Loamly: Pre-launch check

PREPARED FOR
Loamly (fictional)

PREPARED BY
BuildThenTested

TESTED ON / DELIVERED ON
2026-09-14 / 2026-09-16

BUILT WITH / STACK
lovable / supabase, stripe, vercel

A fictional plant-care subscription app: users log plants, get watering reminders, and pay a monthly plan through Stripe. Built with Lovable on Supabase, deployed on Vercel. Every name, path and finding in this report is invented to show the report format; the app does not exist.

Findings describe what we observed on the version you gave us access to, on the date of testing. Changes after that date are not covered until re-checked.

Summary for a non-technical reader

We tested Loamly the way a first wave of users and an opportunistic visitor would, using two test accounts and the public parts of the app. We found seven things worth fixing before launch: two critical, three high, two medium.

The two critical findings both let one user see or change another user's data. One is a database table with no access rules, so anyone with the app's public key can read every plant record. The other is a plan check that runs only in the browser, so a free user can reach paid features by editing one value.

Both critical fixes are small: turn on row-level rules for the plants table and move the plan check to the server. We describe the exact change for each finding and will re-test within 14 days of your fixes at no extra cost.

Nothing in this report means the app is finished being tested. It lists what we found on the version we tested, on the date we tested it, against the pre-launch checklist.

- **Critical: 2**
- **High: 3**
- **Medium: 2**

Findings by severity

ID	Severity	Finding	Affected path
F-01	critical	Row-level rules are off on the plants table	`Supabase table `public.plants`; REST endpoint `/rest/v1/plants``
F-02	critical	Paid plan enforced only in the browser	`Component `PremiumReminders.tsx`; Supabase Edge Function `send-reminders``
F-03	high	Stripe webhook accepts unsigned events	`Edge Function `stripe-webhook``
F-04	high	Admin page reachable by URL for any signed-in user	`Route `/admin`; table `profiles``
F-05	high	No rate limit on password reset	`Supabase Auth `/auth/v1/recover` via the app's reset form`
F-06	medium	Security headers missing on the production site	`All responses from `loamly.example``
F-07	medium	Spinner never stops when the reminders call fails	`Component `PremiumReminders.tsx`, function `handleSchedule``

SCOPE TESTED

- Secrets and keys: JavaScript bundle, public API responses, error messages
- Data access: every Supabase table the app touches, with the anon key and two signed-in roles
- Login and permissions: routes, roles, admin area, password reset
- Reliability: error handling, rate limits on public endpoints
- Deploy and config: headers, debug settings, environment variables reaching the browser

F-01: Row-level rules are off on the plants table

● Critical · pattern: rls-disabled-on-table

MECHANISM

The plants table was created without row-level security (RLS: per-row rules deciding who may read or change a row). The app's public key, which every visitor's browser holds, can therefore read and modify every row in the table.

AFFECTED PATH

`Supabase table public.plants; REST endpoint /rest/v1/plants`

WHAT WE OBSERVED

```
GET /rest/v1/plants?select=id,user_id,nickname,notes  (anon key, not signed in)
HTTP 200
[{"id":1,"user_id":"7c1e...","nickname":"Fern in the kitchen","notes":"Gift from mum"}, ... 1,284 rows]
```

FIX DIRECTION

Enable RLS on plants and add two policies: users may select rows where user_id = auth.uid(), and users may insert, update or delete rows where user_id = auth.uid(). Then repeat the request above; it should return an empty array.

RE-CHECK

We will repeat the anon and cross-user requests and confirm both return no rows.

F-02: Paid plan enforced only in the browser

● Critical · pattern: authorization-only-in-browser

MECHANISM

The premium reminder feature checks profile.plan === 'pro' in the React component and then calls the send-reminders function. The function does not check the plan itself, so a free user can call it directly.

AFFECTED PATH

`Component PremiumReminders.tsx; Supabase Edge Function send-reminders`

WHAT WE OBSERVED

```
POST /functions/v1/send-reminders  (free test account, valid session)
HTTP 200 {"scheduled": 12}
```

FIX DIRECTION

Read the plan from the subscriptions table inside the function using the caller's user id, and return HTTP 403 when the plan is not pro. Keep the browser check for the user interface only.

RE-CHECK

We will call the function with the free account and expect HTTP 403.

F-03: Stripe webhook accepts unsigned events

High · pattern: stripe-webhook-without-signature-verification

MECHANISM

The webhook handler parses the request body as JSON and updates subscriptions without calling Stripe's signature check, so anyone who knows the URL can post a fake checkout.session.completed event.

