Velorum AI

# The First App Playbook

Everything you need to know to build and sell your first app, in one document, by Taku. You need a coding agent, a plan written down before any code, and the right few services around it.

**What I used myself.** I built mine with Claude and Supabase. I used Next.js instead of Swift, so the same app works on both Android and iPhone. I did not use RevenueCat, Sentry or PostHog; they are here because I recommend them in the reel. This playbook takes the reel's path: iPhone first, in Swift.

by Taku. Version 1.0, November 2026.

# How to use this document

- **Part 1, the ten steps,** follows the reel: what to do and why, with the prices and rules linked to the pages they come from.
- **Part 2, the build order,** is the order to do it in, with four prompts, shown here for reading. Copy each with its copy button at velorum-ai.com/resources/app and paste it into your coding agent in your project folder and answer its questions.
- Prices and rules were checked on 26 November 2026. They change, so check the link before you pay.

# The ten steps

| | |
|---|---|
| 01 Your tech stack, a coding agent | 02 Write a PRD before any code |
| 03 Supabase for your backend | 04 RevenueCat, Sentry and PostHog |
| 05 Build it in Swift, for iOS first | 06 Get an Apple developer account |
| 07 Test with real users on TestFlight | 08 Join the Small Business Program |
| 09 Before you submit, run the review skill | 10 Submit, and your first app is live |

# What it costs to get to the App Store

| What | Price | Source |
|---|---|---|
| **A coding agent** | Claude Pro, $20 a month billed monthly (Claude Code included). Or ChatGPT Plus, $20 a month (Codex included) | claude.com/pricing, Codex pricing |
| **More usage, when you need it** | Claude Max, $100 a month, "5x Pro usage". Or ChatGPT Pro, from $100 a month | claude.com/pricing, Codex pricing |
| **A Mac and Xcode** | Xcode is free; it needs a Mac with an Apple M1 chip or later, on macOS 26.6 or later | Xcode on the Mac App Store |
| **Apple Developer Program** | $99 a year (TestFlight and App Store publishing come with it) | developer.apple.com/programs |
| **Supabase (backend)** | Free plan $0. Free projects pause after 1 week with no activity | supabase.com/pricing |
| **RevenueCat (subscriptions)** | Free up to $2,500 in monthly tracked revenue, then 1% | revenuecat.com/pricing |
| **Sentry (crashes)** | Free developer plan, one user, 5k errors a month | sentry.io/pricing |
| **PostHog (what users do)** | Free every month up to 1 million analytics events | posthog.com/pricing |
| **Prompt 1 and the review skill** | Free. The review skill is open source (MIT). Running them uses your agent's usage | Step 2 and Step 9 |

So the least it costs to reach the App Store is $20 a month plus Apple's $99 a year (more if you move to a $100 plan), if you already have a Mac that runs Xcode.

# Part 1: The ten steps

## Step 1: Your tech stack, a coding agent

> *"You need Codex or Claude in order to help you do this."*

A coding agent writes the code with you, in your project folder: Claude Code comes with Claude Pro, and Codex comes with ChatGPT plans.

- **Start on the $20 plan.** Claude Pro is $20 a month billed monthly; ChatGPT Plus is $20 a month.
- **Move up when you hit the limits.** As the app grows, you may hit the $20 plan's usage limits. Claude Max is $100 a month for "5x Pro usage"; ChatGPT Pro starts at $100 a month. In my view that is enough to build your first app.

## Step 2: Write a PRD before any code

> *"Before writing any code with these agents you need a ... product requirement doc."*

A PRD (product requirements document) outlines exactly what the app is, who it is for, the features that come with it, and what it means when the app is done. With one, you and your agent do not build in circles, and you have a solid foundation to start on.

Prompt 1 in Part 2 has your agent interview you and write the PRD for you.

## Step 3: Supabase for your backend

> *"Use Supabase for your back end which is very user friendly and it plugs straight into your coding agent."*

Supabase holds your database, user logins and file storage. Its free plan is $0, with two active projects; a free project is paused after 1 week with no activity, so check it is running before testers or Apple's reviewers use your app.

**Plug it into your agent.** Supabase runs an MCP server, a connection that lets your agent see and work on your Supabase project (Supabase's guide). The commands are in Part 2, Build order step 5. Supabase's guide warns that connecting an AI agent to your projects carries security risks, so connect one project, not your whole account, and once real users' data is in it, switch the connection to read-only (`read_only=true`) or remove it.

