UX CASE STUDY · 2026

Blu-I-Tech

E-commerce wireframes

Seventeen screens covering the full path from catalog to paid order, plus the loading, failure and admin states that usually get drawn last. Structure and behaviour resolved before a single pixel of visual design.

17	5	4	12
screens	flows	user roles / areas	documented states

N. Nkosi · Product design
Kempton Park, South Africa · August 2026

CONTENTS

What is in this deck

The deck follows the order a customer meets the product, then turns around and looks at the same system from the shop owner's side.

CONTEXT

The brief and the constraints

How I worked

Screen map

Design principles

BROWSE AND DISCOVER

Catalog and filtering

Product detail

BASKET AND CHECKOUT

Basket

Checkout: the key decision

Checkout, chosen layout

Payment states

Confirmation

ACCOUNT AND ORDERS

Sign in, register, profile

Order history and detail

ADMIN

Dashboard

Inventory

EDGE STATES

Loading and skeletons

404 and 500

CLOSE

Component notes

What I would do next

Reading the screen pages

Each screen page carries the wireframe on the left and numbered notes on the right. The numbers match the blue markers drawn on the wireframe itself, so a note and the thing it describes are always findable together.

CONTEXT

The brief

Blu-I-Tech sells developer branded gear in South Africa. The site already had a product list and a working payment integration. It did not have a coherent path between them.

WHAT WAS WRONG

Buyers dropped out between the basket and the paid order. The basket did not say what delivery would cost, checkout asked for a sign in only after the buyer had committed, and a declined card left the page silent.

WHAT I WAS ASKED FOR

Wireframes covering the full purchase path, detailed enough that a developer could build from them without inventing behaviour, and honest about what happens when things go wrong.

> **Draw what happens when it fails, or you have not designed the checkout. You have designed a demo of the checkout.**

FIXED CONSTRAINTS

- Payments run through Stripe; card details never touch our server
- Prices in rand, VAT at 15 percent, free delivery over R500
- 21 products across 4 brands and 4 types
- One shop owner does all admin, usually on a laptop
- Existing API; new endpoints had to be justified

CONTEXT

How I worked

Four passes, each one answering a question the next pass depends on. Nothing was styled until the behaviour was settled.

01

Map the path

Wrote out every screen a buyer touches between landing and a paid order, then every screen the owner touches to fulfil it. Seventeen, not the nine I first guessed.

02

Draw the happy path

Catalog, product, basket, checkout, confirmation at low fidelity. Enough to argue about layout without anyone arguing about colour.

03

Break it on purpose

For each screen: what does this look like empty, loading, failed? That question produced the skeletons, the declined card state and the two error pages.

04

Annotate for build

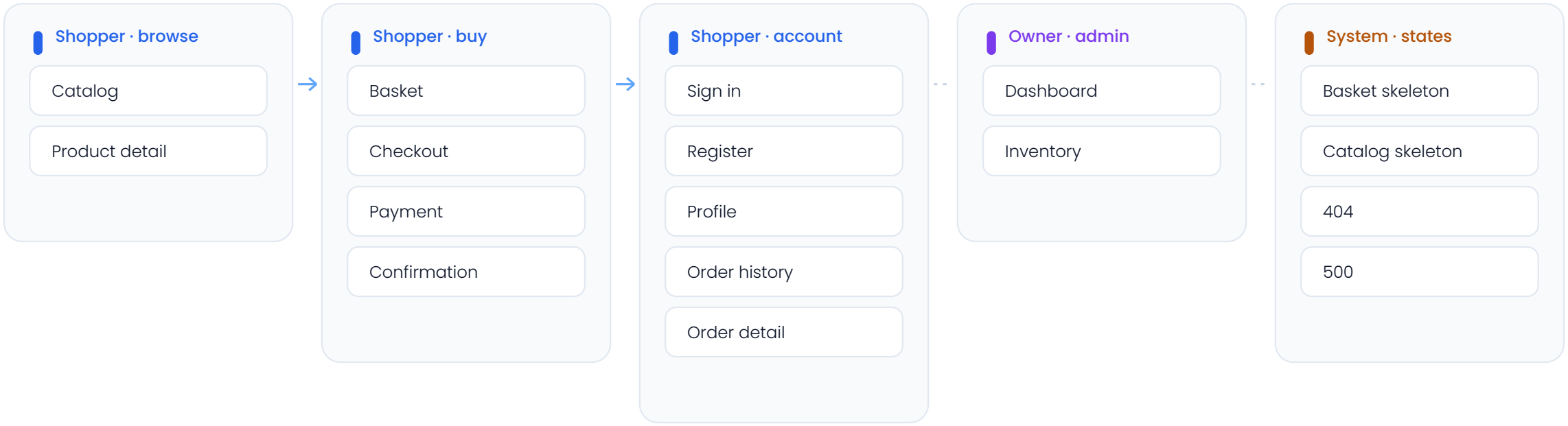
Numbered callouts on every screen carrying the rule a developer cannot read off the layout, and a note where new API work is needed.

Pass 03 is the one that changed the deliverable. Asking "what does this look like when it fails" turned nine screens into seventeen, and every one of the eight extra screens is a state a real customer meets.

CONTEXT

Screen map

Every screen, grouped by who uses it. Solid arrows are the main path; the states column is what each area falls back to when data is slow or a request fails.



Solid arrow the main purchase path — **Dashed** reached from anywhere — **Purple** admin only, gated on role — **Amber** fallen into, never chosen

CONTEXT

Six rules I held the whole way

Written down first so that every later layout argument had something to be settled against.

1 One blue, one job

The accent blue only ever marks the single action this screen wants. Everything else is outline or plain text. If two things are blue, one of them is wrong.

2 Say the number

Buttons carry the amount, not the verb. "Pay R954.50" instead of "Submit". Delivery says "R120 more for free delivery", not "almost there".

3 Every list has four states

Loading, empty, error, loaded. A list drawn only in its loaded state is not finished, it is a quarter finished.

4 Ask why before you ask what

The sign in screen explains that the basket is saved and why sign in is being asked for. Context banner first, form second.

5 Fail out loud, fail kindly

Errors name the cause and offer a route out. A declined card offers EFT. A 500 says plainly that nothing was charged.

6 Never move the furniture

Skeletons mirror the loaded layout. Refetching dims content in place. Nothing reflows under a cursor that is already moving.

01

SECTION

Browse and discover

The two screens that decide whether anyone reaches the basket at all. Both are dense with data, so the work here was mostly about what to leave out.

