RETAIL INNOVATIONS LTD

Digital Retail Management Platform

Task 2 — Activity B: Documenting the Development Process

T Level Technical Qualification in Digital Software Development

Document Contents

1. Testing Overview and Strategy

This document records all testing conducted on the Retail Innovations Ltd digital platform. Testing was carried out iteratively across four development weeks, with a final systematic review in week five. The approach covers four test categories:

- Functional testing — does each feature produce the correct output?
- Boundary/edge-case testing — what happens at data limits (e.g. 0 points, empty arrays, maximum values)?
- Negative/error testing — does the system handle invalid input gracefully and inform the user clearly?
- Security testing — are access controls, XSS protections, and Supabase RLS policies enforced correctly?

Tests are organised by feature area. Each test records a unique ID, a description, the expected result, the actual result, a PASS/FAIL/PARTIAL verdict, and any fix applied. Where a test failed, the root cause is documented, the fix is described, and the test was re-run to confirm resolution. Tests prefixed AUTH cover authentication, PROD covers products, CUST covers customers, ORD covers orders, LOY covers the loyalty programme, DASH covers the dashboard, SEC covers security, and UI covers interface and usability.

Test Category	Total Tests	Initial Pass	Notes
Authentication (AUTH)	14	14	All passed first run
Products (PROD)	18	15	3 failures — all fixed and re-tested
Customers (CUST)	12	11	1 failure — fixed and re-tested
Orders (ORD)	10	9	1 failure — fixed and re-tested
Loyalty (LOY)	10	10	All passed first run
Dashboard (DASH)	8	7	1 failure — fixed and re-tested
Security (SEC)	9	8	1 partial — noted as tech debt
UI / Usability (UI)	12	12	All passed first run
TOTAL	**83**	**77 (93%)**	**6 failures all resolved; 1 partial noted**

2. Authentication Tests

ID	Test Description	Expected Result	Actual Result	Resu lt	Fix / Notes
AUTH 01	Login — valid admin credentials	Admin signed in. All tabs visible. Admin badge displayed.	Admin login successful. Customers tab visible. Admin badge displayed.	**PASS**	—
AUTH 02	Login — valid customer credentials	Customer signed in. Customers tab hidden. Orders subtitle reads 'Your orders'.	Customer login correct. Admin-only elements hidden. Subtitle updated.	**PASS**	—
AUTH 03	Login — wrong password for existing account	Toast: 'Invalid credentials.' User remains on auth screen.	Error toast shown. No login proceeded.	**PASS**	—
AUTH 04	Login — email does not exist	Toast: 'User not found. Register or check credentials.'	Correct error toast shown.	**PASS**	—
AUTH 05	Login — empty email field	Toast: 'Please enter email/username and password.'	Validation toast shown immediately on submit.	**PASS**	—
AUTH 06	Login — empty password field	Toast: 'Please enter email/username and password.'	Correct toast shown.	**PASS**	—
AUTH 07	Register — valid data (email, password, full name)	Account created. Auto signed in as customer. App enters customer view.	Account created, session stored in localStorage, app loaded correctly.	**PASS**	—
AUTH 08	Register — password fewer than 3 characters	Toast: 'Use a longer password.' Registration blocked.	Toast shown correctly. Registration did not proceed.	**PASS**	Noted: min recommended 8 chars for production deployment.
AUTH 09	Register — duplicate email already registered	Toast: 'User already exists.' No duplicate created.	Duplicate check in JS before save. Error shown correctly.	**PASS**	—
AUTH	Register — no full	Registration proceeds.	Name defaulted to	**PASS**	—

ID	Test Description	Expected Result	Actual Result	Resu lt	Fix / Notes
10	name entered	Name defaults to email prefix.	email prefix (e.g. 'jane' from 'jane@test.com'). Acceptable fallback.		
AUTH 11	Press Enter key on login form	handleAuth() called — same as clicking Sign In.	Enter key triggers authentication correctly.	**PASS**	—
AUTH 12	Toggle Login ↔ Register modes via 'Create one' link	Form switches. Full Name and Confirm Password fields appear. Button label and subtitle update.	Toggle works correctly. Fields appear/disappear. Labels update.	**PASS**	—
AUTH 13	Sign out using Sign Out button	Session cleared from localStorage. Auth screen shown.	Session cleared. Auth screen restored.	**PASS**	—
AUTH 14	Reload page — session persistence	User auto signed back in from localStorage session without re-entering credentials.	Session correctly restored from localStorage on page reload.	**PASS**	—

