WDS 디자인 시스템 리서치 요약

React Native 기반 + Web 확장 가능한 모바일 앱 디자인 시스템 구성 & 컴포넌트 규칙

대상: Wellwe(웰위) 디자인 시스템 구축 / Figma 네이밍 및 Claude 기반 코드 생성 연계
생성일: 2026-01-15

WDS (Wellwe Design System) 리서치 요약

> React Native 기반 + Web 확장 가능한 모바일 앱 디자인 시스템 구성 & 컴포넌트 규칙
>
> (젠의 `ComponentType/Purpose/Variant/Size` 체계를 유지하면서 확장성/자동화(Claude)까지 고려)

1) RN + Web 확장형 디자인 시스템 권장 아키텍처

1.1 Tokens 우선주의: Primitive → Semantic → Component

- Primitive tokens: 원자값
- 예: `color.green.500`, `space.4`, `radius.3`
- Semantic tokens: 의미값(용도/맥락)
- 예: `color.bg.brand`, `color.text.primary`, `shadow.elevation.2`
- Component tokens: 컴포넌트 전용 토큰(상태/변형 포함)
- 예: `button.bg.primary`, `button.shadow.primary`

> 포인트
>
- 테마(라이트/다크), 브랜드 확장, 플랫폼별 튜닝을 위해 Primitive vs Semantic 분리는 필수
- “상태(state)”는 네이밍 슬롯에 넣지 말고 토큰/프로퍼티로 관리하는 게 확장성(자동화/코드 생성)에 유리

2) 컴포넌트 모델링 규칙 (Figma ↔ RN ↔ Web)

2.1 Primitives(기초 컴포넌트) 3종을 먼저 고정

RN+Web 확장에서는 아래 3개가 “디자인 시스템 엔진” 역할:

- Box: 레이아웃/스타일 단일 진입점
- 간격/정렬/배경/테두리/모서리/그림자
- Text: 타이포 단일 진입점
- font/size/lineHeight/weight/color 등의 토큰 적용
- Pressable/Button: 인터랙션 단일 진입점
- 상태/피드백/접근성/이벤트 계측

> 단일 진입점이 생기면
>
- Figma Auto Layout → RN Flexbox 매핑이 단순해지고
- 토큰 적용이 강제되어 UI 일관성이 올라감

3) Variant / Size / State 규칙 (젠의 네이밍 체계에 맞춘 확장)

3.1 Variant는 “역할/외형”, State는 “상호작용 상태”

- Variant(외형/역할): `primary | secondary | outline | ghost`
- State(상태): `default | pressed | disabled | loading`
- Web 확장 시: `hover | focus` 추가 고려

3.2 핵심 원칙: State는 “네이밍”이 아니라 “프로퍼티”

- 네이밍은 젠 방식 유지: `ComponentType/Purpose/Variant/Size`
- 상태는 Figma Component properties(Variant properties)로 관리
- 상태를 파일/컴포넌트 이름에 넣으면 조합 폭발(관리 난이도↑), 코드 설계와 불일치 가능성↑

권장 Properties 템플릿 (Button 예시)

- `variant`: `primary | secondary | outline | ghost`
- `size`: `xs | sm | md | lg | xl`
- `state`: `default | pressed | disabled | loading`
- `icon`: `none | leading | trailing | only`

4) RN + Web 확장 대응 포인트 (실무에서 꼭 터지는 지점)

4.1 Web 확장 고려 시 Hover/Focus 표준화는 필수

모바일에는 hover가 없지만 웹에는 존재하므로 WDS에서 규격화 필요:

- `stateLayer`(hover/focus/pressed overlay) 토큰화
- focus ring(접근성) 토큰화

결론

- “state 표현 규칙”이 없다면 웹 확장 시 UX/접근성 파편화가 발생

5) Figma 구조 권장안 (젠의 Hierarchy에 결합)

5.1 현재 Hierarchy 방향은 좋다

젠의 구조 예시:

- `Screen → TopBar/Section/Card/ListItem/TabBar`

5.2 한 가지 보강: Patterns 레이어를 별도로 둔다

젠이 금지한 비즈니스 상태 네이밍(Loading/Error/Empty)은 옳은 판단.
대신 상태는 컴포넌트명이 아니라 패턴 레벨(컴포넌션)에서 표준화:

