CASE STUDY

TheSkinProof

Bangladesh's First Verified Skincare Marketplace

124	779	26
API ENDPOINTS	TESTS PASSING	DATABASE TABLES

Next.js 16 · TypeScript 5 · PostgreSQL · Redis · Tailwind CSS v4 · Zustand 5

E-commerce Platform · Full-Stack · 2024 – 2025

Contents

01 — EXECUTIVE SUMMARY

Overview

TheSkinProof is a full-stack e-commerce marketplace purpose-built to solve the single biggest problem in Bangladesh's online skincare market: **trust**. In a landscape flooded with counterfeit products and unverified sellers, TheSkinProof introduces a verification-first approach — every product listed on the platform undergoes invoice checks, Certificate of Analysis (COA) validation, batch tracking, and expiry monitoring before it reaches a customer.

The platform serves four distinct user roles — buyers, sellers, administrators, and warehouse operators — each with a dedicated portal, tailored workflows, and role-based access controls. A standout feature is the **AI-powered Skin Quiz**, which analyzes a user's skin type, concerns, allergies, and lifestyle to generate personalized product recommendations with match-score explanations.

124 RESTFUL API ROUTES	**26** DATABASE TABLES	**37** TEST SUITES
779 TESTS PASSING	**75+** REACT COMPONENTS	**45+** PAGE-LEVEL VIEWS

02 — PROBLEM STATEMENT

Why This Platform Exists

Bangladesh's skincare market has grown rapidly, but online platforms have failed to keep pace with consumer trust expectations.

Counterfeit Products

An estimated 30–40% of skincare products sold online in South Asia are counterfeit or expired, posing serious health risks including allergic reactions and skin damage.

No Verification Standard

Existing marketplaces treat skincare like any other product — no invoice verification, no batch tracking, no expiry monitoring.

Decision Paralysis

Consumers struggle to choose products suited to their specific skin type and concerns without professional guidance.

COD Fraud

Cash-on-delivery orders are plagued by velocity abuse, fake names, and rapid-fire ordering that create significant operational losses.

> *How do you build a skincare marketplace where every product is verified authentic, every recommendation is personalized, and every transaction is protected against fraud — while keeping the experience seamless for all stakeholders?*

03 — SOLUTION ARCHITECTURE

System Design

High-Level Architecture

```
CLIENT LAYER
┌──────────┐  ┌──────────┐  ┌──────────┐  ┌───────────────┐
│  Buyer   │  │  Seller  │  │  Admin   │  │  Warehouse    │
│  Portal  │  │  Portal  │  │  Portal  │  │  Portal       │
└────┬─────┘  └────┬─────┘  └────┬─────┘  └──────┬────────┘
     └─────────────┴──────┬──────┴───────────────┘
                          │
             Zustand State Management (6 stores)
         cart · wishlist · auth · chat · notifications · ui
                          │
               ┌──────────┴──────────┐
               │    Next.js 16 App   │
               │    Router (SSR)     │
               └──────────┬──────────┘
                          │
API LAYER ── 124 Endpoints · Rate Limiting · RBAC · Session Auth
 ┌─────────┐  ┌─────────┐  ┌─────────┐  ┌──────────┐
 │  Auth   │  │Products │  │ Orders  │  │  Admin   │  ...
 │  APIs   │  │  APIs   │  │  APIs   │  │  APIs    │
 └────┬────┘  └────┬────┘  └────┬────┘  └────┬─────┘
      └────────────┴────────────┴────────────┘
                          │
SERVICE LAYER
 Skin Quiz Engine · Fraud Detection · COD Eligibility
 Notification Service · RBAC Permission Engine
                          │
DATA & INTEGRATION LAYER
 ┌────────────┐ ┌───────┐ ┌───────┐ ┌────────┐ ┌──────────┐
 │ PostgreSQL │ │ Redis │ │  S3   │ │Pathao  │ │  bKash   │
 │ 26 tables  │ │ Cache │ │ Files │ │Courier │ │  Nagad   │
 └────────────┘ └───────┘ └───────┘ └────────┘ │  SSLComm │
                                                └──────────┘
```

