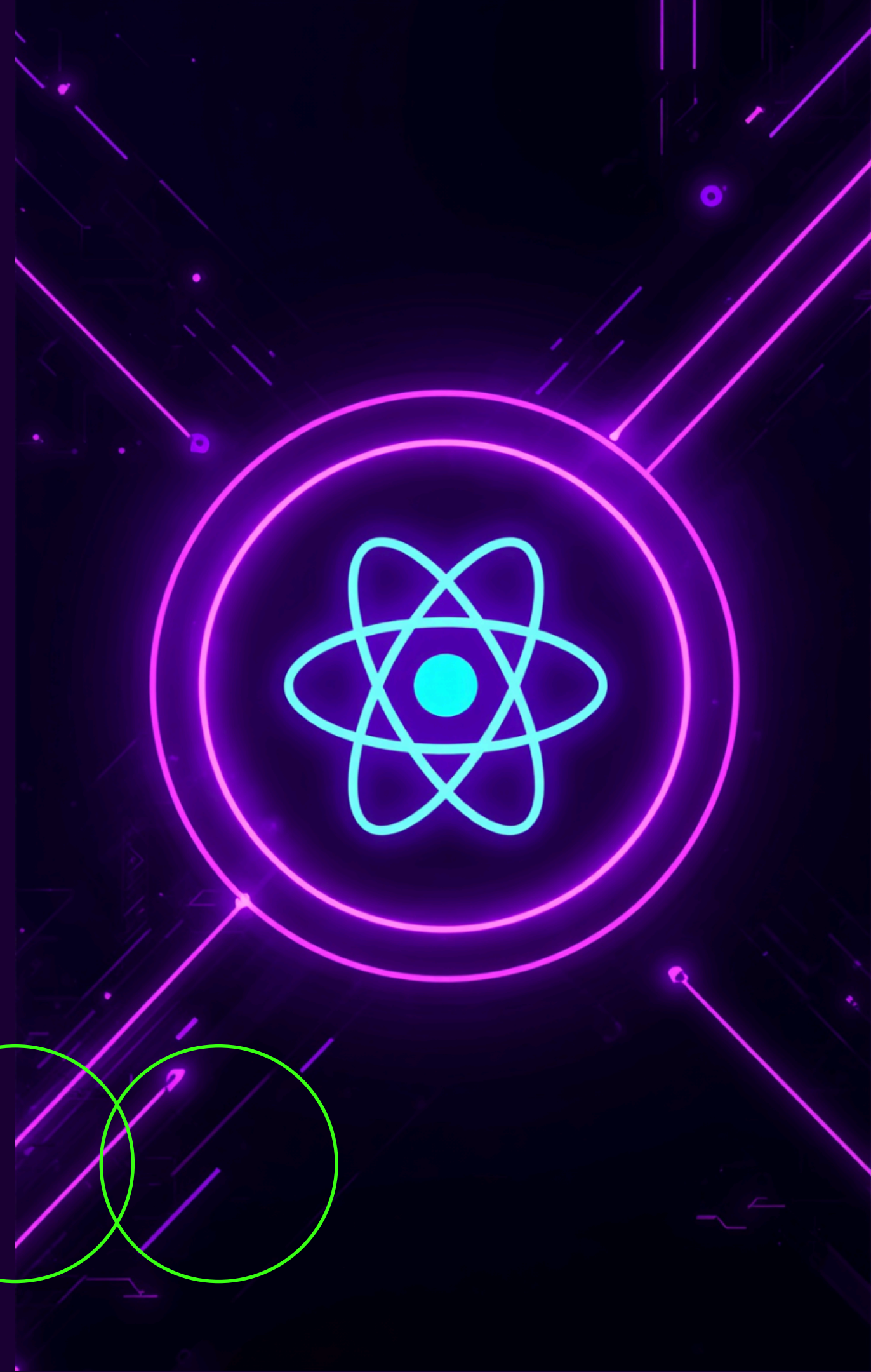


How I Learned React Native

Erik Hanchett, Chain React 2026 • Portland

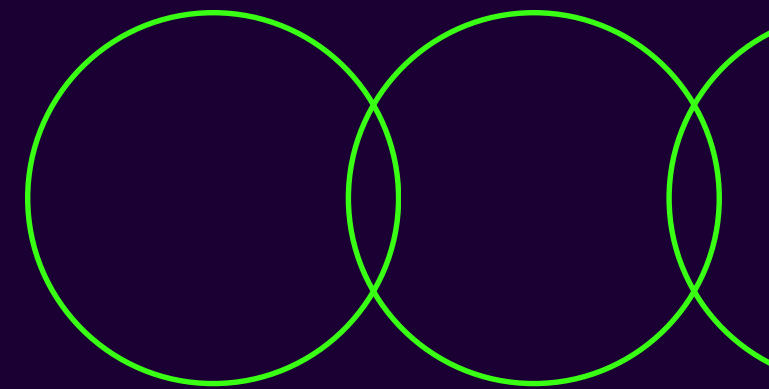


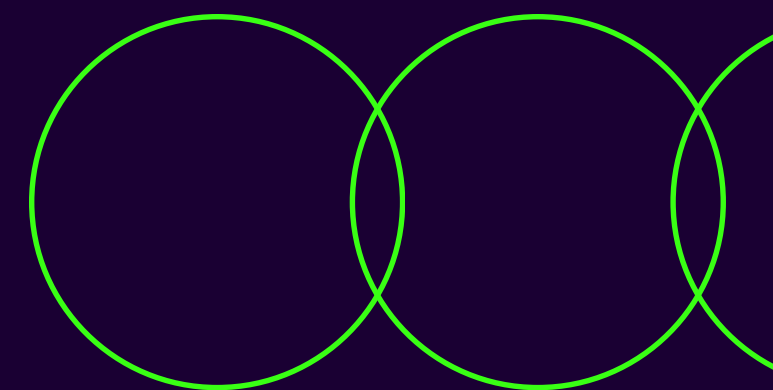
I build websites

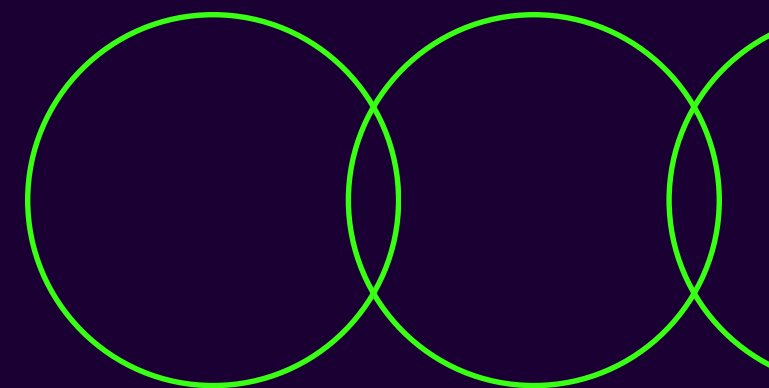
I want to build a fitness app

But where do I start?





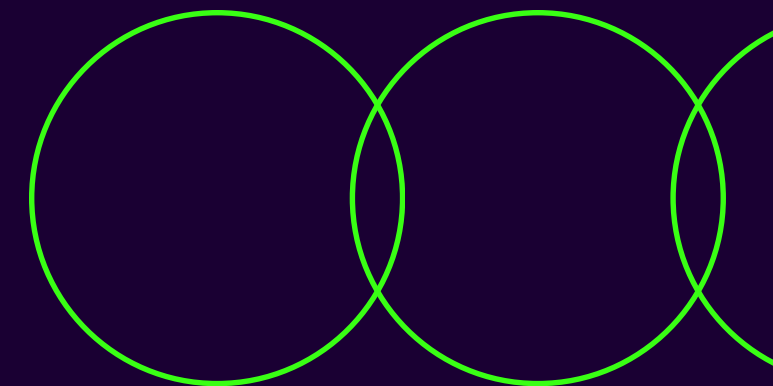






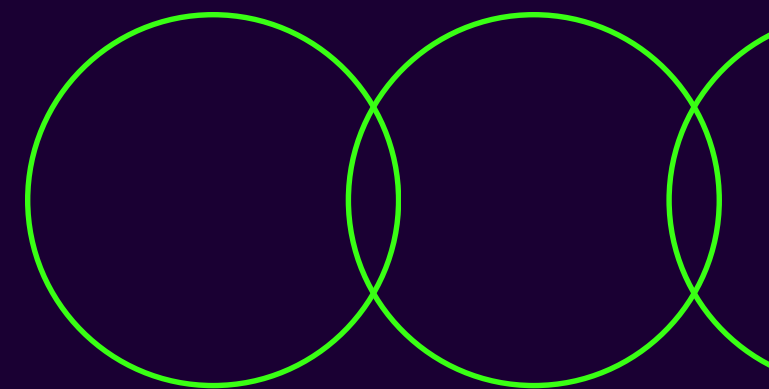
Introduction

Welcome to the very start of your React Native journey! If you're looking for getting started instructions, they've moved to [their own section](#). Continue reading for an introduction to the documentation, Native Components, React, and more!



