App **Tallyroom** Delivered **5 November 2026** Audit **a-sample-01**

SAMPLE Sample report. The app, its code and every finding are invented to show the format; no real client is described.

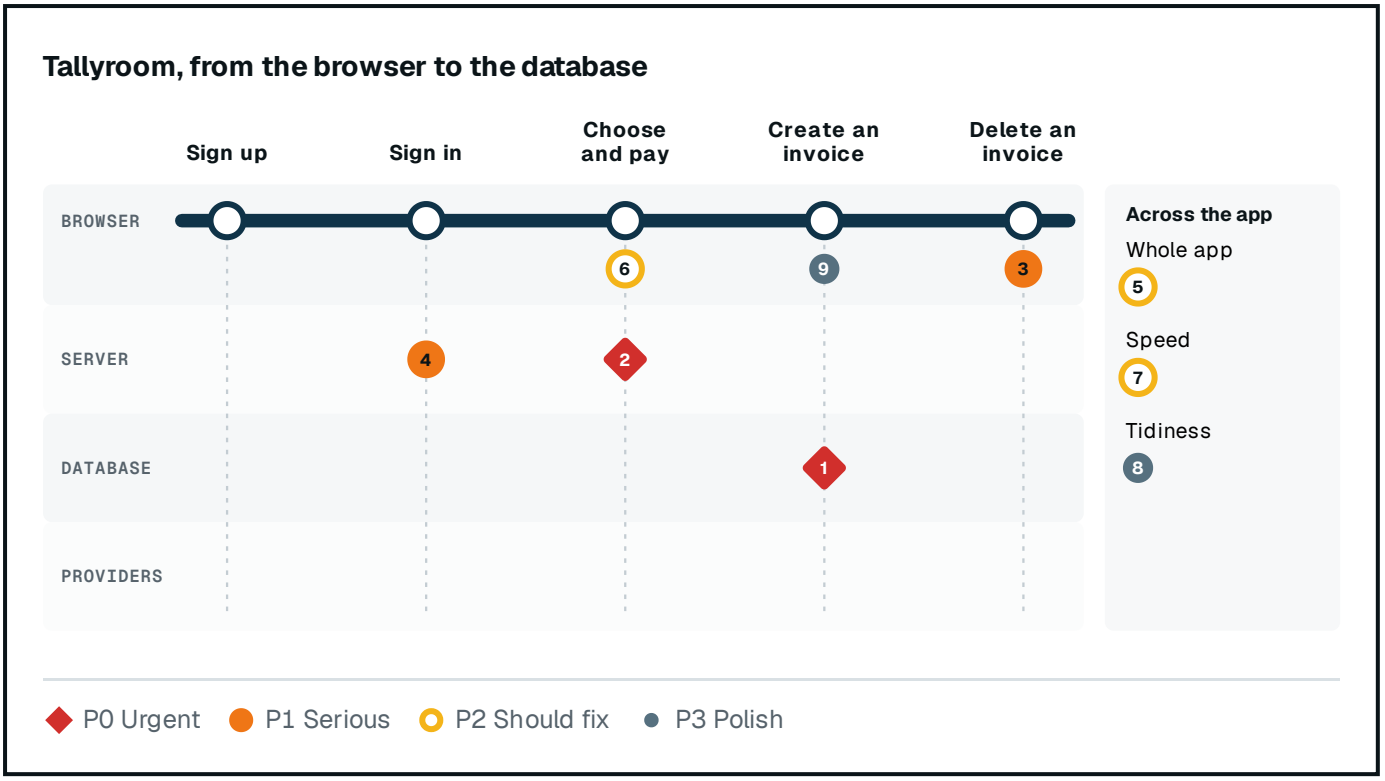
# Tallyroom works, and two problems need fixing this week.

Any visitor can read every customer's invoices without signing in, and the checkout lets the browser set the price. Both are quick to fix. The rest of the app is in reasonable shape: sign-up, invoicing and payments work end to end, and nothing leaks your secret keys.

| 2 | 2 | 3 | 1 |
|---|---|---|---|
| **Urgent** | **Serious** | **Should fix** | **Polish** |
| Act now | Fix within weeks | Quality and upkeep | When convenient |

# Where the findings sit

Choose a marker to read its finding. Numbers match the chapters below.




## Fix these this week

**1** **Anyone can read every customer's invoices without signing in**

P0 Urgent · Under a day

**2** **A customer can change the price they pay at checkout**

P0 Urgent · Under a day

**3** **The Delete invoice button does nothing**

P1 Serious · Under a day

# What is solid

Reported as clearly as what is broken: secrets, payments and the main flows held under every check that touched them.

- No secret keys reach the browser, and the git history is clean.
- Stripe webhooks are verified and cannot be replayed to grant access twice.
- Sign-up, sign-in and paying an invoice all work at desktop and phone width.
- Every money amount is stored as whole pence, so totals never drift.

# Urgent

**2 findings**

Two ways in: anyone can read invoices, and the browser sets the price. Close both this week.

