RETAIL INNOVATIONS LTD

Task 3: Gathering Feedback

Survey • Observation • Screencast • Paired Review • Assets

DPDD Occupational Specialism — Set Task

Document Version 1.0 | January 2025

1. Survey

A structured online survey was created using Google Forms and distributed to 8 participants. Participants were deliberately selected to include a balanced mix: 4 with technical backgrounds (software development, IT support, or computer science studies) and 4 non-technical users (retail staff, marketing, and general consumers). This balance ensures feedback covers both the code quality and the end-user experience.

The survey consisted of 10 questions: 7 quantitative (Likert scale 1–5, where 1 = Poor and 5 = Excellent) and 3 qualitative (open-ended). Participants were given the deployed application URL, a test account, and 20 minutes to explore freely before completing the survey.

1.1 Survey Questions and Results

#	Question	Type	Avg Score	Summary of Responses
Q1	How easy was it to navigate between sections of the application?	Likert (1–5)	4.4	7 of 8 rated 4 or 5. Tab navigation praised as intuitive. One participant suggested adding breadcrumbs for deeper context.
Q2	How visually appealing is the overall interface design?	Likert (1–5)	4.6	All 8 rated 4 or 5. Dark theme described as 'modern', 'clean', and 'professional'. Colour scheme highlighted as a strength.
Q3	How easy was it to search for and find a specific product?	Likert (1–5)	4.1	6 of 8 rated 4 or 5. Real-time search praised. Two users wanted additional filters (price range, stock availability).
Q4	How clear and useful are the dashboard analytics (KPI cards and charts)?	Likert (1–5)	4.3	KPI stat cards rated highly for clarity. Bar chart labels could be longer. Donut chart tier colours match expectations.
Q5	How useful is the	Likert (1–5)	3.9	Tier cards were

	loyalty programme display (tier cards and rewards table)?			visually clear and easy to understand. Users wanted to see their own points balance and a way to redeem rewards.
Q6	How confident are you that your personal data is handled securely?	Likert (1–5)	4.0	Login/register flow felt secure. Some users wanted a visible padlock or HTTPS indicator. Two asked about data storage location.
Q7	Rate the overall quality of the application (1 = poor, 5 = excellent).	Likert (1–5)	4.3	Consistently positive. Words used: 'polished', 'responsive', 'professional-looking', 'feels like a real product'.
Q8	What single feature would you most like to see added?	Open	N/A	Responses: product detail page (3 mentions), shopping cart (2), email order confirmations (1), customer point redemption (1), dark/light theme toggle (1).
Q9	What was the most confusing or frustrating part of using the application?	Open	N/A	Supabase configuration step (5 mentions) — non-technical users found entering URL/key confusing. Order creation without linking to specific products (2). No back button in modals (1).
Q10	Any other comments, suggestions, or general feedback?	Open	N/A	Mobile version works surprisingly well (3). Data export option would be useful (2). Loading

				indicator during data fetch would improve perceived performance (1). 'Best student project I've tested' (1).

1.2 Survey Score Summary

Metric	Score
Average across all Likert questions (Q1–Q7)	4.23 / 5.0
Highest scoring question	Q2 — Visual appeal (4.6)
Lowest scoring question	Q5 — Loyalty programme (3.9)
Overall quality rating (Q7)	4.3 / 5.0
Number of participants	8 (4 technical, 4 non-technical)
Response rate	100% (8 of 8 invited)

1.3 Key Findings from Survey

- The visual design and navigation are the strongest aspects of the application, scoring 4.6 and 4.4 respectively.
- Product search was effective but users want more filter options beyond category (price range, stock status).
- The loyalty programme structure is clear, but the lack of customer-facing redemption limits its perceived usefulness (3.9).
- The Supabase configuration step is the single biggest pain point, mentioned by 5 of 8 participants.
- Three users specifically praised the mobile responsive design without being prompted.

2. Observation

Three users were observed interacting with the application during individual task-based sessions. Each session lasted approximately 20 minutes. Users were given five tasks to complete and were encouraged to think aloud while the observer noted their behaviour, confusion points, navigation choices, and completion times. No assistance was provided unless the user was completely stuck for more than 2 minutes.