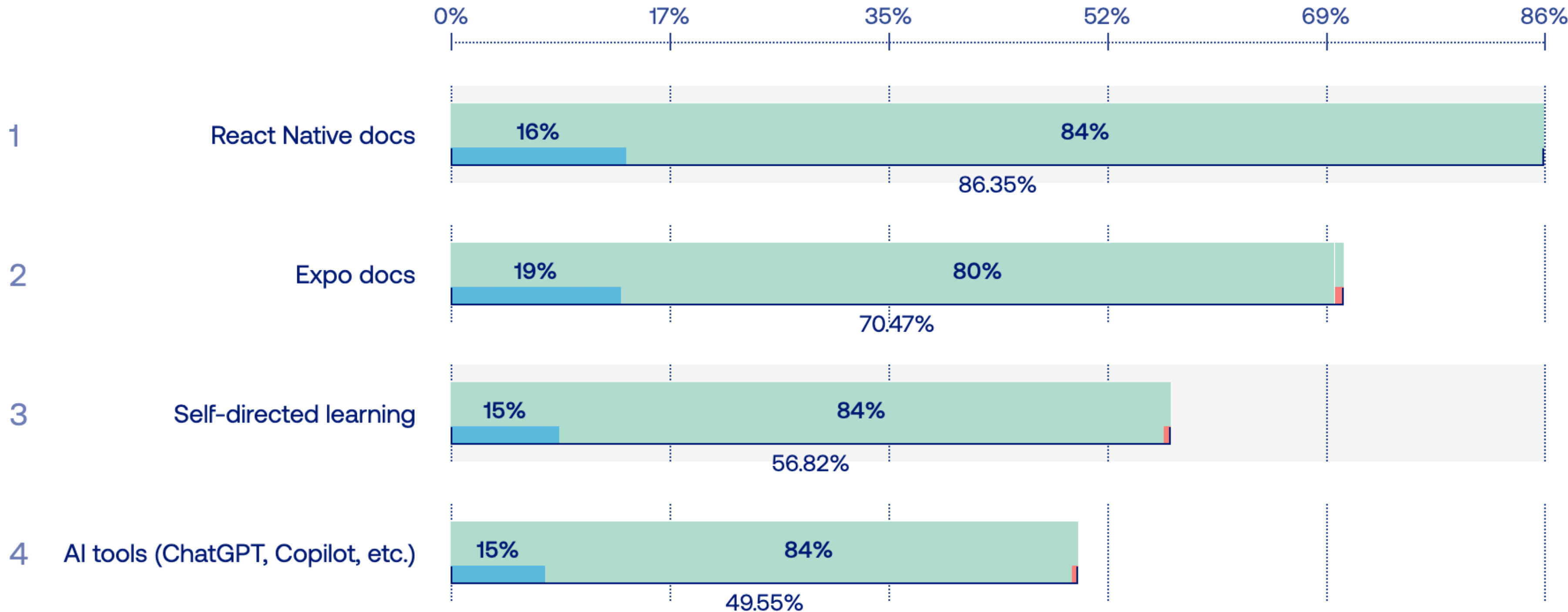


AI



Initial learning methods

② Which resource types have you used to initially learn React Native?



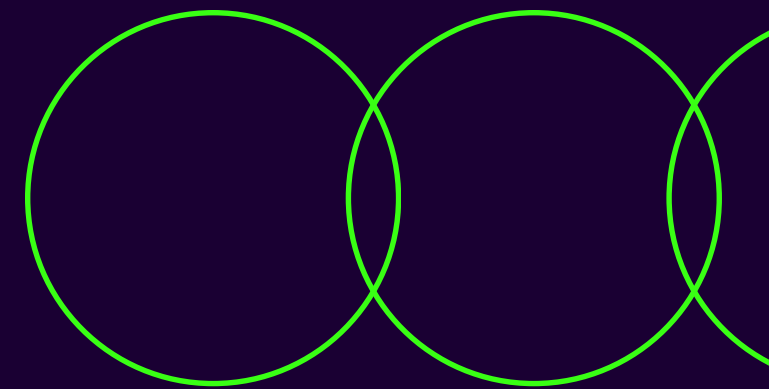
Erik Hanchett

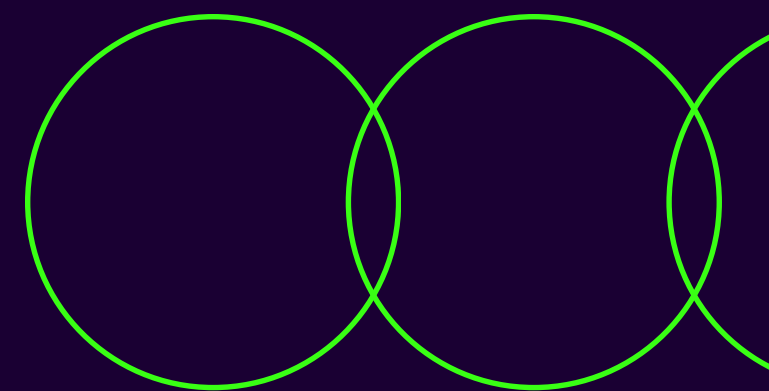
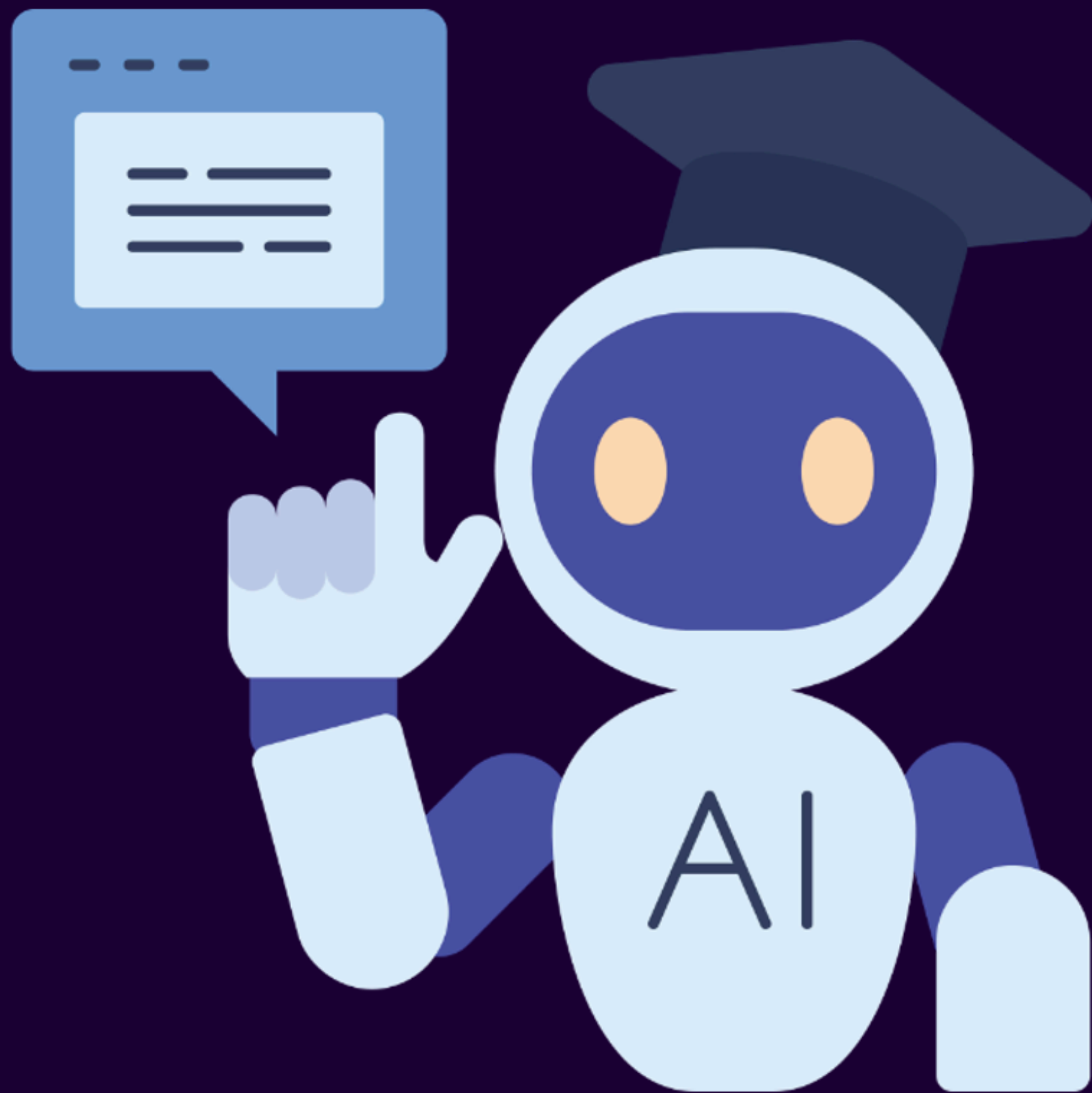
www.programwitherik.com

Senior Developer Advocate at AWS

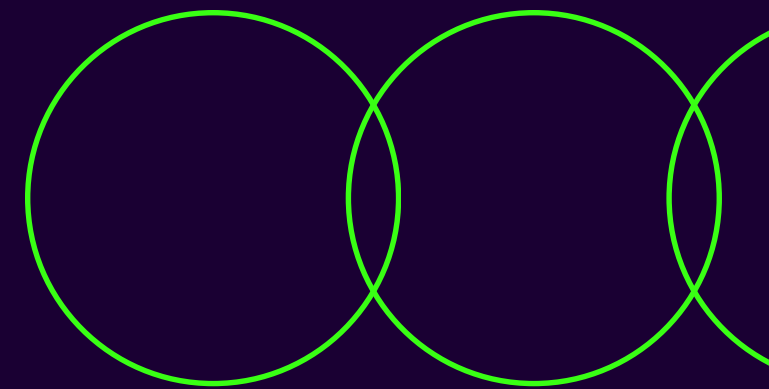


How much do you
really need to know?

