Product Configuration System Guide

Overview

This guide explains the complete product configuration system for the Promo Gifts enhanced designer, including color variations, multiple view support, and print area management.

Table of Contents

System Architecture

The product configuration system consists of three main layers:

1. Database Layer (Supabase)

- **product_templates**: Base product information
- **product_template_variants**: Color and view specific templates
- **print_areas**: Designable areas on products
- **product-templates** (Storage): Template images

2. Service Layer (`supabaseService.js`)

- Product template CRUD operations
- Variant management (color + view combinations)
- Print area management
- Template image uploads
- Configuration persistence

3. UI Layer

- **EnhancedDesigner.jsx**: Main design interface with color/view selection
- **PrintAreaAdmin.jsx**: Admin panel for configuring products
- **PrintAreaSelector.jsx**: UI component for selecting print areas

---

Database Schema

Tables

1. product_templates

Stores base product information.

```sql
CREATE TABLE product_templates (
  id UUID PRIMARY KEY,
  product_key VARCHAR(100) UNIQUE NOT NULL,
  name VARCHAR(255) NOT NULL,
  template_url TEXT,  -- Default template (fallback)
  colors TEXT[],      -- Available colors for this product
  base_price DECIMAL(10,2),
  default_view VARCHAR(50) DEFAULT 'front',
  available_views TEXT[] DEFAULT ARRAY['front'],
  created_at TIMESTAMP,
  updated_at TIMESTAMP,
  created_by UUID REFERENCES auth.users(id)
);
```

Example Data:

```json
{
  "id": "uuid-1",
  "product_key": "tote-bag-5oz",
  "name": "5oz Cotton Tote Bag",
  "template_url": "/templates/bag/default.png",
  "colors": ["#000000", "#FFFFFF", "#FF0000"],
  "base_price": 18.99,
  "default_view": "front",
  "available_views": ["front", "back"]
}
```

2. product_template_variants

Stores color and view specific template images.

```sql
CREATE TABLE product_template_variants (
  id UUID PRIMARY KEY,
  product_template_id UUID REFERENCES product_templates(id) ON DELETE CASCADE,
  color_name VARCHAR(100) NOT NULL,
  color_code VARCHAR(20) NOT NULL,  -- e.g., "#000000"
  view_name VARCHAR(50) NOT NULL,    -- e.g., "front", "back"
  template_url TEXT NOT NULL,
  created_at TIMESTAMP,
  updated_at TIMESTAMP,
  UNIQUE(product_template_id, color_code, view_name)
);
```

Example Data:

```
{
  "id": "uuid-2",
  "product_template_id": "uuid-1",
  "color_name": "Black",
  "color_code": "#000000",
  "view_name": "front",
  "template_url": "https://media.istockphoto.com/id/1197598953/photo/eco-friendly-
black-colour-fashion-canvas-tote-bag-isolated-on-white-background.jpg?
s=612x612&w=0&k=20&c=3HMfU62MrBvLmE0u5DUZ3dt7rbSTO4a99yBE0kOUoLI="
}
```

3. print_areas

Stores printable/designable areas on products.

```
CREATE TABLE print_areas (
  id UUID PRIMARY KEY,
  product_template_id UUID REFERENCES product_templates(id) ON DELETE CASCADE,
  variant_id UUID REFERENCES product_template_variants(id) ON DELETE CASCADE,
  area_key VARCHAR(100) NOT NULL,
  name VARCHAR(255) NOT NULL,
  x INTEGER NOT NULL,
  y INTEGER NOT NULL,
  width INTEGER NOT NULL,
  height INTEGER NOT NULL,
  max_width INTEGER,
  max_height INTEGER,
  shape VARCHAR(50) DEFAULT 'rectangle',  -- 'rectangle', 'circle', 'ellipse'
  created_at TIMESTAMP,
  updated_at TIMESTAMP
);
```

Example Data:

```
{
  "id": "uuid-3",
  "variant_id": "uuid-2",
  "area_key": "front_center",
  "name": "Front Center",
  "x": 200,
  "y": 200,
  "width": 400,
  "height": 400,
  "max_width": 400,
  "max_height": 400,
  "shape": "rectangle"
}
```