Technology Decisions

LAYER	TECHNOLOGY	RATIONALE
Framework	Next.js 16 (App Router)	SSR for SEO, API routes colocation, React Server Components
Language	TypeScript 5 (strict)	Type safety across 6,500+ lines, compile-time bug detection
Database	PostgreSQL 14+	Relational integrity for e-commerce, UUID support, full-text search
Cache	Redis 4.7	Sliding-window rate limiting, session caching, in-memory fallback
State	Zustand 5	Lightweight persistent stores (cart/wishlist survive page reloads)
Auth	HMAC-SHA256 signed cookies	No third-party dependency, tamper-proof, timing-safe validation
Payments	bKash + Nagad + SSLCommerz	Covers 95%+ of Bangladesh's digital payment methods
Courier	Pathao API	Largest courier in Bangladesh, real-time tracking
Styling	Tailwind CSS v4	Utility-first, custom design tokens, responsive mobile-first

04 — KEY FEATURES

Deep Dive

4.1 — Product Verification Pipeline

The core differentiator. Every product goes through a multi-stage verification before appearing on the marketplace:

1. **Seller Submits Product Draft**

   Uploads product details, images, pricing, and ingredient information through the seller portal.

2. **Document Upload (S3 Storage)**

   Seller attaches Invoice, Certificate of Analysis (COA), and Brand Authorization documents.

3. **Admin Review Queue**

   Admin team manually inspects documents, cross-references batch numbers, and validates authenticity.

4. **Approved → Goes Live | Rejected → Revision Required**

   Approved products appear on the marketplace with a verified badge. Rejected items return to the seller with specific reason codes.

What gets verified:

- **Invoice authenticity** — original purchase invoice from authorized distributor
- **Certificate of Analysis (COA)** — lab results confirming product composition
- **Batch information** — manufacturing date, expiry date, batch number
- **Brand authorization** — proof the seller is authorized to sell the brand

4.2 — Personalized Skin Quiz Engine

An 8-question quiz that generates a complete skin profile and product recommendations. The scoring engine has **90% test coverage** — the highest in the codebase.

QUESTION	OUTPUT
Q1: Top concerns (ranked)	Weighted concern tags
Q2: Skin feel	Skin type classification
Q3: Sensitivity check	Sensitivity level (low / medium / high)
Q4: Allergies	Ingredient exclusion list
Q5: Sunscreen usage	SPF recommendation flag
Q6: Lifestyle factors	Environmental concern tags
Q7: Pregnancy status	Safety exclusions (retinol, etc.)
Q8: Additional concerns	Secondary concern tags

Scoring Algorithm (5 stages):

1 **Tag Weight Computation**

Each answer maps to weighted tags (e.g., ranking 1st concern = weight 3, 2nd = weight 1).

2 **Skin Type Determination**

Compares aggregate scores across dry/oily/combination/normal with concern-based adjustments.

3 **Concern Prioritization**

Top 2 concerns become primary, next 2 secondary.

4 **Exclusion Generation**

Allergies and pregnancy flags generate an ingredient blacklist.

5 **Product Matching**

Products scored: skin type match (+20), primary concern (+30), sensitivity compatibility (+15), verified status (+5). Top 2 per slot recommended.

4.3 — Fraud Detection & COD Risk Management

A multi-signal fraud detection system designed for Bangladesh's COD-heavy market:

SIGNAL	THRESHOLD	RISK LEVEL	ACTION
Velocity Abuse	5+ orders / same phone / 24h	**HIGH**	Block order
Name Mismatch	3+ different names / same phone / 30d	**MEDIUM**	Flag for review
Rapid-Fire Orders	2+ orders in 10 minutes	**MEDIUM**	Flag for review
[illegible]	[illegible] 10,000 BDT	[illegible]	[illegible]

COD Eligibility Engine: Checks block rules by phone number, district, and IP pattern. Computes user risk score (0–100). Blocks COD if risk score exceeds 75.

4.4 — Role-Based Access Control (RBAC)

A granular permission system supporting 15+ modules and 50+ discrete actions:

```
Super Admin ──→ Full access to all modules
Verifier    ──→ Products (view, verify), Product Drafts (view, verify)
Operations  ──→ Orders (view, update), Warehouse (all), Payouts (view)
Custom Role ──→ Any combination of module:action permissions
```

Permission Modules:

users sellers products product_drafts orders roles payouts settings audit_logs newsletter warehouse notifications risk_management reports returns

4.5 — Multi-Portal Architecture

PORTAL	PAGES	KEY FEATURES
Buyer	Catalog, quiz, cart, checkout, orders, returns, reviews, account	Personalized recommendations, real-time tracking, WhatsApp notifications
Seller	Dashboard, drafts, orders, payouts, analytics	Revenue stats, draft workflow, commission tracking (15% default)
Admin	30+ pages: dashboard, products, sellers, orders, users, RBAC, risk, warehouse, newsletter, audit logs, settings	Complete operational control, fraud dashboard, batch management
Warehouse	Picking board, inventory, batch management	FIFO allocation, expiry tracking, pick-pack workflow

05 — SECURITY

Architecture

Security is treated as a first-class concern across every layer of the platform.

