RETAIL INNOVATIONS LTD

Task 2: Code Organisation

Comments • Naming • Modularity • Structure

DPDD Occupational Specialism — Set Task

Document Version 1.0 | January 2025

1. Code Commented Clearly (Including DOCSTRINGS)

All four source files use consistent, structured commenting patterns that enable any developer to understand the codebase quickly.

1.1 app.js — Section Block Comments

The JavaScript file (798 lines) is divided into 14 clearly labelled sections, each opened by an ASCII box-drawing comment block:

```
// ============================================================
// SECTION NAME
// ============================================================
```

The sections are:

- State (lines 6–17) — Global variables: supabase, currentUser, currentProfile, isAdmin, 4 data caches, authMode
- Boot (lines 19–22) — DOMContentLoaded listener calling initNavigation()
- Supabase Connection (lines 24–58) — initSupabaseFromConfig(), checkExistingSession()
- Auth: Login/Register/Logout (lines 73–186) — toggleAuthMode(), handleAuth(), handleLogin(), handleRegister(), handleLogout()
- User Profile + Role (lines 188–215) — loadUserProfile() with fallback defaults
- Enter the App (lines 217–256) — enterApp() with role-based UI setup
- Navigation (lines 258–285) — initNavigation() with tab click handlers and Enter key support
- Toasts (lines 287–299) — showToast() notification system
- Modal Helpers (lines 301–312) — openModal(), closeModal(), background click dismiss
- Load All Data (lines 314–327) — loadAllData() with Promise.all()
- Products CRUD (lines 329–440) — load, render, filter, populate filter, open modal, edit, save, delete
- Customers CRUD (lines 442–530) — load, render, filter, open modal, edit, save, delete
- Orders CRUD (lines 532–653) — load, render, filter, open modal, edit, save, delete
- Loyalty Rewards CRUD (lines 655–737) — load, render, open modal, edit, save, delete

Inline comments explain non-obvious logic, such as: '// RLS handles filtering, but for customers we also set user_id filter as a safety net' (line 542), '// Some Supabase projects require email confirmation' (line 159), and '// Profile might not exist yet if trigger hasn't fired' (line 203).

1.2 index.html — HTML Section Comments

Each major section is delimited by a clear HTML comment:

```
<!-- ==================== LOGIN / REGISTER SCREEN ==================== -->
<!-- ==================== MAIN APP (hidden until logged in)
==================== -->
<!-- ==================== DASHBOARD ==================== -->
<!-- ==================== PRODUCTS ==================== -->
<!-- ==================== CUSTOMERS (admin only) ==================== -->
<!-- ==================== ORDERS ==================== -->
<!-- ==================== LOYALTY ==================== -->
<!-- ==================== MODALS ==================== -->
```

1.3 styles.css — CSS Section Comments

The stylesheet (766 lines) uses matching comment blocks for 17 sections: Auth/Login, Admin Visibility, Layout, Top Bar, Navigation Tabs, Main Content, Buttons, Search & Toolbar, Data Table, Stats Cards, Charts, Loyalty Tier Cards, Modal, Forms, Toasts, Empty State, and Responsive.

1.4 schema.sql — Numbered Section Comments

The SQL file uses numbered sections (1–12) with descriptions: User Profiles Table, Products Table, Customers Table, Orders Table, Order Items Table, Loyalty Rewards Table, Loyalty Transactions Table, Auto-Update Timestamps, Admin Helper Function, Row Level Security, Seed Data, and Make Your First Admin. The final section provides step-by-step admin setup instructions as SQL comments.

2. Uses Appropriate Naming Conventions

2.1 JavaScript Naming

Convention	Pattern	Examples
Variables	camelCase	currentUser, currentProfile, productsCache, customersCache, ordersCache, rewardsCache, authMode
Functions	camelCase verb phrases	loadProducts(), renderCustomers(), filterOrders(), saveReward(), deleteProduct()
Booleans	is- prefix	isAdmin
Event handlers	handle- prefix	handleAuth(), handleLogin(), handleRegister(), handleLogout()
DOM references	Descriptive getElementById	getElementById('productSearch'), getElementById('authEmail')
Constants/State	Descriptive names	authMode = 'login' | 'register'

2.2 HTML Naming

Convention	Pattern	Examples
IDs	camelCase	authScreen, loginForm, productSearch, orderStatusFilter, toastContainer
Classes	kebab-case	auth-screen, nav-tab, stat-card, modal-overlay, form-input, data-table
Data attributes	data- prefix	data-tab="dashboard", data-tab="products"
Modifier classes	BEM-inspired	badge-active, badge-bronze, btn-primary, btn-danger, btn-sm
Admin visibility	Consistent class	admin-only (used on 12+ elements)

