



Launching on the App Store

A practical guide to taking your React Native app from finished build to live on the App Store.

WHO THIS GUIDE IS FOR

Beginners — if you've never submitted an app before, this walks through the accounts, tools, and Apple-specific requirements you'll hit along the way.

Technical readers — if you've shipped before, this is a reference checklist for the parts that are easy to forget or easy to get rejected for.

Note: subscription/monetization setup is covered in a separate guide — this one only covers where subscriptions fit in the launch sequence.

SECTION	COVERS
01 · Account setup	EAS CLI, Apple Developer account, App Store Connect, privacy & website requirements
02–03 · Build & submit	The <code>eas build</code> / <code>eas submit</code> commands, export compliance, TestFlight
04 · Listing	Exact screenshot dimensions, description, ASO, Privacy Policy & Terms
05 · Gotchas	A pre-submission checklist, then the avoidable rejections it maps to
06 · Post-launch	Reviews, update submissions, phased rollout



01 · ACCOUNT & APP STORE CONNECT SETUP

Start the **clock** early

Before you can submit anything, you need an active Apple Developer account and an app record in App Store Connect — and one of these has a lead time you can't rush.

DO THIS NOW · TERMINAL · TAKES FIVE MINUTES

```
# 1. Install the EAS CLI and sign in (free Expo account)
npm install -g eas-cli
eas login

# 2. Generate eas.json in your project – build profiles,
#    and the submit config you'll need on page 4
eas build:configure

# 3. Print the bundle ID you're about to register with Apple.
#    It must match App Store Connect exactly, forever.
grep -A2 '"ios"' app.json
```

Do this before the Apple paperwork below. It costs nothing, and it tells you your bundle identifier — which you need in order to create the app record, and which you cannot change afterwards.

Individual vs. Organization

Individual is faster to set up and shows your personal name as the seller. Organization shows your company name but requires a D-U-N-S number, which can take days to weeks to obtain or verify. Start this early — it's often the longest lead time in the whole launch.

Annual fee

The Apple Developer Program is **\$99/year** (local currency and tax vary by region), billed to the account rather than per app — unlimited apps on one membership. Note the renewal date: a lapsed membership pulls your live apps from the store.

Creating the app record

- a** **Bundle ID** must match your Expo/React Native config exactly, and is effectively permanent once a build is submitted against it — plan the naming convention (e.g. `com.yourcompany.yourapp`) before you create it.
- b** **App name** must be unique across the entire App Store, not just your account. Check availability early.
- c** **SKU** is an internal identifier only you see — low stakes, but worth being consistent if you'll have multiple apps.
- d** **App icon** has strict size/format rules — e.g. a 1024x1024 App Store icon with no transparency or baked-in rounded corners. Icon rejections are common and avoidable.



The most **underestimated** step

App Store Connect will not let you submit without a working support page — and this section trips up more first-time submitters than any single technical requirement.

WHAT YOUR APP MUST LINK TO

Your app itself must link to **Privacy Policy, Terms of Use, and Support** (page or email) — typically a small "Legal" or "About" section in Settings with tappable rows. Both documents must be hosted as live pages on your website, not bundled as static text. **Test every link before submitting** — a broken or missing privacy link is a common, entirely avoidable rejection, and reviewers do check.

- 1 App Privacy "Nutrition Label."** A questionnaire asking exactly what data your app collects and why — must match what your code actually does, not what you intend to do. Add an analytics SDK later? Come back and update it.
- 2 Privacy Policy hosting.** App Store Connect requires a live URL, not an uploaded file. Page 7 names a free generator that will also host the page for you.
- 3 Support URL.** A working support page, separate from any general marketing site URL field.
- 4 Support email.** A monitored inbox both users and Apple reviewers can reach — put it on your support page and consider surfacing it in-app too.



02-03 · SUBSCRIPTIONS & SUBMITTING A BUILD

Build, then **submit**

Two separate EAS commands, a required compliance answer, and — if your app has paid features — a subscriptions step that sits between app creation and your first submission.