# Step 4: RevenueCat, Sentry and PostHog

> *"Use RevenueCat for all of your subscriptions so you make sure that you get paid. Use Sentry to see when your app crashes and use PostHog to see and monitor what your users are actually doing."*

- **RevenueCat** runs your subscriptions and in-app purchases on top of Apple's StoreKit and checks the receipts for you. It is free until $2,500 in monthly tracked revenue, then 1%. Purchases only work once you accept Apple's Paid Apps Agreement, add banking and tax details, and create your app's record in App Store Connect with the same bundle ID as your Xcode project (RevenueCat's iOS guide).
- **Sentry** tells you when and where the app crashes. The free developer plan is for one user, with 5k errors a month.
- **PostHog** shows what users actually do in the app, so you can improve each feature. Free each month up to 1 million analytics events.

Prompt 3 in Part 2 adds them one at a time, once your first features work.

# Step 5: Build it in Swift, for iOS first

> *"When building the first app, use Swift to build your app which is native for iOS."*

Swift is Apple's own programming language "for all Apple platforms", and SwiftUI is Apple's way to build the screens. You write and run it in Xcode, which is free and needs a Mac with an Apple M1 chip or later, on macOS 26.6 or later. Xcode can also work with coding agents from Anthropic, Google and OpenAI.

**Why iOS first: go where the money is.** In 2024, iOS made up about a quarter of App Store and Google Play downloads (28.3 of 110 billion, 26%), but about 72% of the money people spent in apps on the two stores was spent on the App Store ($91.6 of $127 billion), according to Appfigures, via TechCrunch.

# Step 6: Get an Apple developer account

> *"Get an Apple developer account which is $99 for one entire year."*

Join the Apple Developer Program for $99 a year (prices may vary by region). It is what lets you test with TestFlight and publish on the App Store.

# Step 7: Test with real users on TestFlight

> *"Nothing will replace feedback that you can get from a real user."*

TestFlight lets you send your app to testers before it is public:

- up to 100 people on your own team, and up to 10,000 external testers by email or a public link (Apple);
- your first build for external testers goes through Apple's review first (Apple);
- each build can be tested for up to 90 days (App Store Connect Help).

Give it to a handful of people who match the "who it is for" in your PRD. Watch Sentry and PostHog while they use it, and ask them what confused them.

# Step 8: Join the Small Business Program

> *"You can sign up for the small business program where Apple will take only 15% of your app subscriptions rather than the full 30%."*

Apple's App Store Small Business Program gives "a reduced commission rate of 15% on paid apps and Apple In-App Purchases". Without it, on a subscription you keep 70% in a subscriber's first year, so Apple takes 30%; in the program you keep 85% from the start. After a subscriber's first year you keep 85% either way.

- **Who qualifies:** developers new to the App Store, and developers who "made up to 1 million USD in proceeds in the prior calendar year for all their apps" (Apple).
- **It is not automatic:** you enrol. You need to be the Account Holder, accept the latest Paid Apps agreement in App Store Connect, and list any associated developer accounts (Apple).

## Step 9: Before you submit, run the review skill

> *"Before you submit, run your app against this free skill which checks your app against Apple's review guidelines."*

`app-store-review` is a free, open-source (MIT) skill that checks your app's code against Apple's App Store Review Guidelines. Its README says it is current through Apple's update of October 16, 2026, which was still the latest update on 26 November 2026 (Apple). It supports Swift, Objective-C, React Native and Expo apps, and works with Claude Code, Codex and other agents. It flags the potential rejection issues it finds, with code references, such as private API use, secrets in the code, an account you can create but not delete, or purchases with no "restore" button (README).

Prompt 4 in Part 2 has the install lines and the prompt.

## Step 10: Submit, and your first app is live

> *"And just like that your first app is live."*

Fix what the review found, run it again, then submit in App Store Connect with clear review notes for Apple, including a test login if your app has one. If Apple asks for a change, fix it, run the review skill again and resubmit.

# Part 2: The build order

This is the order to do it in. Steps marked **you** are yours: accounts, payments and anything with a password. Your agent does the code and tells you what to click.

### 1 Set up your coding agent (you)

Install Claude Code or Codex and sign in with your Claude or ChatGPT account. Make an empty folder for your app, open Terminal, type `cd ` (with a space), drag the folder onto the window, press Enter, then type `claude` or `codex`. Claude Code asks before it runs most commands or changes a file (Claude Code docs); Codex may ask too. Read what it wants to do and allow it if it makes sense.

### 2 Write your PRD (Prompt 1)

In that folder, paste Prompt 1. Your agent interviews you and writes `PRD.md`.

