Safe retries across services

Key takeaways

1. A local database transaction cannot include a remote service’s changes.

2. Save intended actions before sending them and verified results as they arrive.

3. Recover unresolved work under each service’s retry guarantees.

Common misunderstandings of cross-service operations

Which claim would you defend right now? Choose before the evidence.

"Rolling back the database can undo a remote call made inside the transaction."

"A failed request means the service made no change."

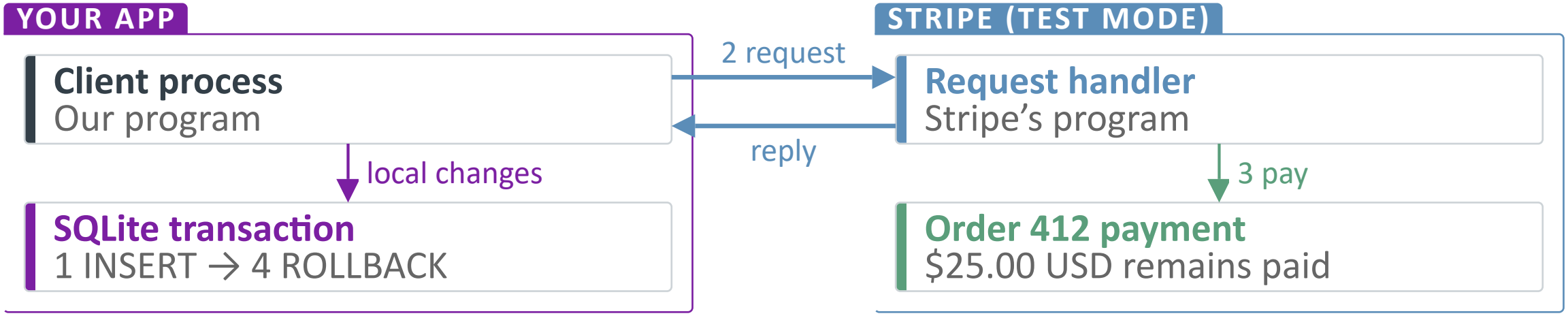
"Saving a request's identity makes it safe to repeat on any service."

Keep your choice. Look for the mechanism or counterexample that would make it fail.

The local transaction stops at SQLite

A SQLite transaction groups local database changes. Stripe owns the payment.

The client is our program; Stripe is the service. A request asks; a reply reports.



A timeout ends the client's local wait for a reply.

ROLLBACK removes the local row. The completed payment remains at Stripe.

Instapoll

Instapoll · Review · Nov 12 B

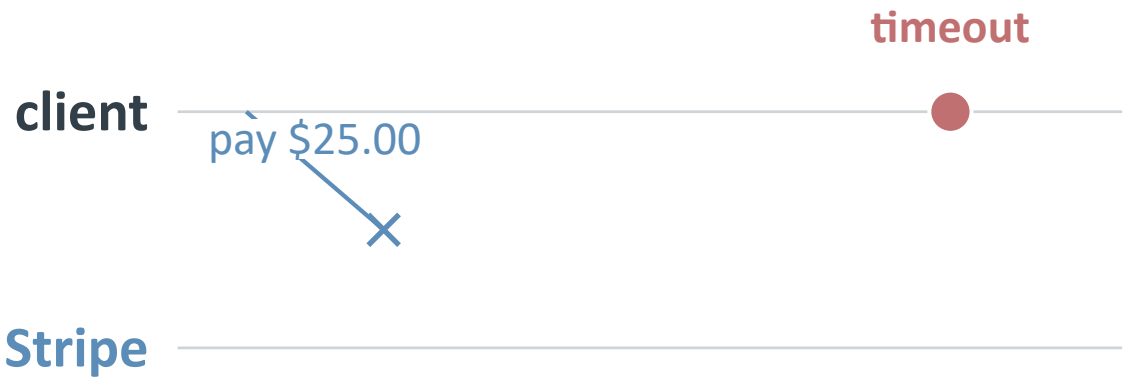
Your agent runs “gh pr create” and the call times out with no response at all. Was the pull request created?