SCREENS IN THIS SECTION

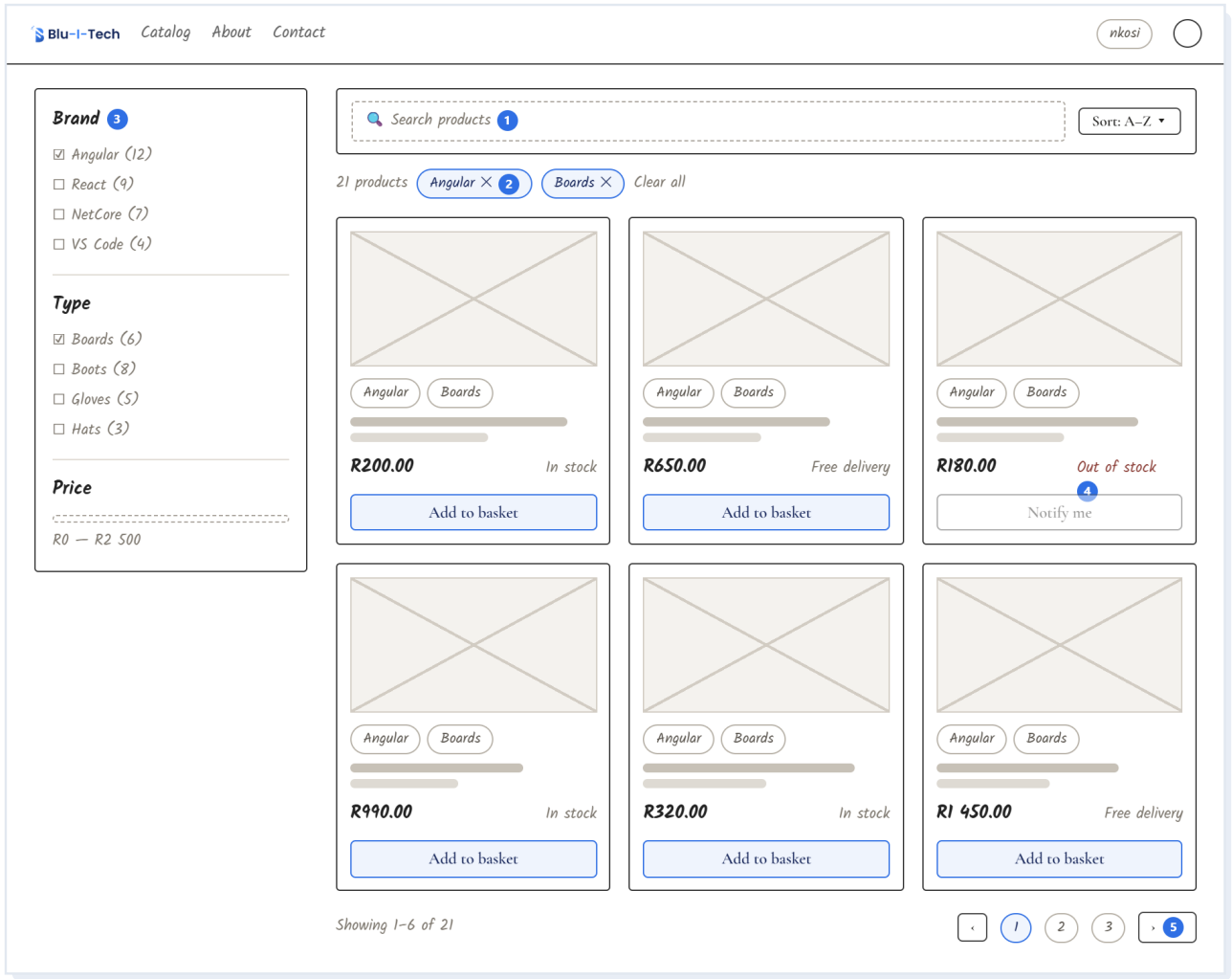
- Catalog with filtering, search and sort
- Product detail with gallery and buy panel

SCREEN 1A

1g-catalog.png

Catalog

Filter sidebar, search, sort and a paginated grid. The screen has to answer "what do you have" and "can I get it" at the same time.



1. **Search writes to the URL**
   Search, filters, sort and page all live in the query string, so any view can be shared, bookmarked or reloaded without losing state.

2. **Filters repeat as chips**
   Active filters appear as removable chips above the grid. The user undoes a choice where they are looking, not by hunting the sidebar.

3. **Counts sit on the facets**
   Angular (12), Boards (6). A filter that would return nothing is visible as a dead end before it is clicked.

4. **Out of stock stays visible**
   Sold out products keep their place in the grid and swap Add to basket for Notify me. Hiding them makes the catalog look thinner than it is.

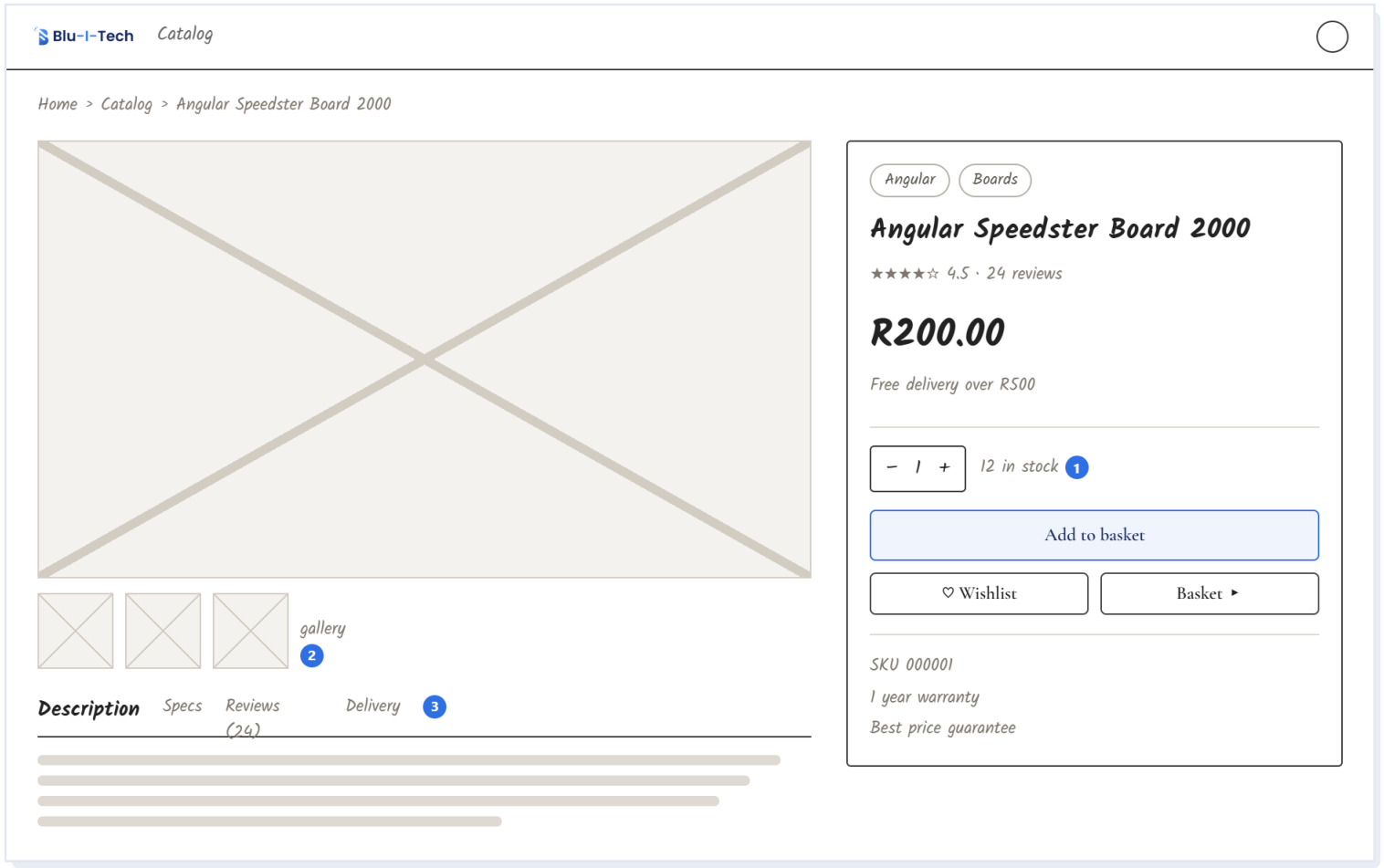
5. **Pages, not infinite scroll**
   "Showing 1 to 6 of 21" stays honest, the footer stays reachable, and the back button returns to the same six products.

SCREEN 1B

Product detail

1i-product-details.png

One product, everything a buyer needs to commit, with the buy panel held on the right so it never scrolls out of reach.



1 Real stock, not a badge
"12 in stock" next to the stepper, and the stepper will not go above it. The buyer learns the limit before the basket rejects them.

2 Gallery under the main image
Thumbnails swap the hero image in place. No lightbox, no modal, nothing that has to be dismissed.

3 Long copy goes in tabs
Description, specs, reviews and delivery are tabbed so the price and the buy button stay above the fold on a laptop.

4 Breadcrumb keeps the search
Home > Catalog > product. Catalog returns to the filtered result set, not to an unfiltered page one.

5 Trust lines, not badges
SKU, warranty and price guarantee sit as quiet text under the buy panel rather than as coloured trust seals.

02

SECTION

Basket and checkout

The part of the product where design decisions cost money. This section includes the one real fork in the road, and why I went the way I did.