## Prompt 1: Write your PRD before any code

Shown here for reading. Copy it with the copy button at velorum-ai.com/resources/app, so nothing from this PDF page comes with it.

**What it does:** your coding agent interviews you about the app, then writes a product requirements document (PRD) into your project folder as `PRD.md`. The PRD says what the app is, who it is for, the features in the first version, and what "done" means. Prompts 2, 3 and 4 tell the agent to read this file first, so you and your agent build toward the same thing.

**When to use it:** first, in your project folder, before the agent writes any code.

**Fill in:** the one line in `[BRACKETS]`. Leave the rest as it is.

```
I want to build my first app, for iPhone first, in Swift. Before you write any code, interview
me and write a product requirements document (PRD).

My idea in one line: [WHAT THE APP DOES AND FOR WHOM]

How to interview me:
- Ask one question at a time, in plain words, with two or three example answers when a question
is abstract.
- Cover these, in this order: the problem it solves; who it is for (one person, described);
what they do today instead; the three to five features of the first version; what is out of
scope for now; how it makes money (free, one-time price, or subscription); whether users log in
and what data the app stores; what "done" means for version 1 (a short list I can check).
- We are building for iPhone first. If I want Android too, note it under out of scope for
later.
- If I am unsure, suggest the simplest option and say why in one line.
- Never invent facts about me, my users or my market. If I do not know, write it as an open
question.

When the interview is finished, write PRD.md in this folder with these sections:
1. One-line summary
2. The problem, and what they do today instead
3. Who it is for
4. Version 1 features, each with a one-line user story ("As a ..., I want ..., so that ...")
5. Out of scope for version 1
6. How it makes money
7. Logins and data stored
8. Done means (a checklist)
9. Open questions

Show me the PRD and ask "anything to change?" before we start building. From then on, read
PRD.md before each new feature, and tell me if a request would go outside it.
```

**Prefer a ready-made skill?** The `prd` skill in GitHub's awesome-copilot collection does the same job in its own format. With the GitHub CLI (version 2.90.0 or later): `gh skills install github/awesome-copilot prd`.

## 3 Make the Xcode project (you)

Install Xcode from the Mac App Store and open it once; if it offers to install extra components, let it finish. Then choose File > New > Project, pick App, Interface SwiftUI, Language Swift (Apple's guide). Product Name is your app's name as people will see it. Organization Identifier is what Xcode uses to make your app's bundle ID, for example com.yourname. Team is optional and can wait until you join the Apple Developer Program. Save it inside your app folder, and if the last sheet offers to create a Git repository, untick it: your agent sets up git in step 4. Xcode makes its own folder for the project inside yours, which is fine: `PRD.md` stays in the top folder, and you keep opening your agent there.

## 4 Build version 1 (Prompt 2)

Paste Prompt 2. The agent sets up git first, then builds one feature at a time. Prompt 2 writes its working rules into CLAUDE.md (or AGENTS.md for Codex), so they last. To stop, close the agent (in Claude Code, type `/exit`); to come back, open it in the same folder and say "read PROGRESS.md and carry on".

## Prompt 2: Start building from your PRD

Shown here for reading. Copy it with the copy button at velorum-ai.com/resources/app, so nothing from this PDF page comes with it.

**What it does:** your coding agent sets up git, reads `PRD.md` and builds version 1 one feature at a time, checking each one before the next, and keeps a `PROGRESS.md` so you can stop and come back.

**When to use it:** after Prompt 1 and after you have made the Xcode project (Build order, step 3). Open your agent in the project folder, the one that holds `PRD.md`.

