مستندات کامل سیستم ساد - فروشگاه آنلاین قطعات خودرو

نسخه: 1.0
تاریخ: 2024
نویسنده: تیم توسعه ساد

فهرست مطالب

1. معرفی و معماری سیستم

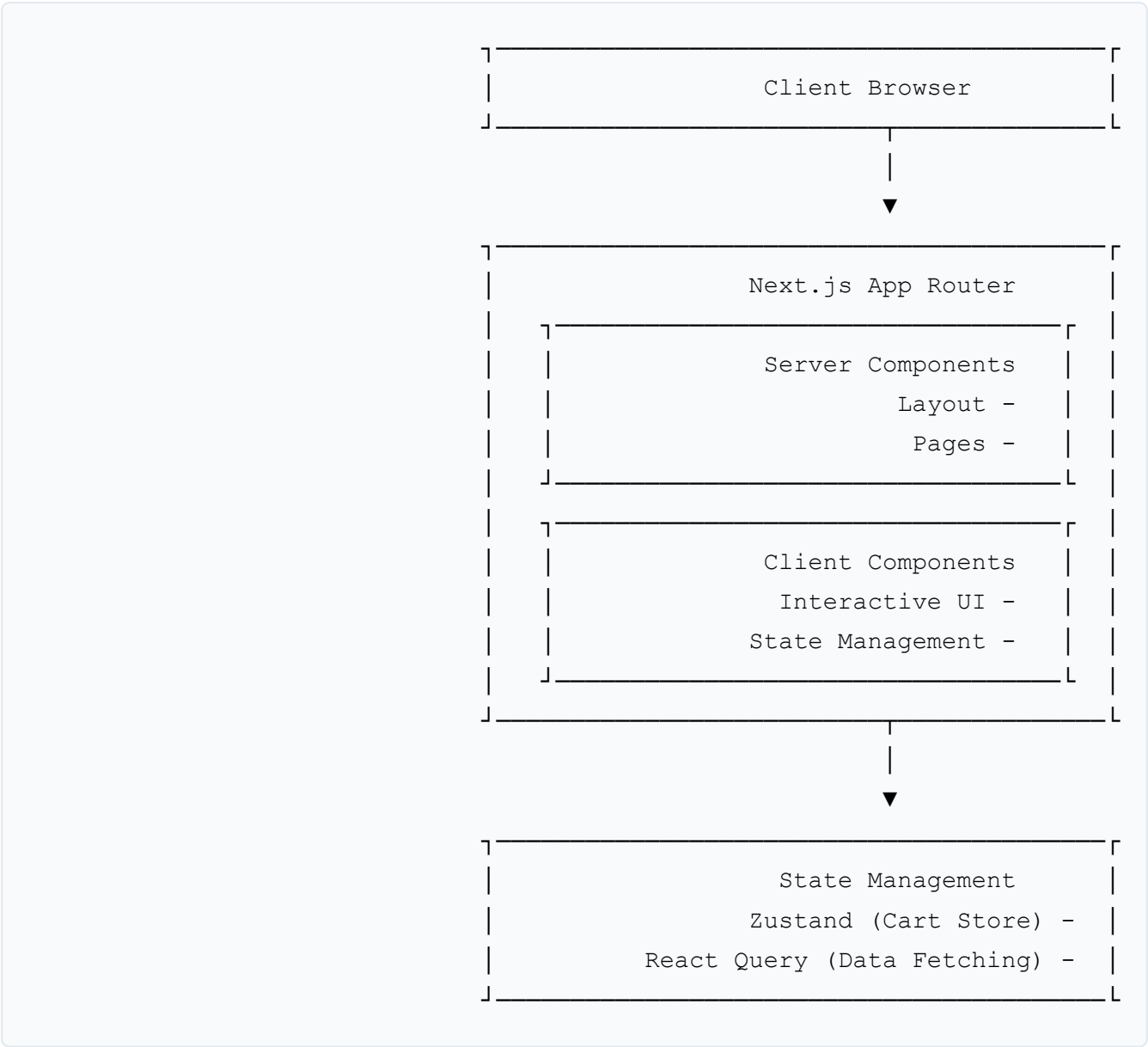
1.1 معرفی پروژه

ساد یک فروشگاه آنلاین مدرن برای فروش قطعات خودرو وارداتی است که با استفاده از آخرین تکنولوژی‌های وب توسعه یافته است. این سیستم به کاربران امکان جستجو، فیلتر، مشاهده جزئیات و خرید قطعات خودرو را می‌دهد.

1.2 معماری کلی

سیستم بر اساس **Next.js 14** با استفاده از **App Router** ساخته شده است. این معماری مدرن امکان استفاده از Server Components، Client Components و API Routes را فراهم می‌کند.