- `Pattern/EmptyState`

- `Pattern/LoadingSkeleton`
- `Pattern/ErrorBanner`
- `Pattern/StickyCTA` (핵심 행동 유도 패턴)

> 컴포넌트는 “부품”, 패턴은 “조합된 사용자 경험”
>
> 상태는 패턴 레벨에서 관리하면 확장성과 일관성이 올라감

6) Auto Layout → RN Flexbox 매핑 (더 정확한 규칙)

6.1 itemSpacing → gap 매핑은 ‘환경 확인’이 필요

- Figma `itemSpacing`을 RN `gap`으로 매핑하는 방향은 좋지만
- RN의 `gap` 지원은 환경/버전에 따라 제약이 있을 수 있음

실전 대응

- `Stack`(VStack/HStack) 래퍼 컴포넌트 도입
- 내부에서 `gap`을 토큰으로 적용
- 필요시 폴백으로 `margin` 분배 방식 구현

> ZIP 제공 후 코드베이스를 확인해
>
> “gap 지원 전제인지/폴백 필요인지”를 기준으로 추천안을 확정하는 게 안전

7) 전환/행동 설계까지 포함한 WDS 운영 관점 (GoodUI / VWO / Growth.Design 활용)

디자인 시스템은 “예쁜 부품”이 아니라 “일관된 행동 유도 시스템”이기도 함.

7.1 Sticky CTA / Bottom Bar 패턴을 Patterns로 표준화

- 핵심 행동(완료/구매/참여)에 영향을 주는 패턴은
- 화면마다 제각각 구현되지 않게 `Pattern/StickyCTA`로 올리는 것이 좋음
- 결과적으로 실험/분석 지표(클릭, 완료, 전환)도 일관되게 계측 가능

7.2 Disabled → Enabled 전환 규칙을 WDS에서 고정

- 약관 체크/조건 충족 전 버튼 disabled → 충족 후 enabled 같은 패턴은
- 전환/완료율과 연결되는 핵심 UX 요소
- 따라서 WDS에서
- disabled 색/대비/피드백
- 입력 유효성(valid/invalid) 표현
- 로딩 상태 표현(스피너, 텍스트, 버튼 폭 유지 등)

를 규칙으로 고정하는 게 좋음

7.3 Growth.Design 심리학 기반 UI 체크리스트를 “리뷰 룰”로 포함

- 진행률/스텝퍼/온보딩 완주 유도
- 행동 가능성(affordance)과 신호(signifier)의 명확성
- 마찰 최소화(인지부하/결정 피로 감소) 관점으로 WDS 리뷰 기준을 잡기

8) 젠의 현재 네이밍 체계에 "딱" 추가하면 좋은 규칙 5개

1. State는 네이밍 금지, property로만 관리
2. Semantic tokens 우선 사용(primitive 직접 사용 최소화)
3. Component-specific tokens 허용(특히 shadow/elevation)
4. Patterns 레이어 신설(Empty/Loading/Error/StickyCTA 등)
5. 웹 확장 대비 hover/focus 토큰 강제

9) 다음 액션 (피그마 정리 단계 / ZIP 분석 단계)

9.1 ZIP 주기 전(피그마 정리 단계)

- Tokens: `primitive/semantic` 2단 분리 (가능하면 Figma Variables로)
- Button/Input/Card/ListItem/TopBar/TabBar:
- `state property` 포함해서 컴포넌트 정리
- Patterns 폴더 생성:
- `Pattern/StickyCTA`, `Pattern/EmptyState`, `Pattern/LoadingSkeleton`, `Pattern/ErrorBanner`

9.2 ZIP 받은 뒤(코드/문서 포함 분석 후)

- 토큰 파일 구조 제안:
- 예: `tokens/primitive.json`, `tokens/semantic.json` + TS 타입 생성 전략
- RN/Web 공용 컴포넌트 API 설계:
- `variant/size/state` props 표준화
- 피그마 네이밍 ↔ 코드 prop 자동 매핑 규칙:
- Claude Code 프롬프트/룰 파일로 고정
- 실험 가능하게:
- 이벤트 네이밍, 계측 포인트(클릭/완료/전환) 표준화 제안