 15, 2026에 액세스, https://dev.to/anhpvbhsoft/design-principles-and-best-practices-in-react-native-app-development-4jak
25. Core Components and APIs - React Native, 1월 15, 2026에 액세스, https://reactnative.dev/docs/components-and-apis
26. Haptics design principles | Views - Android Developers, 1월 15, 2026에 액세스, https://developer.android.com/develop/ui/views/haptics/haptics-principles
27. What is Haptic Feedback? The Ultimate Guide in 2026 - Swovo, 1월 15, 2026에 액세스, https://swovo.com/blog/what-is-haptic-feedback/
28. Haptic UX — The Design Guide for Building Touch Experiences, 1월 15, 2026에 액세스,

https://medium.muz.li/haptic-ux-the-design-guide-for-building-touch-experiences-84639aa4a1b8
29. Responsive Design Breakpoints: 2025 Playbook - DEV Community, 1월 15, 2026에 액세스, https://dev.to/gerryleonugroho/responsive-design-breakpoints-2025-playbook-53ih
30. React Native Responsive Design with NativeWind - Medium, 1월 15, 2026에 액세스, https://medium.com/@CodeCraftMobile/react-native-responsive-design-with-nativewind-467afb75fc74
31. Responsive Web Design: Breakpoints, Layouts & Real Testing Guide, 1월 15, 2026에 액세스, https://dev.to/prateekshaweb/responsive-web-design-breakpoints-layouts-real-testing-guide-5ee5
32. React Native styling tutorial with examples - LogRocket Blog, 1월 15, 2026에 액세스, https://blog.logrocket.com/react-native-styling-tutorial-examples/
33. React Native Accessibility Best Practices: 2025 Guide for Mobile ..., 1월 15, 2026에 액세스, https://www.accessibilitychecker.org/blog/react-native-accessibility/
34. Dark Mode Design: A Practical Guide With Tips and Examples, 1월 15, 2026에 액세스, https://www.uxdesigninstitute.com/blog/dark-mode-design-practical-guide/
35. Dark mode in React: An in-depth guide - LogRocket Blog, 1월 15, 2026에 액세스, https://blog.logrocket.com/dark-mode-react-in-depth-guide/
36. How to Implement Dark Mode in Mobile Apps for Better UX, 1월 15, 2026에 액세스, https://nextbigtechnology.com/how-to-implement-dark-mode-in-mobile-apps-for-better-ux/
37. How Production Apps Handle Dark Mode in React Native - Medium, 1월 15, 2026에 액세스, https://medium.com/@silverskytechnology/how-production-apps-handle-dark-mode-in-react-native-ed38512eb790
38. AI-Markdown-Database/markdown-templates - GitHub, 1월 15, 2026에 액세스, https://github.com/AI-Markdown-Database/markdown-templates
39. Specification Templates for AI Code Generation: From First Draft to ..., 1월 15, 2026에 액세스, https://www.softwareseni.com/specification-templates-for-ai-code-generation-from-first-draft-to-production/
40. selfishprimate/plain: PLAIN (Product Language for AI ... - GitHub, 1월 15, 2026에 액세스, https://github.com/selfishprimate/plain
41. llms-txt: The /llms.txt file, 1월 15, 2026에 액세스, https://llmstxt.org/
42. 6 Tools for Documenting Your React Components Like a Pro, 1월 15, 2026에 액세스, https://dev.to/giteden/6-tools-for-documenting-your-react-components-like-a-pro-1gml
43. Architecture Best Practices for Cross Platform App Development, 1월 15, 2026에

액세스,
https://www.wildnetedge.com/blogs/architecture-best-practices-for-cross-platform-app-development