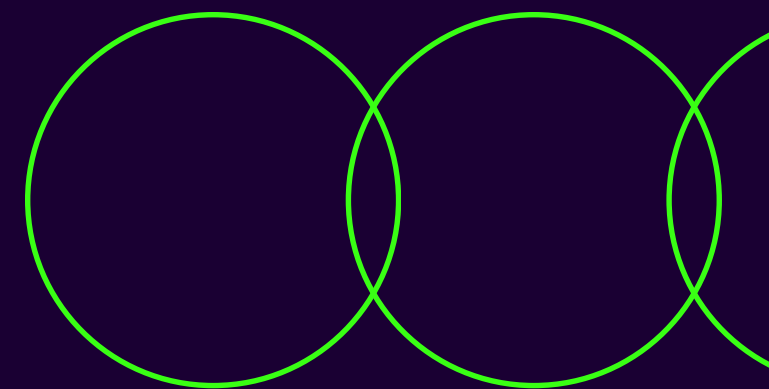




“We are the sytem designers, we are the product engineers, thinking about which primitives are available for our agents to use to solve our users problems” - Kent C. Dodds

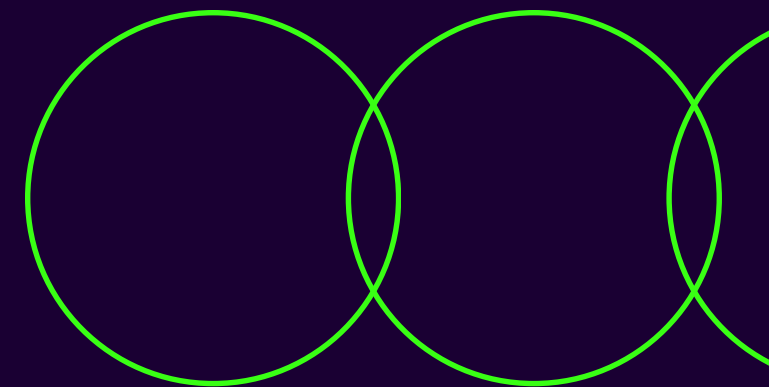


So what does AI
recommend today?



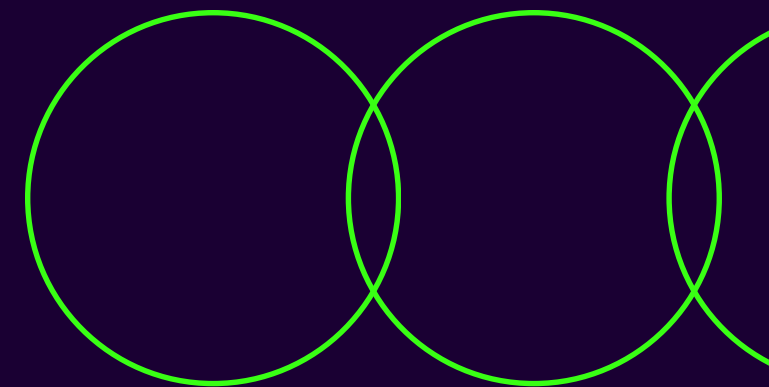
So what does AI recommend today?

Let's try some evals!

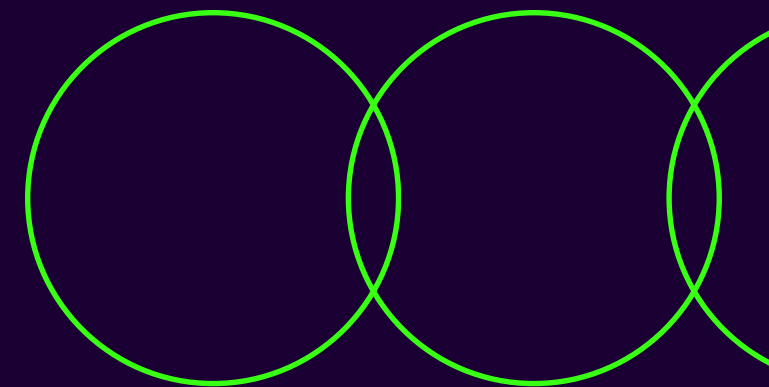


Prereqs

Gemini / Claude



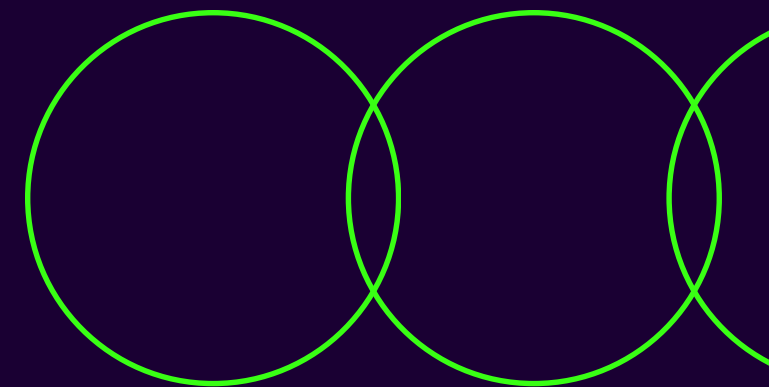
“Build a mobile application for
iOS.”



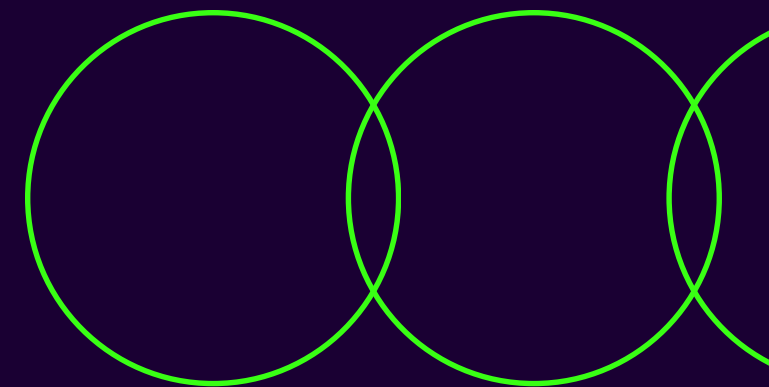
Build a mobile application for iOS



Swift
Programming
Language

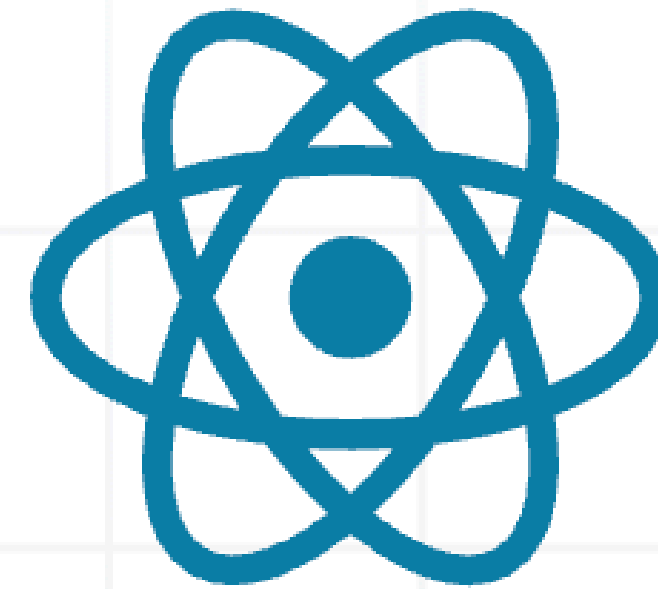


“Build me a cross platform iOS/Android application”

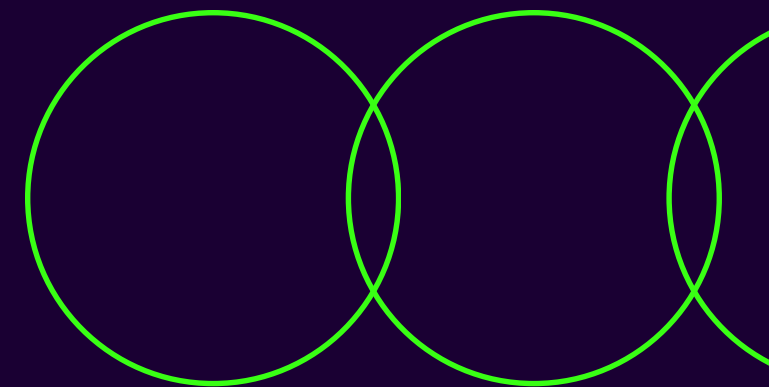




Flutter



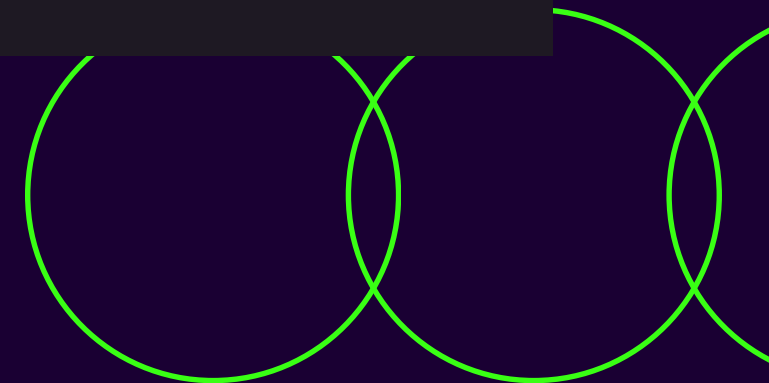
React Native



Here are the top cross-platform and native frameworks worth considering:

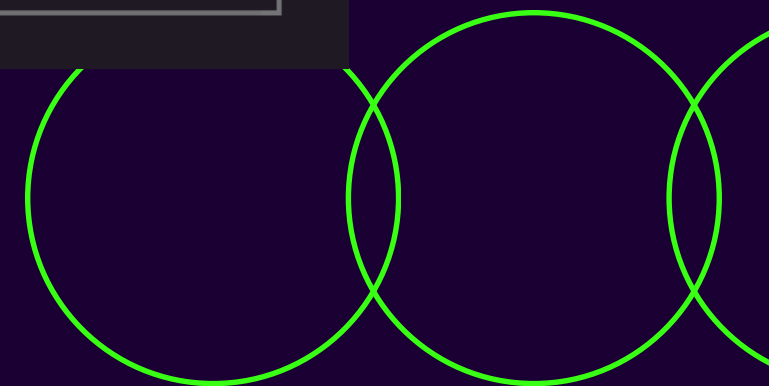
Cross-Platform (single codebase for both iOS & Android)

Framework	Language	Best for
Flutter	Dart	Beautiful custom UIs, fast iteration, strong performance
React Native	JavaScript/TypeScript	Teams with web/React experience, large ecosystem
Kotlin Multiplatform (KMP)	Kotlin	Sharing business logic while keeping native UIs
.NET MAUI	C#	Teams already in the .NET ecosystem



If you're sticking with JavaScript/TypeScript, here are your main options:

Framework	Notes
React Native	Industry standard. Huge ecosystem, Expo makes setup trivial, strong community.
Expo	Built on React Native but handles builds, OTA updates, and native modules for you. Start here unless you have a reason not to.
NativeScript	Direct access to native APIs from JS/TS. Supports Angular, Vue, Svelte, or vanilla JS. Smaller community.
Ionic + Capacitor	Web app wrapped in a native shell. Use your existing HTML/CSS/JS skills (React, Angular, Vue). Good for content-heavy apps, weaker for heavy animations.