3. Product Management Tests

ID	Test Description	Expected Result	Actual Result	Result	Fix / Notes
PROD01	Load Products tab — 10 seeded products render	Table shows all 10 products. All columns populated. Count reads '10 products'.	All 10 seed products loaded and displayed. Count correct.	**PASS**	—
PROD02	Search 'tea' — real-time filter	Only 'Organic Green Tea' shown. Count updates to 1.	Filtered to 1 result. Count updated to '1 product'.	**PASS**	—
PROD03	Search 'TEA' — case-insensitive check	Same result as typing 'tea'.	Case-insensitive filtering confirmed.	**PASS**	—
PROD04	Filter by category 'Sports'	Only Running Shoes and Yoga Mat shown. Count updates.	2 Sports products shown. Filter correct.	**PASS**	—
PROD05	Combine search 'premium' with Sports category filter	Only Yoga Mat shown (description contains 'premium').	Correct single result. Combined filter works.	**PASS**	—
PROD06	Clear search — table returns to full list	All products shown again. Category filter retained if set.	Search cleared. Count returned to full amount.	**PASS**	—
PROD07	Admin: Open Add Product modal	Modal opens with empty fields. Title reads 'Add Product'.	Modal opened. All fields empty.	**PASS**	—
PROD08	Admin: Add valid new product	Product added. Toast 'Product saved (local)'. Count increments by 1.	Product added to localStorage. Toast shown. Count updated.	**PASS**	—
PROD09	Admin: Save product with empty Name field	Toast error. Modal stays open. Product not saved.	Initially no validation — blank name was saved. Fix applied.	**FAIL**	Fix: Added if (!name || price<0 || stock<0 || !sku) validation block with toast and early return. PASS on re-test.
PROD10	Admin: Enter negative price (e.g. -5)	Toast error. Product not saved.	Initially accepted negative price. Fix applied.	**FAIL**	Fix: Added price >= 0 check. PASS on re-test.
PRO	Admin: Edit product —	Modal pre-populates.	Edit modal pre-	**PASS**	—

ID	Test Description	Expected Result	Actual Result	Resu lt	Fix / Notes
D11	change price	Saving updates only changed field.	populated correctly. Price update saved.		
PROD12	Admin: Delete product — confirm dialog	Confirm dialog shown. On confirm, product removed. Toast shown.	Product deleted from localStorage. Table updated.	**PASS**	—
PROD13	Admin: Cancel delete in confirm dialog	Product not deleted. Table unchanged.	Cancel preserved product intact.	**PASS**	—
PROD14	Customer: Add Product button not visible	Button absent for customer-role users.	Button hidden via .is-admin CSS class.	**PASS**	—
PROD15	Category dropdown populates dynamically	All unique categories listed. Selection filters correctly.	8 categories populated from seed data. Filter works.	**PASS**	—
PROD16	Product thumbnail renders for valid image_url	Small 36×36px image shows in table row.	Thumbnails render for all 10 seed products via Unsplash URLs.	**PASS**	—
PROD17	Product with broken/missing image_url — fallback	Failed image handled gracefully. No broken icon or layout break.	Broken image showed broken icon. Fix: onerror handler added.	**FAIL**	Fix: Added onerror to show initials placeholder div. PASS on re-test.
PROD18	Inactive product shows 'Inactive' badge	Orange 'Inactive' badge for products with is_active = false.	Inactive badge rendered with correct orange styling.	**PASS**	—

4. Customer Management Tests

ID	Test Description	Expected Result	Actual Result	Resu lt	Fix / Notes
CUST 01	Customers tab hidden for customer-role users	Tab absent from nav when logged in as customer.	Tab hidden via .admin-only CSS class.	**PASS**	—
CUST 02	Customers panel loads with tier visual cards	Four tier cards shown: Bronze, Silver, Gold, Platinum with point ranges.	Tier visual rendered with emoji icons and point ranges.	**PASS**	—
CUST 03	Admin: Add new customer with valid data	Customer added. Toast confirmation. Count increments.	Customer added to localStorage. Table updated.	**PASS**	—
CUST 04	Admin: Add customer — empty email field	Toast error. Customer not created.	Empty email allowed initially. Fix applied.	**FAIL**	Fix: Added if (!payload.full_name || !payload.email) block with toast and early return. PASS on re-test.
CUST 05	Admin: Add customer — duplicate email	Toast: 'A customer with that email already exists.'	Duplicate check before save. Error shown correctly.	**PASS**	—
CUST 06	Admin: Edit — change loyalty points from 0 to 600	Tier auto-updates to Silver (600 ≥ 500). Saved correctly.	Tier recalculated to Silver. Silver badge displayed.	**PASS**	—
CUST 07	Admin: Enter negative loyalty points (-50)	Toast: 'Points cannot be negative.' Customer not saved.	Negative value rejected. Toast shown.	**PASS**	—
CUST 08	Loyalty tier badge colours correct	Bronze=brown, Silver=grey, Gold=gold, Platinum=blue.	All four tier badge CSS classes render correct colours.	**PASS**	—
CUST 09	Admin: Filter customers by tier dropdown	Only matching-tier customers shown. Count updates.	Tier filter functional.	**PASS**	—
CUST 10	Admin: Search customers by name or email	Real-time filter matches full name or email on each	Search filters table correctly.	**PASS**	—