## 1

**P0 Urgent**

### Anyone can read every customer's invoices without signing in

**Create an invoice, database** | Effort: **Under a day** | Fix in: **week 1**

Check `SEC-RLS-01` `a-sample-01-F-001`

The invoices table has no access rules, so a stranger with the public app key can download every invoice, client name and amount. That is a data breach you would have to report.

#### Evidence

**supabase/migrations/20260811_invoices.sql:14** — cited line highlighted

```
14  create table public.invoices (
15    id uuid primary key default gen_random_uuid(),
16    owner_id uuid references auth.users not null,
```

**policies/export.sql:88** — cited line highlighted

```
88  -- invoices: row level security DISABLED
```

#### Fix

Turn on row-level security for invoices and add policies so each signed-in user reads and writes only rows where owner_id is their own id.

**Fix prompt**

Paste into Lovable, Cursor, Bolt, Claude Code or any coding agent

```
In Supabase, enable and force row-level security on the public.invoices table. Add select, insert, update and delete policies that allow a row only when owner_id = auth.uid(). Before changing anything, list any page or function that reads invoices without a signed-in user, and tell me; then show the before and after policies and confirm signed-in users still see their own invoices.
```

## 2

**P0 Urgent**

### A customer can change the price they pay at checkout

**Choose and pay, server** | Effort: **Under a day** | Fix in: **week 1**

Check `SEC-PRICE-01` `a-sample-01-F-002`

The checkout takes the amount from the browser, so anyone who edits the request can pay one pence for a plan that costs forty pounds. You lose money on every such sign-up.

#### Evidence

**supabase/functions/create-checkout/index.ts:21** cited line highlighted

```
21  const { priceAmount, planId } = await req.json()
22  const session = await stripe.checkout.sessions.create({
23    line_items: [{ price_data: { unit_amount: priceAmount,
```

#### Fix

Look up the price on the server from the plan id, and never read an amount from the request. Add an idempotency key to the session call.

**Fix prompt**

Paste into Lovable, Cursor, Bolt, Claude Code or any coding agent

```
In supabase/functions/create-checkout/index.ts, stop reading priceAmount from the request body. Look up the Stripe price for planId on the server (from a plans table or Stripe price ids), pass it as price, and add an idempotency key built from the user id and plan id. Keep the current checkout flow working for every existing plan.
```

# Serious

**2 findings**

A dead button your customers have already met, and nothing stopping password guessing.

## 3

**P1 Serious**

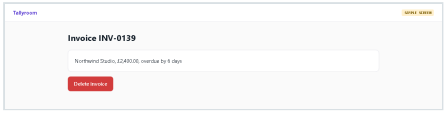
### The Delete invoice button does nothing

**Delete an invoice, browser** | Effort: **Under a day** | Fix in: **week 1**

Check `PROD-STUB-01` `a-sample-01-F-003`

Users click it, nothing happens, and the invoice stays. They lose trust and email you instead, which costs you time on every account.

#### Evidence




https://app.tallyroom.example/invoices/demo ↗
Clicked Delete invoice at 1280 wide: no request sent, no message, the invoice remains.

**src/pages/Invoice.tsx:142** cited line highlighted

```
  142  <Button variant="destructive" onClick={() => {}}>Delete invoice</Button>
```

#### Fix

Wire the button to a delete call with a confirmation step, then return to the invoice list with a message.

**Fix prompt**

Paste into Lovable, Cursor, Bolt, Claude Code or any coding agent

```
In src/pages/Invoice.tsx, make the Delete invoice button open a confirmation dialog, then delete the invoice through Supabase (respecting the new row-level security), show a short confirmation message, and return to the invoices list. Do not change any other button.
```

# 4

**P1 Serious**

## Nothing slows down someone guessing passwords

**Sign in, server** | Effort: **One to three days** | Fix in: **week 2**

Check `SEC-RATE-01` `a-sample-01-F-004`

An attacker can try thousands of passwords against one account without being stopped. Accounts with weak passwords will eventually be taken over.

### Evidence

**src/lib/auth.ts:9** cited line highlighted

```
 9  export const signIn = (email, password) =>
10    supabase.auth.signInWithPassword({ email, password })
```

### Fix

Turn on Supabase's auth rate limits and add a growing cooldown after failed attempts, by IP and account together.

**Fix prompt**

Paste into Lovable, Cursor, Bolt, Claude Code or any coding agent

```
Add brute-force protection to sign-in: enable Supabase auth rate limits
in the dashboard, and add an increasing cooldown after failed attempts,
throttled by IP and account together, so nobody can lock a known user out
on purpose. Show the user a clear message while the cooldown runs.
```

# Should fix

**3** findings

Quality and upkeep: errors nobody hears about, a pricing page that breaks on phones, a slow first page.

## 5

**P2 Should fix**

### When the app breaks, nobody finds out

Across the app: whole app | Effort: **Under a day** | Fix in: **week 2**

Check `SHIP-ERRTRACK-01` `a-sample-01-F-005`

Errors that users hit are not recorded anywhere, so you learn about problems only when someone complains, or leaves.

#### Evidence

**src/main.tsx:1** — cited line highlighted

```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App'
```

#### Fix

Add an error tracker (Sentry has a free tier) to the front end and the Supabase functions.