Authentication

OTP-based login (6-digit, 10-min expiry). HMAC-SHA256 signed cookies with nonces. bcrypt password hashing (12 rounds). Separate session cookies per role.

API Protection

Redis-backed sliding-window rate limiting with 8 presets. Parameterized SQL queries. Field whitelisting for dynamic updates. RBAC on every endpoint.

Session Timeout

Auto-logout after 5 min inactivity. Warning 1 min before expiry. Device sleep detection (heartbeat gaps > 3s). Tracks mouse, keyboard, touch, scroll events.

Audit Trail

Every admin action logged. Captures actor, action, resource, old/new values, IP address, user agent, timestamp. Immutable — cannot be edited or deleted.

Rate Limiting Presets

PRESET	REQUESTS	WINDOW	USE CASE
`strict`	5	15 min	OTP, sensitive operations
`login`	10	15 min	Login attempts
`checkout`	10	5 min	Order placement
`standard`	60	1 min	Authenticated API calls
`relaxed`	100	1 min	Public endpoints
`search`	30	1 min	Search queries
`upload`	10	1 min	File uploads

Security Headers

```
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000 (production)
Referrer-Policy: strict-origin-when-cross-origin
Content-Security-Policy: [configured]
Permissions-Policy: [configured]
```

---

06 — DATABASE

Design (26 Tables)

```
users ──────────┬──→ user_roles ──→ roles ──→ role_permissions ──→ permissions
                │
                ├──→ user_permissions (direct grants)
                ├──→ user_addresses (multiple, Bangladesh geo)
                ├──→ user_settings
                ├──→ user_risk_flags
                ├──→ quiz_profiles
                ├──→ wishlists
                ├──→ orders ──┬──→ order_items
                │             ├──→ return_requests
                │             └──→ pathao_shipments
                ├──→ reviews ──→ review_helpful_votes
                └──→ notifications

sellers ────────┬──→ products ──┬──→ variants
(via user_id)   │               └──→ product_verifications
                ├──→ product_drafts
                └──→ payouts

warehouses ────→ batches

admin_audit_logs
cod_block_rules ──→ cod_block_logs
otp_codes
newsletters ──→ newsletter_subscribers
notification_templates
rejection_codes
```

Key Design Decisions

- **UUID primary keys** on `users` for security (non-sequential)
- **Separate `sellers` table** from `users` — allows a user to be both buyer and seller
- **`product_drafts` as a staging table** — products only go live after admin approval
- **20 incremental migrations** — safe, repeatable schema evolution
- **Performance indexes** — dedicated migration for query optimization

---

07 — INTEGRATIONS

Ecosystem

Payment Gateways

GATEWAY	USE CASE	INTEGRATION TYPE
bKash	Mobile wallet (largest in BD)	Tokenized API — grant token, create/execute/query payment, refund
Nagad	Mobile wallet (2nd largest)	Merchant DFS API
SSLCommerz	Card payments, net banking	Redirect-based with IPN verification

Logistics & Communication

COURIER	FEATURES
Pathao	OAuth auth, order creation, price calculation, city/zone/area lookup, real-time tracking
Steadfast	Status updates, delivery confirmation

CHANNEL	USE CASE
WhatsApp	Order confirmations, shipping updates, delivery notifications
SMS	OTP delivery, critical alerts

Storage

SERVICE	USE CASE
AWS S3	Seller documents (invoices, COA, brand authorization)
Cloudinary	Product images (optimized delivery, transformations)
Local filesystem	User avatars

08 — TESTING

Strategy

AREA	SUITES	TESTS	COVERAGE TARGET
Quiz / Scoring	4	120+	90% lines, 80% branches
Cart Store	3	80+	80% lines
Wishlist Store	2	40+	80% lines
Server Actions	8	150+	—
API Routes	10	200+	—
Middleware	3	60+	—
Hooks	2	40+	—
Components	5	89+	—
Total	**37**	**779**	**Enforced via Jest config**

Testing Patterns

- **Database mocking** — All DB calls mocked via `jest.mock('../../lib/db')`
- **Zustand persist mocking** — `jest.mock('zustand/middleware')` before store imports
- **Clean state** — `jest.resetAllMocks()` in every `beforeEach`
- **Demo user paths** — Separate test paths for demo users (`userId.startsWith('demo-')`)

09 — LOCALIZATION

Bangladesh-Specific Features

Phone Number Handling

Normalized format: `01XXXXXXXXX` (11 digits). Strips international prefixes (+880, 880, 88). Validates operator codes 013–019. Display: `+880 1712-345678` .