- A. No — a timeout means the request never completed
- B. Yes — the server commits before it sends the response
- C. Unknowable from here; the server may or may not have committed it
- D. No — the client cancels the server’s work when the call times out

Did order 412's $25.00 payment happen?

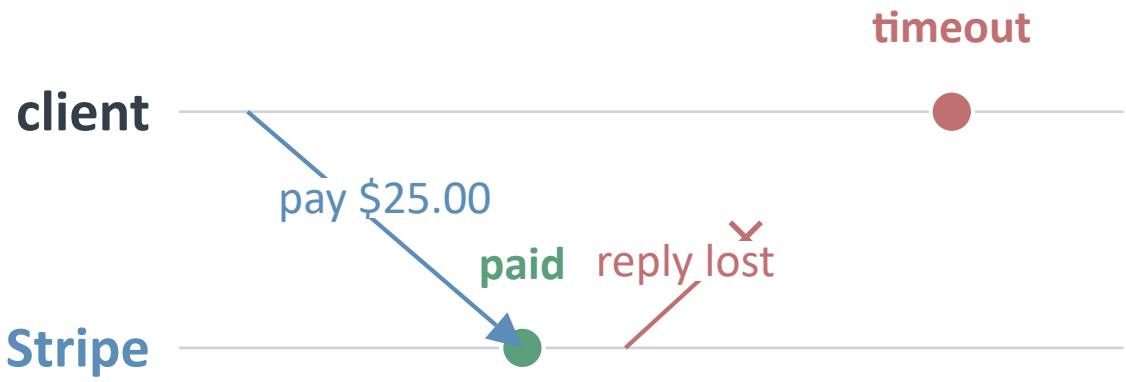
A timeout cannot distinguish these two histories. (Test-mode example.)

A: request never arrives



0 payments exist

B: payment succeeds

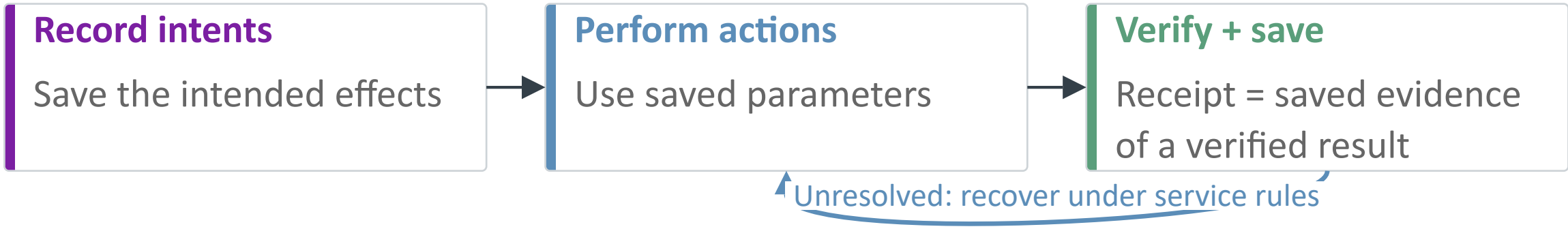


1 payment exists

Same timeout. Should the client ask for another payment?

Record intents, perform actions, verify results

An intent states an effect still to establish. Pending means unresolved locally.