WHERE SUBSCRIPTIONS FIT IN THIS SEQUENCE

If your app sells anything, all of the subscription setup happens **after** the app record exists and **before** your first submission — products can't be created without an app record, and a build with an untested payroll shouldn't be submitted. *Subscriptions with RevenueCat* covers that whole step in detail; come back here when your purchase flow works.

COPY-PASTE · TERMINAL · THE TWO COMMANDS

```
# Build a production iOS binary – EAS handles signing
# and provisioning on its own macOS machines
eas build --platform ios --profile production
```

```
# Upload the finished binary to App Store Connect
eas submit --platform ios --latest
```

```
# Along the way
eas build:list           # status and URLs for past builds
eas credentials         # inspect/repair signing certificates
```

Export compliance is asked once per submission. If your app only uses HTTPS and standard OS-level encryption — true for almost every app — set `"usesNonExemptEncryption": false` under `ios.config` in `app.json` and EAS answers it for you every time.

Version vs. build number: your app version (`1.0.0` , user-facing) and build number (`1` , `2` , `3` , internal) are different fields. Every submission — even TestFlight-only — needs a build number higher than the last one uploaded for that version. Set `"autoIncrement": true` in `eas.json` and stop thinking about it; bump the version yourself only for user-facing releases.



03 · TESTING BUILDS ON TESTFLIGHT

Widen the audience **carefully**

TestFlight gives you a safety net before full App Review — but internal and external testers play by different rules, and crash reporting should already be wired in before you widen the pool.

Internal testers

Up to 100, must be part of your App Store Connect team. Get builds immediately, with no review.

External testers

Can number in the thousands, but require a lightweight Beta App Review before the build goes out. Budget a few days for the first external review — usually faster than full App Review, but not instant.

- a Use the built-in feedback loop.** TestFlight has its own feedback mechanism — use it deliberately rather than relying on testers to message you separately.
- b Wire in crash reporting first.** Sentry, Firebase Crashlytics, or similar should be in place before you widen your TestFlight audience, so early crashes surface with stack traces instead of vague tester reports.



04 · LISTING & MARKETING ASSETS

What shoppers **see**

Screenshots, description, and ASO all have to agree with each other — and with what's actually in the app — on which features are paid.

Screenshot dimensions, exactly

App Store Connect validates pixel dimensions strictly — an image off by a single pixel is rejected at upload. Supply the largest size in each family and Apple scales it down for every smaller device, so in practice you need **one iPhone set**, plus **one iPad set** only if your app supports iPad.

DEVICE CLASS	PORTRAIT (PX)	ALSO ACCEPTED	REQUIRED?
6.9" iPhone	1320 × 2868	1290 × 2796 · 1260 × 2736	Yes — iPhone apps
13" iPad	2064 × 2752	2048 × 2732	Yes — iPad apps
Everything smaller	Scaled automatically from the sizes above		Optional

1–10 screenshots per device class and per localization. JPEG or PNG, **no alpha channel or transparency** — exporting a PNG with transparency is a common upload failure. Landscape is the same numbers reversed. Verify against Apple's screenshot specifications page before a launch: these change as new devices ship.

Paid-feature disclosure

If your app has paid features, the screenshots must clearly indicate which shown features are paid — a required disclosure point, and one a human reviewer specifically looks for.

App preview video

Optional, up to ~30 seconds. Can lift conversion, but a bad preview is worse than none — a reasonable v1 skips it and adds one later from real usage footage.



04 · LISTING & MARKETING ASSETS

Words, and the paperwork

The description sells the app; the legal pages are what let you submit it at all. Both have to agree with what the app actually does.

Description & ASO

The first three lines carry the whole pitch — everything after them is hidden behind a "more" tap. State the paid vs. free split in the description itself, not only in screenshots. For a first launch, free keyword tools are entirely sufficient; an enterprise ASO suite is not a v1 problem.

Privacy Policy & Terms of Use

TermsFeed's free generator produces both, has a mobile-app-specific flow, and will host the result at a public URL — which matters, because App Store Connect requires a live link, not an uploaded file. **app-privacy-policy-generator** (open source, on GitHub) is a solid alternative if you'd rather host the pages yourself.