ID	Test Description	Expected Result	Actual Result	Resu lt	Fix / Notes
		keystroke.			
CUST 11	Admin: Delete customer — confirm dialog	Confirm dialog. Customer removed on confirm.	Customer deleted. Table refreshed.	**PASS**	—
CUST 12	Tier boundary values: 999 pts = Silver, 1000 pts = Gold	calculateLoyaltyTier(999) returns 'Silver'. calculateLoyaltyTier(1000) returns 'Gold'.	Boundary values handled correctly.	**PASS**	—

5. Order Management Tests

ID	Test Description	Expected Result	Actual Result	Resu lt	Fix / Notes
ORD 01	Admin: Orders panel shows all orders with status badges	All orders visible. Status shown as coloured badge.	Admin sees all orders. Badges styled correctly.	**PASS**	—
ORD 02	Customer: Orders shows only own orders	Only orders linked to logged-in user_id displayed.	Customer view filtered by user_id. Other orders not visible.	**PASS**	—
ORD 03	Admin: Create order — valid customer, amount, status	Order created. Toast. Count increments. Dashboard revenue updates.	Order saved. Revenue recalculated via loadDashboard().	**PASS**	—
ORD 04	Admin: Create order — negative total amount	Toast error. Order not saved.	Initially accepted negative amount. Fix applied.	**FAIL**	Fix: Added if (isNaN(total) || total<0) block with toast and early return. PASS on re-test.
ORD 05	Admin: Update order status pending → shipped	Status badge updates. Toast. Change persists on refresh.	Status updated. Badge colour changed to match shipped status.	**PASS**	—
ORD 06	Admin: Delete order — revenue recalculates	Order removed. Dashboard revenue total reduced.	Order deleted. Dashboard total recalculated correctly.	**PASS**	—
ORD 07	All 5 status values display correct badge colour and label	pending/processing/ shipped/delivered/ cancelled all styled.	All 5 statuses display correct badge text and colour class.	**PASS**	—
ORD 08	Order modal pre-populates on edit	All fields pre-filled with current order values.	All fields pre-populated correctly.	**PASS**	—
ORD 09	Admin subtitle reads 'All customer orders (admin view)'	Subtitle differs for admin vs customer users.	Subtitle set correctly based on isAdmin flag.	**PASS**	—
ORD 10	Order date displayed in readable format	created_at shown as e.g. '14 Jan 2025'.	Date rendered using toLocaleDateString(). Correct format.	**PASS**	—

6. Loyalty Programme Tests

ID	Test Description	Expected Result	Actual Result	Result	Fix / Notes
LOY01	Loyalty Rewards tab loads with 4 seeded rewards	10% Off, £5 Off, Free Shipping, 25% VIP Discount shown.	All 4 rewards loaded with correct types and values.	**PASS**	—
LOY02	Reward type labels render human-readable	'discount_percent' → 'Discount (%)'. 'free_shipping' → 'Free Shipping'.	All 4 type labels rendered correctly.	**PASS**	—
LOY03	Reward value formatted: % for percent, £ for fixed	'10% Off' shows 10%. '£5 Off' shows £5.00.	Formatting logic in renderRewards() applied correctly.	**PASS**	—
LOY04	Admin: Add reward — all valid fields	Reward added. Toast. Count increments.	Reward saved to localStorage. Table updated.	**PASS**	—
LOY05	Admin: Add reward — 0 points required	Toast: 'Points Required must be greater than zero.'	Validation catches zero value. Error shown.	**PASS**	—
LOY06	Admin: Add reward — empty name	Toast: 'Please fill in Name and Points Required.'	Validation triggered. Modal stays open.	**PASS**	—
LOY07	Admin: Edit reward — change points required	Modal pre-populates. Saving shows updated value.	Edit worked correctly. New value displayed.	**PASS**	—
LOY08	Admin: Delete reward — confirm dialog	Confirm dialog. Reward removed on confirm.	Reward deleted. Table updated.	**PASS**	—
LOY09	Customer: Loyalty tab visible but read-only	Customer sees rewards table. No Edit or Delete buttons.	Customer view shows rewards without action buttons.	**PASS**	—
LOY10	Inactive reward shows 'Inactive' badge	Reward with is_active=false shows Inactive badge.	Inactive badge renders with muted styling.	**PASS**	—