**Fill in:** nothing.

```
Read PRD.md in this folder. We are building version 1 of this app for iPhone, in Swift and
SwiftUI. The Xcode project is in a subfolder of this folder.

Before any feature:
- Write the working rules below into a file in this folder that you read at the start of every
session (CLAUDE.md for Claude Code, AGENTS.md for Codex), so they still hold if I close you and
come back.
- If this folder is not a git repository yet, set one up here (the folder with PRD.md). If the
Xcode subfolder already has its own .git folder, tell me and ask before changing it. Add a
.gitignore for Xcode projects (xcuserdata, build folders, .DS_Store) plus Secrets.xcconfig,
.env, *.p8 and .sentryclirc. Make a first commit, and check that the Xcode project files are in
it.
- Keys: a key that a service makes for apps (for example Supabase's publishable key,
RevenueCat's public SDK key) can ship inside the app, but put it in Secrets.xcconfig, not in
the code. A secret or admin key (for example Supabase's secret key) never goes into the app or
into git at all: tell me if a feature seems to need one: that call belongs on a server (for
example a Supabase Edge Function), never in the app, not even as an environment variable. Key
files I download (for example .p8 files) stay outside this folder.

How to work:
- Version 1 keeps its data on the phone for now. No accounts, backend, payments or analytics
yet: we add those later, one at a time.
- List the version 1 features from PRD.md in the order you suggest building them, and wait for
my yes.
- Build one feature at a time. Before each one, say in one or two plain sentences what you will
change. After each one, build the project and run the tests, fix what fails, commit, and tell
me how to try it in the simulator.
- Stay inside PRD.md. If I ask for something outside it, say so and ask whether to add it to
the PRD first.
- Keep PROGRESS.md in this folder: the feature we are on, what is done, what is next, decisions
we made. Update it after every feature. If I come back later and say "read PROGRESS.md and
carry on", pick up from there.
- Never delete or overwrite a file I made without asking.
- If you are not sure how an Apple framework works, say so and check Apple's documentation
rather than guessing.
```

## 5 Connect Supabase (you, then the agent)

Once version 1 works in the simulator, sign up at supabase.com and create a project for this app. You need its project ID, which Supabase calls the project ref. Close your agent (type `/exit`), then run this in a terminal in your app folder, with your project ref in place of `YOUR_PROJECT_REF`. It is the command from Supabase's MCP guide, with its feature list taken out and its `project_ref` option added, which scopes the connection to one project:

```
claude mcp add --scope project --transport http supabase \
  "https://mcp.supabase.com/mcp?project_ref=YOUR_PROJECT_REF"
```

Then start Claude Code in the folder (`claude`), type `/mcp`, choose "supabase", then "Authenticate", and sign in to Supabase in your browser (Supabase, Claude Code docs). Supabase says you may need to restart after signing in, so type `/exit`, run `claude` again, and go on to step 6. Supabase's guide also lists the setup for other agents; if yours is not there, check your agent's own MCP docs.

## 6 Join the Apple Developer Program (you)

Enrol at developer.apple.com/programs ($99 a year). In Xcode, set Team to your new developer account. In App Store Connect, accept the Paid Apps Agreement, add your banking and tax details, and create the app record (its bundle ID matches your Xcode project): in-app purchases do not work before that (RevenueCat's iOS guide).

## 7 Add the services (Prompt 3)

Paste Prompt 3. Create the RevenueCat, Sentry and PostHog accounts when the agent asks.

## Prompt 3: Add the services (backend, payments, crashes, analytics)

Shown here for reading. Copy it with the copy button at velorum-ai.com/resources/app, so nothing from this PDF page comes with it.

**What it does:** your coding agent adds each service the playbook recommends, one at a time, using that service's own setup guide, and tells you which accounts and keys you need to create yourself.

**When to use it:** once the first features work (Prompt 2), after you have connected Supabase to your agent and restarted it (Build order, step 5), and after the Paid Apps Agreement, banking, tax and the app record are done in App Store Connect (Build order, step 6). RevenueCat purchases do not work before that.

**Fill in:** delete the lines for any service you do not need yet.

```
Read PRD.md and PROGRESS.md. Now add these services to the app, one at a time, in this order. For each one: tell me what it is for in one line, tell me exactly which account or key I need to create myself (I do the sign-ups and payments, never you), then follow the service's own current setup guide for iOS and Swift Package Manager, build, test, commit, and update PROGRESS.md.

1. Supabase, as the backend: the tables, logins and storage the PRD needs. Use the Supabase connection you have in this session; if you do not see it, stop and tell me. Turn on row level security for every table and explain each policy in one line. If the app has accounts, add a way to delete the account from inside the app. If I want Google or another social login, tell me what Apple's guideline 4.8 asks for first.
2. RevenueCat, for subscriptions and in-app purchases: the products from the PRD, a paywall screen, and a "restore purchases" button.
3. Sentry, to see when the app crashes.
4. PostHog, to see what users do: only the few events that tell me whether the PRD's features are used. Nothing personal in event names or properties.

Keys follow the rule in CLAUDE.md or AGENTS.md: app keys go in Secrets.xcconfig, secret keys never go into the app or git. If a step needs something only I can do in App Store Connect or a service's dashboard, stop and tell me what to click.
```

### 8 Add a privacy policy and privacy details (you, with the agent)

Apple requires a link to your privacy policy both in App Store Connect and inside the app (guideline 5.1.1(i)). Ask your agent to draft one from your PRD and the services you added, read it, then publish it on a public page you control, for example a GitHub Pages site. In App Store Connect, also fill in the App Privacy section about the data your app and the services you added collect (App Store Connect Help); ask your agent to list it from the code. If your app lets people create an account, it must also let them delete it from inside the app (guideline 5.1.1(v)).