Geographic Data

Full Bangladesh division → district → thana hierarchy. Area-specific delivery zones. District-based COD blocking rules.

Currency

All prices in BDT (Bangladeshi Taka). Minimum payout threshold: 500 BDT. Default seller commission: 15%.

Bilingual UI

English/Bangla internationalization. Language toggle accessible site-wide. Localization files in `src/lib/i18n/` .

10 — OPERATIONS

Operational Features

Warehouse Management

- **FIFO Batch Allocation** — First-in-first-out inventory management
- **Expiry Tracking** — Alerts for products approaching expiry
- **Pick-Pack Workflow** — Order picking board with item-by-item confirmation
- **Stock Intake** — Batch receiving with quantity and expiry logging

Seller Payout System

- Automatic commission calculation (15% default)
- Minimum payout threshold: 500 BDT
- Payment methods: bKash, Nagad, bank transfer
- 7-day settlement window after delivery confirmation

Admin Operational Tools

- Bulk order processing and status updates
- Shipping label generation
- Pathao shipment creation
- Newsletter campaign management
- Notification template system (email/SMS with variable substitution)

11 — PERFORMANCE

& Scalability

CONCERN	SOLUTION
Database connections	Connection pooling (5–10 production, configurable)
API response times	Redis caching layer with configurable TTL
Rate limit accuracy	Redis sorted sets for sliding-window algorithm
Redis unavailability	Automatic fallback to in-memory store
Static assets	Cloudinary CDN for images, Next.js static optimization
Build performance	Turbopack for development, standalone output for deployment
State persistence	Zustand + localStorage (cart/wishlist survive reloads)
Feature rollout	Feature flag system for incremental deployment

12 — LESSONS

Learned

What Worked Well

1. **Verification-first architecture** — Building the product verification pipeline as the core flow (not an afterthought) made it natural and non-disruptive for sellers.
2. **Separate draft and product tables** — Keeping `product_drafts` separate from `products` provided a clean approval workflow without complex status columns.
3. **HMAC-signed sessions over JWT** — Simpler, no token expiry headaches, no refresh token flows, and equally secure for a server-rendered application.
4. **Zustand over Redux** — 90% less boilerplate for 6 stores, with built-in persistence that "just works".
5. **Comprehensive test coverage early** — The 90% coverage threshold on the scoring engine caught 3 edge cases (pregnancy exclusions, tied skin type scores, empty concern lists) before they reached production.

Challenges Overcome

1. **UUID vs VARCHAR foreign keys** — The `users.id` (UUID) and `user_roles.user_id` (VARCHAR) mismatch required `::text` casts. Solved by establishing a consistent casting pattern.
2. **Device sleep detection** — Standard `setTimeout` doesn't fire during device sleep. Solved with a heartbeat interval that detects gaps > 3 seconds.
3. **COD fraud in Bangladesh** — High COD rates (70%+) make fraud economically significant. The multi-signal fraud detection system reduced fraudulent orders during testing.
4. **Seller onboarding completeness** — Early versions created sellers in the `users` table but not the `sellers` table. Fixed by ensuring both records are created atomically.

13 — FUTURE

Roadmap

AI-Powered Ingredient Analysis

Scan product ingredient lists and flag potential irritants for user's skin profile.

Subscription Boxes

Monthly curated skincare boxes based on quiz results.

Mobile App

React Native app leveraging existing API layer.

Dermatologist Consultations

In-app video consultations for premium users.

14 — CONCLUSION

TheSkinProof demonstrates that building trust in e-commerce requires **systemic design**, not surface-level badges. By embedding verification into the product lifecycle, personalizing recommendations through a tested scoring engine, and protecting transactions with multi-layered fraud detection, the platform addresses the fundamental trust deficit in Bangladesh's online skincare market.

The technical foundation — 124 API endpoints, 26 database tables, 779 passing tests, and production-grade security — provides a scalable base for expanding into adjacent categories (cosmetics, wellness, personal care) while maintaining the same verification standard.

TheSkinProof is not just a marketplace.
It is a trust infrastructure for skincare.

TheSkinProof

Next.js 16 · TypeScript 5 · PostgreSQL · Redis · Tailwind CSS v4

779 tests passing · 37 test suites · 124 API endpoints · 26 database tables