**Fix prompt**

Paste into Lovable, Cursor, Bolt, Claude Code or any coding agent

```
Add Sentry error tracking to this app: initialise it in src/main.tsx for the front end and in each Supabase edge function, using environment variables for the key. Do not change any app behaviour. Tell me how to see a test error arrive.
```

# 6

**P2 Should fix**

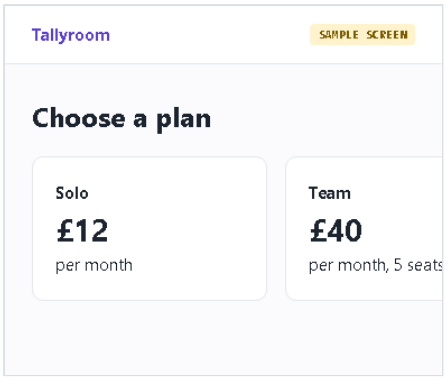
## The pricing table runs off the screen on a phone

**Choose and pay, browser** | Effort: **Under a day** | Fix in: **week 2**

Check `PROD-MOBILE-01` `a-sample-01-F-006`

On a phone the plans scroll sideways and the price of the middle plan is cut off, so visitors on mobile are less likely to sign up.

### Evidence




https://app.tallyroom.example/pricing ↗

At 390 wide the page is 612 pixels wide; the Team plan price is outside the screen.

### Fix

Stack the plans vertically below 640 pixels wide.

**Fix prompt**

Paste into Lovable, Cursor, Bolt, Claude Code or any coding agent

```
On the pricing page, make the plans stack vertically on screens narrower than 640 pixels, with no sideways scroll, and keep the recommended plan first. Check it at 390 wide.
```

7

P2 Should fix

## The first page downloads more code than it needs

Across the app: speed | Effort: **One to three days** | Fix in: **week 3**

Check `PERF-CODESPLIT-01` `a-sample-01-F-007`

Visitors wait about four seconds on a phone before the landing page responds, because the whole charting library loads up front. Slow first pages lose sign-ups.

### Evidence

**scans/lighthouse-landing.json** — scanner output

```
1  "total-byte-weight": 1.34 MB of JavaScript before first interaction
```

### Fix

Load the charts only on the pages that show them.

**Fix prompt**

Paste into Lovable, Cursor, Bolt, Claude Code or any coding agent

```
The landing page loads the charting library although it shows no charts. Load the chart components with a dynamic import only on the dashboard and reports pages, and tell me the JavaScript size of the landing page before and after.
```

# Polish

**1 finding**

Polish: leftover sample pages from the builder that make the app look unfinished.

## 8

P3 Polish

### Leftover sample pages from the builder are still live

Across the app: tidiness | Effort: **Under a day** | Fix in: **week 3**

Check `SHIP-BUILDERLEFTOVER-01` `a-sample-01-F-008`

A demo page with fake customers is reachable at /examples. It looks unfinished to anyone who finds it.

#### Evidence

https://app.tallyroom.example/examples ↗
Page loads with the builder's demo data.

**src/pages/Examples.tsx:1** cited line highlighted

```
1  // Generated example page
```

#### Fix

Delete the examples page and its route.

**Fix prompt**
Paste into Lovable, Cursor, Bolt, Claude Code or any coding agent

```
Remove the builder's example page (src/pages/Examples.tsx) and its
/examples route. Make sure no navigation links point to it.
```

# Make it easier to use

**1 finding**

Not defects: one change that would make sending an invoice feel finished, with the reason behind it.

## 9

P3 Polish

### Sending an invoice ends on a toast that disappears

Create an invoice, browser | Effort: **Under a day** | Fix in: **week 3**

Based on: **A likely cost** | UX law: **Peak-End Rule** | Check `PROD-EASE-01`

`a-sample-01-F-009`

After Send, a two-second message says Sent and the page goes back to the invoice list. There is no record on the invoice of when it went or to which address, so a user who looks away has to open their sent mail to check, or sends it twice.

#### What this rests on

Sending is the moment the app earns its keep. When it ends without a lasting confirmation, users check by hand or send the same invoice again.

#### Evidence

https://app.tallyroom.example/invoices/INV-0142 ↗
After Send, a toast shows for about two seconds, then the list loads. The invoice page shows no sent date or recipient.

#### Fix

End the send on a confirmation that stays: the invoice shows Sent, the date and time, and the address it went to, with a way to resend.

**Fix prompt**

Paste into Lovable, Cursor, Bolt, Claude Code or any coding agent

```
When an invoice is sent, keep the user on the invoice and show a lasting
confirmation: Sent, the date and time, and the recipient's email, with a
Resend button. Store the sent time and recipient on the invoice so the
```

```
confirmation is still there after a reload. Do not change how the email
itself is sent.
```

# Functionality and usability

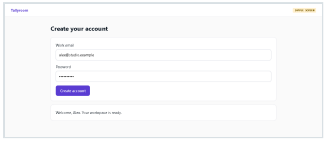
Sign-up works everywhere; choosing a plan and sending an invoice struggle on a phone; deleting an invoice is broken.

