

MINI PACK

Get Paid

Stripe Checkout and webhooks that actually fire.

For Cursor · Claude Code · Claude · ChatGPT · Windsurf · Lovable · Bolt · v0 · Replit

9 prompts · 4 sections

Welcome! How to Use This Pack

Someone clicks "Buy," pays, and... nothing. The webhook never arrives, they never get access, or it all worked in test mode and broke the moment you switched to live keys.

This pack walks you through Stripe payments from start to finish: creating a Checkout Session, adding success and cancel pages, building a webhook endpoint that receives the right events, unlocking what people paid for exactly once, testing with test cards and the Stripe CLI, and switching safely to live mode.

Stripe changes its dashboard and docs over time. The prompts ask your AI to say when it isn't sure, and the tips point you to Stripe's official docs at docs.stripe.com. If your AI's instructions don't match what you see, trust Stripe's dashboard and docs.

How each prompt is laid out

- **Title:** what the prompt is.
- **Use this when...:** one or two sentences on what the prompt is for and when to reach for it.
- **Fill in:** the blanks to replace before you paste.
- **The prompt:** the box you copy.
- **Tips to customize:** optional ways to change it.

Anything in **[BRACKETS IN CAPS]** is a blank to fill in. **Never paste real Stripe keys or webhook secrets into an AI chat.** Use variable names like `STRIPE_SECRET_KEY` instead.

Works best with: Agentic editors like Cursor and Claude Code can create the files, run the Stripe CLI, and read the output for you. App builders like Lovable, Bolt, and v0 work well for the pages and flows. Every prompt also works in any AI chat.

Using Ship It Safe too? Its Prompt 4.3 covers checking that webhook requests really come from Stripe. This pack covers everything else that makes payments work.

Quick Start: Find Your Problem

- **Haven't added payments yet?** Start with Prompts 1.1 and 1.2.
- **Paid in test mode, but your app never heard about it?** Prompt 2.2.
- **Users pay but don't get access, or get it twice?** Prompts 2.3 and 3.1.
- **Ready to take real money?** Prompts 4.1, then 4.2.

Golden rule: Only unlock what someone paid for after your server confirms the payment with Stripe. Never trust the success page alone.

What's Inside

1. Set Up Checkout (2 prompts)
2. Webhooks That Actually Fire (3 prompts)
3. Unlock What They Paid For (2 prompts)
4. Test It, Then Go Live (2 prompts)

SECTION 1

Set Up Checkout

Send people to a Stripe-hosted payment page and bring them back.

1.1 Add Stripe Checkout for One Product

Use this when you want to sell one product or plan and haven't set up payments yet. It's for adding a working "Buy" button that sends people to a Stripe-hosted Checkout page.

Fill in: [SKILL LEVEL], [TECH STACK], [WHAT I'M SELLING] (e.g. "lifetime access to the Pro features" / "a monthly plan"), [ONE-TIME / SUBSCRIPTION], [PAGE].

Help me add Stripe Checkout to my app for one product. I'm a [SKILL LEVEL], so go step by step and explain each part. Use Stripe test mode for now.

Tech stack: [TECH STACK]

What I'm selling: [WHAT I'M SELLING]

One-time payment or subscription: [ONE-TIME / SUBSCRIPTION]

Where the Buy button goes: [PAGE]

Please:

1. Tell me how to create the product and price in my Stripe Dashboard (in a sandbox, with test keys) and which ID my code needs (the price ID, not the name).
2. Create a server-side route that creates a Checkout Session with that price, the right mode ("payment" or "subscription"), a `success_url`, and a `cancel_url`, then redirects the customer to the session's URL.
3. Keep the secret key on the server only, read from an environment variable. Explain which Stripe keys are safe in the browser (publishable) and which aren't (secret and restricted).
4. If users log in, attach my user ID to the session (for example with `client_reference_id` or `metadata`) so I know who paid.
5. Add the Buy button that calls my route.
6. List every file you created or changed, and tell me how to test it with a Stripe test card.

If you're not sure about a Stripe option, say so and point me to the right page on docs.stripe.com.

 **Tips to customize:** For subscriptions, ask the AI to explain what changes compared with a one-time payment before it writes code. Checkout Sessions expire after a while, so create a new one each time someone clicks Buy instead of reusing an old link.

1.2 Success and Cancel Pages That Tell the Truth

Use this when you have Checkout working but your success page just says "Thanks!" no matter what, or the cancel button lands on a broken page. You get pages that show the right message after checkout without pretending payment is confirmed when it might not be.