Relationships

```
product_templates (1) -----> (many) product_template_variants
                                              |
                                              |
                                              v
                                        (many) print_areas
```

Data Flow

1. Loading a Product for Design

```
User selects product → EnhancedDesigner
  ↓
Load product template from database (loadProductConfiguration)
  ↓
User selects color → Load variant for (product, color, front)
  ↓
Load print areas for this variant
  ↓
Display template image + print area overlays
  ↓
User selects different view (back) → Load variant for (product, color, back)
  ↓
Display back template + back print areas
```

Code Example:

```
// Load configuration for Black Tote Bag, Front view
const config = await loadProductConfiguration(
  'tote-bag-5oz',
  '#000000',  // Black
  'front'
);

// config will contain:
{
  name: "5oz Cotton Tote Bag",
  template: "https://vivipins.com/backend/uploads/template_b/photos/18/black-tote-
bag-1714293150.jpg",
  printAreas: {
    front_center: { x: 200, y: 200, width: 400, height: 400, ... }
  },
  currentColor: "#000000",
  currentView: "front",
  variantId: "uuid-2"
}
```

2. Saving Product Configuration

```
Admin opens Print Area Admin
  ↓
Select product + color + view
  ↓
Upload template image (if new)
  ↓
Configure print areas (drag, resize, add, delete)
  ↓
Click Save
  ↓
saveVariantConfiguration(productKey, colorCode, viewName, config)
  ↓
1. Upsert product_template_variants
2. Batch update print_areas
3. Upload template to Supabase Storage
```

Code Example:

```
// Save configuration for Red Tote Bag, Back view
await saveVariantConfiguration(
  'tote-bag-5oz',
  '#FF0000',  // Red
  'back',
  {
    colorName: 'Red',
    templateUrl: 'https://upload.wikimedia.org/wikipedia/commons/b/b9/TeeshirtCo-
pyleft_cadre.jpg',
    printAreas: {
      back_top: { x: 100, y: 50, width: 300, height: 200, ... },
      back_bottom: { x: 100, y: 300, width: 300, height: 200, ... }
    }
  }
);
```

Adding New Products

Step 1: Create Base Product Entry

You can add products in two ways:

Method A: Via Code (enhancedProductCatalog.json)

```
{
  "categories": [
    {
      "name": "Bags",
      "products": [
        {
          "key": "canvas-bag-10oz",
          "name": "10oz Canvas Bag",
          "template": "/templates/bag/canvas-10oz.png",
          "colors": ["#FFFFFF", "#000000", "#001f3f"],
          "basePrice": 24.99,
          "printAreas": {
            "front": { "name": "Front", "x": 200, "y": 200, "width": 400, "height": 40
0 }
          }
        }
      ]
    }
  ]
}
```

Method B: Via Database (SQL)

```sql
INSERT INTO product_templates (
  product_key, name, colors, base_price, default_view, available_views
) VALUES (
  'canvas-bag-10oz',
  '10oz Canvas Bag',
  ARRAY['#FFFFFF', '#000000', '#001f3f'],
  24.99,
  'front',
  ARRAY['front', 'back']
);
```

Step 2: Configure Color Variants

For each color + view combination, you need to:

1. **Prepare Template Images**
   - Front view: `canvas-bag-10oz-white-front.png`
   - Back view: `canvas-bag-10oz-white-back.png`
   - Repeat for each color

2. **Upload via Print Area Admin**
   - Open Enhanced Designer → Sign In (admin account)
   - Click “Print Area Admin” button (gear icon)
   - Select product: “10oz Canvas Bag”
   - Click “Template” button to upload image
   - Configure print areas
   - Save

3. **Or via API**

```javascript
// Create variant for White front view
await saveVariantConfiguration(
  'canvas-bag-10oz',
  '#FFFFFF',
  'front',
  {
    colorName: 'White',
    templateUrl: '/templates/bag/canvas-bag-10oz-white-front.png',
    printAreas: {
      front_center: {
        name: 'Front Center',
        x: 200,
        y: 200,
        width: 400,
        height: 400,
        maxWidth: 400,
        maxHeight: 400,
        shape: 'rectangle'
      }
    }
  }
);
```