SCREENS IN THIS SECTION

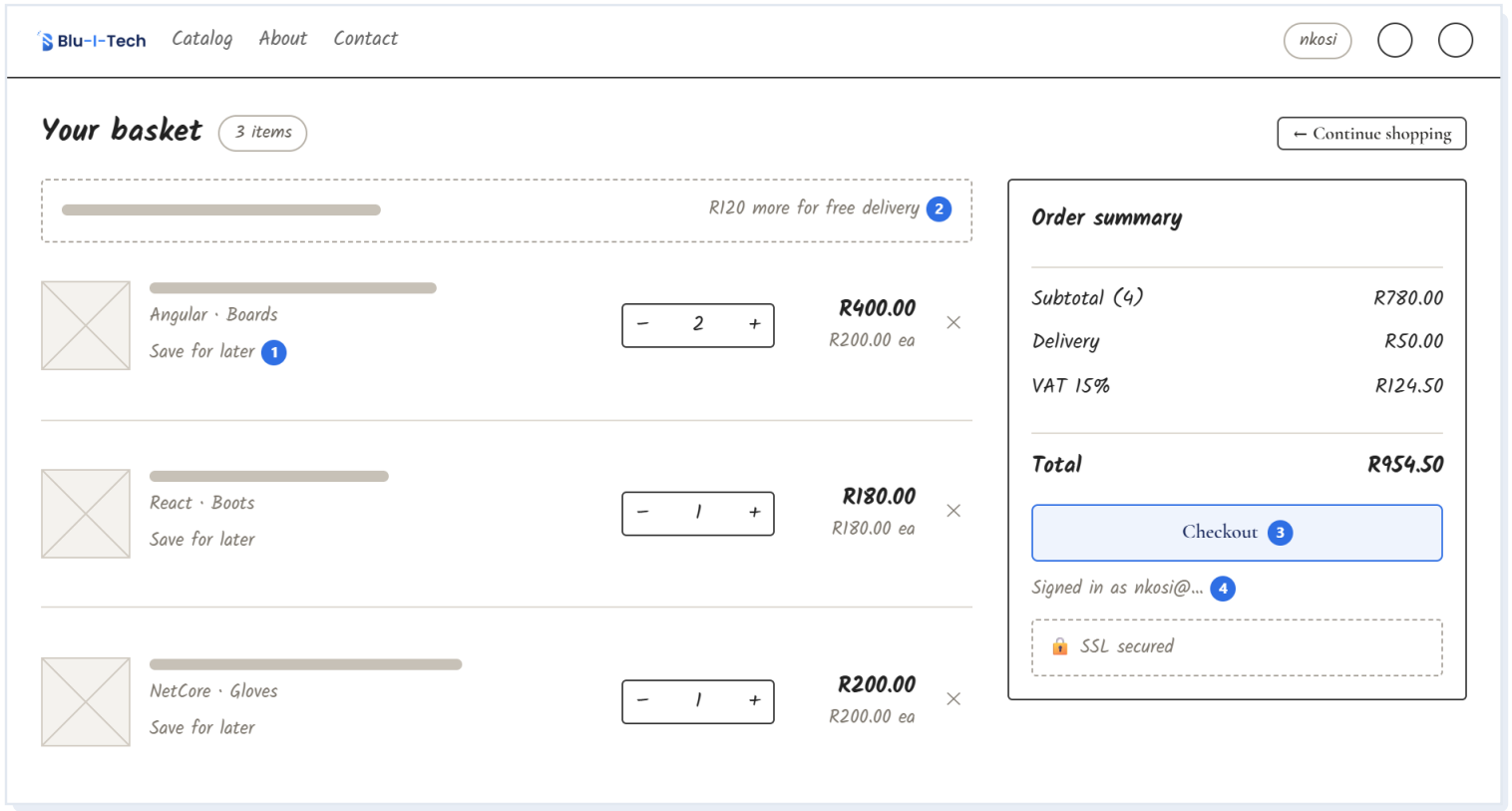
- Basket with save for later and delivery progress
- Checkout: two competing layouts compared
- Payment: setting up, declined, processing
- Order confirmation

SCREEN 2A

1a-basket.png

Basket

Line items on the left, a sticky order summary on the right, and one clear way forward.



1 Save for later, not delete
Moves an item out of the totals without losing it. The most common reason to empty a basket is hesitation about one line, not all of it.

2 Delivery progress in rand
"R120 more for free delivery" with a progress bar. A specific number is actionable; "almost there" is not.

3 One primary button
Checkout is the only filled button on the page. Continue shopping is a quiet outline in the corner, and remove is a plain X.

4 Sign in state shown early
"Signed in as nkosi@..." sits under the checkout button, so nobody hits an unexpected sign in wall on the next click.

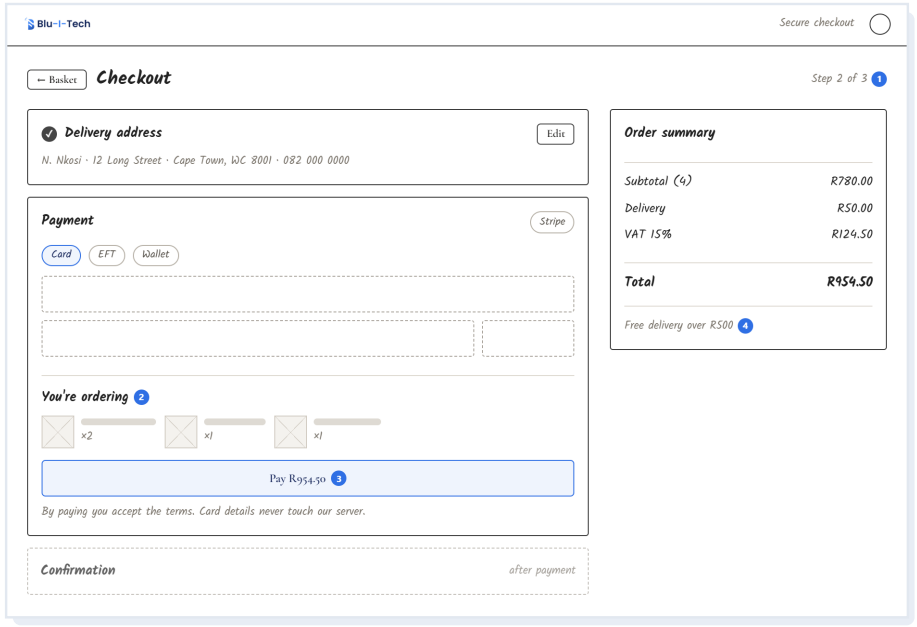
5 Totals broken out
Subtotal, delivery and VAT at 15 percent listed separately before the total. No number appears at checkout that was not already shown here.

DECISION

Checkout: one page or two panes

The only place in this project where two layouts were genuinely defensible. I drew both properly rather than arguing about them.

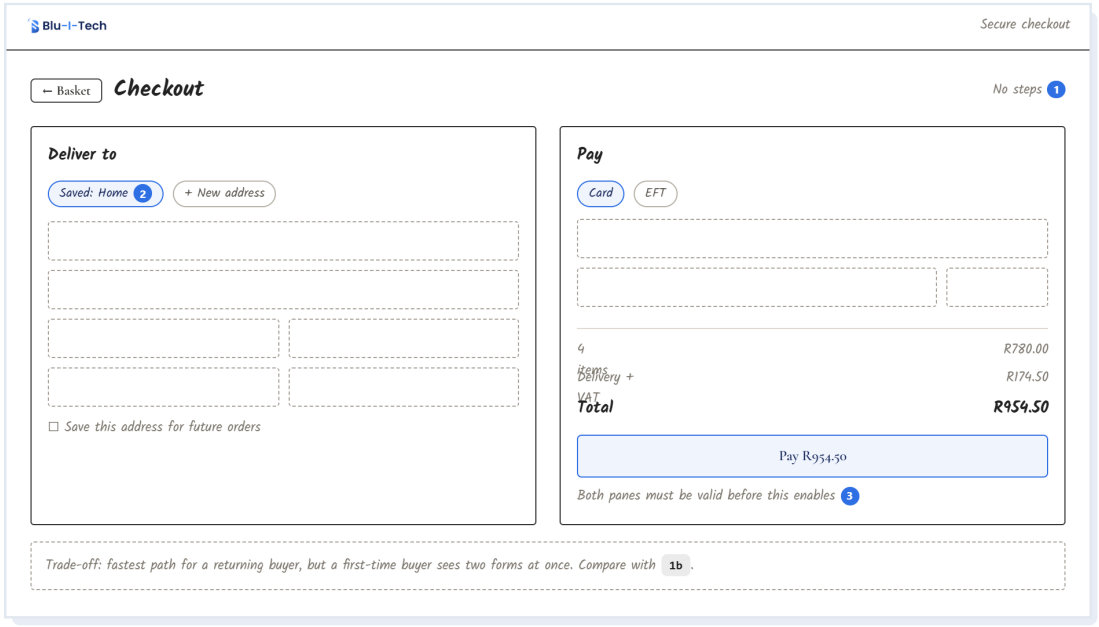
Option A — One page, stepped



For — Address confirmed, then payment. One decision at a time, and "Step 2 of 3" sets the expectation.

