SPEC-1: Custom Artistic Press-On Nails E-commerce Platform

Table of Contents

1. Background

The press-on nail market is growing rapidly as more consumers seek convenience without sacrificing style or individuality. This project aims to launch a user-centric e-commerce platform focused on selling hand-made, artistic, and reusable press-on nails. Unlike standard online beauty stores, this site will offer customizable designs where customers can either drag and drop components to design their own sets or submit reference images for a completely custom set.

The platform will serve two major user groups: beauty-forward femmes (ages 18–40) and beauty professionals needing unique nails for editorial shoots, performances, or promotions. The experience will evoke the feeling of a digital nail salon, offering aesthetic freedom, size inclusivity, and community-driven exclusivity. To foster brand loyalty and recurring revenue, a membership system with tiered discounts and exclusive designs will be integrated.

The site will initially target North America with international expansion planned. Its long-term vision includes being a digitally immersive beauty club that redefines personalization in beauty accessories.

2. Requirements

The platform will prioritize user experience, customizability, and recurring revenue through subscriptions. Requirements are categorized using MoSCoW prioritization:

2.1. Must Have

- User registration and authentication system
- Product catalog with high-resolution images, variants (length, shape, design)
- Purchase flow with shopping cart and secure payment gateway
- Drag-and-drop custom nail design interface
- Custom design submission form with image/file upload
- Tiered membership subscription system with recurring billing
- User dashboard for order tracking, subscription management, and custom design history
- Mobile responsive UI optimized for beauty/fashion standards
- Basic size kit purchase option for trial and sizing
- Admin dashboard for product uploads, order management, and custom design intake

2.2. Should Have

- Inclusive sizing system with selectable nail sizes for each finger
- Product tagging by color, style, shape, and collection
- Email notifications for order status and membership updates
- Integration with social media for user-generated content and promotion

- Membership-exclusive product visibility and pricing
- Discount logic engine tied to membership tiers

2.3. Could Have

- AI-assisted style suggestions based on user preferences or past purchases
- AR/3D preview of nail sets on virtual hand
- Community features (e.g. wishlists, user reviews, shareable designs)
- Affiliate program for beauty professionals or influencers

2.4. Won’t Have (Initial MVP)

- Native mobile app (web-optimized site only)
- Full internationalization/multilingual support (initially North America only)

3. Method

3.1. Overview

The system follows a modern web architecture using a JAMstack approach: - **Frontend**: Next.js + TailwindCSS for responsive UI and drag-and-drop designer - **Backend**: Node.js + Express or tRPC (API layer) - **Database**: PostgreSQL with Prisma ORM - **Auth**: Clerk or Auth0 for user management - **Payments**: Stripe for one-time and subscription payments - **Hosting**: Vercel (frontend) and Railway/Supabase (backend and DB)

3.2. Component Diagram

```
@startuml
package "Frontend" {
  [Landing Page]
  [Shop Catalog]
  [Drag & Drop Designer]
  [User Dashboard]
}

package "Backend API" {
  [Auth Service]
  [Order Service]
  [Design Service]
  [Membership Service]
  [Admin Panel]
}

package "Infrastructure" {
  [PostgreSQL]
  [Stripe]
  [Image CDN]
}

[Frontend] --> [Backend API]
[Backend API] --> [PostgreSQL]
[Order Service] --> [Stripe]
[Drag & Drop Designer] --> [Design Service]
[Admin Panel] --> [PostgreSQL]
@enduml
```

3.3. Database Schema (Simplified)

- `users`: `id`, `email`, `password_hash`, `subscription_tier`, `created_at`
- `products`: `id`, `name`, `price`, `type`, `design_image_url`, `tags[]`
- `nail_designs`: `id`, `name`, `design_type` (e.g., "cat-eye", "french"), `image_url`, `creator_id`
- `custom_sets`: `id`, `user_id`, `name`, `nail_layout` (JSON of 10 nails), `created_at`
- `orders`: `id`, `user_id`, `product_ids[]`, `status`, `total_price`, `created_at`
- `subscriptions`: `id`, `user_id`, `tier`, `start_date`, `end_date`, `stripe_customer_id`

3.4. Drag-and-Drop Design Logic

The nail designer will be implemented as a grid of 10 nail placeholders (2 hands). Each nail has the following configurable slots: - **Shape**: square, almond, stiletto, etc. - **Length**: short, medium, long, XXXL - **Design**: applied from preset assets in `nail_designs`

The state is stored as a client-side object:

```
{
  "thumb": { "shape": "almond", "length": "medium", "design_id": "abc123" },
  ...
}
```

User can save this set as a `custom_set`, which is linked to their account and optionally added to cart for purchase.