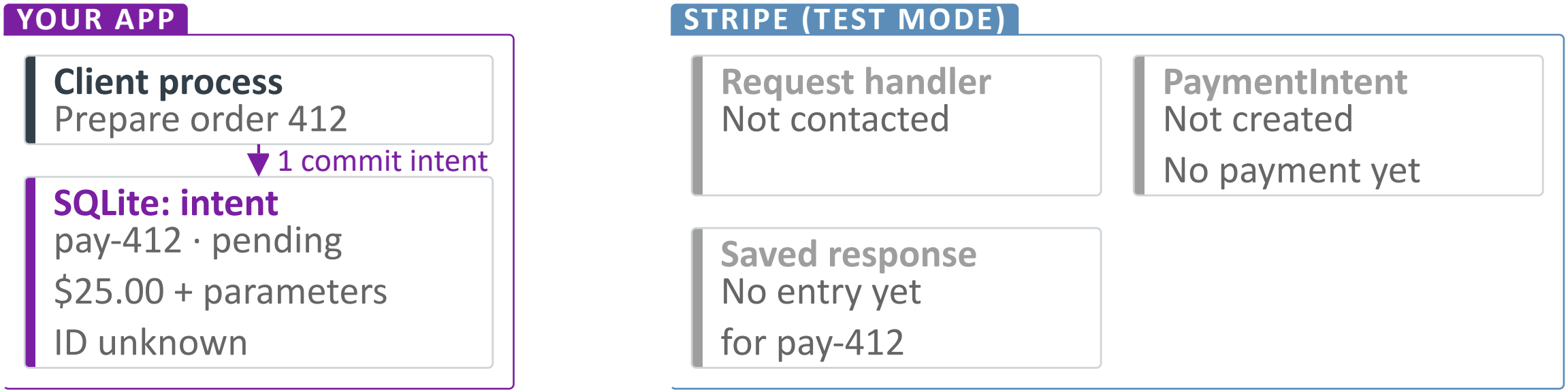


- Write all known intents. Save resolved parameters before each action’s first attempt.
- Wait for prerequisite results; save each verified receipt as soon as it is available.

Remote changes can finish separately. This method does not roll them back together.

Save the operation before sending it

An API is a service’s request interface. An idempotency key names one operation.
A PaymentIntent is Stripe’s payment record. Key pay-412 and returned ID pi_412 differ.

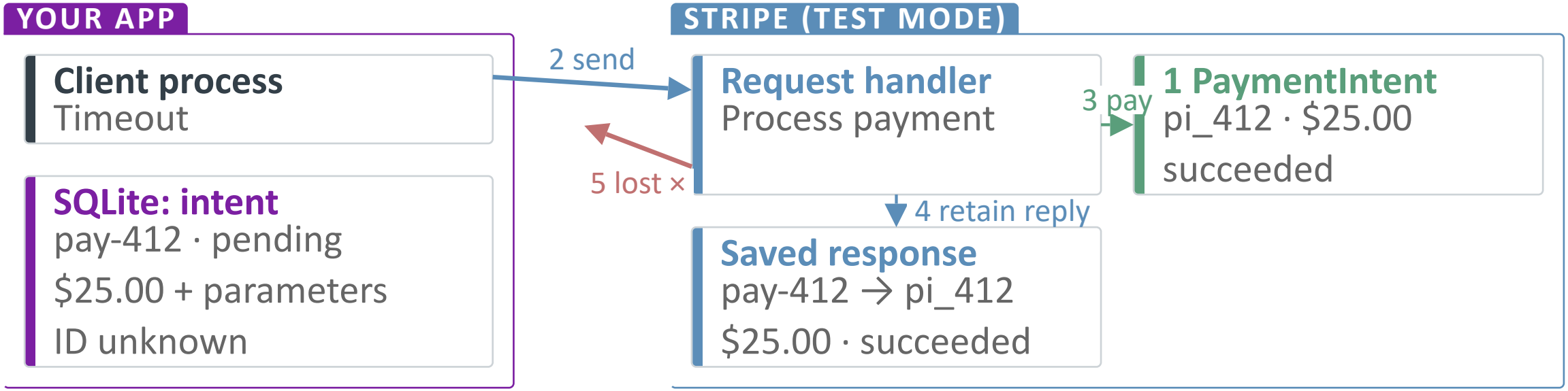


Save the original parameters: $25.00 USD, test card, and request to attempt payment.

source: Illustrative IDs; docs.stripe.com/api/payment_intents/create

The payment succeeds, but the reply is lost

The client sends pay-412 + saved parameters.
The payment succeeds; its reply is lost. The local intent stays pending.