Step 3: Verify Configuration

1. Open Enhanced Designer
2. Select your new product
3. Test color switching
4. Test view switching
5. Verify print areas appear correctly

Managing Color Variations

Understanding Color Variations

Each product can have multiple colors, and each color can have different template images. For example:

- **Black Tote Bag** (front) → black-front.png
- **Black Tote Bag** (back) → black-back.png
- **Red Tote Bag** (front) → red-front.png
- **Red Tote Bag** (back) → red-back.png

Adding a New Color to Existing Product

Via Print Area Admin:
1. Open Print Area Admin
2. Select the product
3. Click "Manage" button
4. Load existing configuration
5. Upload new template for new color
6. Configure print areas
7. Save

Via API:

```
// Add Navy Blue variant
await saveVariantConfiguration(
  'tote-bag-5oz',
  '#001f3f',  // Navy Blue
  'front',
  {
    colorName: 'Navy Blue',
    templateUrl: '/templates/bag/tote-bag-5oz-navy-front.png',
    printAreas: {
      // Same as other colors or different if needed
      front_center: { x: 200, y: 200, width: 400, height: 400, ... }
    }
  }
);
```

Updating Color-Specific Templates

```
// Update template URL for existing variant
await updateProductVariant(variantId, {
  templateUrl: 'https://i.ytimg.com/vi/w6jRj6-AQKo/maxresdefault.jpg'
});
```

Best Practices

1. **Use consistent naming**: `{product}-{color}-{view}.png`
2. **Same print areas**: Keep print area positions consistent across colors
3. **High resolution**: Use high-quality template images (at least 1200x1200px)
4. **Transparent backgrounds**: If applicable, use PNG with transparency
5. **Test thoroughly**: Verify each color+view combination works

---

Managing Multiple Views

Understanding Views

Views represent different angles or sides of a product:
- **front**: Front view (default)
- **back**: Back view
- **left**: Left side
- **right**: Right side
- **top**: Top view
- **bottom**: Bottom view
- **inside**: Interior view
- **custom**: Custom view name

Adding Multiple Views

Step 1: Update Product Template

```
UPDATE product_templates
SET available_views = ARRAY['front', 'back', 'left', 'right']
WHERE product_key = 'tote-bag-5oz';
```

Step 2: Create Variants for Each View

```
// For each color + view combination
const colors = ['#000000', '#FFFFFF', '#FF0000'];
const views = ['front', 'back', 'left', 'right'];

for (const color of colors) {
  for (const view of views) {
    await saveVariantConfiguration(
      'tote-bag-5oz',
      color,
      view,
      {
        colorName: getColorName(color),
        templateUrl: `/templates/bag/tote-bag-5oz-${getColorName(color)}-${view}.png`,
        printAreas: getPrintAreasForView(view)
      }
    );
  }
}
```

Step 3: View-Specific Print Areas

Different views can have different print areas:

```
function getPrintAreasForView(view) {
  switch(view) {
    case 'front':
      return {
        front_center: { x: 200, y: 200, width: 400, height: 400 },
        front_top_logo: { x: 350, y: 100, width: 100, height: 100 }
      };
    case 'back':
      return {
        back_full: { x: 150, y: 150, width: 500, height: 500 }
      };
    case 'left':
      return {
        left_side: { x: 250, y: 200, width: 300, height: 400 }
      };
    case 'right':
      return {
        right_side: { x: 250, y: 200, width: 300, height: 400 }
      };
    default:
      return {};
  }
}
```

UI Integration

The Enhanced Designer automatically shows view tabs when multiple views are available:

```
// In EnhancedDesigner.jsx
{availableViews.length > 1 && (
  <div className="view-tabs">
    {availableViews.map(view => (
      <button
        key={view}
        onClick={() => setSelectedView(view)}
        className={selectedView === view ? 'active' : ''}
      >
        {view.charAt(0).toUpperCase() + view.slice(1)}
      </button>
    ))}
  </div>
)}
```