Against — A returning buyer with a saved address scrolls past a section they did not need to think about.

Option B — Two panes, no steps



For — Fastest path for a returning buyer. Everything on one screen, no scroll at all.

Against — A first time buyer meets two forms at once, and Pay stays disabled until both panes validate, which reads as broken.

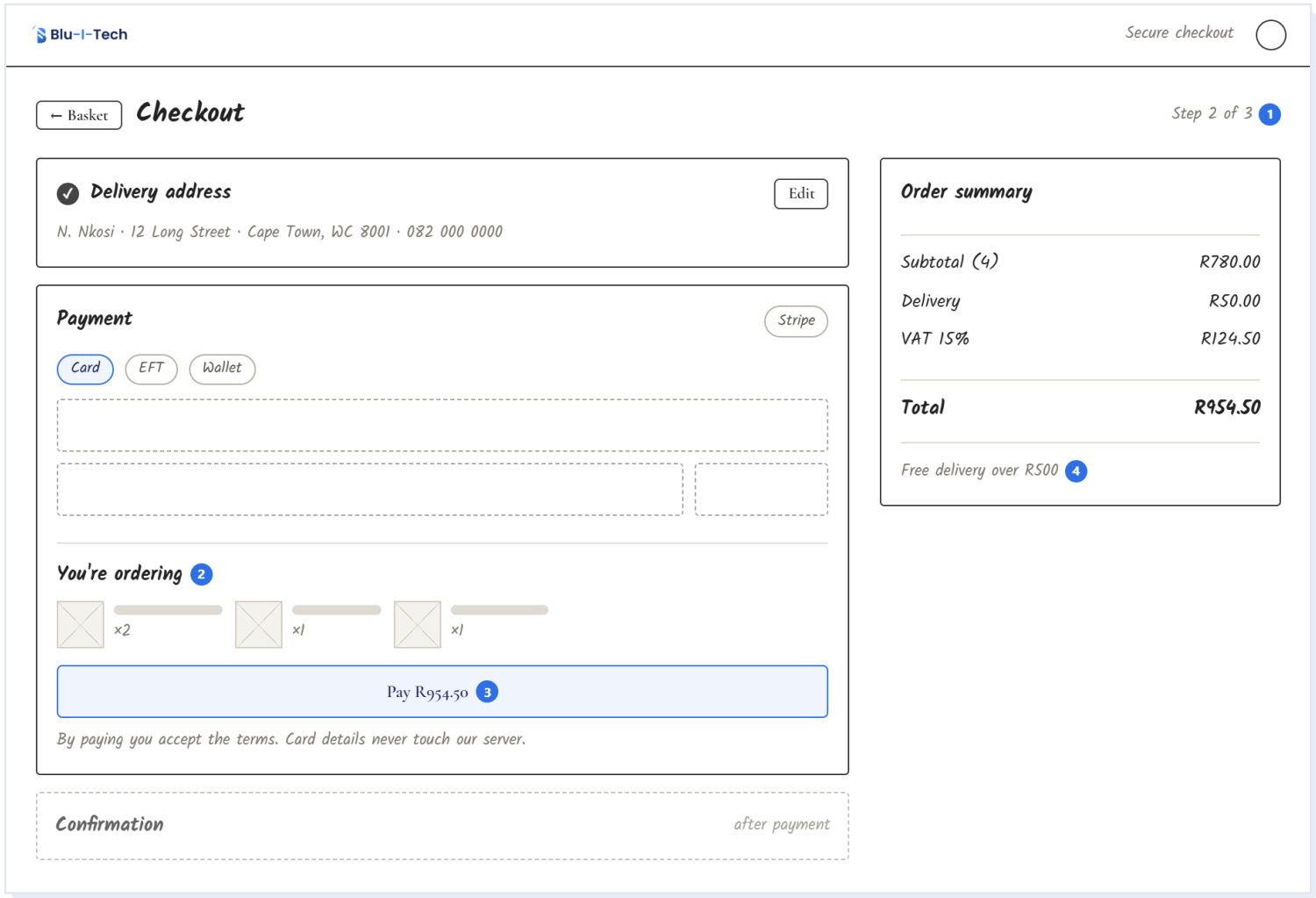
Chose A — First time buyers are where the money leaks, and checkout is three fields deep either way. A step counter is worth the one scroll it costs a returning customer, and Option B hides its own progress: a disabled Pay button with no explanation is the worst silence in a payment flow.

SCREEN 2B

1b-checkout-one-page.png

Checkout, chosen layout

Delivery address confirmed and collapsed, payment open, order summary pinned. Step 2 of 3.



1 Step counter, always
"Step 2 of 3" in the header. Buyers tolerate a long form far better when they can see where it ends.

2 Confirmed sections collapse
The delivery address shrinks to one line with an Edit button once it is valid, so the open section is always the one that needs attention.

3 You're ordering, right above Pay
Item thumbnails and quantities sit directly above the pay button. Last chance to catch a wrong size without leaving the page.

4 The button carries the amount
"Pay R954.50", not "Place order". The number on the button matches the number in the summary, every time.

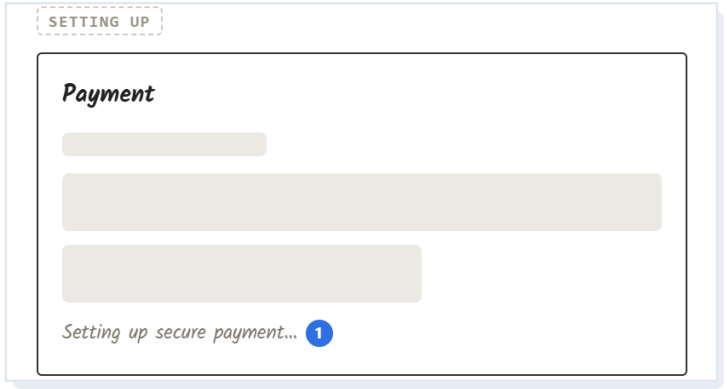
5 Say where the card goes
"Card details never touch our server" in plain words under the button, next to the Stripe mark. Reassurance where the hesitation is.

SCREEN 2C

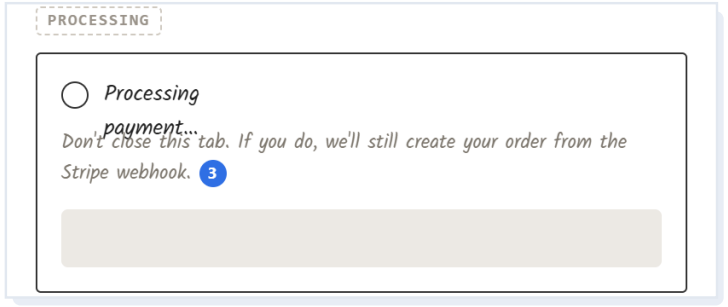
1d-payment-states.png

Payment states

The three moments between pressing Pay and knowing the outcome. Drawing these was the point at which the checkout stopped being a form and started being a transaction.







1 Setting up

A skeleton shaped like the card form while the Stripe intent is created. The fields land exactly where the grey blocks already are, so nothing jumps under a cursor that is already moving.

2 Declined

The error sits above the form, names what happened, and offers EFT as a real alternative. A bare "try again" just sends the buyer back to the card that has already failed once.

3 Processing

Says the order is safe even if the tab closes, because the Stripe webhook creates the order, not the browser. A backend truth the interface has to say out loud.