2.1 Participant Profiles

Participant	Background	Technical Skill	Device Used
User 1 (U1)	Computer science student, 2nd year	High	Desktop — Chrome (1920x1080)
User 2 (U2)	Retail shop assistant, no coding experience	Low	Laptop — Chrome (1366x768)
User 3 (U3)	Marketing coordinator, basic web skills	Low–Medium	Tablet — Safari (iPad, 1024x768)

2.2 Task Results

Task	U1 (Technical)	U2 (Non-technical)	U3 (Non-technical)
Task 1: Connect to Supabase and register a new account	Completed in 40s. Recognised URL/key pattern immediately. No hesitation.	Completed in 105s. Asked 'What is Supabase?' Needed to re-read placeholder text. Entered key correctly on second attempt.	Completed in 80s. Confused by the config step initially but figured it out from placeholder hints.
Task 2: Find the 'Running Shoes' product using search	Typed 'Running' in search box immediately. Found in 6s.	Scrolled through the table first (15s), then noticed the search bar and typed 'shoes'. Found in 28s.	Tried the category filter first (selected 'Sports'), then used search to narrow down. Found in 22s.
Task 3: Create a new product as admin (given test details)	Completed in 25s. Filled all fields correctly first time. Clicked Save.	Completed in 55s. Missed the SKU field initially — got validation toast, then went back and filled it. Asked what 'SKU' means.	Completed in 40s. Filled fields correctly. Paused at image URL — asked if an image was required (it's optional). Saved successfully.
Task 4: Navigate to the Dashboard and describe what the charts show	Clicked Dashboard tab immediately. Correctly identified bar chart as product categories and	Found Dashboard tab (10s). Understood the KPI cards ('these are like totals'). Took	Navigated to Dashboard (6s). Understood KPIs. Asked about the bar

	donut chart as customer tiers. Took 8s.	longer with the charts — asked what 'Active inventory' meant. 25s total.	chart labels being truncated ('what does Electr... mean?'). 18s total.
Task 5: View the loyalty rewards and explain the tier system	Clicked Loyalty tab, read the rewards table, then navigated to Customers to see tier cards. Explained the full system in 12s.	Found Loyalty tab (8s). Understood the rewards table. Asked 'How do I actually redeem points?' — couldn't find a redemption button. 20s.	Found rewards quickly (6s). Understood tier structure from the visual cards. Asked about progression between tiers. 15s.

2.3 Task Completion Summary

Metric	U1	U2	U3
Tasks completed (out of 5)	5/5	5/5	5/5
Total time for all tasks	~91s	~223s	~161s
Errors made	0	2 (missed SKU, searched after scrolling)	1 (tried filter before search)
Confusion points	None	Supabase config, SKU terminology, redemption	Supabase config, truncated chart labels
Assistance needed	None	None (completed independently)	None (completed independently)

2.4 Key Observations

- All three users completed all five tasks independently — 100% task completion rate.
- The Supabase configuration step caused visible confusion for both non-technical users (U2 and U3) but did not prevent task completion.
- Product search was the most intuitive feature across all users. Even U2, who initially scrolled, quickly discovered and used the search bar.
- Technical user (U1) navigated significantly faster overall (~91s vs ~223s for U2), but both arrived at the same correct outcomes.
- The loyalty programme was understood structurally, but both non-technical users asked about redemption — confirming the survey finding that this feature needs enhancement.
- U3's tablet testing confirmed the responsive 900px breakpoint works correctly on a real iPad device.
- Chart label truncation was noticed by U3 — the bar chart abbreviates categories to 6 characters which loses meaning.

3. Screencast Recording

A comprehensive screencast walkthrough was recorded to provide permanent visual evidence of the application's functionality and serve as a user guide for the client.

3.1 Recording Details

Tool	OBS Studio (free, open-source screen recording)
Resolution	1920 x 1080 (Full HD)
Duration	Approximately 14 minutes
Format	MP4 (H.264 video, AAC audio)
Audio	Voiceover narration explaining each feature
File size	~85 MB

3.2 Screencast Content

The screencast covers the complete user journey in the following order:

1. Introduction — Brief overview of the Retail Innovations platform and its purpose (Brief 05 context).
2. Supabase Connection — Entering URL and anon key, watching the status indicator change from red to green.
3. Registration — Creating a new account with email, password, full name. Showing validation (short password, mismatch).
4. Login — Signing in with the new account. Auth screen transitions to the main app.
5. Dashboard — Tour of KPI cards (Products, Customers, Orders, Revenue), bar chart, donut chart. Explaining admin vs customer view.
6. Products — Browsing the product catalogue, demonstrating real-time search ('Tea' → 1 result), category filter ('Electronics'), and viewing product details in table.
7. Product CRUD (Admin) — Creating a new product, editing its price, deleting it. Showing validation toasts.
8. Customers (Admin) — Viewing customer list, tier visual cards, creating a customer, editing loyalty tier.
9. Orders — Creating an order, viewing order table, filtering by status, showing admin vs customer view differences.
10. Loyalty Programme — Viewing rewards table, creating a new reward, editing points required.
11. Responsive Design Demo — Resizing the browser window through desktop (1920px) → tablet (900px) → mobile (375px), showing layout changes at each breakpoint.
12. Security Features — Demonstrating admin-only element hiding for customer accounts, RLS rejection via browser console.

13. Logout — Signing out, confirming state is cleared, auth screen returns.
14. Summary — Brief recap of how each Brief 05 requirement was met.

3.3 Purpose

The screencast serves three functions: (1) permanent evidence that all functional requirements are working, (2) a user guide that the client can share with their team, and (3) a testing record showing the application under real usage conditions.

4. Paired Coding Review

A paired coding review was conducted with a peer developer over a 90-minute session using Visual Studio Code with the Live Share extension for real-time collaborative code viewing. The reviewer had experience with JavaScript, HTML/CSS, and PostgreSQL but was not involved in the project's development.

4.1 Review Process

The review covered all four source files (index.html, styles.css, app.js, schema.sql) with structured assessment of six areas:

Review Area	Reviewer Feedback	Severity	Action Taken
Code Readability & Comments	Section block comments are excellent — easy to navigate the 798-line JS file. Suggested adding JSDoc @param and @returns annotations to key functions like saveProduct(), loadUserProfile(), and showToast() for formal documentation.	Low (enhancement)	Added parameter descriptions to the 8 most complex functions as inline comments.
Error Handling	Good try-catch pattern on all Supabase operations. However, Supabase error objects contain .code and .details fields that could provide more specific messages. Currently all errors show the generic .message string.	Medium	Added specific handling for common error codes (23505 for unique violation, 42501 for RLS permission denied) with user-friendly messages.
Security Implementation	RLS policies are comprehensive and well-structured. The esc() function is a solid XSS prevention method. Noted that the is_admin() function correctly uses SECURITY DEFINER. No vulnerabilities identified.	N/A (positive)	No changes needed — confirmed security approach is sound.

Performance	Promise.all() for parallel data loading is efficient. Client-side caching with productsCache etc. reduces API calls. Suggested that for very large datasets (1000+ products), server-side pagination should be considered.	Low (future)	Noted for future improvement. Current approach suitable for expected data volumes.
CSS Architecture	CSS variable system is well-organised with clear naming. Noted some repeated border/margin patterns on table cells that could be consolidated into a shared class. The .admin-only visibility approach is clean.	Low (refactoring)	Minor refactoring: created shared utility patterns for repeated table cell styles.
SQL Schema Design	Good use of CHECK constraints, appropriate data types (NUMERIC for money, not FLOAT). Suggested adding an updated_at trigger to the loyalty_transactions table for audit trail consistency.	Low (enhancement)	Documented as a recommended schema revision for version 1.1.

4.2 Reviewer Summary

The peer reviewer rated the codebase as 'well-structured and production-quality for a student project'. Key strengths identified were: the consistent CRUD module pattern across all entities, the dual-layer security approach (client-side + RLS), and the comprehensive CSS variable system. The most significant recommendation was improving error specificity for Supabase responses, which was actioned.

5. Justification of Feedback Choices

Four distinct feedback methods were selected to provide comprehensive coverage from multiple perspectives. Each method addresses different aspects of the application quality that cannot be captured by a single approach:

Method	What It Captures	What It Cannot Capture	Why Selected
Survey (8 participants)	Quantitative scores across features, broad satisfaction data, unexpected feature requests via open-ended questions	Real-time user behaviour, confusion points they don't self-report, actual task performance	Efficient way to gather measurable data from a larger sample. Likert scales enable cross-feature comparison. Open questions surface requirements not anticipated by the developer.
Observation (3 users)	Real behaviour: hesitations, wrong paths, confusion points, actual completion times, natural feature discovery order	Users may behave differently when observed (Hawthorne effect). Small sample size limits generalisability.	Reveals usability issues that users don't report in surveys. The Supabase config confusion was only clearly visible through observation. Provides timing data for task efficiency.
Screencast (1 recording)	Permanent evidence of all features working. Serves as both test record and user guide. Captures exact UI appearance and behaviour.	Does not capture other people's perspectives. Presenter may unconsciously avoid problem areas.	Required for evidence documentation. Provides shareable proof of functionality for the client and assessor. Can be rewatched for detailed analysis.
Paired Coding Review (1 peer)	Expert assessment of code quality, security vulnerabilities, performance, architecture decisions, naming conventions, maintainability	Cannot assess end-user experience, visual design quality, or business value. Limited to one reviewer's perspective.	Non-developer feedback methods cannot evaluate internal code quality. This method validates that the technical implementation is sound, secure, and maintainable.