## Sign up

**Desktop:** Account created, welcome screen shown.

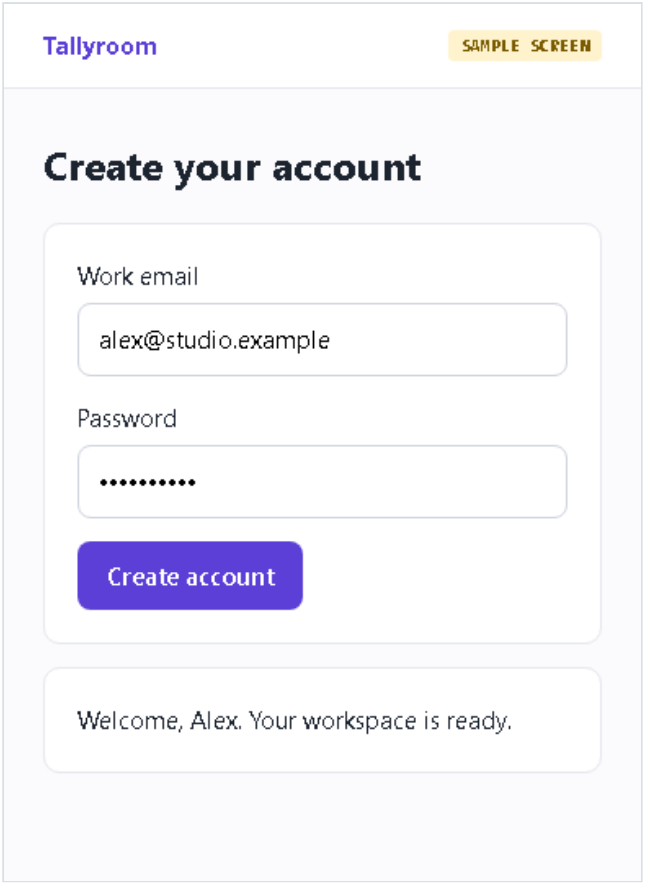
**Phone:** Works; the form fits the screen.

Checks: `PROD-FLOW-01`



Desktop, 1280px

**Works**




Phone, 390px

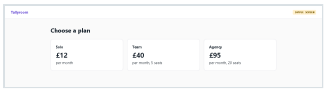
**Works**

## Choose a plan

**Desktop:** Plans readable side by side.

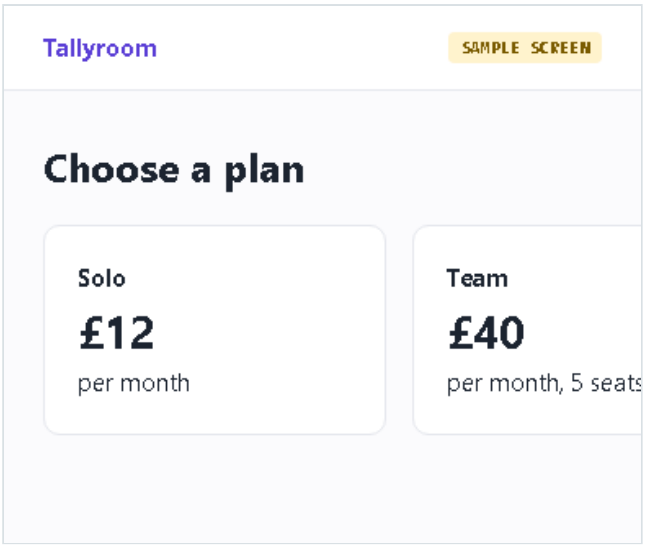
**Phone:** Table runs off the screen; see finding F-006.

Checks: `PROD-MOBILE-01`



Desktop, 1280px

Works




Phone, 390px

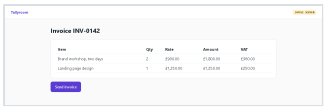
Broken

## Create and send an invoice

**Desktop:** Created and sent in under a minute.

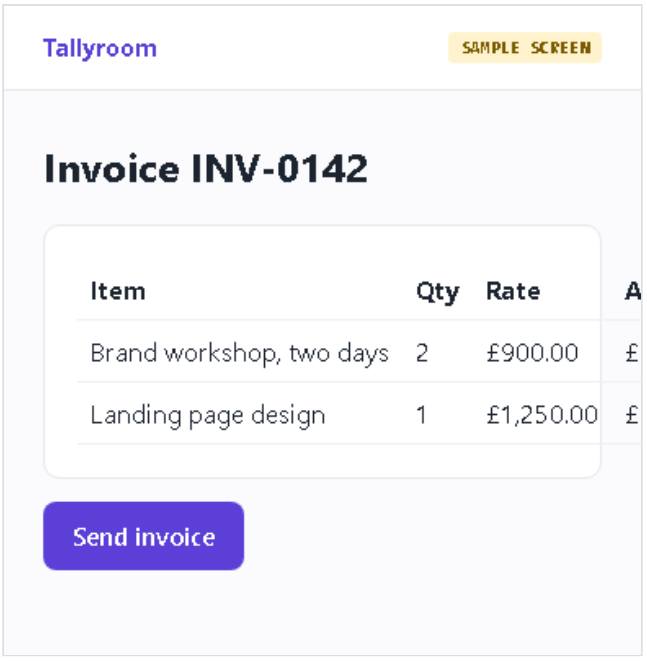
**Phone:** Works, but the line-item table needs sideways scrolling.