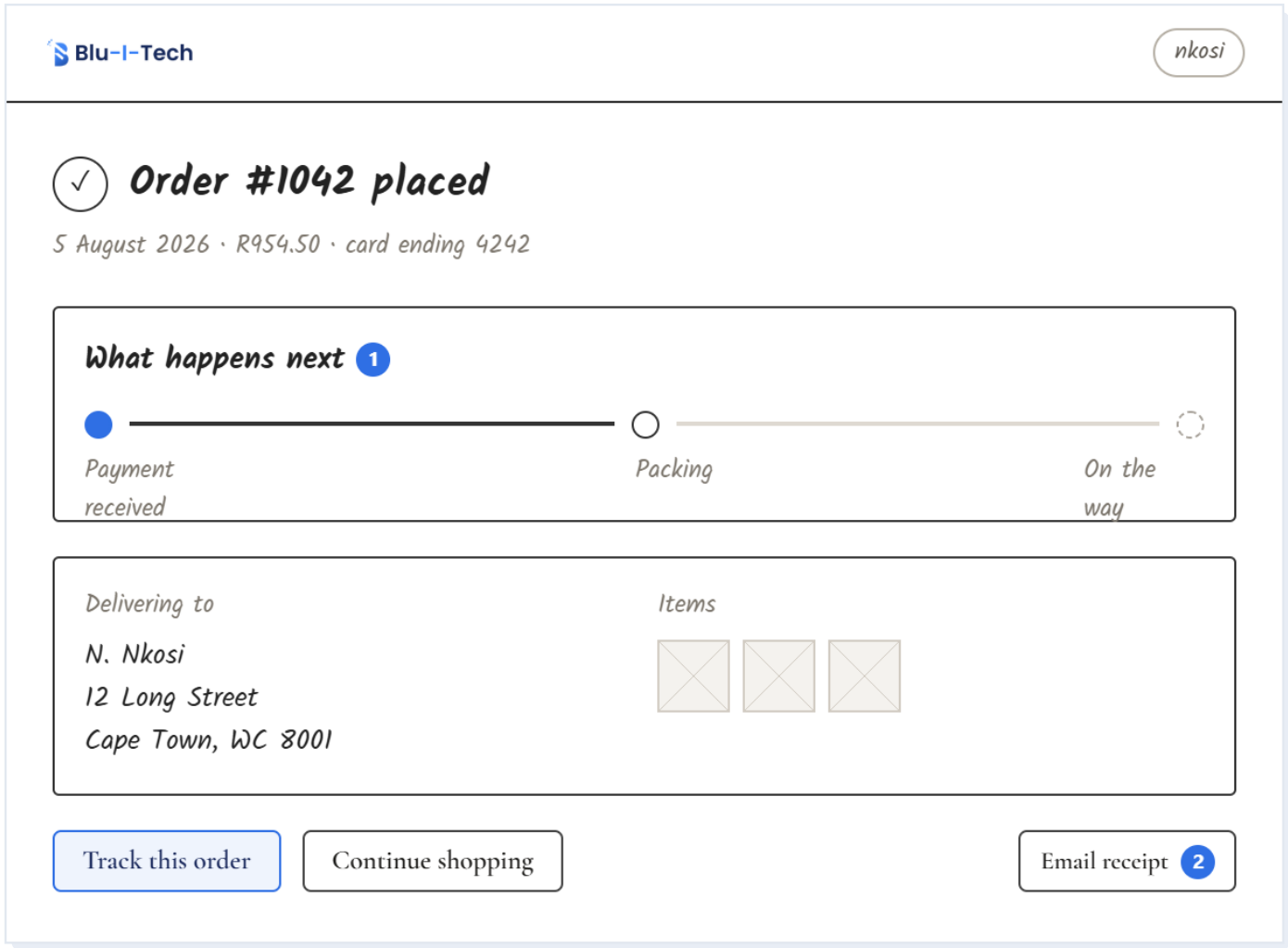
Why it matters These three states are the difference between a checkout that loses money quietly and one that recovers. Each was drawn as a full state, not as a note on the happy path, because they are the frames where a buyer decides whether the site can be trusted with a card number.

SCREEN 2D

1e-confirmation.png

Confirmation

The order exists. Now the job is to answer the question the buyer actually has, which is not "did it work" but "when does it arrive".



1 What happens next, not thank you
A three step tracker: payment received, packing, on the way. The page answers the next question instead of closing the conversation.

2 Receipt as an action
Email receipt is an explicit button rather than a promise that something was already sent. The buyer controls where the record goes.

3 Facts in one line
Order number, date, amount and card last four across the top. Enough to quote in a support message without opening anything.

4 Two ways out
Track this order for the anxious, continue shopping for the rest. No dead end.

03

SECTION

Account and orders

Everything a returning customer needs, kept deliberately thin. Account screens earn their keep by getting out of the way.

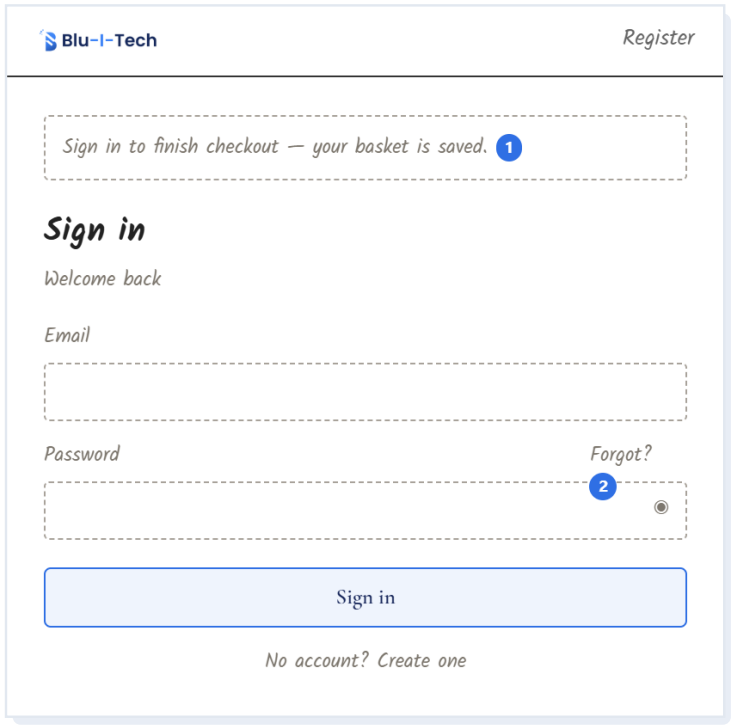
SCREENS IN THIS SECTION

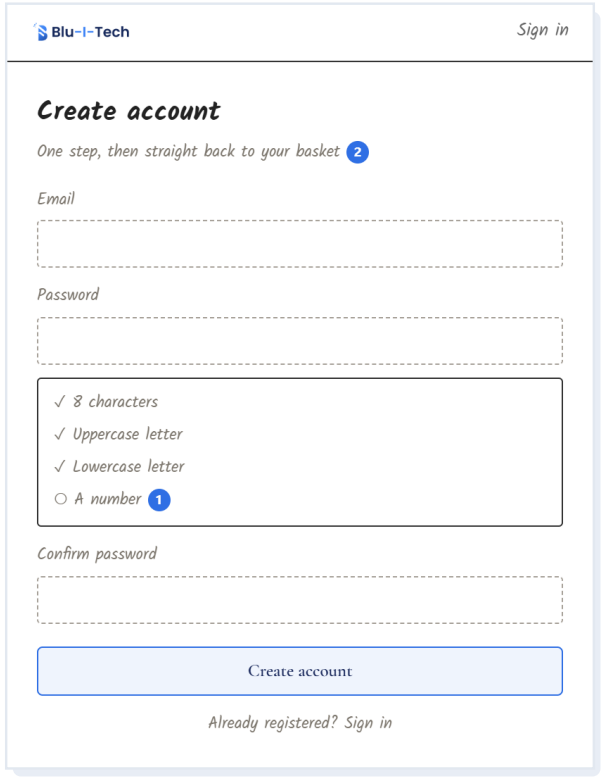
- Sign in with checkout context
- Register with live password rules
- Profile and saved addresses
- Order history
- Order detail

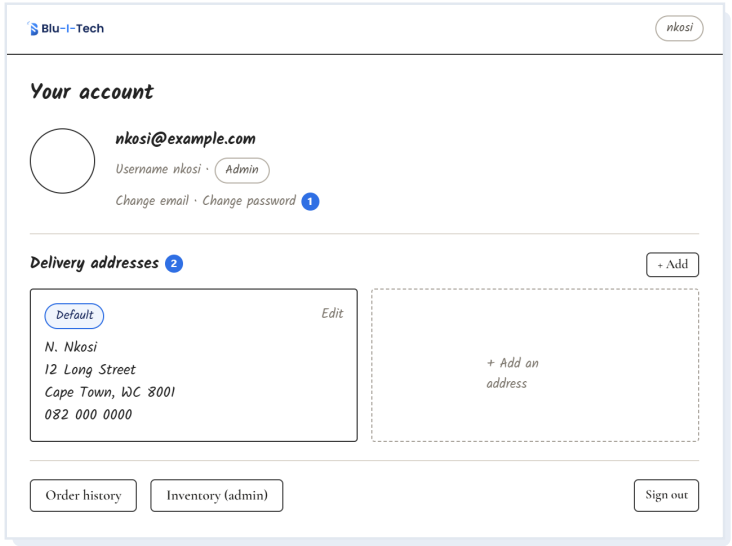
SCREENS 3A TO 3C

Sign in, register, profile

Three screens, one rule: never ask for something without saying why, and never let a form fail after submission when it could have failed while typing.







