Mini-Commerce — 3 - 5 Day Front-End Technical Assessment

Target role: Junior–Mid React / Next.js Developer
Stack to use (mandatory): Next.js 14 (App Router) · React · **React Query** · **Zustand** · **Tailwind CSS** · **TypeScript (strict mode)**
Data layer: localStorage only (no backend)

---

1. Product Goal

Deliver a client-side prototype of a tiny e-commerce shop called **Mini-Commerce**. A visitor can browse products, manage a cart, and finish a mock checkout. All state must survive page reloads.

---

2. Required Features

#	Feature	Route	Key Details
1	**Catalogue**	`/`	List at least 8 dummy products (image, name, price). Fetch via React Query from a local JSON file and seed to localStorage.
2	**Product Detail**	`/product/[slug]`	Full product info + *Add to Cart* button.
3	**Cart**	`/cart`	View items, change quantity, remove, show subtotal & total. State lives in Zustand and persists to localStorage.
4	**Checkout Flow**	`/checkout` → success page	Show order summary → *Place Order* clears cart and displays “Thank you” with a random order ID.

Freedom to enhance: search, filters, dark-mode, animations, etc., are welcome but optional.

---

3. Design & UX Expectations

- Clean, modern look; mobile-first responsive grid/flex layouts.
- Use Tailwind utilities; feel free to add custom classes, variables, or design tokens.

- Accessibility: semantic HTML, keyboard-reachable actions, alt text on images.

---

4. Technical Requirements

1. **TypeScript** with `"strict": true`; zero `any` types.
2. **React Query** for catalogue fetching, caching, refetching, and built-in loading/error states.
3. **Zustand** global store for cart (derive totals with selectors; persist via middleware).
4. **Error Handling**: graceful UI for failed catalogue fetch, cart edge cases, and unknown routes.
5. **SEO** *(added value)*: meta tags, Open Graph, structured data, and image optimisation with `next/image`.
6. **Testing**: at least one React component test (Jest + RTL) or Playwright happy-path e2e.
7. **Linting / Formatting**: ESLint & Prettier must pass.

---

5. Deliverables

Item	Requirement
Live Demo	Host on a free, public platform (e.g., Vercel, Netlify, Cloudflare Pages). Provide the URL.
GitHub Repository	Push all source code; keep a clear, incremental commit history.
README.md (max 2 pages)	Include: • **Project Overview** — what you built • **Design Approach** — layout, color, responsiveness decisions • **Tools & Techniques** — libraries, patterns, testing, CI (if any) • **SEO Strategy** — tags, structured data, performance tweaks • **Error-Handling Technique** — how errors are surfaced, logged, recovered
tsconfig & ESLint configs	Show strict typing and code-quality rules.

---

6. Evaluation Rubric (100 pts)

Category	Points	Focus
React / Next fundamentals	25	App Router usage, dynamic routing, data-fetch patterns
State management (Zustand)	20	Store structure, selectors, persistence logic
Data handling (React Query)	15	Query setup, caching, loading/error UI
Styling & UX (Tailwind)	10	Responsiveness, visual polish, accessibility
Code Quality & Types	15	Strict typing, lint-free, test(s) passing
Documentation & DX	10	Setup ease, clear README, commit hygiene
Extra polish (SEO, stretch)	5	Dark mode, search, CI/CD, performance touches

---

8. Submission

Send us:

1. GitHub repo link.

2. Live demo URL.
3. Submit assessment to using this google form https://forms.gle/RPfh7vxXqVJ9Rwk3A