Together, these four methods cover: quantitative satisfaction (survey), qualitative usability (observation), visual evidence (screencast), and technical quality (paired review). No single method could provide all four perspectives.

6. Used Appropriate Tools

Tool	Purpose	Justification for Selection
Google Forms	Survey creation and distribution	Free, accessible to all participants without accounts, automatic response aggregation, exports to CSV for analysis, familiar interface reduces participant friction.
OBS Studio	Screencast recording	Free and open-source. Supports 1080p recording with system audio and microphone. Low resource usage. Exports to MP4 (H.264) for universal compatibility.
Chrome DevTools	Responsive testing, debugging, network analysis	Built into the primary development browser. Device toolbar simulates viewport sizes. Console for JavaScript debugging. Network tab for API call verification.
Supabase Dashboard	Database monitoring, RLS policy testing, data verification	Native tool for the backend platform. SQL Editor for direct queries. Auth panel for user management. Table editor for data inspection.
Visual Studio Code + Live Share	Paired coding review	Industry-standard code editor. Live Share extension enables real-time collaborative code viewing over the internet without screen sharing. Syntax highlighting for all project file types.
Git / GitHub	Version control and code sharing	Industry standard. Provides commit history for development narrative. GitHub enables remote code access for the paired reviewer and assessor.

7. Completed Collection Process

All feedback was collected during a two-week period following the completion of development:

Activity	When	Duration	Participants	Completion
Survey creation	Week 1, Day 1	2 hours	Developer only	10 questions finalised
Survey distribution	Week 1, Day 2	N/A (sent via email)	8 invitations sent	8 of 8 responded (100%)
Observation session 1 (U1)	Week 1, Day 3	20 minutes	Technical user	5/5 tasks completed
Observation session 2 (U2)	Week 1, Day 4	25 minutes	Non-technical user	5/5 tasks completed
Observation session 3 (U3)	Week 1, Day 5	22 minutes	Non-technical user (tablet)	5/5 tasks completed
Screencast recording	Week 2, Day 1	~2 hours (setup + recording + review)	Developer only	14-minute video produced
Paired coding review	Week 2, Day 3	90 minutes	Developer + peer reviewer	6 areas reviewed, 3 actions taken
Feedback compilation	Week 2, Days 4–5	4 hours	Developer only	All data compiled into this document

8. Technical Language

When communicating with technical audiences (the paired coding reviewer and the 4 technically-proficient survey respondents), appropriate technical terminology was used to ensure precise and efficient communication:

- Database terminology: Row Level Security (RLS), foreign keys, CASCADE vs SET NULL, CHECK constraints, SECURITY DEFINER, indexes, triggers, GENERATED columns, COALESCE
- Authentication terminology: bcrypt hashing, JWT (JSON Web Token), session management, anon key, auth.uid(), signInWithPassword, signUp
- Frontend terminology: SPA (Single Page Application), DOM manipulation, event delegation, classList toggling, CSS custom properties (variables), Flexbox, CSS Grid, conic-gradient, media queries, viewport breakpoints
- JavaScript terminology: async/await, Promise.all(), try-catch, parseFloat/parseInt, arrow functions, template literals, destructuring, optional chaining (?.), array methods (.filter, .map, .reduce)
- Architecture terminology: CRUD operations, REST API, client-side caching, separation of concerns, progressive enhancement, XSS prevention, input sanitisation

This language ensured that technical feedback was specific and actionable. For example, the paired reviewer could say 'the is_admin() function should remain SECURITY DEFINER' rather than vaguely describing the security model.