3.5. Membership Tiers

Stripe will manage recurring subscriptions. Tiers will be: - **Basic** (Free): Access to size kit and public products only - **Club** ($10/mo): Access to exclusive designs and 10% discount - **Elite** ($25/mo): All Club benefits + 25% discount + priority for custom requests

Business logic will gate product visibility and discounts based on tier.

4. Implementation

The implementation will be staged to support rapid MVP delivery using low-cost, developer-friendly platforms that allow seamless migration later.

4.1. 1. Project Setup

- Initialize GitHub monorepo: `/frontend` (Next.js), `/backend` (Express API)
- Setup hosting accounts:
- Vercel for frontend
- Railway for backend
- Supabase for PostgreSQL and file storage
- Configure CI/CD:
- GitHub → Vercel (auto deploy frontend)
- GitHub → Railway (auto deploy backend)

4.2. 2. Database Setup (Supabase)

Use Supabase UI or SQL to set up the following tables: - `users`, `products`, `nail_designs`, `custom_sets`, `orders`, `subscriptions` - Create indexes on `user_id`, `created_at` fields - Use Prisma or Supabase client for integration

4.3. 3. Frontend Development (Next.js)

- Build public pages: Home, Shop, About, Trial Kit
- Build user-only pages: Designer, Dashboard, My Orders, Subscription
- Implement drag-and-drop builder using `react-dnd` or `@dnd-kit/core`
- 10 static nail slots → update state on drop
- Save result as JSON and send to backend for persistence

4.4. 4. Backend Development (Railway + Express)

- REST or tRPC API:
- `/auth`, `/products`, `/custom-sets`, `/orders`, `/subscriptions`
- Implement Stripe integration for:
- Checkout session (one-time and recurring)
- Webhooks to sync subscription tiers
- Image uploads → Supabase Storage
- Limit file size, validate types

4.5. 5. Authentication

- Use Clerk.dev or Supabase Auth
- Social logins + email/password
- Protect user-specific routes and custom content

4.6. 6. Admin Portal (basic MVP)

- Add `/admin` route restricted by user role
- Upload new products or designs
- Review custom submissions

4.7. 7. Membership Gating

- Gate premium designs and discounts by checking `subscription_tier` from API
- Implement discount logic on frontend cart and backend total calculation

5. Milestones

5.1. Milestone 1: Foundation Setup (Week 1–2)

- Set up GitHub repo and monorepo structure (`/frontend`, `/backend`)
- Configure Vercel (frontend), Railway (backend), Supabase (DB + Storage)
- Initialize Prisma schema and DB tables on Supabase
- Implement static pages: Home, About, Trial Kit info
- Deploy working landing site

5.2. Milestone 2: Core E-Commerce (Week 3–4)

- Build product catalog and product detail pages
- Implement shopping cart and Stripe checkout for guest purchases
- Develop user authentication (Clerk or Supabase)
- Setup admin UI for adding products

5.3. Milestone 3: User Dashboard + Membership (Week 5–6)

- Develop user dashboard (order history, profile, membership status)
- Integrate Stripe recurring subscriptions

- Implement membership-based pricing logic
- Email notifications for orders and subscriptions

5.4. Milestone 4: Drag-and-Drop Designer MVP (Week 7–8)

- Implement drag-and-drop UI with 10-slot nail canvas
- Load and apply preset designs from database
- Allow saving and submitting custom sets
- Backend endpoint to store custom design JSON and preview image (optional)

5.5. Milestone 5: Final QA + Soft Launch (Week 9)

- Cross-device QA
- SEO and performance tuning
- Soft launch to limited audience with feedback loop

5.6. Milestone 6: Public Launch + Marketing (Week 10)

- Open to general public
- Launch newsletter + social campaigns
- Begin collecting analytics for product-market fit

6. Gathering Results

Post-launch, the platform’s performance and alignment with user needs will be evaluated through both quantitative metrics and qualitative feedback.

6.1. Success Metrics

- **Conversion Rate**: % of visitors who become customers or members
- **Custom Set Creation Rate**: # of users who save or submit a nail design
- **Monthly Recurring Revenue (MRR)**: Revenue from subscription tiers
- **Repeat Purchase Rate**: % of users making 2+ orders within 30 days
- **Average Order Value (AOV)**: Track spending trends by tier
- **Drop-off Points**: Use Hotjar or PostHog to identify where users abandon flows

6.2. Feedback Collection

- **Post-purchase surveys** (automated emails)

- **Custom design feedback forms** embedded in dashboard
- **Social media tracking** (hashtags, reposts, DMs)
- **Manual outreach** to first 50–100 customers for deeper insights

6.3. Technical Evaluation

- **Page load times** (Lighthouse or Vercel Analytics)
- **API response latency** and error rates
- **Storage and bandwidth usage** (Supabase dashboards)
- **Payment success/failure rates** (Stripe Dashboard)

Based on this data, the next development cycle will prioritize expanding community features, refining design tools, and optimizing fulfillment workflows.