GENERATED IS FINE — UNREAD IS NOT

Whatever you generate, read it and make it true. The policy has to match both what your code actually collects and the answers you gave in the App Privacy questionnaire. A generated policy claiming you collect nothing, sitting next to a questionnaire declaring analytics, is exactly the kind of inconsistency reviewers do catch — and the fix costs a submission cycle.

DON'T MISADVERTISE PAID FEATURES

Any feature that isn't free must not be described or implied as included when it actually sits behind a paywall. This is both an App Review risk and a user-trust/refund-request risk — and it's checked in three places at once: your description, your screenshots, and the app itself.



05 · PRE-SUBMISSION CHECKLIST

Tick this while you read

The next three pages are the rejection gotchas. Work through them with this page open — every line here maps to something a reviewer checks, and nearly every rejection is one of these left undone.

- ☐ **Accounts:** Apple Developer membership active, app record created, bundle ID matches `app.json`
- ☐ **Icon:** 1024×1024, no transparency, no baked-in rounded corners
- ☐ **Legal:** Privacy Policy and Terms live at public URLs, Support URL and monitored support email set
- ☐ **In-app links** to Privacy Policy, Terms and Support all open correctly — tested on the submitted build, not the dev build
- ☐ **App Privacy questionnaire** matches what the code actually collects, including any analytics SDK
- ☐ **Sign in with Apple** present if you offer any other third-party sign-in, styled per the HIG
- ☐ **Account deletion** reachable from inside the app, not just by emailing support
- ☐ **Permission strings** custom and specific for every prompt; ATT wording present if you track across apps
- ☐ **Paid/free split** consistent across description, screenshots and in-app presentation
- ☐ **Purchases** route through StoreKit/IAP, tested end to end
- ☐ **Reviewer access:** demo credentials or a bypass note in the App Review notes field
- ☐ **Age rating** answered honestly against actual content, including any user-generated content
- ☐ **Screenshots** at 1320×2868 (and 2064×2752 if iPad), no transparency
- ☐ **Build number** higher than the last upload for this version; export compliance answered
- ☐ **Crash reporting** wired in and verified before external testers see it

Every unticked box above is a page in Section 05. If they're all ticked and you still get rejected, page 11 covers what to do about it.



What your app must

Three requirements about the app's own behaviour. All three are checked by a human opening your app, and all three are cheap to satisfy in advance and expensive to retrofit after a rejection.

5.1 · Minimum functionality

5.1 Guideline 4.2. Apps that feel like a thin website wrapper get rejected. If your app is largely a webview, expect scrutiny — there needs to be a genuine native experience: real navigation, offline behaviour, notifications, something the browser couldn't already do.

5.2 · Sign in with Apple

5.2 Required conditionally, not universally. You need it specifically when you offer sign-in through any *other* third-party service (Google, Facebook, and so on) — not merely because your app has login. Email-and-password only? Not required.

5.2b The button itself is reviewed. Apple's own styling, clearly visible, and not buried beneath the other options. A correctly implemented flow behind a de-emphasised button still gets flagged.

5.3 · Account deletion

5.3 Guideline 5.1.1(v). If users can create an account, they must be able to delete it *from inside the app* — a support email address does not satisfy this, and it is one of the most consistently enforced items on this list.

5.3b What "delete" can mean technically. Nulling out identifying fields while keeping the row for referential integrity is acceptable, as long as the user-facing effect genuinely removes their personal data and their access.



Permissions, money, **access**

What you ask users for, how you take their money, and whether the reviewer can actually get into your app.

5.4 · Permissions & privacy strings

5.4 Custom, specific text for every prompt. Notifications, microphone, camera, location — each needs a string saying what *your* app uses it for. Generic or missing strings are a common and entirely avoidable rejection.

5.4b App Tracking Transparency. Required for any cross-app tracking or ad targeting, with its own mandated wording and its own prompt. If you added an analytics or ads SDK, check whether this applies to you.