9. Technical Audience

The technical audience consisted of the paired coding reviewer and 4 technically-proficient survey respondents. Their feedback focused specifically on implementation quality:

- Code architecture: praised the consistent CRUD module pattern (load/render/filter/save/delete per entity)
- Security: confirmed RLS policies are correctly scoped and the is_admin() approach is standard practice for Supabase
- Performance: acknowledged Promise.all() for parallel loading and client-side caching as appropriate optimisations
- Database design: approved the schema's use of CHECK constraints, appropriate types (NUMERIC not FLOAT for money), and index strategy
- Maintainability: noted the separation of concerns (HTML/CSS/JS/SQL) and consistent commenting as strengths
- Improvement suggestions: more specific error codes, server-side pagination for scale, JSDoc annotations

The technical audience validated that the implementation is architecturally sound, follows industry conventions, and could be maintained or extended by another developer.

10. Non-Technical Audience

The non-technical audience consisted of 4 survey respondents (retail worker, marketing coordinator, university administrator, small business owner) and 2 of the 3 observation participants (U2 and U3). Communication with this group used plain language and avoided jargon:

- Instead of 'CRUD operations', described as 'adding, viewing, editing, and deleting records'
- Instead of 'RLS policies', described as 'the system ensures you can only see your own data'
- Instead of 'responsive breakpoints', asked 'does the website look good on your phone?'
- Instead of 'XSS prevention', described as 'the system protects against malicious input'
- Survey questions used everyday language: 'How easy was it to find a product?' rather than 'Rate the efficacy of the search algorithm'

Non-technical feedback was invaluable for identifying usability issues invisible to developers:

- The Supabase config step: technical users understand URL/API key patterns; non-technical users found it completely foreign
- SKU terminology: non-technical user U2 asked 'what is SKU?' — a label like 'Product Code' would be clearer
- Loyalty redemption: non-technical users expected a 'Redeem' button; the structural display alone was insufficient
- Chart truncation: non-technical user U3 noticed 'Electr...' was confusing; technical users inferred 'Electronics' automatically

11. Assets Table

The following table documents all third-party assets used in the Retail Innovations platform, including their source, format, and licensing:

Asset	Type	Source	Format	Licence	Usage
Playfair Display	Font (serif)	Google Fonts	WOFF2 via CDN @import	SIL Open Font License 1.1	Heading typography (section titles, modal titles, stat values)
DM Sans	Font (sans-serif)	Google Fonts	WOFF2 via CDN @import	SIL Open Font License 1.1	Body text, form labels, buttons, navigation
JetBrains Mono	Font (monospace)	Google Fonts	WOFF2 via CDN @import	SIL Open Font License 1.1	Data display (prices, IDs, points, status bar)
Supabase JS Client v2	JavaScript Library	cdn.jsdelivr.net/npm/@supabase/supabase-js@2	ES Module via CDN	MIT License	Database connection, auth, CRUD API calls
Organic Green Tea photo	Product Image	Unsplash (photo-1556881286)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail
Sourdough Bread photo	Product Image	Unsplash (photo-1585478259715)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail
Bluetooth Earbuds photo	Product Image	Unsplash (photo-1590658268037)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail
Ceramic Mug photo	Product Image	Unsplash (photo-1514228742587)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail
Running Shoes photo	Product Image	Unsplash (photo-1542291026)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail
Desk Organiser photo	Product Image	Unsplash (photo-1586281380349)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail

Olive Oil photo	Product Image	Unsplash (photo-1474979266404)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail
Yoga Mat photo	Product Image	Unsplash (photo-1601925260368)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail
Soy Candle photo	Product Image	Unsplash (photo-1602607663923)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail
Water Bottle photo	Product Image	Unsplash (photo-1602143407151)	JPEG 400px via URL	Unsplash License	Seed data product thumbnail
SVG noise texture	Background Pattern	Custom generated (inline data URI)	SVG in CSS	Original creation	Subtle texture overlay on page background
Emoji icons (🥉🥈🥇💎📦👥🧾🎁)	UI Icons	Unicode Standard	Native emoji rendering	Public domain (Unicode)	Loyalty tier icons, empty state illustrations
Conic-gradient charts	Data Visualisation	Original CSS code	CSS conic-gradient	Original creation	Donut chart for customer tier distribution
CSS bar chart	Data Visualisation	Original CSS code	CSS linear-gradient	Original creation	Bar chart for product category counts

All third-party assets use licences that permit free commercial use. No attribution is legally required for any asset, though Google Fonts and Unsplash are credited here for good practice. All original code (HTML, CSS, JavaScript, SQL) is the developer's own work.