Print Area Configuration

Print Area Properties

```
{
  name: "Front Center",        // Display name
  x: 200,                       // X position on template (px)
  y: 200,                       // Y position on template (px)
  width: 400,                   // Width of print area (px)
  height: 400,                  // Height of print area (px)
  maxWidth: 400,                // Maximum width for designs (px)
  maxHeight: 400,               // Maximum height for designs (px)
  shape: "rectangle"            // Shape: rectangle, circle, ellipse
}
```

Configuring Print Areas via Admin Panel

1. **Open Print Area Admin**
   - Click gear icon in designer
   - Admin authentication required

2. **Select Product + Color + View**
   - Choose the product
   - Select color variant
   - Select view (front/back/etc.)

3. **Add Print Areas**
   - Click "+ Add" button
   - Enter area name
   - Choose shape (rectangle/circle/ellipse)
   - Click "Add Area"

4. **Adjust Print Areas**
   - **Drag** to move position
   - **Resize** using corner handles
   - **Arrow keys** for fine positioning (1px, 10px with Shift)
   - **Grid** for alignment

5. **Configure Settings**
   - Enable/disable grid
   - Enable/disable snap to grid
   - Adjust grid size

6. **Save Configuration**
   - Click “Save” button
   - Configuration saved to database
   - Linked to specific color+view variant

Print Area Shapes

Rectangle

```
{
  shape: 'rectangle',
  width: 400,
  height: 300
}
```

Used for: Most common designs, logos, text

Circle

```
{
  shape: 'circle',
  width: 300,  // diameter
  height: 300  // diameter (same as width)
}
```

Used for: Round badges, circular logos

Ellipse

```
{
  shape: 'ellipse',
  width: 400,   // horizontal diameter
  height: 300   // vertical diameter
}
```

Used for: Oval designs, stretched circular areas

Print Area Best Practices

1. **Consistent Naming**: Use descriptive names like “Front Center”, “Back Top Logo”
2. **Realistic Sizes**: Match actual printable area on physical product
3. **Safe Margins**: Leave 20-50px margin from edges
4. **Test Designs**: Upload test designs to verify print area accuracy
5. **Document Specs**: Keep notes on print area dimensions and DPI requirements

API Reference

Product Configuration Functions

loadProductConfiguration(productKey, colorCode, viewName)

Load configuration for a specific product variant.

```
const config = await loadProductConfiguration('tote-bag-5oz', '#000000', 'front');
```

Returns:

```
{
  name: "Product Name",
  template: "template URL",
  printAreas: { /* print areas object */ },
  colors: ["#000000", "#FFFFFF"],
  basePrice: 18.99,
  currentColor: "#000000",
  currentView: "front",
  variantId: "uuid"
}
```

saveVariantConfiguration(productKey, colorCode, viewName, config)

Save configuration for a specific variant.

```
await saveVariantConfiguration('tote-bag-5oz', '#FF0000', 'back', {
  colorName: 'Red',
  templateUrl: '/path/to/template.png',
  printAreas: {
    back_center: { x: 200, y: 200, width: 400, height: 400 }
  }
});
```

loadProductVariants(productKey)

Load all variants for a product.

```
const variants = await loadProductVariants('tote-bag-5oz');
```

Returns:

```
{
  colors: {
    "#000000": {
      name: "Black",
      code: "#000000",
      views: {
        front: { variantId: "uuid", templateUrl: "...", printAreas: [...] },
        back: { variantId: "uuid", templateUrl: "...", printAreas: [...] }
      }
    }
  },
  views: ["front", "back"],
  availableColors: ["#000000", "#FFFFFF", "#FF0000"]
}
```

Variant Management Functions

getProductVariants(productTemplateId)

Get all variants for a product template.

getProductVariant(productTemplateId, colorCode, viewName)

Get specific variant by color and view.

createProductVariant(variant)

Create a new product variant.

updateProductVariant(variantId, updates)

Update existing variant.

deleteProductVariant(variantId)

Delete a variant.

upsertProductVariant(productTemplateId, colorCode, viewName, variantData)

Create or update variant.

Print Area Functions

getPrintAreasByVariant(variantId)

Get print areas for a variant.

createPrintAreaForVariant(variantId, printArea)

Create print area for variant.

batchUpdatePrintAreasForVariant(variantId, printAreasConfig)

Batch update all print areas for a variant.