Fill in: [TECH STACK], [WHAT THE CUSTOMER GETS] (e.g. "Pro features unlocked on their account").

Help me build proper success and cancel pages for my Stripe Checkout flow.

Tech stack: [TECH STACK]

What the customer gets: [WHAT THE CUSTOMER GETS]

Please:

1. Set the `success_url` to include the literal `{CHECKOUT_SESSION_ID}` placeholder (for example `/success?session_id={CHECKOUT_SESSION_ID}`) and explain that Stripe fills in the real ID when it redirects.
2. On the success page, read the session ID on the server, retrieve the Checkout Session from Stripe, and check its `payment_status` before saying "You're all set."
3. If payment is still processing, show a friendly "We're confirming your payment" message instead of an error.
4. Never unlock access just because someone loaded the success page. Explain why (people can close the tab, lose their connection, or visit the URL directly).
5. Build a cancel page that says nothing was charged and links back to the Buy button.
6. Make both pages work if someone refreshes them or opens them twice.



Tips to customize: Pair this with Prompt 3.1 so the success page and the webhook both use the same fulfillment function.

SECTION 2

Webhooks That Actually Fire

How Stripe tells your server that someone paid.

2.1 Build the Webhook Endpoint

Use this when you have Checkout working and need your app to find out when a payment actually succeeds. It's for creating the server route that receives Stripe's events and responds correctly.

Fill in: [TECH STACK], [HOST] (e.g. Vercel / Netlify / Render / my own server), [ONE-TIME / SUBSCRIPTION].

Build a Stripe webhook endpoint for my app. Explain each part simply.

Tech stack: [TECH STACK]

Host: [HOST]

One-time payment or subscription: [ONE-TIME / SUBSCRIPTION]

Please:

1. Create a POST route (for example /api/webhooks/stripe) that reads the RAW request body. Tell me exactly how to do that in my framework, because many frameworks parse the body automatically and that breaks Stripe's signature check.
2. Verify the event using Stripe's official library, the Stripe-Signature header, and a webhook signing secret from an environment variable.
3. Handle only the events I need. For one-time Checkout payments, start with checkout.session.completed, plus checkout.session.async_payment_succeeded and checkout.session.async_payment_failed if I might accept delayed payment methods. For subscriptions, tell me which events to add and point me to Stripe's subscription webhook docs.
4. Return a 2xx response quickly. If the work is slow, explain how to respond first and do the work after.
5. Log the event type and event ID (never secrets or full card details) so I can debug.
6. Tell me which events to select when I register this endpoint in my Stripe Dashboard.



Tips to customize: For a deeper check of signature verification, see Ship It Safe Prompt 4.3. If your framework has CSRF protection on all POST routes, ask the AI whether the webhook route needs an exemption.

2.2 "My Webhook Never Fires"

Use this when payments succeed in Stripe, but your app never reacts, or your logs show errors on the webhook route. It's for tracing why Stripe events aren't reaching your app, or are reaching it and failing.

Fill in: [TECH STACK], [HOST, URL], [LOCAL / LIVE / BOTH], [WEBHOOK ROUTE IN MY CODE] (e.g. /api/webhooks/stripe), [WHAT I SEE] (e.g. "nothing in my logs", "400 errors", "signature verification failed", "works locally, not live").

Stripe payments succeed, but my webhook isn't working. Help me find out why, step by step.

Tech stack: [TECH STACK]

Host and live URL: [HOST, URL]

Testing locally or on the live site: [LOCAL / LIVE / BOTH]

Webhook route in my code: [WEBHOOK ROUTE IN MY CODE]

What I see: [WHAT I SEE]

Check these in order and explain each in plain English:

1. Is the endpoint registered in Stripe for the right mode (test vs live), with the exact URL (https, domain, and path) that matches my route?
2. Is it subscribed to the right event types (for example `checkout.session.completed`)?
3. In Stripe's dashboard, what do the event delivery attempts for this endpoint show (delivered, pending, or failed, plus the HTTP status code)? Tell me where to find this.
4. Status code clues: a redirect (3xx) counts as a failure, a 404 means the URL doesn't exist, a 401, 403, or 405 usually means access is restricted or the route doesn't accept POST, a 5xx means my code is crashing, and a timeout means I'm responding too slowly.
5. Signature errors: am I using the right signing secret? The one from "stripe listen" is different from the dashboard endpoint's secret, and test and live endpoints have different secrets too. Is my framework changing the raw body?
6. Is the environment variable for the signing secret set on my host for the right environment, and did I redeploy after adding it?
7. Are preview deployments protected by a login wall that Stripe can't get through?

