3D Outfit Builder - Application Architecture

> NOTE: This repository's documentation has been consolidated into `README.md`. Please consult the main `README.md` for the canonical, up-to-date guide. The original content follows below for reference.

🏗️ Overall Architecture

Tech Stack

- **Frontend**: Next.js 14 (App Router)
- **Styling**: Tailwind CSS + shadcn/ui
- **Database**: Supabase (PostgreSQL)
- **Authentication**: Supabase Auth
- **State Management**: React Context + Local Storage
- **Animations**: Framer Motion + GSAP
- **3D Visualization**: CSS-based 3D transforms

🗂️ Project Structure

Main Directory Layout:

- **app/** — Next.js App Router pages
    - **(auth)/** — Auth route group
        - login/
        - register/
    - **(dashboard)/** — Protected dashboard routes
        - dashboard/
        - profile/
        - settings/
    - **(shop)/** — Shopping experience
        - gallery/
        - customize/
        - cart/
        - checkout/
    - **(studio)/** — Creative tools
        - 3d-playground/
        - outfit-picker/
    - **api/** — API routes
    - globals.css
    - layout.tsx
    - page.tsx — Landing page
- **components/** — Reusable components
    - ui/ — shadcn/ui components

  - auth/ — Authentication components
  - dashboard/ — Dashboard-specific components
  - gallery/ — Gallery components
  - profile/ — Profile components
  - cart/ — Shopping cart components
  - 3d/ — 3D visualization components
  - layout/ — Layout components (navbar, footer)

- **contexts/** — React contexts
- **hooks/** — Custom hooks
- **lib/** — Utility libraries
- **types/** — TypeScript type definitions
- **scripts/** — Database scripts

🗄 Database Schema

Core Tables

1. User Management

```sql -- Profiles (extends Supabase auth.users) profiles ( id UUID PRIMARY KEY REFERENCES auth.users(id), username TEXT UNIQUE, full_name TEXT, avatar_url TEXT, bio TEXT, created_at TIMESTAMP, updated_at TIMESTAMP )

-- User preferences user_preferences ( id UUID PRIMARY KEY, user_id UUID REFERENCES profiles(id), theme TEXT DEFAULT 'dark', notifications JSONB, privacy_settings JSONB ) ```

#### 2. Product Catalog

```sql -- Categories categories ( id UUID PRIMARY KEY, name TEXT NOT NULL, slug TEXT UNIQUE, description TEXT, image_url TEXT, parent_id UUID REFERENCES categories(id), sort_order INTEGER )

-- Products products ( id UUID PRIMARY KEY, name TEXT NOT NULL, description TEXT, price DECIMAL(10,2), category_id UUID REFERENCES categories(id), brand TEXT, colors JSONB, sizes JSONB, tags TEXT[], model_url TEXT, images JSONB, is_active BOOLEAN DEFAULT true, created_at TIMESTAMP )

-- Product variants (colors, sizes) product_variants ( id UUID PRIMARY KEY, product_id UUID REFERENCES products(id), sku TEXT UNIQUE, color TEXT, size TEXT, price DECIMAL(10,2), stock_quantity INTEGER, model_url TEXT ) ```

#### 3. 3D Avatar System

```sql -- User avatars avatars ( id UUID PRIMARY KEY, user_id UUID REFERENCES profiles(id), name TEXT NOT NULL, avatar_data JSONB, -- 3D model configuration measurements JSONB, is_default BOOLEAN DEFAULT false, created_at TIMESTAMP )
```

-- Avatar measurements avatar_measurements (id UUID PRIMARY KEY, avatar_id UUID REFERENCES avatars(id), measurement_type TEXT, -- height, chest, waist, etc. value DECIMAL(5,2), unit TEXT DEFAULT 'cm') ```

4. Outfit System

```sql -- Saved outfits saved_outfits ( id UUID PRIMARY KEY, user_id UUID REFERENCES profiles(id), avatar_id UUID REFERENCES avatars(id), name TEXT NOT NULL, description TEXT, is_public BOOLEAN DEFAULT false, tags TEXT[], thumbnail_url TEXT, created_at TIMESTAMP, updated_at TIMESTAMP )

-- Outfit items (products in an outfit) outfit_items ( id UUID PRIMARY KEY, outfit_id UUID REFERENCES saved_outfits(id), product_id UUID REFERENCES products(id), variant_id UUID REFERENCES product_variants(id), position_data JSONB, -- 3D positioning customization_data JSONB -- colors, adjustments ) ```

**5. Shopping System**

```sql -- Shopping cart (persistent) cart_items ( id UUID PRIMARY KEY, user_id UUID REFERENCES profiles(id), product_id UUID REFERENCES products(id), variant_id UUID REFERENCES product_variants(id), quantity INTEGER DEFAULT 1, added_at TIMESTAMP )

-- Orders orders ( id UUID PRIMARY KEY, user_id UUID REFERENCES profiles(id), status TEXT DEFAULT 'pending', total_amount DECIMAL(10,2), shipping_address JSONB, billing_address JSONB, created_at TIMESTAMP )

-- Order items order_items ( id UUID PRIMARY KEY, order_id UUID REFERENCES orders(id), product_id UUID REFERENCES products(id), variant_id UUID REFERENCES product_variants(id), quantity INTEGER, unit_price DECIMAL(10,2), total_price DECIMAL(10,2) ) ```