Checks: `PROD-FLOW-04`



Desktop, 1280px

**Works**




Phone, 390px

**Works with problems**

## Delete an invoice

**Desktop:** Button does nothing; see finding F-003.

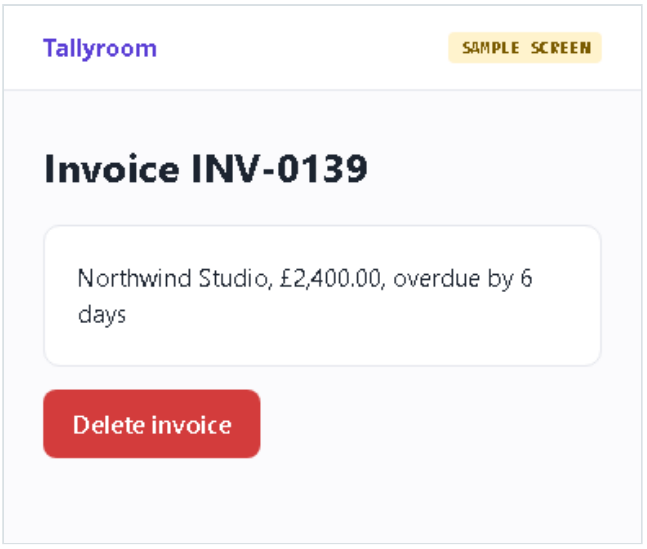
**Phone:** Same as desktop.

Checks: `PROD-STUB-01`



Desktop, 1280px

**Broken**




Phone, 390px

**Broken**

# Fix plan

Three weeks, in an order where each fix clears the way for the next.

## This week

Close the two ways in: data readable without login, and the price set by the browser. Then the dead button.

- 1. Anyone can read every customer's invoices without signing in
- 2. A customer can change the price they pay at checkout
- 3. The Delete invoice button does nothing

## Next week

Stop password guessing, start hearing about errors, fix the phone pricing page.

- 4. Nothing slows down someone guessing passwords
- 5. When the app breaks, nobody finds out
- 6. The pricing table runs off the screen on a phone

## When convenient

Speed, tidy-up, and a clearer end to sending an invoice.

- 7. The first page downloads more code than it needs
- 8. Leftover sample pages from the builder are still live
- 9. Sending an invoice ends on a toast that disappears

# Every check, one mark each

| | | |
|---|---|---|
| **Security** |  | 15 of 20 passed |
| **Data** |  | 7 of 8 passed |
| **Reliability** |  | 6 of 9 passed |
| **Performance** |  | 4 of 7 passed |
| **Shipping and monitoring** |  | 3 of 6 passed |
| **AI features** |  | Not applicable |
| **Functionality and usability** |  | 17 of 26 passed |

Passed · Partly · Failed · Not checked · Not applicable

Graded against the Qodeup Check Catalogue v1.4: 76 of 90 checks apply to this app.