Sign in

The banner explains that sign in is being asked for because checkout needs it, and that the basket is already saved. Forgot sits inline with the password label, where it is needed.

Register

Password rules tick live as you type rather than rejecting the form after submit. The sub heading promises a return straight to the basket, so the detour has a stated end.

Profile

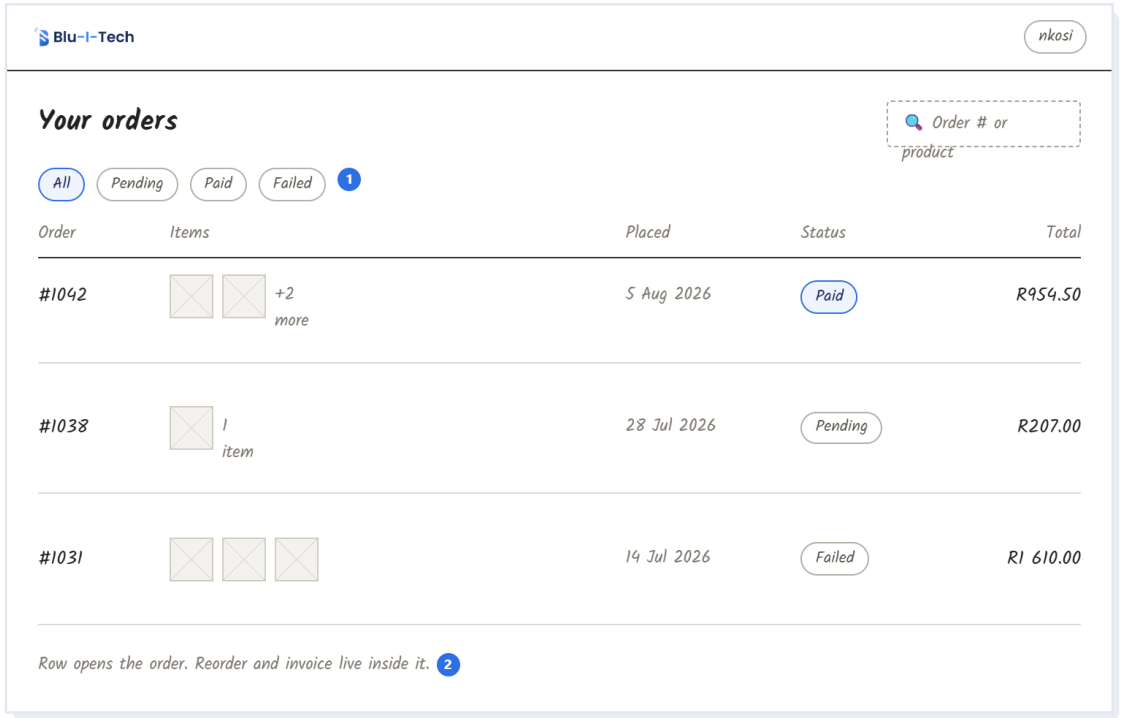
Account details, saved delivery addresses with one marked default, and role aware entry points. Inventory only appears for an admin account.

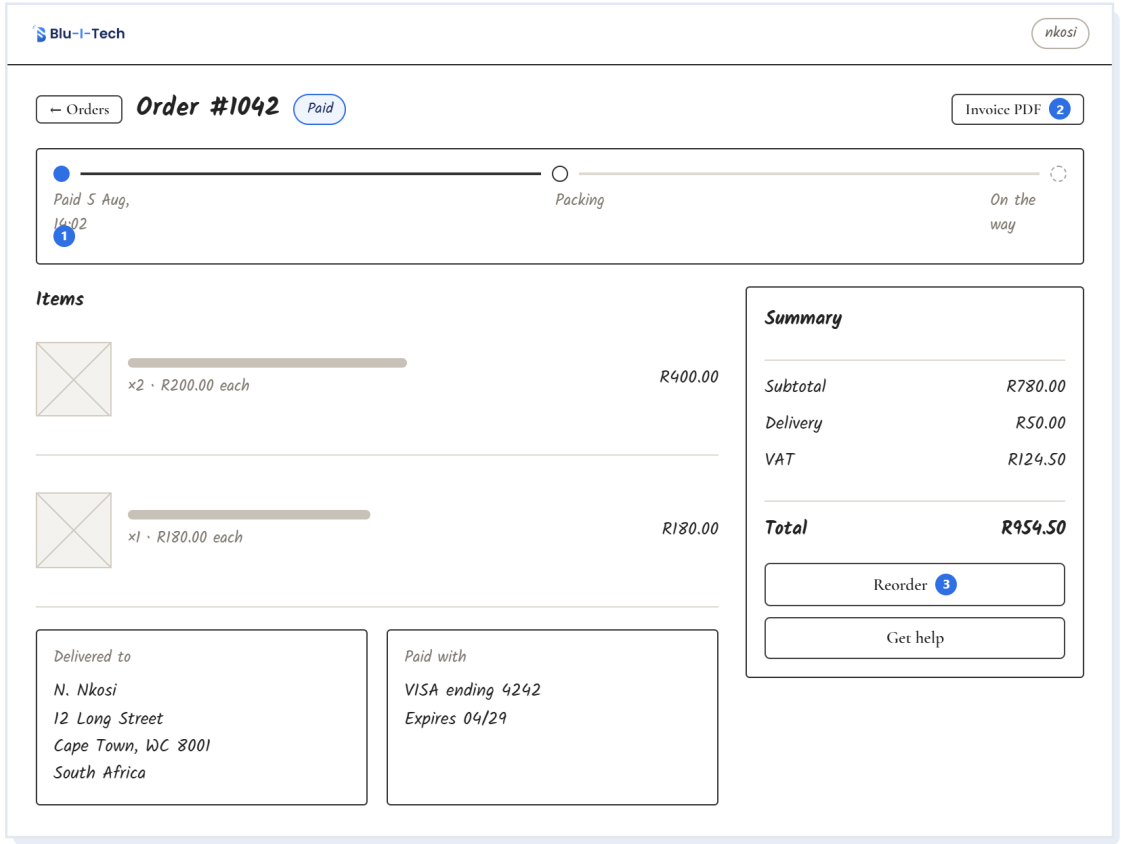
All three reuse the same field, button and card components as checkout. No screen in this project introduces a control that exists nowhere else.

SCREENS 3D AND 3E

Order history and order detail

A list built for scanning and a detail page built for the two things people come back for: proof of purchase and buying it again.





Order history

Status filter chips across the top, search by order number or product, and the whole row is the target rather than a small View link. Status uses the same chip component as the admin table, so the two views never disagree about wording.

Order detail

The same three step tracker the buyer saw on confirmation, so the mental model carries. Invoice PDF, reorder and get help are the three real jobs, grouped in the summary card rather than scattered across the page.

04

SECTION

Admin

The same system seen by the person who has to fulfil the orders. One owner, one laptop, a handful of jobs done many times a day.