AI as My Teacher

Crafting a Personalized Learning Path



**"Build me a curriculum to
learn React Native.
Assume I already have
basic knowledge of React,
HTML/CSS/JavaScript."**

AI as My Teacher

Crafting a Personalized Learning Path

**MODEL SELECTION
MATTERS!**



AI as My Teacher

Crafting a Personalized Learning Path

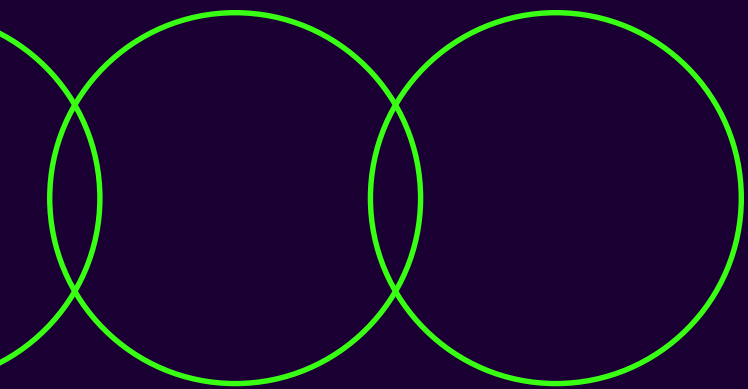
**MODEL SELECTION
MATTERS!**

Sonnet 4 vs Opus 5



All the wrong things

2023 era advice

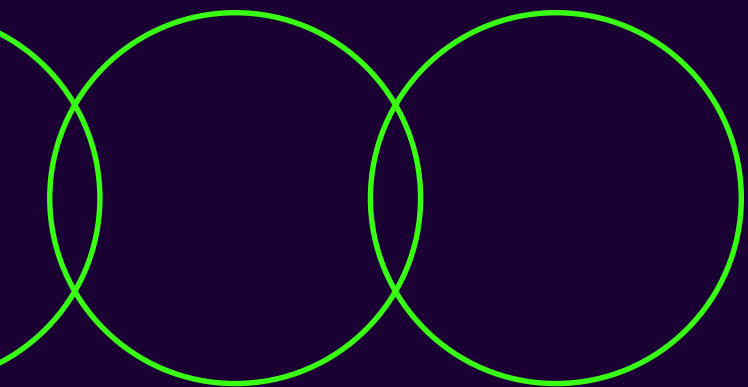


All the wrong things

2023 era advice

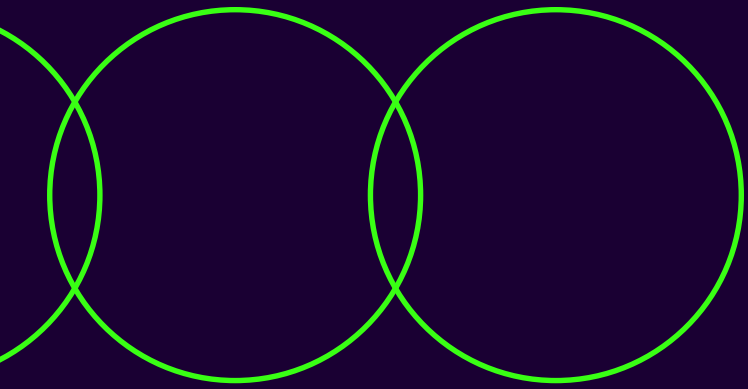
Node 18

✗ Node 22 is the minimum since RN 0.84
(Feb 2026)



All the wrong things

2023 era advice



Node 18

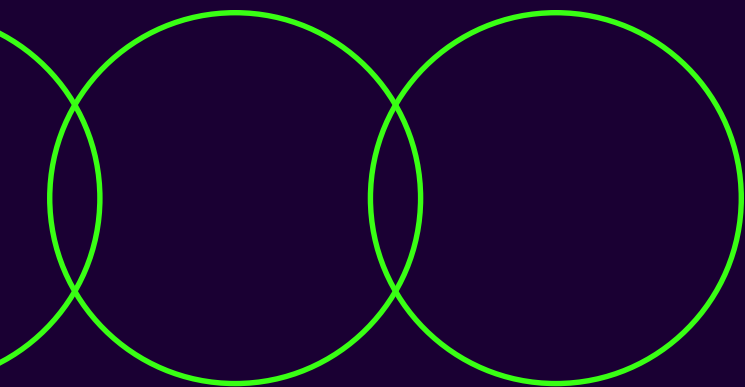
✗ Node 22 is the minimum since RN 0.84 (Feb 2026)

react-native init

✗ Deprecated in 0.75, sunset 2024-12-31, replaced by create-expo-app

All the wrong things

2023 era advice



Node 18

✗ Node 22 is the minimum since RN 0.84 (Feb 2026)

react-native init

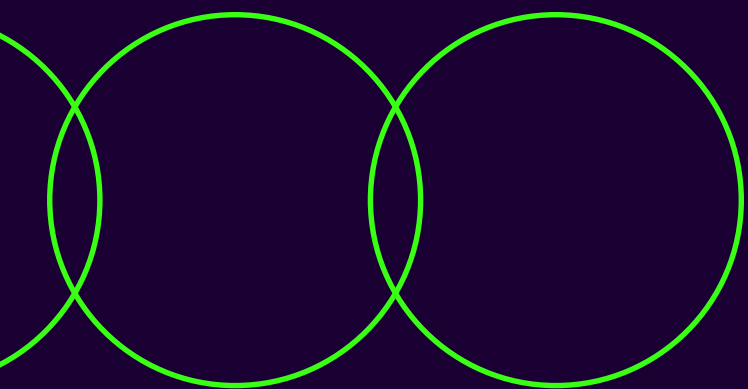
✗ Deprecated in 0.75, sunset 2024-12-31, replaced by create-expo-app

Old deprecated testing libraries

✗ @testing-library/jest-native
Legacy Architecture (sonnet-4, unmentioned)

Better but not perfect

2026 advice

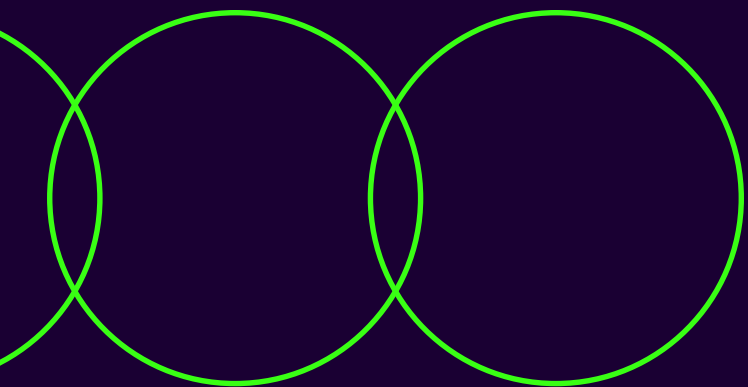


Better but not perfect

2026 advice

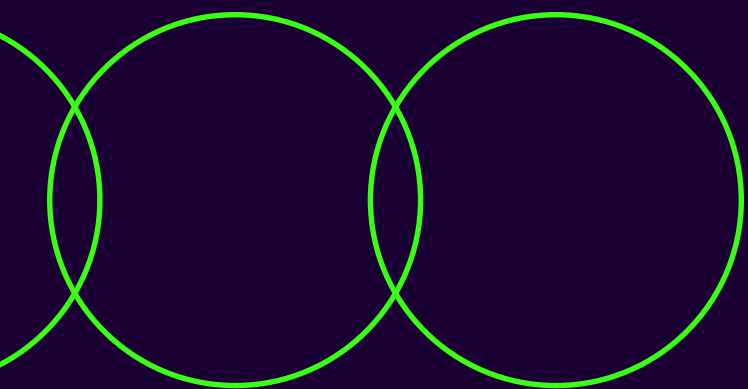
Data loaders

Teaches Expo Router data loaders (alpha, web-only, useless for native) but asks for an offline indicator without mentioning NetInfo.



Better but not perfect

2026 advice



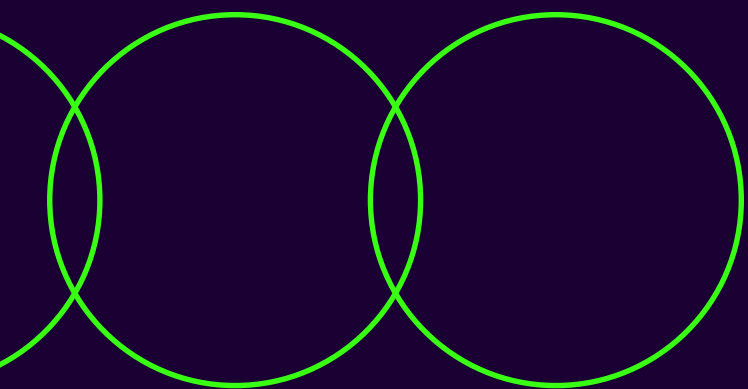
Data loaders

Teaches Expo Router data loaders (alpha, web-only, useless for native) but asks for an offline indicator without mentioning NetInfo.