7. Dashboard Tests

ID	Test Description	Expected Result	Actual Result	Result	Fix / Notes
DASH01	Dashboard loads after admin login	Stat cards: Products=10, Customers=0, Revenue=£0.00 on fresh load.	Stats loaded correctly from cached data.	**PASS**	—
DASH02	Revenue stat updates when order added	After £50 order added, Revenue stat includes £50.	Revenue recalculated via loadDashboard() after save.	**PASS**	—
DASH03	Bar chart renders proportional bars for product categories	Chart shows bars per category. Tallest bar = most products. Labels visible.	Bar chart rendered correctly using CSS height percentages.	**PASS**	—
DASH04	Donut chart shows customer tier distribution	Conic-gradient donut with Bronze/Silver/Gold/Platinum segments and legend.	Donut renders with correct colours. Centre shows total count.	**PASS**	—
DASH05	Dashboard charts hidden for customer-role users	Charts section not visible to customers.	Charts wrapped in .admin-only class. Not visible to customers.	**PASS**	—
DASH06	Refresh button reloads dashboard data	↻ Refresh calls loadDashboard(). All stats and charts re-render.	Refresh triggers full dashboard reload.	**PASS**	—
DASH07	Customer dashboard shows personal stat cards only	Products count and own orders/spend shown. No customer count or total revenue.	Admin-only stat cards hidden. Customer stats correct.	**PASS**	—
DASH08	Bar chart handles empty product list gracefully	'No data' message shown. No JavaScript crash.	Crashed: Math.max() on empty array returns -Infinity causing NaN bar heights. Fix applied.	**FAIL**	Fix: Changed to Math.max(...entries.map(e=>e[1]), 1) ensuring max >= 1 always. PASS on re-test.

8. Security and Access Control Tests

ID	Test Description	Expected Result	Actual Result	Result	Fix / Notes
SEC01	Customer cannot access Customers section via UI or JS console	Tab hidden. openCustomerModal() guards with isAdmin check.	Confirmed: tab not in DOM. Console call shows 'Admin access required' toast.	**PASS**	—
SEC02	Customer cannot call admin functions from browser console	openProductModal(), openRewardModal() check isAdmin before proceeding.	isAdmin guard confirmed in all three admin modal functions.	**PASS**	—
SEC03	XSS: <script>alert('xss')</script> in product name field	Script rendered as escaped text. No alert fires.	esc() function escapes all HTML entities before DOM insertion. No execution.	**PASS**	—
SEC04	XSS: <img src=x onerror=alert(1)> in description field	Rendered as escaped text. No script executes.	esc() correctly escapes onerror attribute. No execution.	**PASS**	—
SEC05	Admin password not stored in localStorage	Password not persisted to localStorage.	Admin password only checked in-memory during login. Never written to storage.	**PASS**	—
SEC06	Customer cannot see other users' orders	loadOrders() filters by user_id for customer-role users.	Confirmed: customer only sees own user_id-matched orders.	**PASS**	—
SEC07	Supabase RLS: customer cannot read other users' profiles	RLS policy 'Users see own profile' enforces id=auth.uid().	Tested via Supabase SQL Editor: non-admin SELECT returns only own row.	**PASS**	—
SEC08	Supabase RLS: customer cannot INSERT products	'Admins insert products' RLS blocks non-admin INSERT.	Non-admin insert attempt returns 403 permission denied in Supabase.	**PASS**	—
SEC09	Hardcoded admin credentials in localLogin() —	Hardcoded email/password is a security risk outside	Present in localLogin(). Acceptable for offline	**PARTIAL**	Noted as tech debt (Improvement I-01 in Task 3). Production

ID	Test Description	Expected Result	Actual Result	Resu lt	Fix / Notes
	production risk	dev context.	demo only. Must be replaced in production.		requires full Supabase Auth with no hardcoded credentials.