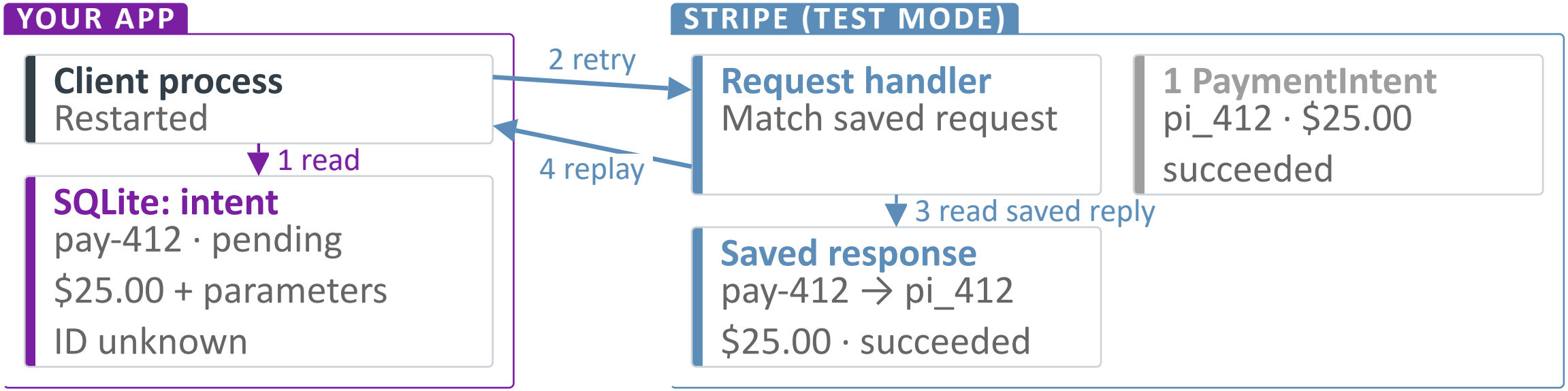


Stripe retains the reply under pay-412. This is the API contract, not its internal storage.

source: Stripe API v1; docs.stripe.com/api/idempotent_requests

Retry reads the result without paying again

Stripe API v1: same retained key + identical parameters returns the saved reply.
The retry does not execute the payment again.

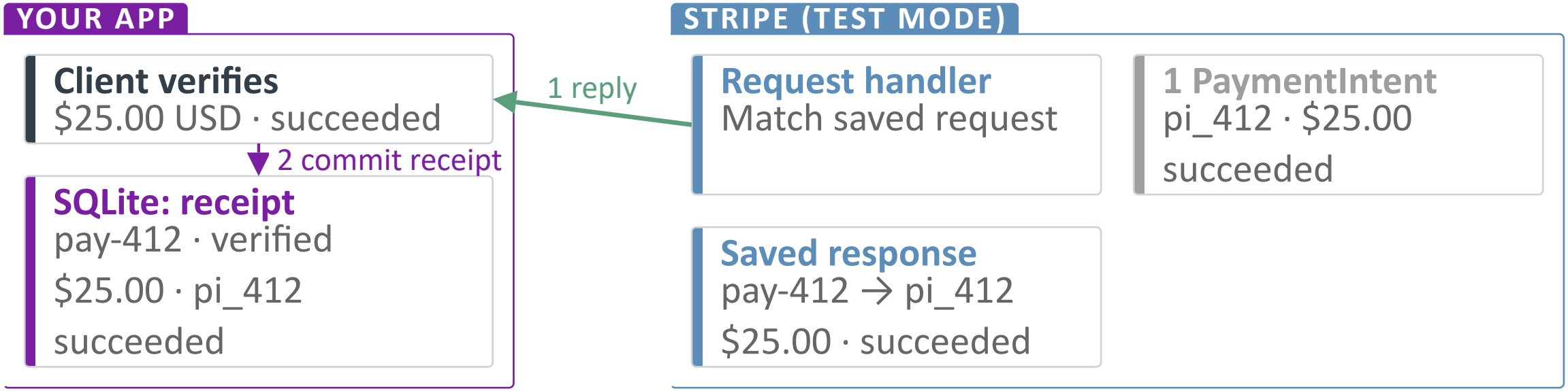


The first arrival executes. A new or discarded key can execute again; keys may expire after ≥24 h.
A replayed HTTP 500 (server error) leaves the payment unresolved.

source: docs.stripe.com/api/idempotent_requests; docs.stripe.com/error-low-level

Commit the verified payment receipt locally

Check the reply: amount $25.00, currency USD, status succeeded.
Commit pi_412 as the verified result for pay-412.