90 of 90 checks shown

| Check | Area | What it asks | Result |
|---|---|---|---|
| `SEC-AUTHZ-01` | Security | Does every server-side write take the caller's identity from the verified session or token, rather than from a user id the client sends in the request body or a hidden form field? | Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| SEC-AUTHZ-02 | Security | Does every endpoint that takes a record id check that the calling account is actually allowed to reach that specific record, not just that the caller is logged in? | Passed |
| SEC-CORS-01 | Security | Does every API that requires authentication answer CORS with an explicit origin list, rather than a wildcard? | Passed |
| SEC-CSP-01 | Security | Does the app send a Content-Security-Policy header, rather than relying only on the framework's default HTML escaping to stop injected scripts? | Partly |
| SEC-DEPS-01 | Security | Does the app's dependency tree carry any known-exploited, critical or high-severity vulnerability in a package the running app actually uses on a reachable code path? | Passed |
| SEC-KEYS-01 | Security | Is the service-role key, or any other database or admin secret key (a private API token that isn't a model-provider key), kept out of every file the browser bundle can reach, including public env var prefixes? | Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| SEC-KEYS-02 | Security | Does the app refuse to start, rather than fall back to a hardcoded or generated default, when a required secret key is missing from the environment? | Passed |
| SEC-RATE-01 | Security | Does the login endpoint specifically lock out or slow down repeated failed attempts, rather than relying only on a generic site-wide rate limit? | Failed |
| SEC-RATE-02 | Security | Are public, unauthenticated write endpoints (contact forms, sign-up, password reset, comment posting) rate limited and returning 429 once the limit is hit? | Passed |
| SEC-REDIRECT-01 | Security | Is the redirect target after login, logout or an auth callback checked against an allowlist of the app's own paths, rather than built directly from a client-supplied parameter? | Passed |
| SEC-RLS-01 | Security | Does every table in an exposed schema have row-level security turned on and forced, with a policy for each operation the app actually performs? | Failed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| SEC-RLS-02 | Security | Do the row-level security policies scope rows to the record's owner, rather than granting access to anyone who is simply signed in? | Partly |
| SEC-SECRET-01 | Security | Is the working tree and git history free of committed secrets, keys and database dumps? | Passed |
| SEC-SESSION-01 | Security | Are session cookies set with HttpOnly, Secure and an appropriate SameSite value? | Passed |
| SEC-CALLBACK-01 | Security | Does the sign-in callback (OAuth, or an emailed sign-in link) check the provider's code and state, or the link's token, on the server before it creates a session, instead of trusting whatever the browser sends back? | Passed |
| SEC-CSRF-01 | Security | Are state-changing requests authenticated by a cookie protected by a CSRF token, a SameSite setting that blocks cross-site posts, or an Origin check, rather than the cookie alone? | Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| SEC-PRICE-01 | Security | Does the server decide the amount, currency and product for every charge, with nothing the browser sends able to change the price, and does charge creation carry an idempotency key so a retry cannot charge twice? | ■ Failed |
| SEC-SESSION-02 | Security | Does a session end on logout, expire on its own, and get revoked when the password changes, rather than staying valid indefinitely on an old token? | ■ Passed |
| SEC-STORAGE-01 | Security | Are storage buckets private by default, with upload and read policies scoped so a user only reaches their own files, and public buckets used only where the content is genuinely meant to be public? | ☐ Not applicable |
| SEC-VIEWS-01 | Security | Do database views and SECURITY DEFINER functions carry a stated reason and their own access check, rather than silently running with wider rights and bypassing row-level security underneath? | ■ Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| SEC-WEBHOOK-01 | Security | Does every server function correctly classify its caller (a signed-in user, a provider webhook, or a scheduled job) and apply the matching check, instead of one shared authentication path for all three? | Passed |
| DATA-BACKUP-01 | Data | Are automated backups running, and has a restore actually been performed and timed at least once? | Passed |
| DATA-MIGRATION-01 | Data | Are all schema changes made through versioned migration files, rather than edited directly in a dashboard? | Passed |
| DATA-PII-01 | Data | Is the personal data this app collects classified somewhere written down: what is collected and why? | Partly |
| DATA-RETENTION-01 | Data | Is there a stated retention and deletion policy, and can a delete request actually be honoured against the database, not just the UI? | Passed |
| DATA-TIME-01 | Data | Is every timestamp stored in UTC and converted to the viewer's local time only for display? | Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| DATA-TXN-01 | Data | Do multi-step writes that must all succeed together (checkout, account deletion, a role change) run inside one transaction or RPC call, rather than as several sequential client calls? | Passed |
| DATA-CONFIG-01 | Data | Is configuration that lives only in a dashboard (auth providers, redirect URLs, custom SMTP, storage limits, scheduled job schedules) written down somewhere the next session or audit can see? | Passed |
| DATA-MONEY-01 | Data | Is every monetary amount stored as an integer in the currency's smallest unit, rather than as a float or decimal that gets formatted at storage time? | Passed |
| REL-ENV-01 | Reliability | Does a complete .env.example exist, and does every key in it actually exist in the production deployment target, not just locally? | Passed |
| REL-ENVSPLIT-01 | Reliability | Do staging and production use separate databases and separate provider keys, rather than sharing one? | Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| REL-ERRHANDLE-01 | Reliability | Where a failure is caught, is it logged and surfaced with a safe message, or is the catch block empty and the failure disappears? | ■ Failed |
| REL-KILLSWITCH-01 | Reliability | Can any feature that spends money, calls an AI model, or writes to the database be turned off instantly without a code deploy? | ■ Passed |
| REL-RETRY-01 | Reliability | Do transient failures on safe-to-repeat outbound calls get retried with backoff, instead of failing the whole request on the first blip? | ■ Partly |
| REL-TIMEOUT-01 | Reliability | Does every outbound call the app makes (fetch, an API client, a database driver, a third-party SDK) carry an explicit timeout? | ■ Partly |
| REL-BGJOB-01 | Reliability | Does slow work (sending an email, calling a third-party API, processing an upload) run inline in the request, or is it handed off to a queue so the user isn't stuck waiting? | ☐ Not applicable |