9. UI and Usability Tests

ID	Test Description	Expected Result	Actual Result	Result	Fix / Notes
UI01	Toast notifications — success, error, and info types	Toasts appear top-right, show 3.5s, fade and are removed from DOM.	All three types appear with correct icons. Auto-removed after timeout.	**PASS**	—
UI02	Modal closes on overlay background click	Clicking outside panel closes modal without saving.	Overlay click listener closes modal correctly.	**PASS**	—
UI03	Modal closes on Cancel button click	Modal dismissed. No data saved.	Cancel correctly calls closeModal(). No side effects.	**PASS**	—
UI04	Active tab highlighted in navigation	Clicking tab highlights it. Previous active tab loses highlight.	Active class toggled correctly on all nav tabs.	**PASS**	—
UI05	Nav tabs switch section panels correctly	Each tab reveals correct panel. Only one panel visible at a time.	Section panels toggled via active class. All transitions correct.	**PASS**	—
UI06	User badge shows correct name and role	Top bar shows user name, avatar initial, and Admin or Customer label.	User badge populated from currentProfile on login.	**PASS**	—
UI07	Platform usable on mobile (375px width)	Layout readable. Tables scroll horizontally. Buttons accessible.	Tables have overflow-x: auto. Cards stack. Nav scrolls on mobile.	**PASS**	—
UI08	Platform usable on tablet (768px width)	Two-column stat grid. Tables readable. No overflow issues.	Responsive grid collapses at 768px breakpoint correctly.	**PASS**	—
UI09	Empty state messages when tables have no data	Empty state icon and message shown for each empty table.	Empty state rows render correctly for all tables.	**PASS**	—
UI10	Confirm dialog shown before any delete action	window.confirm() before delete. Cancel aborts.	All delete functions check confirm() return before proceeding.	**PASS**	—
UI11	Colour contrast passes	#1F3864 on #F0F2F5	Measured ratio	**PASS**	—

ID	Test Description	Expected Result	Actual Result	Resu lt	Fix / Notes
	WCAG AA (4.5:1 minimum)	meets minimum contrast ratio.	approximately 9.3:1. Passes WCAG 2.1 AA.		
UI12	Status badges use text label alongside colour (not colour alone)	Each badge contains text (e.g. 'pending', 'shipped').	All badges include text. Not colour-only communication. Meets WCAG 1.4.1.	**PASS**	—

10. Iteration Log — Changes Made During Testing

The following table documents all changes made to source files as a result of test failures. Each entry records the affected file and version, the root cause, the specific code change applied, and the outcome on re-test.

Ref	Date	File / Version	Failed Test	Root Cause	Code Change Applied	Outcome
IT-01	Week 3	`app.js v1.1`	PROD09	saveProduct() had no required-field validation before calling the DB insert function	`Added: if (!name || price<0 || stock<0 || !sku) { showToast(...); return; }`	PROD09 re-tested — PASS. Empty name now blocked with clear error message.
IT-02	Week 3	`app.js v1.1`	PROD10	Price parsed with parseFloat but no minimum value check — negative values accepted silently	Extended same validation block: added price < 0 and stock < 0 checks simultaneously	PROD10 and ORD04 both re-tested — PASS.
IT-03	Week 3	`app.js v1.2`	PROD17	No onerror handler on img elements in renderProducts(). Broken URLs showed default broken-image browser icon.	`Added onerror='this.style.display="none"; this.nextElementSibling.style.display="flex"' to img tag; sibling div shows product initial.`	PROD17 re-tested — PASS. Clean initials placeholder shown.
IT-04	Week 4	`app.js v1.2`	CUST04	saveCustomer() lacked email and full_name validation before calling DB insert	`Added: if (!payload.full_name || !payload.email) { showToast(...); return; }`	CUST04 re-tested — PASS. Both fields validated before save.
IT-05	Week 4	`app.js v1.3`	DASH08	Math.max() called on empty array returns -Infinity, causing NaN bar heights and a JS TypeError crash	`Changed: const max = Math.max(...entries.map(e=>e[1]), 1); — ensures max is always at minimum 1`	DASH08 re-tested — PASS. Empty list shows 'No data' without crash.
IT-06	Week 4	`app.js v1.3`	ORD04	saveOrder() parsed total_amount with parseFloat but did not validate for negative values or NaN	`Added: if (isNaN(total) || total < 0) { showToast('Amount must be a positive number.', 'error'); return; }`	ORD04 re-tested — PASS. Negative amounts rejected.