5.6 · In-app purchase vs. external payment

5.6 Digital goods go through StoreKit. Anything consumed inside the app — subscriptions, unlocks, credits — must use In-App Purchase; you generally cannot link out to your own checkout for digital content. Strictly enforced. Don't try to route around it on a first launch.

5.7 · Reviewer access

5.7 Give them a way in. If any part of your app sits behind login or a paywall, put working demo credentials — or a bypass note — in the App Review notes field. "Unable to test" is a rejection you caused, and it costs a full review cycle to fix.

THE PATTERN BEHIND ALL OF THESE

A reviewer has your build, your description, and about ten minutes. Every item on these pages is a way of making sure that in those ten minutes they can get in, see what you claimed, and find nothing that contradicts it.



Metadata, and getting **rejected**

The last two consistency checks — then what to actually do when a rejection lands, which it may well do on your first submission.

5.5 · Paid feature disclosure

5.5 Three places, one answer. The app description, the screenshots, and the in-app presentation must agree on what is paid. A feature marked free in one and paid in another is a red flag to reviewers and a refund request from users.

5.8 · Age rating accuracy

5.8 Answer honestly, from actual content. User-generated content, messaging between users, and mature themes all push the rating up. An inaccurate rating discovered later can trigger *removal*, not merely a rejection — this is the one item here with consequences past the review queue.

5.9 · Support & legal links

5.9 Test them on the submitted build. Confirm the Support URL and email in App Store Connect, and tap through every in-app link to Privacy Policy, Terms and Support on the actual binary you uploaded — not in dev, where a localhost URL will happily resolve.

5.10 · HANDLING A REJECTION

Read the guideline number they cite. Apple almost always points at the exact clause, and the fix is usually narrower than the rejection message feels.

Check whether it's metadata-only. Description, screenshots, age rating and review notes can often be fixed and resubmitted *without a new binary* — which turns a week into an afternoon.

Push back when you're right. Reply in the Resolution Center, or request a call with App Review for ambiguous cases. A first rejection is routine, not a verdict on the app.



06 · POST-LAUNCH

Shipping is not **done**

Launch is the start of a maintenance cadence, not the finish line — reviews, updates, and rollout strategy all keep running after day one.

6.1**Monitor reviews**

Set a routine for checking and responding to App Store reviews, especially bug reports — a public "fixed in the next update" reply is genuinely persuasive to future readers.

**6.2****Submit updates**

Update review is typically faster than the initial review, but the same guidelines apply in full — a previously-approved app gets a fresh check on every new build.

**6.3****Phased rollout**

App Store Connect can roll an update out to a percentage of users at a time, so a bad update only affects a fraction of your userbase before you can pause it.



SUMMARY

Launch checklist at a glance

The whole sequence, condensed to one reference table.

STAGE	KEY ITEMS
Account setup	Apple Developer account, App Store Connect app record, privacy disclosure
Website & support	Support page, support email, hosted Privacy Policy + Terms of Use, in-app links to all three
Subscriptions	RevenueCat + StoreKit setup (see dedicated guide), sandbox testing
Build & submit	EAS build/submit, export compliance, TestFlight
Listing	Screenshots, description, ASO, copyright/ToU
Review gotchas	Sign in with Apple, account deletion, permission strings, paid feature disclosure, support/legal links
Post-launch	Review monitoring, update submissions, phased rollout

BEFORE YOU HIT SUBMIT

Almost every rejection in Section 05 traces back to a mismatch: what your app description says vs. what the app does, what your privacy questionnaire says vs. what your code collects, what your in-app links promise vs. what they actually open. Test the submitted build itself, not the dev version, and re-run the checklist on page 8 one more time before you submit.

Where to go next

GUIDE	PICKS UP WITH
Subscriptions with RevenueCat	The monetization step this guide only places in the sequence — paywall code and StoreKit testing
The AI Coding Stack	The build stack underneath all of this, if you arrived here from someone else's project
Idea to App Store	The combined guide — including the analytics stage that starts the day you launch