



FASTER PATH TO COMMERCIALISATION

What Actually Drives the Cost of Building an App

A founder's guide to scoping, briefing and comparing quotes

Inside

1. The five things that move the number
2. Scope: what an MVP is, and what it is not
3. The features that look cheap and are not
4. Native, cross-platform or web
5. Design: how much is enough
6. The part founders forget: after launch
7. Who builds it
8. How to brief so the quotes are comparable
9. Questions worth asking anyone who quotes you
10. How AppCurate approaches it

Why this guide has no numbers in it

Every founder asks the same first question, and it is the right question: what will this cost? The honest answer is that nobody can tell you from a one-line description, and anyone who does is either padding heavily to cover what they do not yet know, or quoting something narrower than what you have in your head.

That is not evasion. A build is not a product you buy off a shelf; it is a quantity of work, and the quantity is set by decisions most founders have not made yet. Which platforms. What happens offline. Who can see whose data. What the thing must talk to. How much of it has to exist before the first user does.

So this guide does something more useful than quoting a number. It walks through the decisions that move the number, so that when you do ask for a quote, you can describe what you want precisely enough to get answers you can actually compare — and so you can tell the difference between a cheap quote and a quote for less product.

1. The five things that move the number

Almost every difference between two quotes for “the same app” traces back to one of five things. Get specific about these and most of the uncertainty disappears.

Scope	How much has to work on day one. The single biggest lever, and the one most under your control.
Platforms	iOS, Android, web, or some combination — and whether they share a codebase.
Integrations	Every external system you connect to is someone else's rules, someone else's downtime and someone else's changes.
Design depth	The gap between a product that works and a product that feels considered is real work, and it is not all visual.
Data and compliance	Who the data belongs to, where it lives, and what you are obliged to do with it. Health, finance and anything involving children change the job.

2. Scope: what an MVP is, and what it is not

A minimum viable product is not a cheap version of the full product. It is the smallest thing that puts your central assumption in front of real users and gets you an answer. Those are different objects, and confusing them is the most expensive mistake in this whole document.

The useful test is to write down the one thing you are not sure about — that people will pay, that they will come back, that the workflow saves them time — and then ask of every feature whether its absence would stop you learning the answer. Most features fail that test. They are things you will want later, which is not the same as things you need now.

Scope cut early is cheap. Scope cut after it is built is the most expensive kind of money a start-up spends, because you paid for it twice: once to build it and once in the time you did not spend on what mattered.

3. The features that look cheap and are not

Some things sound like one line in a brief and are weeks of work underneath. These are the ones that most often blow a budget that was set before anyone looked closely.

Sign-in and accounts	Email, social, password resets, sessions, someone losing their phone. “Users can log in” is a system, not a screen.
Payments	Taking money is the easy part. Refunds, failed cards, receipts, disputes, tax and what the app does while a payment is pending are the rest of it.
Notifications	Sending one is simple. Deciding who gets what, when, how often, and how they turn it off is product design.
Working offline	An app that must function without a connection needs to store, queue and reconcile — and decide what wins when two versions disagree.
Search	Matching exact words is quick. Tolerating typos, ranking results and staying fast as the data grows is a different problem.
Photos, video and files	Capture, upload on a bad connection, resize, store, serve and moderate. Each step is small; together they are not.
Admin and back office	Someone has to run this thing. The screens your own team needs are real product, and they are routinely left out of briefs.

4. Native, cross-platform or web

This choice changes the bill more than almost any other technical decision, and it is usually made for the wrong reason — because a developer prefers one, rather than because the product needs it.

Web first	Fastest to put in front of people and to change. No app store review standing between you and a fix. Right more often than founders expect, particularly for anything used at a desk.
One cross-platform codebase	Two platforms from mostly one body of work. The compromise shows at the edges — deep hardware access, heavy animation, anything that must feel native in the hand.
Fully native	The best result per platform and roughly two of everything. Worth it when the product lives or dies on feel, performance or hardware.

5. Design: how much is enough

Design is not the coat of paint applied at the end. The expensive version of design is the one that happens after the build, when a screen turns out not to work and the thing underneath it has to change too.