11. Test Summary

Category	Total	Initial Pass	After Fixes	Notes
Authentication (AUTH)	14	14	14	All passed on first run
Products (PROD)	18	15	18	3 failures fixed: validation, negative price, broken image
Customers (CUST)	12	11	12	1 failure fixed: empty email validation
Orders (ORD)	10	9	10	1 failure fixed: negative amount validation
Loyalty (LOY)	10	10	10	All passed on first run
Dashboard (DASH)	8	7	8	1 failure fixed: Math.max() empty array crash
Security (SEC)	9	8	8	1 partial noted: hardcoded credentials (tech debt)
UI / Usability (UI)	12	12	12	All passed on first run
TOTAL	**83**	**77 (93%)**	**82 (99%)**	**6 failures all resolved; 1 partial noted as tech debt**

12. Source and Asset Log

This log records all content used in the solution that was created by others. For each asset or resource, the log records: the type of content, its source, how it was used in the project, any modifications made, and the relevant intellectual property or legal considerations. This log covers the four main categories specified in the T Level assessment guidance: code snippets, images/media, data content, and third-party libraries.

12.1 Third-Party Libraries and SDKs

Asset ID	Asset / Library	Source	How Used in Project	Modifications Made	IP / Legal Considerations
LIB-01	Supabase JavaScript Client Library (supabase-js v2)	`cdn.jsdelivr.net/npm/@supabase/supabase-js@2`	Imported via CDN <script> tag in index.html. Used to initialise the Supabase client with project URL and anon key, and to call supabase.auth.signInWithPassword(), supabase.from().select(), .insert(), .update(), .delete().	No modifications to the library itself. Custom wrapper functions (loadProducts(), saveProduct() etc.) call the SDK methods.	Apache 2.0 licence — free to use in commercial and educational projects. No attribution required in the application UI, but the licence permits use with or without source distribution. Supabase is open-source. Appropriate for educational T Level use.
LIB-02	Google Fonts — DM Sans and DM Mono (in wireframes)	`fonts.googleapis.com`	Imported in wireframe HTML file (Task 1B) via @import URL. DM Sans used for body text; DM Mono for code snippets and monospace data. Not used in app.js or index.html (which use system fonts).	No modifications. Only specific weights imported (300, 400, 500, 600).	SIL Open Font Licence (OFL) — free to use in any project including commercial use. No payment or attribution required. Fonts served by Google, so Google's privacy policy applies to font requests made by users' browsers.

12.2 Images and Media

Asset ID	Asset	Source / URL	How Used	Modifications	IP / Legal Considerations
IMG-01	Organic Green Tea product photo	`images.unsplash.com /photo-1556881286...`	Stored as image_url in the products seed data (schema.sql). Retrieved in renderProducts() and displayed as a 36×36px thumbnail using an <img> tag with onerror fallback.	Resized to w=400 via Unsplash URL parameter. No other edits.	Unsplash Licence — free for commercial and non-commercial use. No attribution required. Images may not be sold without significant modification. The licence is perpetual and irrevocable. All 10 product images use Unsplash-hosted URLs under the same licence.
IMG-02	Artisan Sourdough Bread product photo	`images.unsplash.com /photo-1585478259715...`	As IMG-01. Seed data only.	Resized to w=400 via URL parameter.	Unsplash Licence — same as IMG-01. All product images in seed data fall under this licence category and are free to use for educational portfolio purposes.

Asset ID	Asset	Source / URL	How Used	Modifications	IP / Legal Considerations
IMG-03–10	Remaining 8 product photos (Electronics, Homeware, Sports ×2, Office, Food, Accessories)	`images.unsplash.com — see schema.sql image_url fields for full URLs`	All used identically to IMG-01 — stored in seed data, rendered as thumbnails in the products table with onerror fallback.	Resized to w=400 via URL parameter only.	Unsplash Licence — as above. These images are illustrative/placeholder content for a student assessment portfolio. No commercial deployment is intended. If the platform were deployed commercially, image licencing would need to be reviewed for each photograph individually.