Expo Go

Not much information on how to go beyond Expo Go for development builds

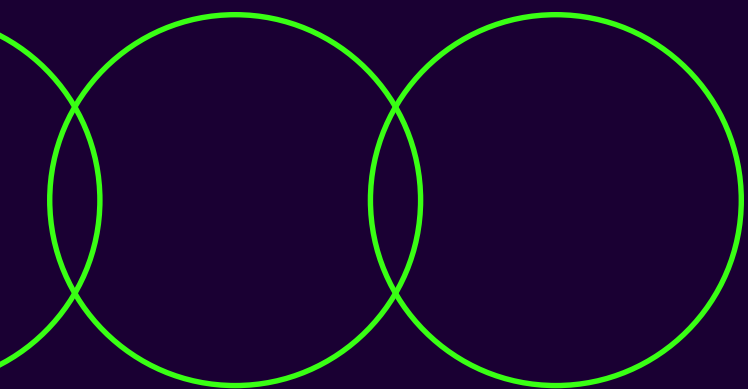
What was
correct in
both?



What was correct in both?

“There is no DOM” is lesson one

Both plans correctly diagnose that this reader's problem isn't React, it's the assumption that the browser is still there.



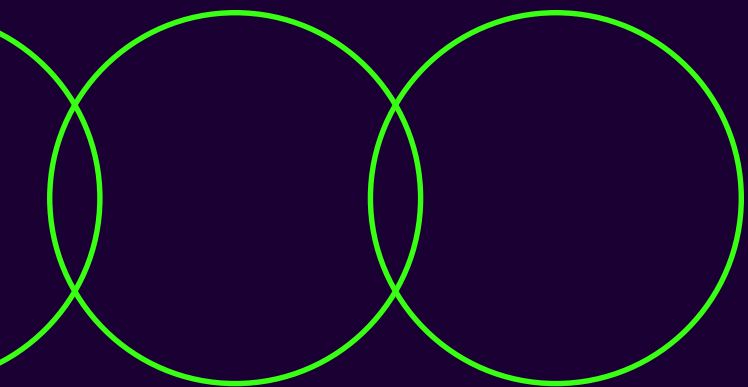
What was correct in both?

“There is no DOM” is lesson one

Both plans correctly diagnose that this reader's problem isn't React, it's the assumption that the browser is still there.

Loading, error, and empty states

These are basics it goes over right away



What was correct in both?

Raw API and patterns

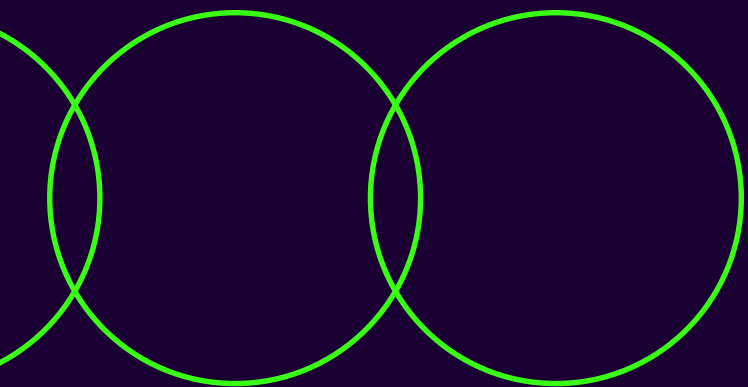
Both teach StyleSheet and flexbox before any styling layer, and neither makes Redux a prerequisite.

“There is no DOM” is lesson one

Both plans correctly diagnose that this reader's problem isn't React, it's the assumption that the browser is still there.

Loading, error, and empty states

These are basics it goes over right away



How to correct this older models?



Adversarial reviews...



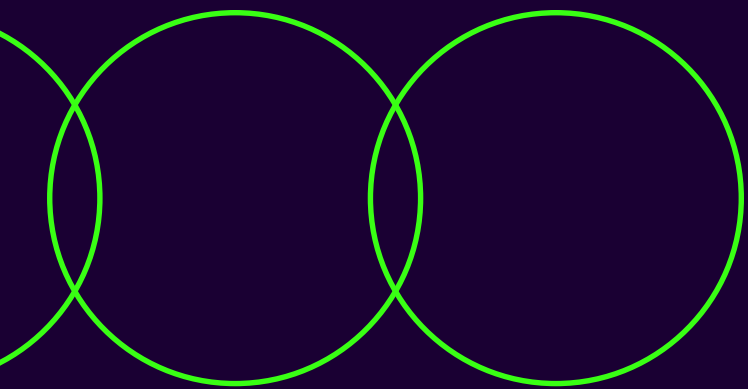
Use Skills, built in search, or MCP servers



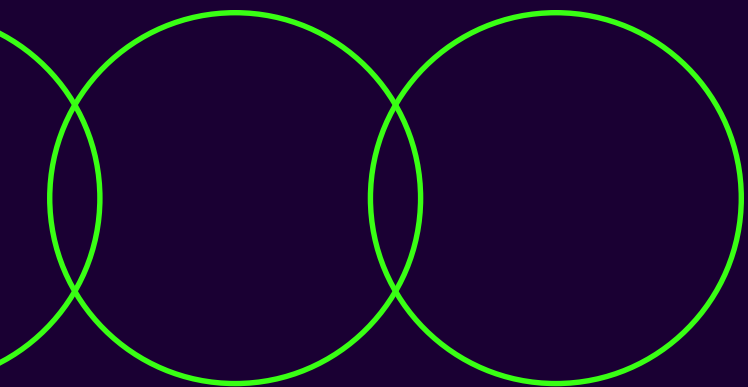
"Build a curriculum to help me learn React Native. Assume I have some knowledge of React, HTML/CSS, and JavaScript. Use search tools to verify each step. Use Context7 and Expo MCP to make sure the APIs still exist."

Did a little better...

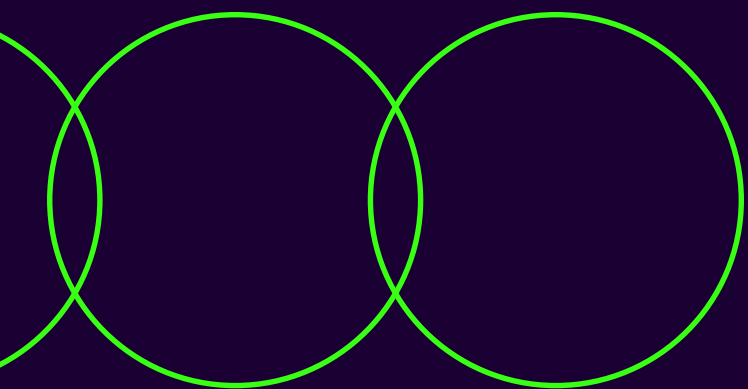
The API was still not perfect...



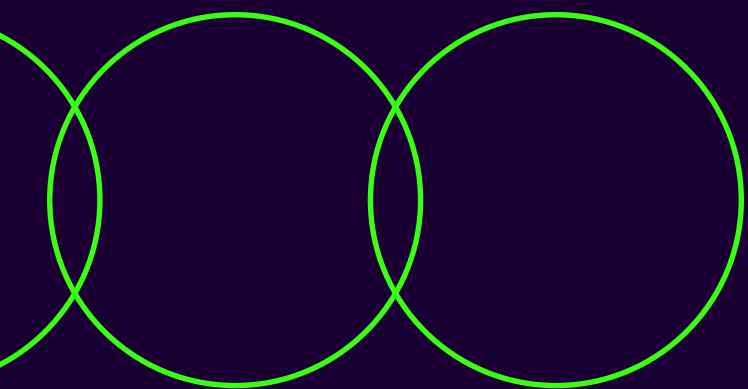
**Conclusion: Use SOA
models**



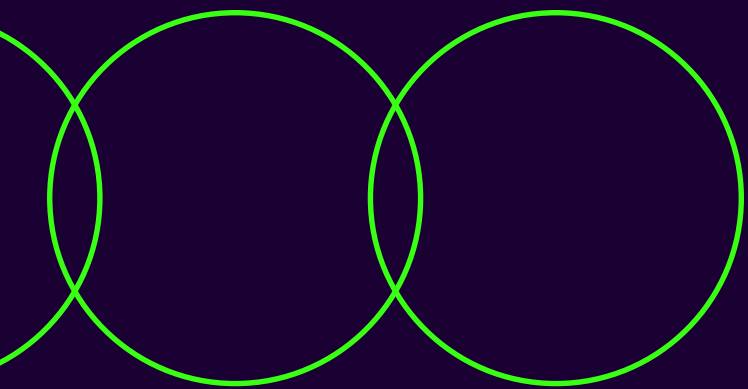
What does Expo give out of the box?



