Système de Design JaelleShop

Ce document sert de référence pour l'utilisation du système de design uniforme dans le projet JaelleShop.

Principes fondamentaux

Le système de design JaelleShop est basé sur les principes suivants :

1. **Cohérence** : Tous les composants et pages doivent partager un langage visuel commun
2. **Accessibilité** : Les composants sont conçus pour être accessibles à tous les utilisateurs
3. **Réutilisabilité** : Des composants modulaires qui peuvent être utilisés dans différents contextes
4. **Responsive** : Adaptation fluide à tous les appareils
5. **Animation subtile** : Animations légères pour améliorer l'expérience utilisateur

Palette de couleurs

Notre palette de couleurs est définie dans le fichier `tailwind.config.js` et accessible via le système de design :

```
import { colors } from '../utils/designSystem';

// Utilisation
<div className="bg-primary-600"></div>
```

Couleurs principales

- **Primary** : Nuances de bleu (50-950)
- **Secondary** : Nuances de rose (50-950)

Couleurs d'accentuation

- **Success** : Vert
- **Danger** : Rouge
- **Warning** : Jaune/Orange
- **Info** : Bleu clair

Typographie

La hiérarchie typographique est définie dans le système de design :

```
import { typography } from '../utils/designSystem';

// Utilisation
<h1 className={typography.headings.h1}>Titre principal</h1>
<p className={typography.body.regular}>Texte normal</p>
```

Composants UI

Button

```
import { Button } from '../components/ui';

// Variantes
<Button>Bouton primaire</Button>
<Button variant="secondary">Bouton secondaire</Button>
<Button variant="accent">Bouton d'accent</Button>
<Button variant="outlined">Bouton contour</Button>
<Button variant="text">Bouton texte</Button>

// Tailles
<Button size="sm">Petit</Button>
<Button size="md">Moyen</Button>
<Button size="lg">Grand</Button>

// Avec icône
<Button
  icon={<svg>...</svg>}
  iconPosition="left"
>
  Avec icône
</Button>

// Comme lien
<Button to="/products">Aller aux produits</Button>

// État de chargement
<Button isLoading={true}>Chargement</Button>
```

Card

```
import { Card } from '../components/ui';

// Basique
<Card>
  Contenu de la carte
</Card>

// Variantes
<Card padding="lg" elevation="high" rounded="xl">
  Carte avec grand padding, ombre forte et coins très arrondis
</Card>

// Comme lien
```

```
<Card to="/product/123">
  Carte cliquable
</Card>

// Avec badge
<Card
  badge={<Badge>Nouveau</Badge>}
  badgePosition="top-right"
>
  Carte avec badge
</Card>
```

Input

```
import { Input } from '../components/ui';

// Basique
<Input
  label="Email"
  placeholder="Entrez votre email"
  helperText="Nous ne partagerons jamais votre email"
/>

// Avec icône
<Input
  leftIcon={<svg>...</svg>}
  rightIcon={<svg>...</svg>}
  iconClickable={true}
  onIconClick={() => console.log('Icon clicked')}
/>

// Avec erreur
<Input
  error="Ce champ est requis"
/>

// Variantes
<Input variant="filled" />
<Input variant="outlined" />

// Tailles
<Input size="sm" />
<Input size="md" />
<Input size="lg" />
```

Badge

```
import { Badge } from '../components/ui';

// Variantes
<Badge>Par défaut</Badge>
<Badge variant="secondary">Secondaire</Badge>
<Badge variant="success">Succès</Badge>
<Badge variant="danger">Danger</Badge>
<Badge variant="warning">Attention</Badge>
<Badge variant="info">Info</Badge>

// Contour
<Badge outline>Contour</Badge>

// Tailles
<Badge size="xs">Très petit</Badge>
<Badge size="sm">Petit</Badge>
<Badge size="md">Moyen</Badge>

// Avec icône
<Badge icon={<svg>...</svg>}>Avec icône</Badge>

// Alternative via les composants prédéfinis
<Badge.Success>Succès</Badge.Success>
```

Layouts et grilles

Le système de design inclut des classes prédéfinies pour les mises en page courantes :

```
import { components } from '../utils/designSystem';

// Grille de produits
<div className={components.gridLayouts.products}>
  {/* Cartes de produits */}
</div>

// Grille de fonctionnalités
<div className={components.gridLayouts.features}>
  {/* Fonctionnalités */}
</div>

// Layout de formulaire
<div className={components.gridLayouts.form}>
  {/* Champs de formulaire */}
</div>
```

Espacement

L'espacement est standardisé à travers l'application :

```jsx
import { spacing } from '../utils/designSystem';

// Espacement entre sections
<div className={spacing.section}>
  {/* Une section */}
</div>

// Espacement entre composants
<div className={spacing.component}>
  {/* Un composant */}
</div>

// Espacement entre éléments
<div className={spacing.element}>
  {/* Un élément */}
</div>

// Padding interne
<div className={spacing.inner}>
  {/* Contenu avec padding */}
</div>
```

Animations

Le système utilise Framer Motion pour des animations cohérentes :

```jsx
import { motion } from 'framer-motion';
import { animations } from '../utils/designSystem';

// Animation de fondu
<motion.div
  variants={animations.fadeIn}
  initial="hidden"
  animate="visible"
>
  Contenu animé
</motion.div>

// Animation de fondu vers le haut
<motion.div
  variants={animations.fadeInUp}
  initial="hidden"
  animate="visible"
>
  Monte en apparaissant
</motion.div>

// Animation séquentielle des enfants
<motion.div
```

```
  variants={animations.staggerChildren}
  initial="hidden"
  animate="visible"
>
  <motion.div variants={animations.fadeIn}>Élément 1</motion.div>
  <motion.div variants={animations.fadeIn}>Élément 2</motion.div>
  <motion.div variants={animations.fadeIn}>Élément 3</motion.div>
</motion.div>
```

Utilisation de l'utilitaire `classNames`

Pour combiner des classes de manière conditionnelle :

```
import { classNames } from '../utils/designSystem';

// Exemple d'utilisation
const buttonClasses = classNames(
  'base-class',
  isActive && 'active-class',
  size === 'large' ? 'large-class' : 'small-class',
  customClass
);
```

Thèmes

Pour les sections spéciales ou des pages thématiques :

```
import { themes } from '../utils/designSystem';

// Thème par défaut
<div className={themes.default.background}>
  <div className={themes.default.card}>
    <h2 className={themes.default.text}>Titre</h2>
    <p className={themes.default.accent}>Texte d'accent</p>
  </div>
</div>

// Thème sombre
<div className={themes.dark.background}>
  <div className={themes.dark.card}>
    <h2 className={themes.dark.text}>Titre</h2>
    <p className={themes.dark.accent}>Texte d'accent</p>
  </div>
</div>
```

Conseils d'implémentation

1. Privilégiez toujours les composants UI prédéfinis plutôt que de recréer des éléments
2. Utilisez le système de design pour toutes les nouvelles fonctionnalités
3. Quand vous modifiez des composants existants, alignez-les avec le système
4. Pour les cas particuliers, étendez le système plutôt que de contourner ses règles
5. Consultez cette documentation avant de créer de nouveaux composants

Contribution au système de design

Pour ajouter ou modifier des éléments du système de design :

1. Discutez des changements avec l'équipe
2. Mettez à jour le fichier `designSystem.ts`
3. Créez ou modifiez les composants concernés
4. Mettez à jour cette documentation
5. Communiquez les changements à l'équipe

Importations

```
import { typography, spacing, components } from '../utils/designSystem';
```