| Check | Area | What it asks | Result |
|---|---|---|---|
| REL-CIRCUIT-01 | Reliability | When a critical upstream (payment provider, email sender, the app's own backend service) goes down, does the app stop hammering it, or does every page load keep firing requests at a dead service? | ☐ Not applicable |
| REL-CONNPOOL-01 | Reliability | Is the app's total database connection use, counted across every concurrent build, deploy preview and serverless worker, budgeted against the pooler's project-wide cap? | ■ Passed |
| REL-RACE-01 | Reliability | Where two requests can update the same record at once (a stock count, a role, a balance, or an 'at least one must remain' rule like the last admin or last owner), is the update atomic and run inside one locked transaction, rather than a read followed by a separate write? | ■ Passed |
| REL-WEBHOOK-01 | Reliability | Does every inbound webhook verify the sender's signature against the raw body, and does replaying the same event twice change nothing the second time? | ■ Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| PERF-CACHE-01 | Performance | Are expensive, repeated reads cached with an explicit TTL, and does a write to the underlying data actually invalidate the cached version? | Partly |
| PERF-CODESPLIT-01 | Performance | Does the first screen ship only the JavaScript it needs, or is a heavy component (a chart library, a rich editor, an admin-only feature) bundled into every visitor's initial load? | Failed |
| PERF-COMPRESS-01 | Performance | Are responses and static assets actually served compressed and with correct cache headers, or is that left to whatever the platform happens to default to? | Passed |
| PERF-IMAGE-01 | Performance | Are images served in a modern format, sized for their container, and lazy-loaded below the fold? | Partly |
| PERF-PAGINATE-01 | Performance | Do list endpoints and pages (a blog index, a product catalog, a search result set) load a bounded page of results, or do they return everything at once? | Passed |
| PERF-INDEX-01 | Performance | Are the columns the app actually filters, sorts, or joins on covered by an index? | Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| PERF-NQUERY-01 | Performance | When the app renders a list with related data (posts with authors, orders with line items), does it fetch the related data in one batched query, or does it fire one query per row? | Passed |
| SHIP-BUILDERLEFTOVER-01 | Shipping and monitoring | Does the repo carry leftover scaffolding, sample data, or duplicate instruction files from the AI builder that made it (a demo route, seed data still live, conflicting CLAUDE.md/cursorrules/AGENTS.md files that disagree with each other)? | Failed |
| SHIP-CI-01 | Shipping and monitoring | Does the gate before deploy run the build, a type check, lint and the test suite, and block the deploy when any of them fails? | Passed |
| SHIP-ERRTRACK-01 | Shipping and monitoring | Do exceptions from every part of the app (frontend, backend, background jobs) land in one place with user and request context, or does the operator find out about bugs only when a user reports one? | Failed |
| SHIP-LOCKFILE-01 | Shipping and monitoring | Is a lockfile committed and used, so the exact dependency versions installed locally match what the build server installs? | Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| SHIP-LOGS-01 | Shipping and monitoring | Are logs structured (JSON, with a request id) so a specific failed request can be traced, or is it free-form text nobody can search? | ■ Passed |
| SHIP-TEST-01 | Shipping and monitoring | Do the app's critical paths (checkout, auth, contact/signup) have automated tests that run on every change, or is verification manual only? | ■ Partly |
| SHIP-DEPLOYSURFACE-01 | Shipping and monitoring | When the app has more than one deployable surface (a web app, a separate admin backend, database migrations, scheduled jobs), is it documented which of them a push actually redeploys automatically? | ☐ Not applicable |
| AI-DATA-01 | AI features | Is what user or tenant data reaches the model provider known and kept to the minimum needed, is the provider's data retention setting stated, and does retrieval return only the caller's own tenant's documents? | ☐ Not applicable |
| AI-EVAL-01 | AI features | Is there an eval set the AI feature's prompt or model gets checked against, or is 'it looked better' the entire test suite for every prompt change? | ☐ Not applicable |

| Check | Area | What it asks | Result |
|---|---|---|---|
| AI-INJECT-01 | AI features | Everywhere untrusted text reaches the model (a user message, an uploaded file, a fetched page, another user's record), what stops an instruction hidden in that text from being followed? | ☐ Not applicable |
| AI-KEY-01 | AI features | Does the AI provider's API key ever reach the browser, in client-side code, a public env var, or a network response? | ☐ Not applicable |
| AI-MODELPIN-01 | AI features | Is the model version pinned to a specific identifier, or does the code point at a floating alias that can change underneath the app? | ☐ Not applicable |
| AI-OUTPUTCHECK-01 | AI features | Before the model's output is saved, shown, or acted on, does anything check its shape and content, or does it flow straight through? | ☐ Not applicable |
| AI-RATELIMIT-01 | AI features | Does a paid AI route limit how often a single user can call it, separately from the spend cap, so one user can't monopolise it or trigger abuse patterns even while staying under the money ceiling? | ☐ Not applicable |