12.3 Code Snippets and Patterns

Asset ID	Content	Source	How Used	Modifications	IP / Legal Considerations
CODE-01	Supabase Row Level Security policy patterns	`Supabase official documentation: supabase.com/docs/guides/auth/row-level-security`	RLS policy syntax and the is_admin() helper function structure were adapted from the Supabase documentation examples. Applied to all 7 tables in schema.sql to enforce access control at the database level.	Adapted to use a custom is_admin() function rather than the JWT-claim approach in the docs. Policy names and table names changed to match the Retail Innovations schema.	Supabase documentation content is licensed under the Apache 2.0 and MIT licences. Code patterns from documentation are intended for developer use and adaptation. No copyright infringement — this is the intended use of technical documentation.
CODE-02	PostgreSQL generated column syntax (subtotal field)	`PostgreSQL 12+ official documentation: postgresql.org/docs/current/ddl-generated-columns.html`	The GENERATED ALWAYS AS (quantity * unit_price) STORED syntax for the order_items.subtotal column was confirmed against PostgreSQL documentation to ensure correct DDL syntax for Supabase's PostgreSQL instance.	No modification — syntax used verbatim as per the standard. Applied to a single column in the order_items table.	PostgreSQL documentation is freely available and the SQL standard itself is not copyright-protectable. Use of standard SQL syntax in a schema file raises no IP concerns.
CODE-03	CSS conic-gradient donut chart pattern	`MDN Web Docs: developer.mozilla.org/en-US/docs/Web/CSS/conic-gradient`	The conic-gradient approach for the customer tier donut chart in renderDonutChart() was informed by the MDN documentation example for creating pie/donut charts using CSS. The degree calculation logic was written independently to dynamically calculate segment sizes from customer tier counts.	Significant adaptation: the MDN example shows a static gradient. The implementation calculates degree values dynamically from the customer tier distribution data, then applies them as a CSS style string.	MDN Web Docs content is licensed under CC-BY-SA 2.5. The CSS property itself (conic-gradient) is a web standard. No proprietary code was copied — the technique was learned from documentation and independently implemented.
CODE-04	HTML entity escaping	`OWASP XSS`	The esc() function used	Written independently	OWASP content is licensed under CC-BY-SA 4.0.

Asset ID	Content	Source	How Used	Modifications	IP / Legal Considerations
	function (esc())	Prevention Cheat Sheet: owasp.org/www-community/attacks/xss	throughout app.js to sanitise user-supplied strings before inserting them into the DOM was informed by OWASP guidance on preventing XSS via HTML entity encoding. The specific character replacements (&, <, >, ", ') follow the OWASP-recommended minimum.	using the OWASP guidance as a reference. The actual function code was not copied — only the list of characters requiring escaping was used as a checklist.	Security guidance documents are intended to be freely followed and implemented. No copyright concerns.

12.4 Synthetic Test Data

Asset ID	Content	Source	How Used	Modifications	IP / Legal Considerations
DATA-01	10 seed product records (names, prices, SKUs, descriptions)	Authored independently by developer. Product names and descriptions are original fictional content appropriate to a multi-category retail context.	Inserted via INSERT INTO products statement in schema.sql. Loaded into productsCache array on application start. Used across all product-related tests.	N/A — original content.	Entirely original content — no third-party IP involved. Product descriptions are fictional and do not reproduce any real brand's marketing copy. Pricing is realistic for illustrative purposes only and is not taken from any commercial source.
DATA-02	4 seed loyalty reward records	Authored independently. Reward types, point thresholds, and values are original figures created to represent a plausible retail loyalty scheme.	Inserted via INSERT INTO loyalty_rewards statement in schema.sql. Used in Loyalty tab rendering and tested in LOY01–LOY10.	N/A — original content.	Original content. Point thresholds (200, 300, 500, 2000) and discount values (5%, 10%, 25%, £5) are illustrative figures with no basis in any real commercial loyalty scheme.
DATA-03	Test user credentials used during testing	Created by developer during testing. Admin account (admin@retail.com) and customer test accounts are	Used during manual testing (AUTH01–AUTH14 and all role-separation tests). Stored in localStorage only — not in schema.sql.	N/A — original content.	All test credentials are fictional. No real personal data was used at any point during testing. This is consistent with GDPR principles of data minimisation and purpose limitation.

Asset ID	Content	Source	How Used	Modifications	IP / Legal Considerations
		fictional.			

13. Change Log

This change log records all notable changes made to the source files throughout the development of the Retail Innovations Ltd platform. It is not a step-by-step narrative of the build process, but a record of meaningful changes to file versions. Each entry records the date, the file name and version number to which the change applies, a brief description of what changed, and the reason for the change.