### 9 Review before testers (Prompt 4)

Install the review skill, start a new session and paste Prompt 4. Fix what it finds.

## Prompt 4: Check your app against Apple's review guidelines

Shown here for reading. Copy it with the copy button at velorum-ai.com/resources/app, so nothing from this PDF page comes with it.

**What it does:** with the free `app-store-review` skill installed, your coding agent reads your app's code against Apple's App Store Review Guidelines and lists the potential rejection issues it finds, with code references, so you can fix them before Apple's review does.

**When to use it:** twice. Before you send your first build to external testers on TestFlight (Apple reviews that build too), and again right before you submit. Run it again after you fix anything.

**Install the skill first** (pick the line for your agent, adapted from the skill's README):

- Claude Code, typed inside Claude Code: `/plugin marketplace add safaiyeh/app-store-review-skill`, then `/plugin install app-store-review@app-store-review`
- Codex, in your terminal: `codex plugin marketplace add safaiyeh/app-store-review-skill`, then in Codex run `/plugins` and install `app-store-review`
- Other agents that support skills, in a terminal in your project folder: `DO_NOT_TRACK=1 npx skills@latest add safaiyeh/app-store-review-skill`. It needs Node.js 22.20 or later. It asks which agents to install for; pick yours. `DO_NOT_TRACK=1` turns off the installer's anonymous usage data. It adds a `.agents` folder and a `skills-lock.json` file to your project (and a `.claude/skills` link if you pick Claude Code); that is expected.

Then start a new session in your project folder, so the agent sees the skill, and paste Prompt 4, copied with its copy button at velorum-ai.com/resources/app.

**Fill in:** nothing.

```
Read PRD.md. Then review this app for App Store compliance, using the app-store-review skill.

Then:
- List the findings from most to least serious, each with the guideline number, the file and line, and one plain sentence on why Apple could reject it.
- For each finding, suggest the smallest fix, and wait for my yes before you change any code. A fix never puts a secret key into the app, not even as an environment variable: a call that needs one moves to a server (for example a Supabase Edge Function).
- After I approve fixes and you make them, build the app, run the tests, commit, and run the review again.
- Tell me which checks you could not do from the code alone (for example App Store Connect metadata, the privacy policy link, screenshots, or the review notes), so I can check those myself.
```

**Good to know:** it reads your code, so it cannot see everything App Review looks at, and a clean result is not an approval. Apple's App Store Review Guidelines are the final word. If the skill misses something, it may offer to draft a feedback issue for its GitHub page; its README says it asks first and shows you the text, so you can say no.

### 10 Test on TestFlight (you, with the agent)

In Xcode, archive the app and upload it to the app record you made in step 6. Apple lists Xcode as one way to upload (App Store Connect Help). The first upload is processed before it appears in App Store Connect, and Apple emails you when it is ready. Add yourself as an internal tester, then a group of external testers (TestFlight). Ask your agent to walk you through each click if it is your first time.

### 11 Enrol in the Small Business Program (you)

Apply on Apple's page as soon as step 6 is done: Apple adjusts your proceeds 15 days after the end of the fiscal calendar month in which your enrolment is approved.

### 12 Review again and submit

Run Prompt 4 again on the final build, fix what it finds, and submit with review notes.

## The checklist

- [ ] Coding agent installed, app folder made (Build order 1)
- [ ] `PRD.md` written and agreed (Prompt 1)
- [ ] Xcode project made inside the app folder
- [ ] Git set up and version 1 features built and tested (Prompt 2)
- [ ] Supabase project made and connected to the agent, project only
- [ ] Apple Developer Program joined; Paid Apps Agreement, banking, tax and app record done
- [ ] RevenueCat, Sentry and PostHog added (Prompt 3)
- [ ] Privacy policy published and linked; App Privacy filled in; in-app account deletion if you have accounts
- [ ] Review skill run before external testers, findings fixed (Prompt 4)
- [ ] Tested on TestFlight with real users, feedback fixed
- [ ] Enrolled in the Small Business Program
- [ ] Review skill run again, submitted with review notes

Want help building your first app? Book a call with Velorum at velorum-ai.com.