| Check | Area | What it asks | Result |
|---|---|---|---|
| AI-SPENDCAP-01 | AI features | Does the AI feature have a per-user and a global spend cap that hard-stops the calls, plus a billing alert on the provider account, or can it run up an unbounded bill? | ☐ Not applicable |
| PROD-A11Y-01 | Functionality and usability | Can the sign-up or main feature flow be completed using only the keyboard, with no mouse? | ■ Passed |
| PROD-A11Y-02 | Functionality and usability | Does every input on the sign-up, sign-in, and main data-entry forms have a label a screen reader can announce? | ■ Passed |
| PROD-A11Y-03 | Functionality and usability | Does the body text and the primary call-to-action button meet at least 4.5:1 contrast against their background (3:1 for large text), as rendered on the page? | ■ Partly |
| PROD-A11Y-04 | Functionality and usability | Is there a visible focus indicator on every interactive element when navigating by keyboard? | ■ Passed |
| PROD-A11Y-05 | Functionality and usability | When text is enlarged to 200 percent, can the landing page, the sign-up form and the main feature screen still be read and used, with nothing cut off or overlapping? | ■ Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| PROD-A11Y-06 | Functionality and usability | At 320 CSS pixels wide, does the content reflow into one column, so nothing needs sideways scrolling to read or use? | Failed |
| PROD-EASE-01 | Functionality and usability | Is the core flow easy to use as well as working, judged against the UX laws index, with a cost or a standard behind every finding? | Partly |
| PROD-DECEPTIVE-01 | Functionality and usability | Does the product work against the person using it anywhere, such as a box ticked before they see it, a cost that appears only at the last step, a decline written to shame, cancelling harder than signing up, or urgency that is not true? | Passed |
| PROD-CONSOLE-01 | Functionality and usability | Does the browser console stay free of errors (not warnings) on the landing page, the sign-up or sign-in screen, and the main feature screen? | Passed |
| PROD-FLOW-04 | Functionality and usability | Can the app's one main feature, the thing the product exists to do, be completed end to end at desktop and phone width? | Partly |

| Check | Area | What it asks | Result |
|---|---|---|---|
| PROD-MOBILE-01 | Functionality and usability | Do the landing page, the main feature screen, and any form pages hold their layout at phone width with no sideways scroll? | Failed |
| PROD-MOBILE-02 | Functionality and usability | Are the primary tap targets (buttons, nav items, form controls) at least 24 by 24 CSS pixels, or spaced as the WCAG 2.2 target-size rule allows, at 390px phone width? | Failed |
| PROD-STATE-01 | Functionality and usability | Does every list, dashboard, or search result screen show a real empty state when there is nothing to show, instead of a blank area or a frozen loader? | Partly |
| PROD-STATE-02 | Functionality and usability | Does the app show a visible loading indicator during every action that takes more than an instant, rather than leaving the screen looking frozen or unresponsive? | Passed |
| PROD-STATE-03 | Functionality and usability | When a request fails, does the user see a plain-language message that says what happened and what to do next, instead of a blank screen, a stuck spinner, or a raw error? | Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| PROD-STUB-01 | Functionality and usability | Do the buttons and links on the app's primary screens actually do something, rather than being visual stubs left over from the build? | Failed |
| PROD-STUB-02 | Functionality and usability | Do the links in the footer and any secondary navigation, not the primary header navigation checked in PROD-STUB-01, all resolve to a real page rather than a 404 or a placeholder? | Passed |
| PROD-FLOW-01 | Functionality and usability | Can a new visitor complete sign-up end to end, in a real browser, at desktop and phone width? | Passed |
| PROD-FLOW-02 | Functionality and usability | Can a returning user sign in end to end, in a real browser, at desktop and phone width? | Passed |
| PROD-FLOW-03 | Functionality and usability | Can a test purchase be completed end to end, from the pricing or product page through to a confirmed paid state, with no dead end? | Passed |
| PROD-FLOW-05 | Functionality and usability | When a user deletes their own account, is their personal data actually deleted or anonymised in the database, not only the account marked deactivated? | Passed |

| Check | Area | What it asks | Result |
|---|---|---|---|
| PROD-FORM-01 | Functionality and usability | Do the app's forms validate input inline and block submission with a clear message, rather than accepting bad data or failing only after a full round trip? | ■ Passed |
| PROD-FORM-02 | Functionality and usability | When a form submission fails, does the user's already-typed input stay in place, or does the form clear and force them to start over? | ■ Passed |
| PROD-PAY-02 | Functionality and usability | When a test payment is declined, does the app tell the user clearly and let them retry, rather than leaving them stuck with no explanation or silently charging nothing while showing success? | ■ Passed |
| PROD-SEO-01 | Functionality and usability | Does every public page carry a unique title, a unique meta description, and a canonical URL, and do internal-only pages carry noindex? | ☐ Not applicable |
| PROD-SEO-02 | Functionality and usability | When the homepage and a content page are shared on WhatsApp, LinkedIn, or Twitter/X, does the link show a real title, description, and image? | ■ Partly |

| Check | Area | What it asks | Result |
|---|---|---|---|
| PROD-SEO-03 | Functionality and usability | Do sitemap.xml and robots.txt exist at the live domain, load without error, and stay in sync as content is added? | ☐ Not applicable |
| PROD-SESSION-01 | Functionality and usability | Does refreshing the browser mid-flow, or closing and reopening the tab, keep the user signed in and their in-progress work intact? | ■ Passed |

# Auditor's statement

Qodeup examined Tallyroom's code, database rules and live app against the Qodeup Check Catalogue v1.4.

Every finding was found by the audit crew and challenged by a verifier against its evidence. Every urgent and serious finding was reviewed again, blind, by an independent second model. A senior engineer read the report and signed it.

**Orezi Emeotu, founder, Qodeup Africa**

Signed 5 November 2026

Report version 1, audit `a-sample-01`

## What happens next

- → Free re-check of the P0 items by 5 December 2026.
- → 30-minute walkthrough call, booked for 6 November.
- → Qodeup can fix everything in weeks 1 and 2 as a fixed-price sprint; the quote is attached separately.