**6. Social Features**

```sql -- Outfit likes outfit_likes ( id UUID PRIMARY KEY, user_id UUID REFERENCES profiles(id), outfit_id UUID REFERENCES saved_outfits(id), created_at TIMESTAMP, UNIQUE(user_id, outfit_id) )

-- Comments outfit_comments ( id UUID PRIMARY KEY, user_id UUID REFERENCES profiles(id), outfit_id UUID REFERENCES saved_outfits(id), content TEXT NOT NULL, created_at TIMESTAMP )

-- User follows user_follows ( id UUID PRIMARY KEY, follower_id UUID REFERENCES profiles(id), following_id UUID REFERENCES profiles(id), created_at TIMESTAMP, UNIQUE(follower_id, following_id) ) ```

## 🔗 Page Flow & Navigation

### 1. Authentication Flow

**User Authentication Journey:**

1. **Landing Page** (`/`)
2. **Register** (`/auth/register`)
    - Email Confirmation
3. **Login** (`/auth/login`)
```

4. **Dashboard** (`/dashboard`)

---

2. Main User Journey

Dashboard (`/dashboard`) — Main hub for users

Profile Management:

- Profile (`/profile`)
- Settings (`/profile/settings`)
- Saved Outfits (`/profile/outfits`)

Shopping Experience:

- Gallery (`/gallery`) → Product Details
- Customize (`/customize`) → 3D Builder
- Cart (`/cart`) → Checkout (`/checkout`)
- Order History (`/orders`)

Creative Tools:

- 3D Playground (`/3d-playground`)
- Outfit Picker (`/outfit-picker`)

---

3. Navigation Hierarchy

Primary Navigation:

- Home (`/`)
- Gallery (`/gallery`)
- Customize (`/customize`)
- 3D Studio (`/3d-playground`)
- **Profile Menu:**
  - Dashboard (`/dashboard`)
  - Profile (`/profile`)
  - Settings (`/profile/settings`)
  - Cart (`/cart`)
  - Sign Out

Secondary Navigation:

- Gallery Filters & Search
- Product Categories
- Outfit Collections
- User-Generated Content

🎯 Component Architecture

1. Layout Components

Global Layout (`app/layout.tsx`):

- Navbar (`components/layout/navbar.tsx`)
- Main Content
- Footer (`components/layout/footer.tsx`)

Page-specific Layouts:

- AuthLayout (for login/register)
- DashboardLayout (for protected pages)
- ShopLayout (for shopping pages)

---

2. Feature Components

Authentication (`components/auth/`):

- LoginForm
- RegisterForm
- ProtectedRoute
- AuthProvider

3D Visualization (`components/3d/`):

- AvatarModel
- ProductModelViewer
- OutfitCombinationView
- ThreeScene

Shopping (`components/shop/`):

- ProductCard
- ProductGrid
- CartItem
- CheckoutForm
- OrderSummary

🔄 State Management Strategy

1. Global State (React Context)

AuthContext — Authentication State:

- Properties: `user`, `session`, `isAuthenticated`
- Methods: `signIn`, `signUp`, `signOut`, `updateProfile`

CartContext — Shopping Cart State:

- Properties: `items`, `totalItems`, `totalPrice`
- Methods: `addItem`, `removeItem`, `updateQuantity`, `clearCart`

BuilderContext — 3D Builder State:

- Properties: `selectedAvatar`, `selectedItems`, `currentOutfit`
- Methods: `updateAvatar`, `addItem`, `removeItem`, `saveOutfit`

---

2. Local State (Component Level)

- Form states
- UI states (modals, dropdowns)
- Temporary selections
- Animation states

---

3. Server State (Supabase)

- User profiles
- Product catalog
- Saved outfits
- Order history

🛡 Security & Permissions

Row Level Security (RLS) Policies

Security Rules:

- **Profiles:** Users can only see/edit their own profile
- **Outfits:** Users can see public outfits + their own private ones
- **Cart:** Users can only access their own cart
- **Orders:** Users can only see their own orders

API Route Protection

Middleware Protected Routes:

- `/dashboard/:path*`
- `/profile/:path*`
- `/api/protected/:path*`

📱 Responsive Design Strategy

Breakpoints

Breakpoint	Width	Device Type
sm	640px	Mobile landscape
md	768px	Tablet

Breakpoint	Width	Device Type
lg	1024px	Desktop
xl	1280px	Large desktop
2xl	1536px	Extra large

Mobile-First Components

- Collapsible navigation
- Touch-friendly 3D controls
- Optimized image loading
- Gesture support

🚀 Performance Optimization

1. Code Splitting

Route-based splitting:

- Automatic with Next.js App Router
- Each route loads only necessary code

Component-based splitting:

- Use dynamic imports for heavy components
- Example: `const HeavyComponent = lazy(() => import('./HeavyComponent'))`

2. Image Optimization

Strategies implemented:

- Next.js Image component for automatic optimization
- Supabase Storage for user-uploaded content
- CDN delivery for product images
- Lazy loading and responsive image sizes

3. Database Optimization

Query optimization:

- Indexes on frequently queried columns (user_id, product_id, created_at)
- Pagination for large datasets
- Caching strategies for static content
- Connection pooling for high traffic

🔧 Development Workflow

1. Environment Setup

Development server:

- Command: `npm run dev`
- Runs Next.js on http://localhost:3000

Database migrations:

- Command: `npm run db:migrate`
- Applies schema changes to Supabase

Type generation:

- Command: `npm run types:generate`
- Generates TypeScript types from database schema

2. Testing Strategy

Test coverage areas:

- Unit tests for utilities (utils, customization logic)
- Integration tests for API routes (auth, cart, products)
- E2E tests for critical user flows (signup, customization, checkout)

3. Deployment

Commands:

- Vercel deployment: `vercel --prod`
- Database migrations on production: `npm run db:migrate:prod`

Deployment checklist:

- ✓ Environment variables configured in Vercel
- ✓ Supabase project linked
- ✓ Database migrations applied
- ✓ Build completed successfully

End of Architecture Documentation