- AGENTS.md/ Claude.md
- MCP
- Skills

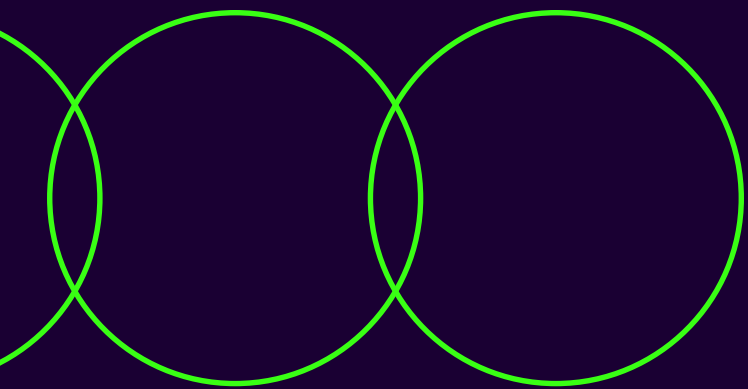


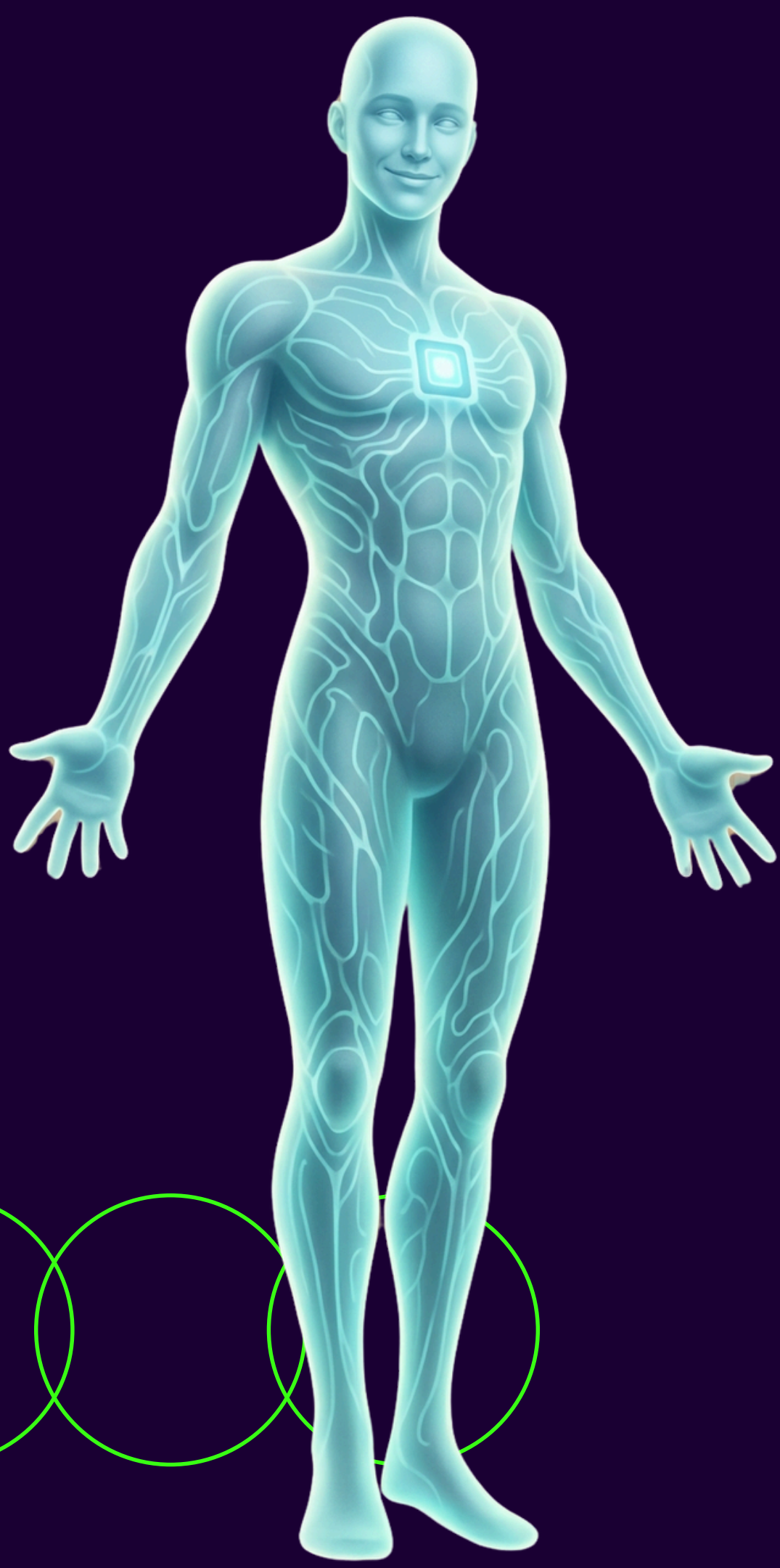
**Best Way of Learning
is Building!**



I do, we do, you do

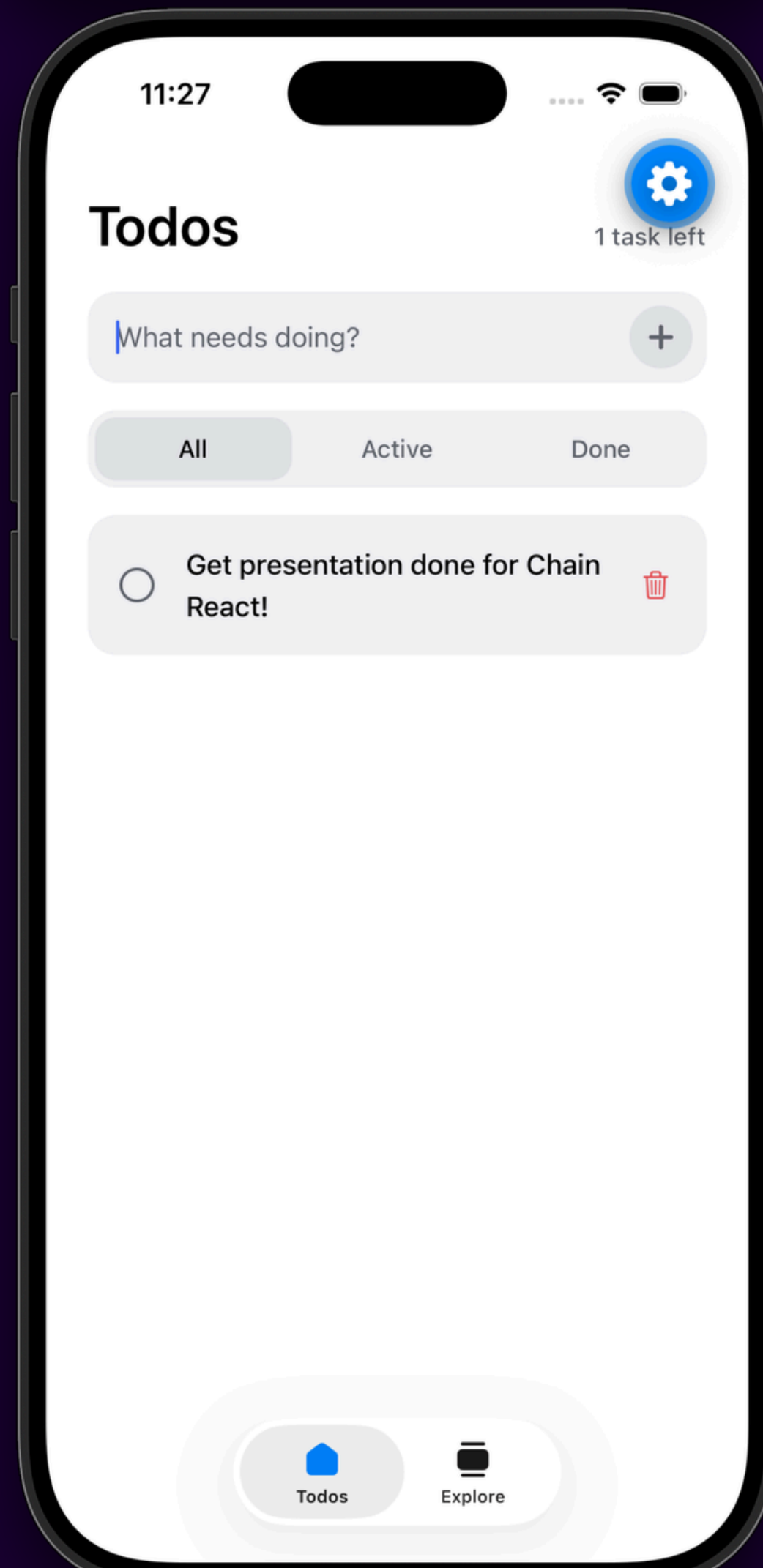
Gradual release of responsibility



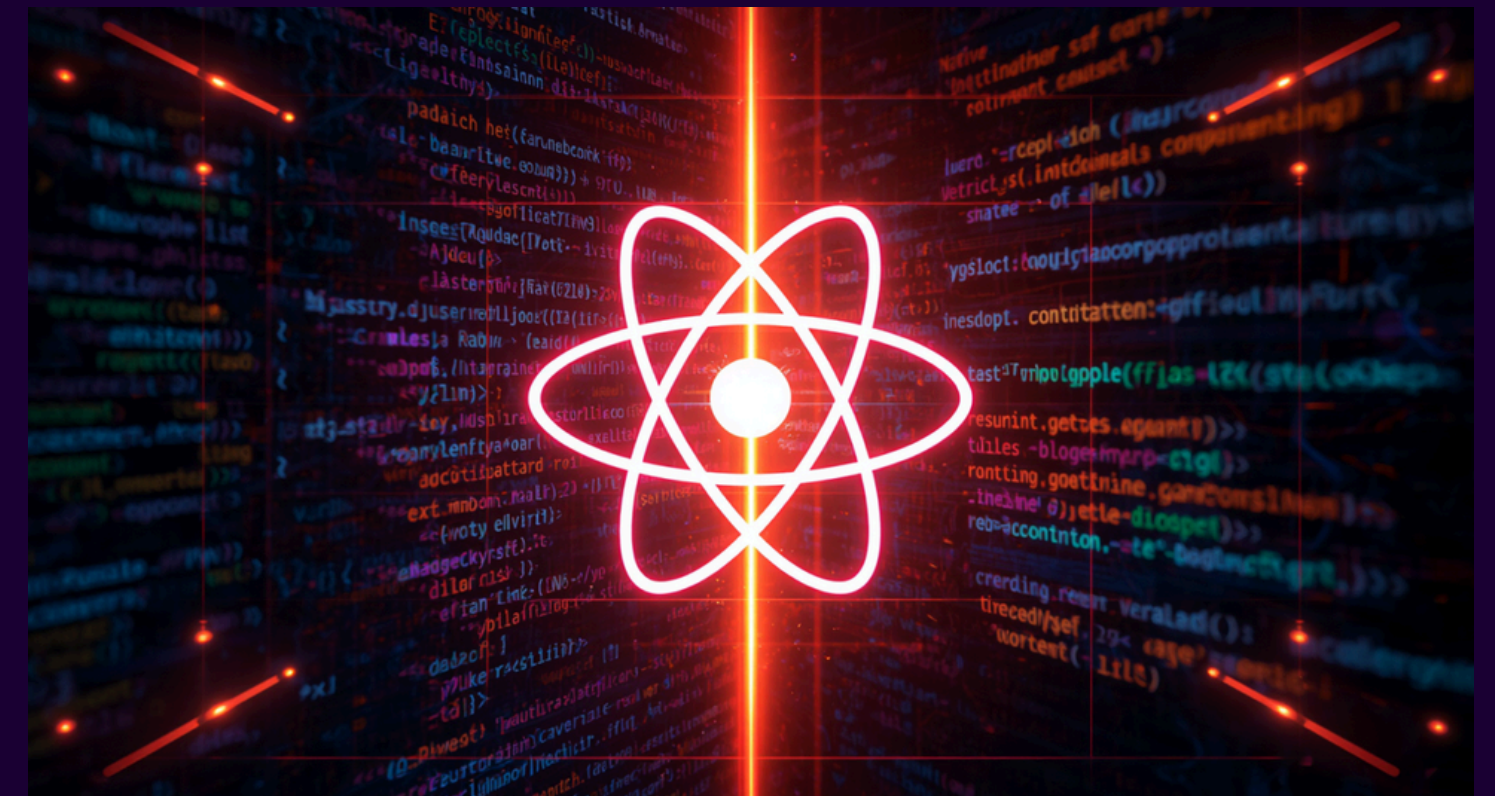
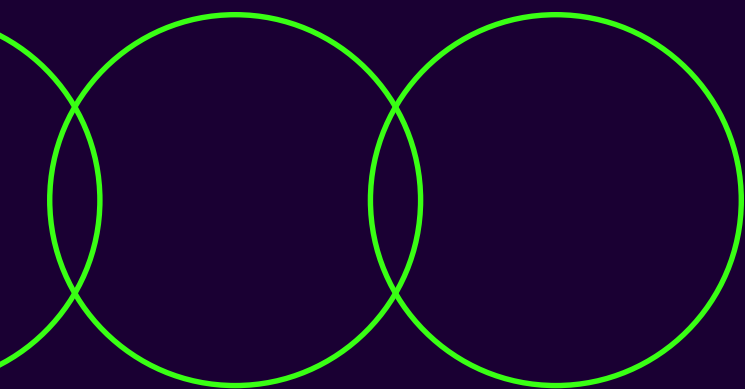