# Sources

Every source this playbook relies on, and when it was checked. Prices and rules change, so check the link before you pay.


1. Claude pricing: Pro and Max, Claude Code included (checked 26 November 2026)
   https://claude.com/pricing
2. Codex pricing: ChatGPT Plus and Pro (checked 26 November 2026)
   https://developers.openai.com/codex/pricing
3. Xcode on the Mac App Store: free, requirements, agent support (checked 26 November 2026)
   https://apps.apple.com/us/app/xcode/id497799835?mt=12
4. Apple Developer Program (checked 26 November 2026)
   https://developer.apple.com/programs/
5. Supabase pricing: free plan, paused projects (checked 26 November 2026)
   https://supabase.com/pricing
6. RevenueCat pricing (checked 26 November 2026)
   https://www.revenuecat.com/pricing/
7. Sentry pricing: developer plan (checked 26 November 2026)
   https://sentry.io/pricing/
8. PostHog pricing (checked 26 November 2026)
   https://posthog.com/pricing
9. Supabase MCP guide: the command, sign-in, security advice (checked 26 November 2026)
   https://supabase.com/docs/guides/getting-started/mcp
10. RevenueCat (checked 26 November 2026)
    https://www.revenuecat.com/
11. RevenueCat iOS installation guide (checked 26 November 2026)
    https://www.revenuecat.com/docs/getting-started/installation/ios
12. Sentry for iOS (checked 26 November 2026)
    https://sentry.io/for/ios/
13. PostHog iOS library (checked 26 November 2026)
    https://posthog.com/docs/libraries/ios
14. Apple: Swift (checked 26 November 2026)
    https://developer.apple.com/swift/
15. TechCrunch, Appfigures figures for 2024 downloads and spending (18 December 2024) (checked 26 November 2026)
    https://techcrunch.com/2024/12/18/app-downloads-decline-2-3-in-2024-but-consumer-spending-grows-to-127b/
16. Apple Developer Program enrollment: $99 a year, prices by region (checked 26 November 2026)
    https://developer.apple.com/support/enrollment/

17. Apple: TestFlight (checked 26 November 2026)
https://developer.apple.com/testflight/
18. App Store Connect Help: TestFlight overview (checked 26 November 2026)
https://developer.apple.com/help/app-store-connect/test-a-beta-version/testflight-overview/
19. Apple: App Store Small Business Program (checked 26 November 2026)
https://developer.apple.com/app-store/small-business-program/
20. Apple: auto-renewable subscriptions, proceeds (checked 26 November 2026)
https://developer.apple.com/app-store/subscriptions/
21. The app-store-review skill, README (MIT) (checked 26 November 2026)
https://github.com/safaiyeh/app-store-review-skill
22. Apple: App Store Review Guidelines (checked 26 November 2026)
https://developer.apple.com/app-store/review/guidelines/
23. Claude Code docs: permissions (checked 26 November 2026)
https://code.claude.com/docs/en/permissions
24. The prd skill in GitHub's awesome-copilot (MIT) (checked 26 November 2026)
https://github.com/github/awesome-copilot/tree/main/skills/prd
25. Apple: creating an Xcode project for an app (checked 26 November 2026)
https://developer.apple.com/documentation/xcode/creating-an-xcode-project-for-an-app
26. Claude Code docs: MCP (checked 26 November 2026)
https://code.claude.com/docs/en/mcp
27. App Store Connect Help: manage app privacy (checked 26 November 2026)
https://developer.apple.com/help/app-store-connect/manage-app-information/manage-app-privacy/
28. App Store Connect Help: upload builds (checked 26 November 2026)
https://developer.apple.com/help/app-store-connect/manage-builds/upload-builds/
29. Supabase API keys: publishable and secret keys (checked 26 November 2026)
https://supabase.com/docs/guides/api/api-keys
30. Apple: SwiftUI (checked 26 November 2026)
https://developer.apple.com/swiftui/
31. The skills installer adds a .agents folder and a skills-lock.json file (and a .claude/skills link for Claude Code): observed in two install runs (CLI 1.5.18 and 1.9.0) (26 November 2026)
32. The Skills CLI: DO_NOT_TRACK=1 turns off its usage data, and it needs Node.js 22.20 or later: npm package skills 1.9.0 (README "Telemetry"; package.json engines) (26 November 2026)

Where to get the tools (sign-up and install pages, not sources): Claude Code setup code.claude.com/docs/en/setup, Codex developers.openai.com/codex, Node.js nodejs.org.