Give me the most likely cause first, the fix, and how to resend a test event to confirm.

 **Tips to customize:** Stripe lets you resend an event from the dashboard, and the Stripe CLI can resend events too. Check Stripe's webhook docs for the current options and time limits.

2.3 Handle Duplicate and Out-of-Order Events

Use this when you've seen double emails or double credits, or you want to make your webhook safe before launch. It's for making sure a retried or repeated event never unlocks something twice, sends two emails, or breaks your data.

Fill in: [TECH STACK], [DATABASE], [WHAT MY HANDLER DOES WHEN SOMEONE PAYS] (e.g. "sets is_pro = true and sends a welcome email").

Make my Stripe webhook handler safe against duplicate and out-of-order events.

Tech stack: [TECH STACK]

Database: [DATABASE]

What my handler does when someone pays: [WHAT MY HANDLER DOES WHEN SOMEONE PAYS]

Please:

1. Explain in plain English why the same event can arrive more than once (Stripe retries if my server doesn't respond with a 2xx in time) and why events can arrive in a different order than they happened.
2. Add a table (or column) that records each processed Stripe event ID, and skip events I've already handled.
3. Make the actual work idempotent too: for example, "set is_pro to true" is safe to repeat, but "add 100 credits" is not. Rewrite anything that isn't safe to repeat.
4. Make sure two copies of the same event arriving at the same moment can't both slip through (explain the simplest way to do this in my database, such as a unique constraint).
5. Don't rely on events arriving in order. If I need the latest state, fetch the object from Stripe instead of trusting the event's order.
6. Give me a way to test this by sending the same event twice.

SECTION 3

Unlock What They Paid For

Fulfillment: giving people what they bought, exactly once.

3.1 Fulfill Once, From Both Places

Use this when users pay but sometimes don't get access, or get it only after a delay. You get one fulfillment function that the webhook and the success page can both call safely, so access unlocks fast and never twice.

Fill in: [TECH STACK], [DATABASE], [WHAT FULFILLMENT MEANS FOR MY APP] (e.g. "mark the user as Pro", "add the course to their library", "email a download link"), [HOW I LINK A PAYMENT TO A USER] (e.g. "client_reference_id is my user ID" / "match by email" / "not sure").

Help me build one fulfillment function for my Stripe Checkout payments that unlocks what the customer paid for exactly once.

Tech stack: [TECH STACK]

Database: [DATABASE]

What fulfillment means for my app: [WHAT FULFILLMENT MEANS FOR MY APP]

How I link a payment to a user: [HOW I LINK A PAYMENT TO A USER]

Please:

1. Create a function like `fulfillCheckout(sessionId)` that retrieves the Checkout Session from Stripe (with line items if I need them), checks `payment_status`, and only fulfills if it's paid.
2. Make it safe to call more than once, even at the same time, with the same session ID. Record that the session was fulfilled.
3. Call it from my webhook handler for `checkout.session.completed` (and `checkout.session.async_payment_succeeded` if I accept delayed payments).
4. Also call it from my success page, so customers who are still on the page get access right away. Explain why the webhook is still required (they might never reach the success page).
5. Find the right user reliably. Warn me if matching only by email could unlock the wrong account.
6. Show the customer a clear message about what they now have.



Tips to customize: Stripe's fulfillment guide (search "Fulfill orders" on docs.stripe.com) describes this pattern. Ask your AI to compare its code with that guide.

3.2 Refunds, Failed Payments, and Cancellations

Use this when you've handled the happy path and want to cover what happens when money goes the other way. It's for deciding what happens to a user's access when a payment fails, gets refunded, or a subscription ends.

Fill in: [TECH STACK], [ONE-TIME / SUBSCRIPTION], [MY POLICY] (e.g. "remove Pro access after a refund", "keep access until the end of the paid period if they cancel").

Help me handle the "money goes the other way" cases in my Stripe integration without breaking user access unfairly.

Tech stack: [TECH STACK]

One-time payment or subscription: [ONE-TIME / SUBSCRIPTION]

My policy: [MY POLICY]

Please:

1. List the Stripe events that matter for my setup and what each one means in plain English. Consider `checkout.session.async_payment_failed` and `charge.refunded`, plus `invoice.payment_failed`, `invoice.paid`, `customer.subscription.updated`, and `customer.subscription.deleted` for subscriptions. Check Stripe's event docs and tell me if any of these don't fit my case.
2. For each event, write what my app should do, based on my policy.
3. Add those events to my webhook handler, using the same duplicate-safe pattern as my other events.
4. Make sure a failed renewal doesn't instantly lock someone out if my policy gives them time to update their card.
5. Write the short, friendly messages or emails users should see in each case.
6. Tell me how to test each case in a sandbox (test cards that decline, and refunds from the dashboard).