Date	File	Version	Description of Change	Reason / Trigger
Week 1	`schema.sql`	`v1.0`	Initial schema created: user_profiles, products, customers, orders, order_items, loyalty_rewards, loyalty_transactions tables. All FK relationships, CHECK constraints, and auto-timestamp triggers defined.	Initial development — schema designed from Task 1B data dictionary.
Week 1	`schema.sql`	`v1.1`	Added Row Level Security policies for all 7 tables. Added is_admin() helper function. Added on_auth_user_created trigger to auto-populate user_profiles on sign-up.	Security requirement from FR8 (Role-Based Access Control). is_admin() centralises the admin check to avoid repeating the full subquery in every RLS policy.
Week 1	`schema.sql`	`v1.2`	Added seed data: 10 product INSERT statements and 4 loyalty reward INSERT statements. Added ON CONFLICT (sku) DO NOTHING clause to prevent duplicate errors on re-run.	Seed data required for testing and demonstration. ON CONFLICT clause added after encountering duplicate key error on second schema execution.
Week 2	`index.html`	`v1.0`	Initial HTML structure created: authentication screen, main app layout with top bar and tab navigation, section panels for Dashboard, Products, Customers, Orders, Loyalty. Modal overlay structure added.	Initial development — translating Task 1B wireframes into HTML markup.
Week 2	`styles.css`	`v1.0`	Initial stylesheet: CSS custom properties for colour scheme (#1F3864, #2E5FAC, #F0F2F5). Typography (system font stack), layout (CSS Grid and Flexbox), responsive breakpoints at 768px and 480px.	Initial development — implementing Task 1B style guide.
Week 2	`app.js`	`v1.0`	Initial JavaScript: Supabase client initialisation, dual-mode detection (if(supabase) conditional), handleAuth() for login/register, enterApp() for role-based UI setup, loadProducts() and renderProducts(), showToast(), closeModal().	Core application logic implementing FR1 (auth), FR2 (products), and FR8 (RBAC) from Task 1A requirements.
Week 3	`app.js`	`v1.1`	Added loadCustomers(), renderCustomers(), openCustomerModal(), saveCustomer(), deleteCustomer(). Added calculateLoyaltyTier() function. Added loadOrders(), renderOrders(), saveOrder(). Added loadRewards(), renderRewards(), saveReward().	Implementing FR3 (Orders), FR4 (Customer Management), FR5 (Loyalty Programme) from requirements.
Week 3	`app.js`	`v1.1`	Added dashboard analytics: loadDashboard(), renderBarChart(), renderDonutChart(). Added local storage fallback implementations for all CRUD operations	Implementing FR6 (Dashboard Analytics). Local storage fallback added to satisfy dual-mode architecture design decision from Task 1A.

Date	File	Version	Description of Change	Reason / Trigger
			(localInsert, localUpdate, localDelete helpers).	
Week 3	`app.js`	`v1.2`	Bug fixes from initial testing (IT-01 to IT-03): added required-field validation in saveProduct(), added price >= 0 and stock >= 0 checks, added onerror fallback handler on product thumbnail img elements.	Test failures PROD09, PROD10, PROD17 identified during testing phase. See Iteration Log IT-01, IT-02, IT-03.
Week 4	`app.js`	`v1.3`	Bug fixes from continued testing (IT-04 to IT-06): added full_name and email validation in saveCustomer(), fixed Math.max() crash in renderBarChart() with empty product array, added negative-value validation in saveOrder().	Test failures CUST04, DASH08, ORD04 identified. See Iteration Log IT-04, IT-05, IT-06.
Week 4	`styles.css`	`v1.1`	Added responsive styles for admin-only elements (.admin-only, body.is-admin .admin-only). Added loyalty tier badge colour classes (.badge-bronze, .badge-silver, .badge-gold, .badge-platinum). Added low-stock warning colour for stock quantity cells.	Admin/customer role separation required CSS-level enforcement to complement JS isAdmin guard. Tier badge colours required for CUST08 test.
Week 5	`app.js`	`v1.4`	Final refinements: improved filterProducts() to combine search text and category filter simultaneously (previously each reset the other). Added category dropdown dynamic population from productsCache. Minor UX improvements to toast message wording.	PROD05 combined filter test revealed that search and category filters were not working together. UX improvements based on user feedback review.
Week 5	`index.html`	`v1.1`	Added aria-label attributes to key form inputs. Added role='alert' to toast container for screen reader support. Added Enter key event listener to auth form. Improved button label specificity (dynamic labels for Add/Edit modal buttons).	Accessibility improvements to partially address WCAG 2.1 AA requirements (NFR5). Enter key handler added in response to AUTH11 test confirmation.
Week 5	`schema.sql`	`v1.3 (FINAL)`	Final version: all tables, RLS policies, triggers, helper functions, seed data confirmed. Added admin promotion comment block (UPDATE user_profiles SET role='admin' WHERE email=...) to guide first-time setup.	Final preparation for submission. Admin promotion comment added as setup guidance per Supabase Auth requirements.