2.3 CSS Naming

Convention	Pattern	Examples
Custom properties	Prefixed categories	--color-accent, --font-body, --radius-lg, --shadow-md, --transition
Component classes	Descriptive kebab	.top-bar, .nav-tabs, .stat-card, .chart-card, .data-table-wrap
State modifiers	Suffix pattern	.active, .connected, .leaving, .hidden
Badge variants	Component-modifier	.badge-active, .badge-pending, .badge-bronze, .badge-gold

2.4 SQL Naming

Convention	Pattern	Examples
Tables	snake_case plural nouns	user_profiles, products, customers, orders, order_items, loyalty_rewards, loyalty_transactions
Columns	snake_case descriptive	full_name, loyalty_points, stock_quantity, created_at, updated_at, total_spent
Foreign keys	_id suffix	customer_id, user_id, order_id, product_id
Indexes	idx_ prefix	idx_profiles_role, idx_products_category, idx_customers_email, idx_orders_status
Triggers	trg_ prefix	trg_products_updated, trg_customers_updated, trg_orders_updated, on_auth_user_created
Functions	Descriptive verbs	handle_new_user(), update_updated_at(), is_admin()

3. Modular JavaScript with Local Scope

3.1 State Management

Global state is declared in a single block at the top of app.js (lines 7–17), clearly separated from all function definitions. This makes it immediately obvious what state the application maintains:

```
let supabase = null;          // Supabase client instance
let currentUser = null;       // Supabase auth user object
let currentProfile = null;    // user_profiles row
let isAdmin = false;          // Derived from profile.role
let productsCache = [];       // Local data caches
let customersCache = [];
let ordersCache = [];
let rewardsCache = [];
let authMode = 'login';       // 'login' | 'register'
```

3.2 Functional Module Pattern

Each entity follows an identical self-contained module pattern with consistent function naming:

Function Pattern	Products	Customers	Orders	Rewards
load{Entity}()	loadProducts()	loadCustomers()	loadOrders()	loadRewards()
render{Entity}(data)	renderProducts()	renderCustomers()	renderOrders()	renderRewards()
filter{Entity}()	filterProducts()	filterCustomers()	filterOrders()	—
open{Entity}Modal(item?)	openProductModal()	openCustomerModal()	openOrderModal()	openRewardModal()
edit{Entity}(id)	editProduct()	editCustomer()	editOrder()	editReward()
save{Entity}()	saveProduct()	saveCustomer()	saveOrder()	saveReward()
delete{Entity}(id)	deleteProduct()	deleteCustomer()	deleteOrder()	deleteReward()

3.3 Encapsulated Rendering

Each render function takes its data as a parameter rather than reading directly from the global cache. This enables the same render function to display both full datasets and filtered subsets:

```
// filterProducts() passes filtered subset to same render function:
renderProducts(productsCache.filter(p => ...))
```

This design keeps rendering logic separate from filtering logic, making each independently testable and modifiable.

3.4 Utility Functions

esc(text) (line 790): HTML escaping utility. Used by all render functions to sanitise user content.

cap(s) (line 797): Capitalises first letter of a string. Used for status badge display.

showToast(message, type) (line 291): Creates, displays, and auto-removes toast notifications. Three types: success, error, info.

openModal(id) / closeModal(id) (line 305–306): Toggle .active class on modal overlays. Background click dismissal registered at lines 308–312.

4. Clean Folder Structure (HTML/CSS/JS Separated)

File	Responsibility
index.html (502 lines)	Structure and content — All HTML markup, semantic elements, page templates, modal forms, data attributes
styles.css (766 lines)	Presentation — All visual styling, responsive breakpoints, animations, CSS custom properties, dark theme
app.js (798 lines)	Behaviour — All application logic: Supabase client, auth, CRUD, DOM manipulation, events, data caching
schema.sql (199 lines)	Database — Table definitions, constraints, indexes, triggers, RLS policies, seed data, admin setup

This separation follows the principle of separation of concerns:

- HTML contains zero <style> or inline styles — all styling is in styles.css
- HTML contains zero inline <script> blocks — all logic is in app.js (onclick attributes reference functions defined in app.js)
- CSS contains zero JavaScript logic — dynamic behaviour is handled via class toggling (.active, .hidden, .is-admin)
- app.js references CSS classes by name only, never modifying CSS rules directly
- schema.sql is independent of all frontend code — can be run in any PostgreSQL environment

The index.html file loads its dependencies cleanly: <link rel="stylesheet" href="styles.css"> in <head>, <script src="app.js"></script> at the bottom of <body> (after DOM is rendered), and the Supabase library from CDN with async attribute to prevent blocking.