SCREENS IN THIS SECTION

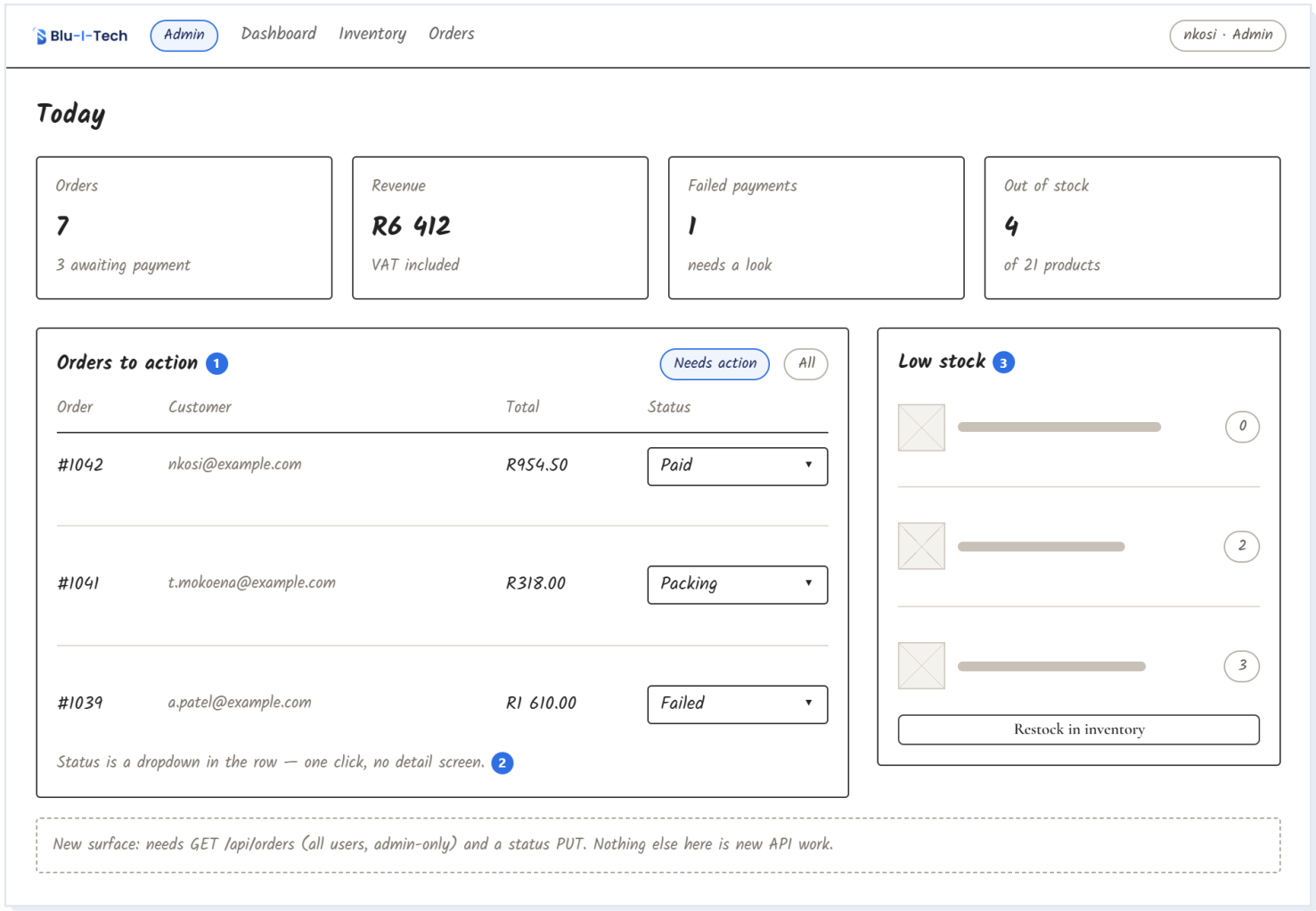
- Dashboard: is anything wrong today
- Inventory: table plus edit panel

SCREEN 4A

Admin dashboard

Built to answer one question on open: is anything wrong right now. Everything else is one click away.

1o-admin-dashboard.png



1 Four numbers, chosen carefully
Orders, revenue, failed payments, out of stock. Two are health, two are alarms. No vanity metrics and no charts nobody acts on.

2 Status changes in the row
The status dropdown lives in the table row. Marking an order as packing is one click with no detail screen, because it is the job done most often.

3 Needs action is the default filter
The table opens filtered to orders that require something, with All one tap away. The default view matches the reason for opening it.

4 Low stock links to the fix
The panel lists what is running out and hands off straight to inventory, rather than reporting a problem it cannot resolve.

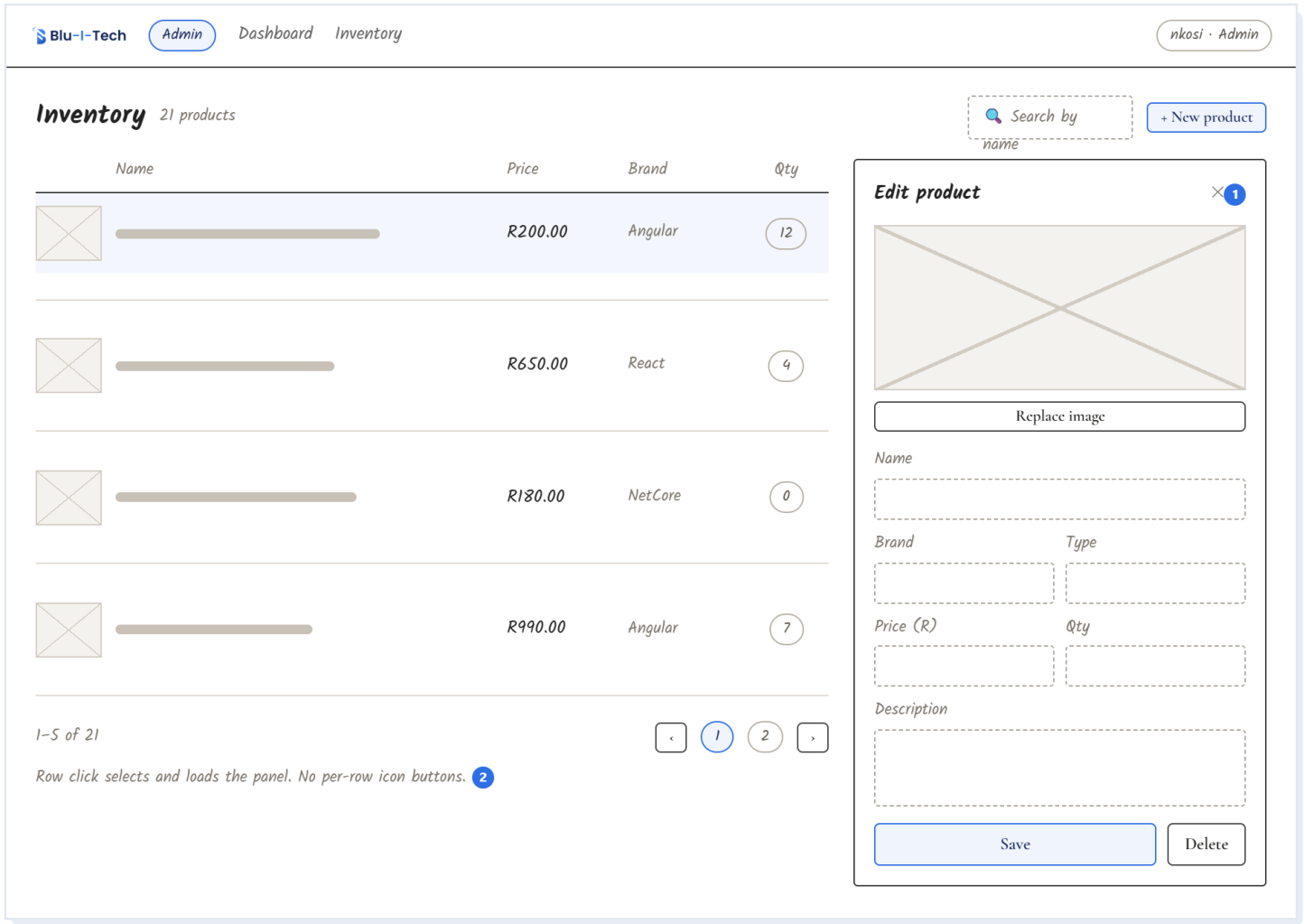
5 Honest about new API work
The footnote flags exactly what this screen needs that does not exist yet: an admin scoped GET on orders and a status PUT. Nothing else here is new backend work.

SCREEN 4B

1p-inventory.png

Inventory

Table on the left, edit panel on the right, one selection model. The most frequent job here is changing a quantity, so that is the shortest path.



1 Row click, not row icons
Selecting a row loads it into the panel. No per row pencil and bin icons crowding the table, and no ambiguity about what is selected.

2 One panel for new and edit
"New product" opens the same panel with empty fields. One form to build, one form to learn, one place bugs can live.

3 Quantity is the last column
Rightmost and set as a pill, because restocking is the job that brings the owner here. Zero reads as a warning without needing red.

4 Delete is quiet
Outline, bottom right, away from Save. Destructive actions should be reachable, never adjacent to the thing you press every time.

05

SECTION

Edge states

Loading and failure, drawn as first class screens. These are the states a user meets on a bad connection, which is most of them.