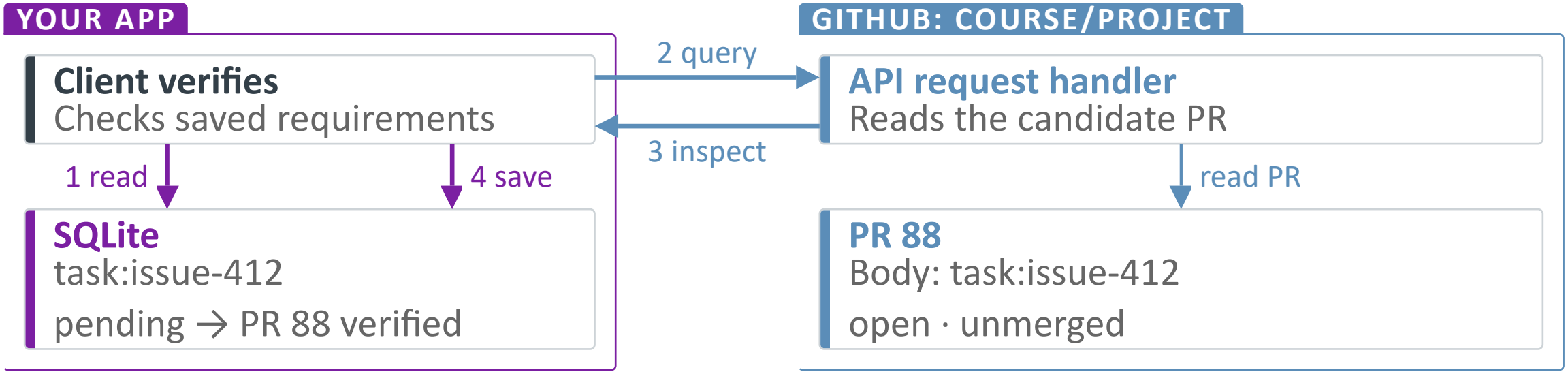


Durable means surviving restart. A crash before this commit leaves the intent pending.
With the key retained, restart and repeat recovery; a valid reply can supply the evidence.

source: docs.stripe.com/api/payment_intents/object; docs.stripe.com/api/idempotent_requests

Recover a GitHub PR by reading it

A marker is a saved label included in the PR body so our client can recognize it.




```
gh pr list --repo course/project --state all
gh api repos/course/project/pulls/88
```

source: cli.github.com/manual/gh_pr_list; docs.github.com/rest/pulls/pulls#get-a-pull-request

The service determines which retry is safe

GitHub does not enforce marker uniqueness. Stripe recognizes a retained request key.

	GitHub PR	Stripe API v1 payment
Saved identity	Marker in the PR body	Idempotency key
Recovery	Find and check an existing PR	Replay a retained request's result

Head = proposed branch; base = destination; full hash = head commit identity.

- Check marker, repository, head branch/full hash, base branch, and open/unmerged state.
- A list miss can hide an in-flight create or incomplete visibility; mismatches and multiple matches need a decision.

A missing match does not make another create safe.

source: docs.github.com/rest/pulls/pulls; docs.stripe.com/api/idempotent_requests

Track each action’s result separately

Order 412: verify payment before buying a shipping label. The label’s reply is lost.

YOUR APP

Client reads SQLite on restart

Intent	Saved result
Pay $25.00 USD	verified pi_412
Buy label	pending ID unknown

STRIPE

Payment pi_412
$25.00 USD
succeeded

SHIPPING SERVICE

Label L-73
Order 412
purchased
reply lost ×

Restart skips the verified payment; recover the label under the shipping service’s rules.

source: Illustrative workflow; the shipping service’s retry rules are not specified.

Key Takeaways

1. Save every known intent and each action’s resolved parameters before its first attempt.

2. Perform actions when their prerequisites are verified. Save each verified result as it becomes available.

3. Recover pending work under that service’s contract. The workflow is complete only when every required result is verified and saved.