## Requirements

The following outlines the core product and technical requirements for Aureve’s skincare e-commerce platform using the MoSCoW method:

### Must Have

* User registration and authentication (email, password, social login optional)
* Product catalog with filters (brand, skin type, concerns, ingredients)
* Product detail page (description, reviews, how-to-use)
* Cart and checkout (local shipping, one payment provider e.g., Stripe)
* Order history and status tracking
* Review and rating system for verified purchases
* Mobile-first responsive web design

### Should Have

* Skin-type quiz for personalized product recommendations
* Wishlist / favorites
* Admin panel for managing products, orders, users
* SEO-optimized product pages
* Basic analytics (product views, purchases, bounce rate)

### Could Have

* Routine builder (e.g., AM/PM suggestions)
* Ingredients knowledge base
* Referral or loyalty program
* Limited-time offers and discount codes

### Won’t Have (in MVP)

* International shipping or multi-currency support
* Third-party seller onboarding (marketplace style)

## Method

### 🏗️ System Architecture

plantuml

CopyEdit

@startuml

package "Client" {

[React Frontend (Next.js)]

}

package "Backend" {

[Node.js API (Express.js)] --> [PostgreSQL Database]

[Node.js API (Express.js)] --> [Cloudinary (Images)]

[Node.js API (Express.js)] --> [Stripe API]

}

[React Frontend (Next.js)] --> [Node.js API (Express.js)]

[Admin Dashboard] --> [Node.js API (Express.js)]

@enduml

### 🔧 Core Components

#### 1. Frontend (React w/ Next.js)

* Homepage with hero, categories, featured products
* Product listing pages with filter sidebar
* Product detail page (images, details, reviews)
* Auth pages (login, signup)
* Cart + checkout flow
* Account section (orders, reviews, profile)

#### 2. Backend (Node.js API)

* Auth module (JWT + bcrypt)
* Product catalog (CRUD for admins)
* Orders & cart logic
* Review system (only for completed orders)
* Stripe integration (checkout, webhooks)
* Skin-type quiz module
* Admin routes with auth guard

#### 3. Database Schema (PostgreSQL)

Key tables:

plaintext

CopyEdit

users(id, name, email, password\_hash, skin\_type, created\_at)

products(id, name, brand, category, price, description, image\_urls[], ingredients[], skin\_types[], created\_at)

orders(id, user\_id, status, total\_price, payment\_id, created\_at)

order\_items(id, order\_id, product\_id, quantity, price)

reviews(id, user\_id, product\_id, rating, comment, created\_at)

cart\_items(id, user\_id, product\_id, quantity)

admin\_users(id, email, password\_hash)

## Implementation

### Phase 1: Environment Setup & Tooling

* Set up monorepo (e.g. using Nx or Turborepo) or separate folders for frontend and backend.
* Configure CI/CD pipeline (GitHub Actions or Vercel for frontend, Render/Heroku for backend).
* Set up database (PostgreSQL) and image storage (Cloudinary or S3).
* Initialize Stripe test environment.

### Phase 2: Core Backend Services (Node.js)

* Auth (register, login, JWT tokens)
* Product API: list, filter, detail (GET only at this point)
* Cart and Checkout logic
* Stripe integration: create checkout session + webhook for payment confirmation
* Orders and order history
* Reviews (only for completed orders)
* Admin APIs: product CRUD, user management

### Phase 3: Frontend (React + Next.js)

* Responsive layout with header, footer, nav
* Homepage, product listings (filters), product detail page
* Authentication flow (login/register)
* Cart and checkout pages (Stripe redirect)
* Account area: orders, reviews
* Skin-type quiz (simple questionnaire mapped to tags)

### Phase 4: Admin Panel (can be same React app or separate)

* Login with admin-only credentials
* Product management UI (upload, edit, delete)
* View orders and users
* Basic metrics dashboard (optional)

### Phase 5: Testing & QA

* Unit tests for critical backend routes
* Integration test for checkout and review flow
* End-to-end test using Cypress or Playwright
* Mobile responsiveness & cross-browser testing

### Phase 6: Deployment & Launch

* Frontend: Deploy to Vercel or Netlify
* Backend: Deploy to Render/Fly.io with database attached
* Domain setup, analytics (e.g. Google Analytics, Hotjar)
* Post-launch monitoring: uptime, errors, logs

## Milestones

| Milestone | Description | Duration |
| --- | --- | --- |
| M1. Setup & Planning | Finalize architecture, tech stack, and development environments | 1 week |
| M2. Backend MVP | User auth, product catalog APIs, cart & orders, Stripe checkout | 2–3 weeks |
| M3. Frontend MVP | Home, product listings, detail page, cart/checkout, auth | 2–3 weeks |
| M4. Admin Panel | Admin login, product management, user & order view | 1–2 weeks |
| M5. Personalization Features | Skin-type quiz, product filtering by concerns/ingredients | 1 week |
| M6. Testing & QA | End-to-end and regression tests, bug fixing | 1 week |
| M7. Deployment & Launch | Staging + production deployment, DNS setup, Stripe live | 1 week |
| Total Time (Est.) | ~8–10 weeks for MVP launch |  |

## Gathering Results

To evaluate whether Aureve meets its goals, we'll track both technical performance and business outcomes.

### 🎯 Product Metrics

* Conversion Rate: % of visitors who complete a purchase
* Average Order Value (AOV): Total revenue / number of orders
* Customer Retention: Repeat purchase rate within 30 and 90 days
* Skin-Type Quiz Engagement: % of visitors using the quiz and converting
* Top-Selling Products: By brand and skin concern

### 🛠️ System Performance

* Page Load Time: Under 2s on mobile (Lighthouse audit)
* Uptime: >99.9% via uptime monitor
* API Response Time: <300ms average
* Checkout Failures: <1% failure rate

### 🔁 Feedback Loops

* Collect user feedback via post-purchase surveys
* Monitor product reviews for insight into quality and satisfaction
* Track drop-off points in the checkout funnel with tools like Hotjar