Tips to customize: If you sell subscriptions, read Stripe's "Using webhooks with subscriptions" page. It lists which events to use for granting and removing access.

SECTION 4

Test It, Then Go Live

Prove it works with fake money before you take real money.

4.1 Test Everything with Test Cards and the Stripe CLI

Use this when you've built Checkout and the webhook and want to make sure they work together. You get a full test run of your payment flow on your own computer, including webhooks, before anyone pays for real.

Fill in: [TECH STACK], [MY LOCAL APP URL AND WEBHOOK ROUTE] (e.g.

localhost:3000/api/webhooks/stripe), [WINDOWS / MAC / LINUX], [MY LOCAL WEBHOOK URL].

Help me test my whole Stripe payment flow locally in test mode. Give me exact commands.

Tech stack: [TECH STACK]

My local app URL and webhook route: [MY LOCAL APP URL AND WEBHOOK ROUTE]

My computer: [WINDOWS / MAC / LINUX]

Please:

1. Tell me how to install the Stripe CLI and log in (point me to Stripe's install page for my system).
2. Show me how to run "stripe listen --forward-to [MY LOCAL WEBHOOK URL]", and explain that it prints a webhook signing secret (starting with whsec_) that I should put in my local environment variables. Remind me it's different from my dashboard endpoint's secret.
3. Walk me through a real test purchase: click my Buy button and pay with Stripe's success test card (4242 4242 4242 4242, any future expiry date, any CVC). Tell me what I should see in the "stripe listen" terminal and in my app.
4. Show me how to use "stripe trigger checkout.session.completed" to send a test event, and explain that it creates its own test objects, so it won't be linked to one of my real users.
5. Give me a checklist: a successful payment, a declined card (from Stripe's testing docs), a cancelled checkout, the same event delivered twice, and a refund made from the dashboard.
6. After each test, tell me what to check in my database to confirm the right thing happened.

 **Tips to customize:** Stripe's testing page (docs.stripe.com/testing) lists test cards for declines and for 3D Secure authentication. Use those instead of making up card numbers.

4.2 Switch from Test Keys to Live Keys

Use this when everything works in test mode and you're ready to accept real money. You get a careful checklist for going from test mode to real payments without breaking anything.

Fill in: [TECH STACK], [HOST], [WHAT I SET UP IN TEST MODE] (e.g. "one product and price, a webhook endpoint, Checkout").

I'm ready to switch my Stripe integration from test mode to live mode. Give me a careful, step-by-step checklist.

Tech stack: [TECH STACK]

Host: [HOST]

What I set up in test mode: [WHAT I SET UP IN TEST MODE]

Please cover:

1. Activating my Stripe account for live payments, if I haven't yet (point me to Stripe's dashboard rather than guessing the steps).
2. Recreating my products and prices in live mode, because test objects don't exist in live mode, and updating the price IDs in my code or environment variables.
3. Swapping my test API keys for live keys in my host's PRODUCTION environment only. Keep test keys in preview and development. Suggest a restricted key if it fits my use.
4. Registering a NEW live webhook endpoint with the same events as my test one, and adding its signing secret (different from the test one) to production.
5. Redeploying after changing environment variables, so the new values take effect.
6. Making a small real purchase myself, confirming the webhook delivered and access unlocked, then refunding it.
7. Following Stripe's go-live checklist (search "go-live checklist" on docs.stripe.com) and telling me anything on it I haven't covered.

Don't ask me to paste any keys into this chat.



Tips to customize: Before switching, ask: "Search my code for any hard-coded test keys, test price IDs, or localhost URLs." Ship It Live has more help with environment variables and redeployments.

Bonus: Magic Phrases for Payments

- "Only unlock access after the server confirms payment with Stripe."
- "Make this safe if the same event arrives twice."
- "Which mode is this for, test or live?"
- "Read the raw body. Don't parse it before the signature check."
- "Check docs.stripe.com and tell me if you're not sure."
- "Show me what to check in my database after this runs."

Get Paid

That's the whole Get Paid pack: **9 prompts across 4 sections**. Create the session on the server, confirm payment through the webhook, fulfill exactly once, and test everything with test cards before you switch on live keys.

Happy selling!