Before you have users, you need enough design to answer three questions: can someone find the thing they came for, do they understand what happened after they act, and does it look like it was made on purpose. That is achievable without a full visual system, a brand refresh or a library of states you do not yet need.

What is genuinely worth paying for early is the thinking: the flows, the empty states, the error cases, what the app says when something goes wrong. Those are where products feel cheap, and they cost almost nothing to get right up front.

6. The part founders forget: after launch

A build budget that stops at launch day is not a budget. The product goes on existing, and most of these are recurring.

Hosting and infrastructure	Scales with use, so it is small until it is not.
Store accounts and fees	Apple and Google both charge to be listed, and take a share of what you sell inside the app.
Third-party services	Maps, messaging, email, analytics, identity. Each has its own meter.
Keeping up with the platforms	iOS and Android ship new versions every year. An app nobody maintains quietly stops working.
Support	Someone answers the emails, resets the accounts and chases the bugs.
The next version	The first release is where you start learning, which means it is where the real roadmap begins.

7. Who builds it

There is no universally right answer here, only trade-offs that suit different stages. Be honest about which stage you are at.

A development agency	Capacity and process from day one. You are a client, so you get what is specified — which puts the weight on specifying well.
-----------------------------	---

Independent contractors	Flexible and direct, and you are the project manager. Works when you know exactly what you want and can tell good work from bad.
Hiring in-house	The right end state for a product company, and slow and expensive to reach before you have proven anything or raised against it.
A venture partner	A build team that also has a stake in whether the product works. Fewer firms do this, and it only makes sense if their interests genuinely sit alongside yours.

8. How to brief so the quotes are comparable

Most founders get quotes that cannot be compared, because each firm filled the gaps in the brief differently. Put these in writing and the spread between quotes narrows dramatically — and what is left is real difference rather than guesswork.

The one assumption you are testing	In a sentence. Everything else is negotiable against it.
Who the user is, and where they are	On a laptop at a desk and on a phone in a warehouse are different products.
The handful of things they must be able to do	Listed as actions, not features.
What it has to connect to	Name the systems. If you do not know whether they have an API, say so — that is a finding, not a failure.
What you already have	Brand, designs, an existing back end, a data set, a prior attempt.
What happens to the data	Who owns it, what is sensitive, what is regulated.
Your deadline, and what is driving it	A raise, a pilot, a season, a competitor.
What success looks like	The number that would tell you it worked.

9. Questions worth asking anyone who quotes you

The answers tell you more about how a build will go than the quote itself does.

Who owns the code and the accounts at the end?	The only acceptable answer is that you do.
---	--

Who exactly is building it?	Names and locations, not a company brochure.
What is deliberately not in this quote?	A good partner has a list ready. No list means the scope has not been thought about.
What will you show me, and how often?	Working software on a regular cadence, or slides.
What happens when something takes longer than you thought?	It will. Ask now how it is handled.
What do you think I should cut?	Anyone who says nothing is selling, not advising.
Can I talk to a founder you built for who did not succeed?	The most revealing question on this list.

10. How AppCurate approaches it

AppCurate builds the product and backs the raise, which is why this guide argues for cutting scope rather than for selling more of it. A bigger first build is not in our interest if it means less runway for the version that follows.

The MVP Accelerator runs in three stages, and each one ends with a decision rather than an invoice:

Idea Validation	Pressure-test the assumption before anything is built. The output is a scope worth building, or a clear reason not to.
Shape-It	Turn that scope into flows, screens and a technical approach, so what gets built is already decided rather than discovered mid-build.
Build-It	Build and ship the first version, with the code and accounts in your name from the start.

If you want a figure for your own build, the fastest route to an accurate one is a conversation about scope — not a form. Bring the eight points from section 8, even half-answered, and the estimate you get back will be worth something.



Ready to put a number on your own build?

Roland Kulen, Co-Founder

Email info@appcurate.com

Phone [+61 414 429 986](tel:+61414429986)

Web appcurate.com

Library appcurate.com/library/

This guide is general information for founders planning a software build. It is not financial, legal or investment advice, and it does not take your particular circumstances into account.