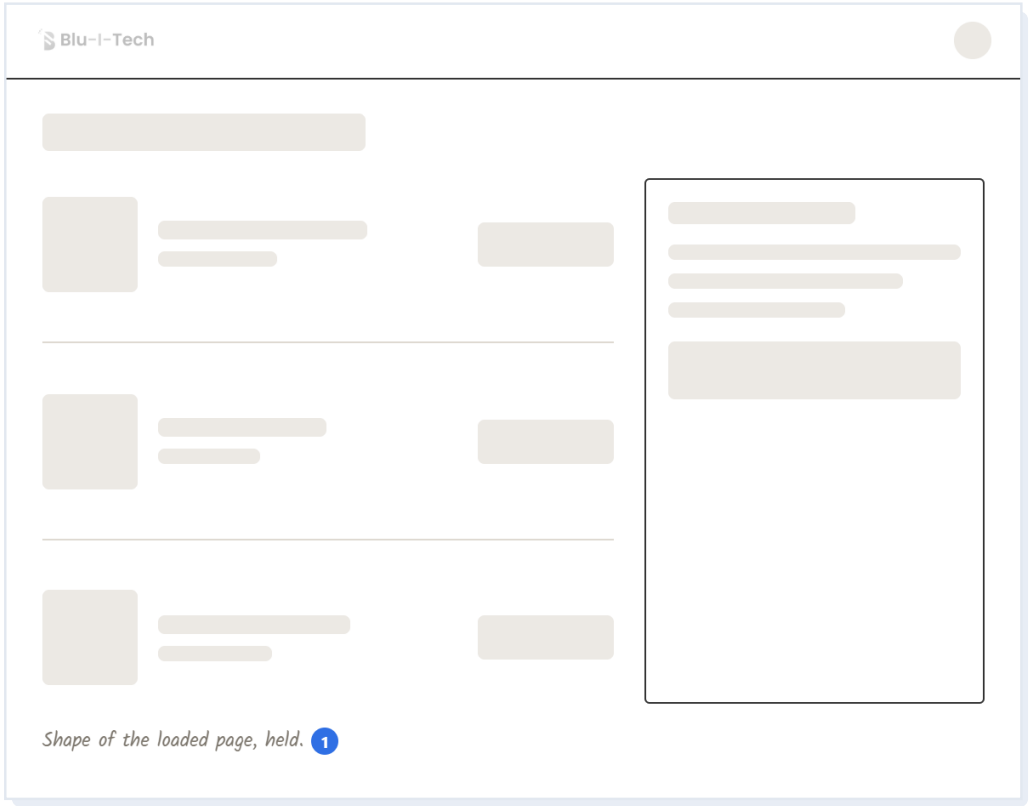
SCREENS IN THIS SECTION

- Basket skeleton
- Catalog skeleton and refetch
- 404 not found
- 500 server error

SCREENS 5A AND 5B

Loading, without the jump

Both skeletons trace the shape of the page they are standing in for, so when the data lands nothing moves.



Basket skeleton

Three line item rows and the summary card, held at the exact dimensions of the loaded page. The header and its border render immediately, because those never depend on data.



Catalog skeleton and refetch

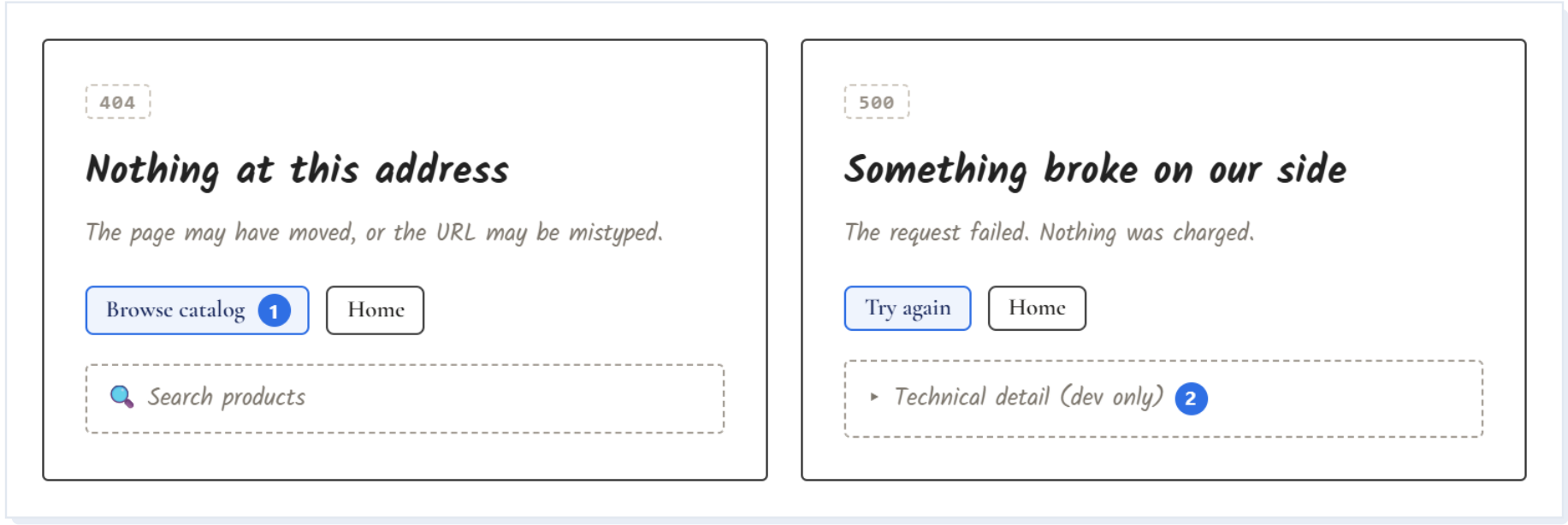
First load shows card shaped placeholders. A refetch, from a page or filter change, dims the existing grid in place instead: no blur overlay, no spinner, no scroll jump.

Rule applied on both: a skeleton is a layout, not a spinner. If it does not match the shape of what arrives, it is doing more harm than an empty box would.

SCREEN 5C

When it breaks

Two error pages that do the same two things: say plainly what happened, and give a way forward that is not the back button.



404 · nothing at this address

Offers a route rather than an apology. Browse catalog is the primary action and a search field sits right underneath, because a mistyped URL usually means the person knows what they wanted.

500 · something broke on our side

Says "nothing was charged" in the first line. On a commerce site that is the only sentence that matters. Technical detail is collapsed and shown to developers only.

Neither page shows a stack trace, an error code the user cannot act on, or a sad illustration. An error page has one job: end the dead end.

CLOSE

What the seventeen screens are made of

The whole set resolves to a small number of repeated parts. That was the point of wireframing it as one system rather than screen by screen.

Buttons

Primary filled, secondary outline, quiet text. One primary per view, always.

Status chip

Paid, pending, packing, failed, on the way. Same five words, customer side and admin side.

Money

Always R with two decimals and a thin space in thousands. R1 450.00, never R1450.

Field

Label above, dashed placeholder while empty, inline validation message below.

Card

1px border, 8px radius. Used for summary, address, panel and product alike.

Stepper

Minus, value, plus. Capped at live stock on the product page, at basket quantity elsewhere.

Skeleton

Grey blocks at the dimensions of the real content. Never a spinner on its own.

Tracker

Three node progress line, shared by confirmation and order detail so the model carries.

Filter chip

Removable, mirrors an active sidebar or list filter, always sits above the results.

Nine parts cover all seventeen screens. Nothing in this deck introduces a control that appears only once, which is the cheapest test I know for whether a set of wireframes is a system or a pile of drawings.

CLOSE

What I would do next

The set is complete enough to build from. It is not complete.

Next

Mobile layouts for basket and checkout

Everything here is drawn at desktop width. The basket summary and the checkout order panel both need a different arrangement below 768px, and the sticky pay button becomes a bar rather than a card.

Next

Empty states

An empty basket and a search that returns nothing are both missing. Both are common, and both are places to recover a session rather than end it.

Next

Accessibility pass

Focus order through checkout, error announcement on the declined card state, and a check that status is never carried by colour alone.

Open question

Test guest checkout against forced sign in

The current flow requires an account. That is a business decision as much as a design one, and it is worth the experiment.

Open question

Real product content

Every card here holds placeholder bars. Real names and photography will change how the grid reads, and the card may need a fixed title height.

THANK YOU

Seventeen screens, one system.

Happy to walk through any screen in detail, including the ones that did not make it into this deck.



N. Nkosi

nkosin361@gmail.com

Kempton Park · August 2026