AI will do it all...

Vibe Code (AI Do)

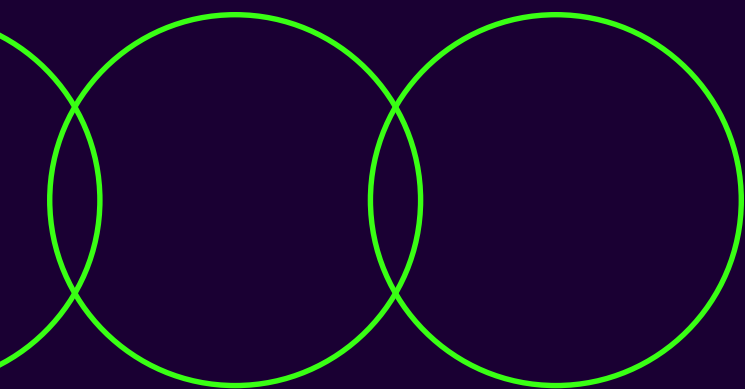
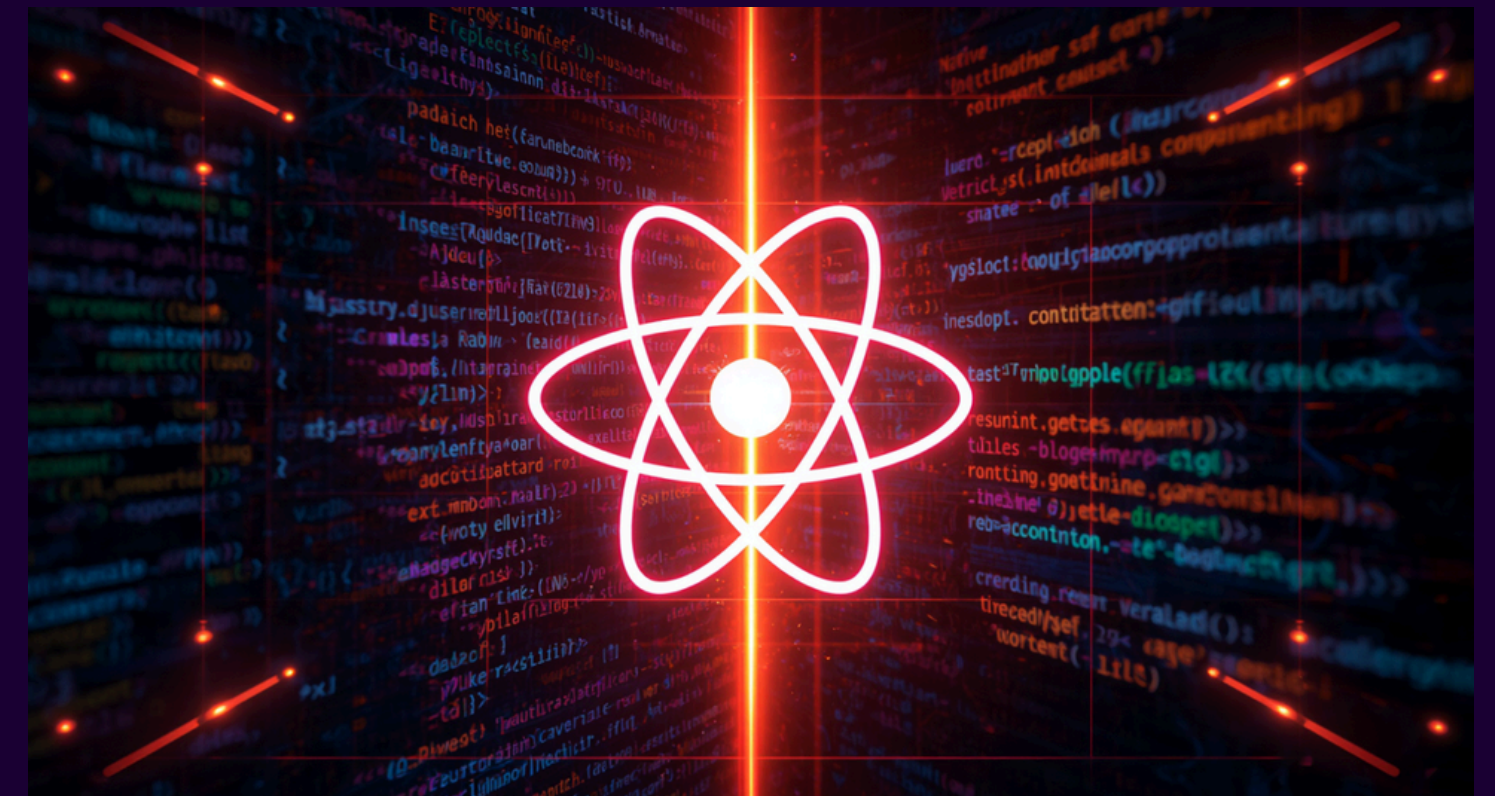


Spec-Driven Development (we do)



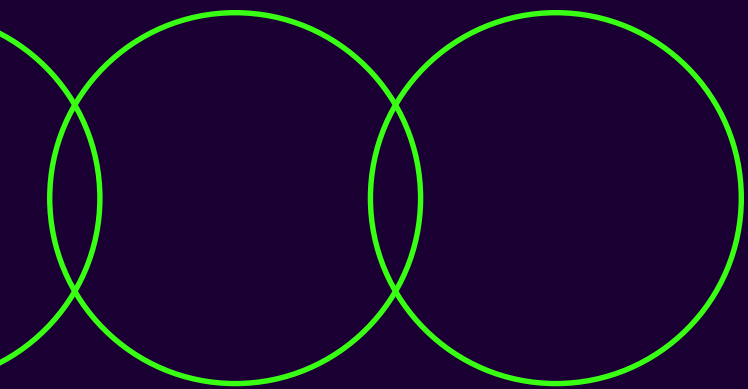
Spec-Driven Development

Spec-Driven Development (SDD) is writing structured specifications first so AI can automatically build and test the code.



“Create a spec that helps build a Retro workout planner app...”

(Made sure my app had the correct MCP and skills files, and used latest models)



Design — Retro 75 Hard Tracker

Overview

A local-first Expo app. All state lives in SQLite on device; the UI is a thin render of a pure domain layer. The single hard problem is **day resolution**: given a run, its day entries, the clock, and the configured rollover hour, decide whether the user is on track, owes a redemption, or has failed. That logic is isolated in pure functions so it can be unit tested and reasoned about, because a bug there costs a user 75 days of work.

Everything else — the arcade skin, the chiptune, the animations — is presentation over that core.

Technology decisions

Concern	Choice	Why
Routing	<code>expo-router</code> 57 with typed routes	Already installed and configured in <code>app.json</code>
Tab bar	<code>expo-router/ui</code> headless tabs (<code>Tabs</code> , <code>TabList</code> , <code>TabTrigger</code> , <code>TabSlot</code>)	A native tab bar cannot carry the CRT aesthetic. Headless tabs give a fully custom neon bar while keeping router semantics. Deliberate deviation from the usual <code>NativeTabs</code> preference.
Styling	NativeWind v5 preview + Tailwind v4 (CSS-first)	Required by the user. SDK 57 pairs with NativeWind v5, which needs no babel config.
Persistence	<code>expo-sqlite</code> ~57.0.1 with <code>SQLiteProvider</code> + <code>PRAGMA user_version</code> migrations	First-party, offline, relational shape fits runs → days → tasks
Photos	<code>expo-image-picker</code> ~57.0.6 + <code>expo-file-system</code> ~57.0.1 (<code>File/Directory/Paths</code>)	Copy picked image into app documents, store relative path
Audio	<code>expo-audio</code> ~57.0.3 (<code>useAudioPlayer</code> , <code>setAudioModeAsync</code>)	Current audio API for SDK 57 (<code>expo-av</code> is retired)
Notifications	<code>expo-notifications</code> ~57.0.7, local only	Scheduled <code>DATE</code> triggers for nudge and final warning
Animation	<code>react-native-reanimated</code> 4.5.0 + <code>react-native-worklets</code>	Already installed

Requirements

R1 — Start, resume, and restart a run

User story: As a user, I want to start a 75 HARD run and have the app remember where I am, so I can pick up each day without re-entering anything.