Troubleshooting

Issue: Template Image Not Loading

Symptoms:
- Blank canvas or fallback rectangle
- Console error: “Failed to load template image”

Solutions:
1. **Check file path**:
- Verify template URL is correct
- Ensure file exists in public folder or Supabase Storage

1. **CORS issues** (for external URLs):
   `javascript`
   ```
      // Add crossOrigin attribute
      fabric.Image.fromURL(url, callback, { crossOrigin: 'anonymous' });
   ```

2. **Cache issues**:
   - Clear browser cache
   - Use cache-busting parameter: `?_cb=${Date.now()}`

Issue: Print Areas Not Showing

Symptoms:
- No blue overlay rectangles
- Print area selector shows “No print areas”

Solutions:
1. **Check variant ID**:
`javascript`
```
   const variant = await getProductVariant(templateId, colorCode, viewName);
   console.log('Variant:', variant);
```

1. **Verify print areas exist**:
   `sql`
   ```
      SELECT * FROM print_areas WHERE variant_id = 'your-variant-id';
   ```

2. **Check print area configuration**:
   - Open Print Area Admin
   - Verify print areas are configured for this color+view

Issue: Color/View Switch Not Working

Symptoms:
- Clicking color/view doesn’t change template
- Template loads but print areas don’t update

Solutions:
1. **Check variant exists**:
`javascript`

```
   const variants = await loadProductVariants('product-key');
   console.log('Available variants:', variants);
```

1. **Verify template URL**:
   - Check if variant has template_url set
   - Ensure template image exists

2. **Console logging**:
   `javascript`
   ```
      // Add to EnhancedDesigner.jsx
      useEffect(() => {
        console.log('Loading variant:', { selectedProduct, selectedColor, selectedView });
        loadProductTemplate();
      }, [selectedProduct, selectedColor, selectedView]);
   ```

Issue: Save Configuration Fails

Symptoms:
- “Failed to save configuration” error
- Changes not persisted to database

Solutions:
1. **Check authentication**:
`javascript`
```
   const { data: { user } } = await supabase.auth.getUser();
   console.log('Current user:', user);
```

1. **Verify admin access**:
   `javascript`
   ```
      const isAdmin = await isCurrentUserAdmin();
      console.log('Is admin:', isAdmin);
   ```

2. **Check database permissions**:
   - Verify RLS policies allow insert/update
   - Check user has correct role

3. **Network issues**:
   - Check browser console for network errors
   - Verify Supabase connection

Issue: Template Upload Fails

Symptoms:
- “Failed to upload template” error
- Upload appears to work but image doesn’t display

Solutions:
1. **Check file size**:
- Maximum file size: 5MB recommended
- Compress large images before uploading

1. **Check file type**:
   - Supported: PNG, JPG, JPEG, GIF
   - Recommended: PNG for transparency

2. **Storage bucket settings**:
   `sql`
   ```
     -- Verify bucket is public
     UPDATE storage.buckets
     SET public = true
     WHERE name = 'product-templates';
   ```

3. **Storage policies**:
   `sql`
   ```
     -- Check upload policy exists
     SELECT * FROM storage.policies
     WHERE bucket_id = 'product-templates';
   ```

Summary

This product configuration system provides a comprehensive solution for managing:
- ✅ Multiple color variations per product
- ✅ Multiple views per product (front, back, sides)
- ✅ View-specific print areas
- ✅ Flexible print area shapes (rectangle, circle, ellipse)
- ✅ Template image uploads to Supabase Storage
- ✅ Complete configuration persistence in database

Key Features:
- **Flexible**: Supports any number of colors and views
- **Scalable**: Database-driven configuration
- **User-friendly**: Visual admin panel for configuration
- **Robust**: Comprehensive error handling and validation
- **Well-documented**: Clear API and data structures

For additional support or questions, refer to the inline code documentation in:
- `/src/services/supabaseService.js`
- `/src/pages/EnhancedDesigner.jsx`
- `/src/components/PrintAreaAdmin.jsx`