Shopify Integration Specification

Technical Documentation for Backend Integration

Generated: 2/5/2026

1. Project Overview

This document provides the technical specifications for building a Shopify app that connects to our existing wholesale fashion product management system. The goal is to enable direct product export from our iOS app to Shopify stores, creating products as drafts for merchant review.

2. System Architecture

```
% % % % % % % % % % % % % % % % % % %        % % % % % % % % % % % % % % % % % % %        % %
%    iOS App      %  % % % % % % %º %   Shared Backend %  % % % % % % %º %    Shopify   %
%   (Expo/React)   %         %    (Express.js)  %          %    Store     %
% % % % % % % % % % % % % % % % % % %        % % % % % % % % % % % % % % % % % % %        % %
                              %²
                              %
                     % % % % % % % % %4% % % % % % % %
                     %   Shopify App  %
                     %   (Web Admin)  %
                     % % % % % % % % % % % % % % % % %
```

3. Current Backend Details

Technology Stack:

- Server: Express.js (Node.js/TypeScript)
- Database: PostgreSQL (Drizzle ORM)
- Port: 5000
- AI Integration: OpenAI API for label scanning
- CORS: Configured for Replit domains and localhost

Current API Endpoints:

POST /api/scan-label

Scans product label images using AI vision

4. Product Data Schema

Products scanned from labels have the following structure:

```
{
  "styleName": "Product name or style name",
  "styleNumber": "Style number, SKU, or product code",
  "wholesalePrice": 29.99,
  "retailPrice": 59.99,
  "colors": ["Black", "White", "Light Blue"],
  "sizes": ["XS", "S", "M", "L", "XL"],
  "category": "Tops | Bottoms | Dresses | Outerwear | Accessories",
  "brandName": "Vendor/Brand name",
  "season": "Fall 2026, Resort 2027, etc.",
  "notes": "Fabric content, care instructions, etc."
}
```

5. Required Shopify App Features

5.1 Shopify OAuth Integration

- Implement Shopify OAuth 2.0 flow
- Store access tokens securely in database
- Handle token refresh automatically
- Link Shopify store to user account

5.2 Required API Scopes

- write_products - Create and update products
- read_products - Read existing products (duplicate detection)

5.3 New Database Tables Needed

```
shopify_stores
% % %  id: UUID (primary key)
% % %  user_id: UUID (foreign key !' users)
% % %  shop_domain: VARCHAR (e.g., "store-name.myshopify.com")
% % %  access_token: TEXT (encrypted)
% % %  scope: TEXT
% % %  created_at: TIMESTAMP
% % %  updated_at: TIMESTAMP

products
% % %  id: UUID (primary key)
% % %  user_id: UUID (foreign key !' users)
% % %  style_name: TEXT
% % %  style_number: VARCHAR
% % %  wholesale_price: DECIMAL
% % %  retail_price: DECIMAL
% % %  colors: JSONB
% % %  sizes: JSONB
% % %  category: VARCHAR
% % %  brand_name: TEXT
% % %  season: VARCHAR
% % %  notes: TEXT
% % %  image_url: TEXT
% % %  shopify_product_id: VARCHAR (nullable)
% % %  synced_at: TIMESTAMP (nullable)
% % %  created_at: TIMESTAMP
% % %  updated_at: TIMESTAMP
```

6. New API Endpoints Required

6.1 Shopify Connection

GET /api/shopify/auth
Initiates OAuth flow, redirects to Shopify

GET /api/shopify/callback
Handles OAuth callback, stores access token

GET /api/shopify/status
Returns connection status for current user

DELETE /api/shopify/disconnect
Removes Shopify store connection

6.2 Product Management

POST /api/products
  Save scanned product to database

GET /api/products
  List all products for current user

POST /api/products/export-to-shopify
  Push selected products to Shopify as drafts

7. Shopify Product Creation

When exporting to Shopify, create products with:

```
{
  "product": {
    "title": "{styleName}",
    "body_html": "{notes}",
    "vendor": "{brandName}",
    "product_type": "{category}",
    "tags": ["{season}"],
    "status": "draft",
    "variants": [
      {
        "option1": "{color}",
        "option2": "{size}",
        "price": "{retailPrice}",
        "sku": "{styleNumber}-{color}-{size}",
        "inventory_management": "shopify",
        "inventory_quantity": 0
      }
    ],
    "options": [
      { "name": "Color", "values": ["{colors}"] },
      { "name": "Size", "values": ["{sizes}"] }
    ]
  }
}
```

Note: Products are created with status "draft" so merchants can review and finalize before publishing.

8. Authentication Flow

1. User logs into iOS app with their account
2. User taps "Connect Shopify Store" in settings
3. App opens Shopify OAuth URL in browser
4. User authorizes the app on Shopify
5. Shopify redirects back with authorization code
6. Backend exchanges code for access token
7. Access token stored linked to user account
8. User can now export products to their Shopify store

9. Environment Variables Needed

```
SHOPIFY_API_KEY=your_shopify_api_key
SHOPIFY_API_SECRET=your_shopify_api_secret
SHOPIFY_SCOPES=write_products,read_products
SHOPIFY_REDIRECT_URI=https://your-domain.com/api/shopify/callback
```

10. Recommended Tech Stack for Shopify App

- Remix or Next.js (Shopify recommends Remix)

- Shopify App Bridge for embedded experience
- @shopify/shopify-api for API interactions
- Prisma for database ORM
- Polaris design system for UI

11. Resources

- Shopify App Development: https://shopify.dev/docs/apps
- Product API: https://shopify.dev/docs/api/admin-rest/current/resources/product
- OAuth: https://shopify.dev/docs/apps/auth/oauth
- Shopify CLI: https://shopify.dev/docs/apps/tools/cli

--- End of Specification ---