Acceptance criteria

1. WHEN the app launches AND no run exists THEN the system SHALL present a start screen showing the five tasks, the active ruleset, and a "BEGIN" action.
2. WHEN the user starts a run THEN the system SHALL create a run with `startDate` = current program date, `status = active`, `dayNumber` 1 mapped to that date, and 75 total day slots.
3. WHEN the app launches AND an active run exists THEN the system SHALL open directly to today's checklist for the current program day.
4. WHEN the current program day is day 75 AND all five tasks are complete THEN the system SHALL mark the run `completed` and show a victory screen.
5. WHEN a run has status `failed` or `completed` THEN the system SHALL offer "START OVER AT DAY 1" which creates a new run and SHALL retain the prior run in history.
6. WHILE a run is active THE system SHALL never allow more than one active run to exist.

R2 — Daily task check-off

User story: As a user, I want to check off each of the five tasks as I finish them, so today's progress is always visible.

Acceptance criteria

1. WHEN viewing today's checklist THEN the system SHALL display six checkable items: diet, outdoor workout, second workout, gallon of water, 10 pages read, progress photo — where the two workouts satisfy task 2 of the program.
2. WHEN the user taps a task THEN the system SHALL toggle its completion, persist it immediately, and give haptic + audio feedback.
3. WHEN the user completes all six items for the current program day THEN the system SHALL mark the day `complete`, play a completion fanfare, and show the incremented day count.
4. WHEN a task supports detail THEN the system SHALL let the user optionally record it: workout duration in minutes and activity label, water

Implementation Plan — Retro 75 Hard Tracker

Sequenced so **Phase 1 ships a running app you can hold and use**, then later phases layer on the sensory and convenience features. Build order inside each phase runs bottom-up: styling foundation, pure domain, persistence, then UI, so day-resolution bugs surface in tests rather than through screens.

Definition of done for every phase: `npx tsc --noEmit` clean, `npm run lint` clean, `npm test` green.

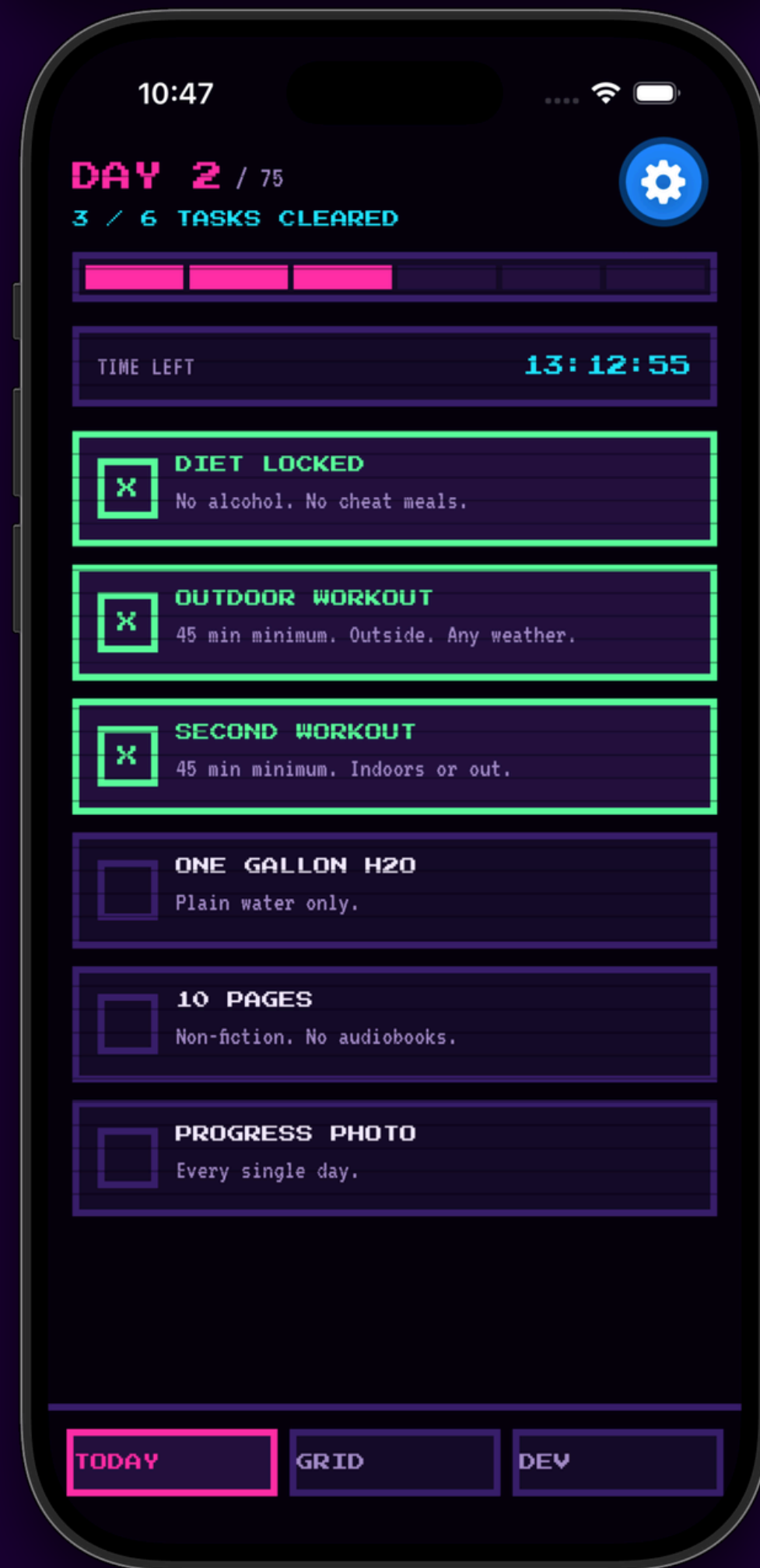
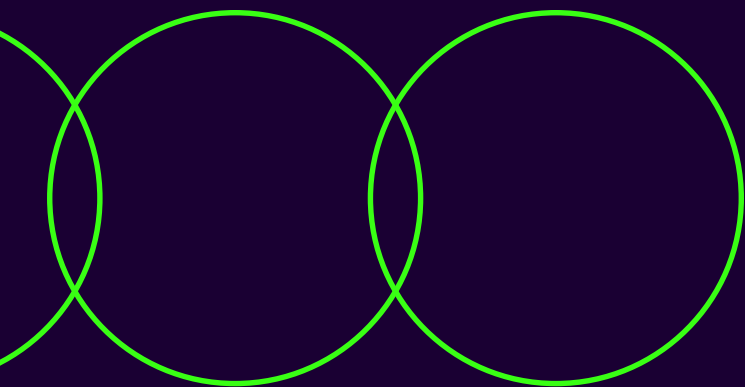
Phase 1 — MVP complete

Goal: start a run, check off all six tasks each day, see a neon 75-day grid of what you completed, get warned and redeem or fail when you miss a day. Persists across restarts. Looks like an arcade cabinet.

Deliberately deferred out of the MVP, with the shape left in place for later:

Deferred	MVP behavior
Music and sound effects	Silent
Progress photo capture	<code>PROGRESS PHOTO</code> is a plain checkbox, no camera or library
Local notifications	In-app countdown and danger banner only
Animated sequences	Static state changes
Day detail, month view, progress timeline, settings screen	Not routed yet; grace mode and midnight rollover are the fixed defaults

- [x] 1. Install and wire the NativeWind v5 / Tailwind v4 foundation
 - Install `tailwindcss@^4`, `nativewind`, `react-native-css`, `@tailwindcss/postcss`, `tailwind-merge`, `clsx` with `npx expo install`
 - Add a `lightningcss` version pin (`overrides` for npm, `resolutions` for yarn)
 - Create `metro.config.js` with `withNativewind(config, { inlineVariables: false, globalClassNamePolyfill: false })` and `postcss.config.mjs`
 - Confirm no `babel.config.js` exists for NativeWind; do not create one



I do patterns

Turn AI Off

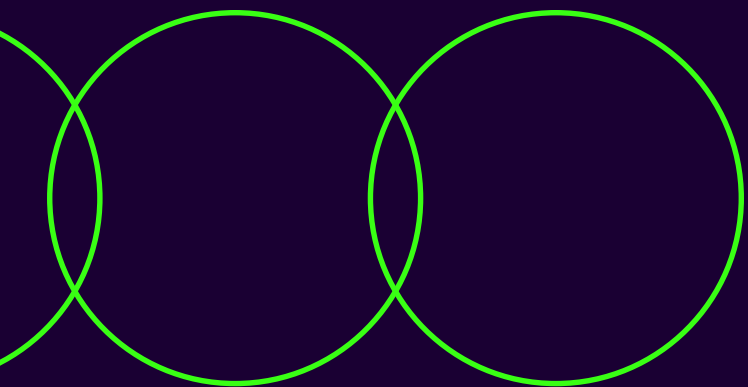
Only use AI when you get stuck

AI Quiz You

Let AI Quiz and grade you as practice.

Learn in public

Build in public



Conclusion...

Start out with a goal!

Use the latest models

**Create a plan - Learn at least the architecture
and primitives**

You do, we do, I do



Amazon Developer

Build for Amazon devices
with React Native



Sign up for updates



Thank you!

www.kiro.dev