AFFECTED PATH

`Edge Function stripe-webhook`

WHAT WE OBSERVED

```
POST /functions/v1/stripe-webhook  (no Stripe-Signature header, hand-written event)
HTTP 200; subscriptions.plan for test user changed to 'pro'
```

FIX DIRECTION

Verify the Stripe-Signature header with your webhook signing secret before reading the event, and reject requests that fail with HTTP 400. Stripe's docs show the exact call for every SDK.

RE-CHECK

We will replay the unsigned event and expect HTTP 400 with no change to the table.

F-04: Admin page reachable by URL for any signed-in user

● High · pattern: admin-routes-without-auth

MECHANISM

The /admin route is hidden from the menu for non-admins but has no server-side role check. A signed-in free user who types the URL sees the user list. Data on the page is limited by RLS on profiles, so names and emails are visible but not editable.

AFFECTED PATH

`Route /admin; table profiles`

WHAT WE OBSERVED

```
GET /admin  (free test account)
HTTP 200, page renders 1,284 profile rows (name, email, plan)
```

FIX DIRECTION

Add a server-side check that the caller's role is admin before the page loads, and add an RLS policy so profiles returns only the caller's own row unless the caller is an admin.

RE-CHECK

We will open /admin with the free account and expect a redirect or HTTP 403, and query profiles expecting one row.

F-05: No rate limit on password reset

● High · pattern: no-rate-limiting

MECHANISM

The password reset endpoint accepted a burst of requests for the same address within one minute without slowing down or blocking. This allows email flooding of any user and makes reset-token guessing cheaper.

AFFECTED PATH

`Supabase Auth /auth/v1/recover via the app's reset form`

WHAT WE OBSERVED

`300 POST requests in 60 s from one IP: 300 x HTTP 200`

FIX DIRECTION

Enable Supabase Auth rate limits for email sending in the project dashboard and set a per-IP limit on the reset form at the edge (Vercel Firewall or middleware).

RE-CHECK

We will repeat the burst and expect HTTP 429 after the configured limit.

F-06: Security headers missing on the production site

● Medium · pattern: missing-security-headers

MECHANISM

Responses carry no Content-Security-Policy, X-Frame-Options or Strict-Transport-Security headers, so the site can be framed by another site and browsers do not enforce HTTPS-only connections.

AFFECTED PATH

`All responses from loamly.example`

WHAT WE OBSERVED

```
curl -sI https://loamly.example/ | grep -iE 'content-security|frame-options|strict-transport'
(no output)
```

FIX DIRECTION

Add the headers in next.config (or vercel.json) starting with X-Frame-Options: DENY and Strict-Transport-Security, then introduce a Content-Security-Policy in report-only mode and tighten it.

RE-CHECK

We will fetch the homepage headers and confirm the three are present.

F-07: Spinner never stops when the reminders call fails

● Medium · pattern: errors-swallowed-ui-spins-forever

MECHANISM

When send-reminders returns an error, the catch block logs to the console and never resets the loading state, so the user sees an endless spinner and often retries, scheduling duplicate reminders.

AFFECTED PATH

`Component PremiumReminders.tsx, function handleSchedule`

WHAT WE OBSERVED

`Blocked the function URL in devtools, clicked Schedule: spinner persisted 5 minutes; console shows 'TypeError: Failed to fetch'`

FIX DIRECTION

Reset the loading state in a finally block and show a short message with a retry button. Make the function idempotent per user and day so a retry cannot double-schedule.

RE-CHECK

We will repeat the blocked call and expect a visible error state within a few seconds.

Tested and not found, and next steps

CHECKED AND NOT FOUND ON THE TESTED VERSION

- No Supabase service-role key or Stripe secret key in the JavaScript bundle or in any response we could reach
- No secrets in the public GitHub repository history that was shared with us
- Password reset tokens expired after use
- Debug mode off in production; environment variables reaching the browser were limited to the public Supabase URL and anon key
- Payment amounts are taken from the server-side price id, not from the browser

NEXT STEPS

1. Fix F-01 and F-02 first; both are small changes in the Supabase dashboard and one function
2. Fix F-03 before enabling live payments
3. Tell us when the fixes are deployed; we re-test the critical and high findings within 14 days, included in the package
4. Consider the ready-to-charge check before switching Stripe to live mode

We test against a defined checklist and report what we find. We do not guarantee that an app is free of vulnerabilities. We are not a certified penetration-testing firm and do not issue compliance attestations. We do not rebuild your app; fixes are yours or your developer's, and we re-check them.