معماری کلی سیستم:



1.3 تکنولوژی‌های استفاده شده

Frontend Framework

- **Next.js 16.0.3**: فریمورک React با قابلیت‌های SSR و SSG
- **React 19.2.0**: کتابخانه UI
- **TypeScript 5**: برای type safety

Styling

- **Tailwind CSS 4**: Framework CSS utility-first
- **Custom Theme**: تم رنگی سفارشی (آبی تیره، قرمز، طلایی)

- **RTL Support**: پشتیبانی کامل از راست به چپ

State Management

- **Zustand 4.5.0**: مدیریت state سبد خرید
- **React Query (TanStack Query) 5.17.0**: مدیریت data fetching و caching

UI Components

- **Radix UI**: کامپوننت‌های دسترسی‌پذیر
- **Shadcn/ui**: کامپوننت‌های UI آماده
- **Lucide React**: آیکون‌ها

Animations

- **Framer Motion 11.0.5**: انیمیشن‌های روان

Other Libraries

- **Next-Auth 4.24.5**: احراز هویت (آماده برای استفاده)
- **React Hook Form 7.49.3**: مدیریت فرم‌ها
- **Zod 3.22.4**: Validation

1.4 ساختار دایرکتوری‌ها

```
/saded
├── app/                          # Next.js App Router
│   ├── layout.tsx                # Layout اصلی با Metadata
│   ├── page.tsx                  # صفحه اصلی
│   ├── globals.css               # استایل‌های全局
│   ├── /products                 # صفحات محصولات
│   │   ├── page.tsx              # لیست محصولات
│   │   └── /[id]                 # صفحه جزئیات محصول
│   │       └── page.tsx
│   └── /cart                     # صفحه سبد خرید
│       └── page.tsx
│
├── /components                   # کامپونت‌های React
│   ├── /ui                       # کامپونت‌های پایه UI
│   │   ├── button.tsx
│   │   ├── card.tsx
│   │   ├── input.tsx
│   │   └── ...
│   ├── # Layout Components       layout/
│   │   ├── # Header              header.tsx با جستجو و سبد
│   │   ├── # Footer              footer.tsx
│   │   └── # Navigation Menu     navigation.tsx
│   ├── /home                     # کامپونت‌های صفحه اصلی
│   │   ├── # Hero                hero-section.tsx با جستجو
│   │   ├── featured-stats.tsx    # آمارها
│   │   ├── category-grid.tsx     # شبکه دسته‌بندی‌ها
│   │   └── featured-products.tsx # محصولات پرفروش
│   ├── /product                  # کامپونت‌های محصولات
│   │   ├── product-grid.tsx      # نمایش Grid
│   │   ├── product-list.tsx      # نمایش List
│   │   ├── product-filters.tsx   # فیلترها
│   │   └── product-detail.tsx    # جزئیات محصول
│   ├── /cart                     # کامپونت‌های سبد خرید
│   │   └── cart-content.tsx
│   └── # React Query Provider    providers.tsx
│
├── # State Management            store/
│   └── # Zustand Store           cart-store.ts برای سبد خرید
│
```

```
├── /lib                    # توابع کمکی
│   └── utils.ts            # توابع utility
│
├── /public                 # فایل‌های استاتیک
│   └── ...
│
├── /__tests__              # تست‌ها
│   └── /components
│
├── package.json            # وابستگی‌ها
├── next.config.ts          # تنظیمات Next.js
├── tsconfig.json           # تنظیمات TypeScript
└── tailwind.config.js      # تنظیمات Tailwind
```

1.5 جریان داده (Data Flow)

```
User Action
    │
    ▼
Client Component (UI)
    │
    ├──► Zustand Store (Cart State)
    │        │
    │        └──► localStorage (Persistence)
    │
    └──► React Query
             │
             └──► API Call (Future)
                      │
                      └──► Backend/Database
```

---

2. توضیح دقیق نحوه کارکرد

2.1 صفحه اصلی (Homepage)

2.1.1 Hero Section

فایل: `components/home/hero-section.tsx`

عملکرد:

- نمایش عنوان و توضیحات اصلی سایت

- جستجوی پیشرفته با قابلیت‌های:
  - جستجوی متنی
  - جستجوی تصویری (آماده برای پیاده‌سازی)
  - جستجو با VIN (آماده برای پیاده‌سازی)
  - جستجوی صوتی (آماده برای پیاده‌سازی)
- دکمه‌های CTA برای هدایت کاربر

تکنولوژی:

- Framer Motion برای انیمیشن‌های ورودی
- Responsive design با Tailwind
- State management محلی با useState

Featured Stats 2.1.2

فایل: `components/home/featured-stats.tsx`

عملکرد:

- نمایش آمارهای کلیدی:
  - تعداد قطعات موجود
  - تعداد مشتریان
  - امتیاز کلی
  - زمان ارسال
- انیمیشن‌های scroll-triggered

Category Grid 2.1.3

فایل: `components/home/category-grid.tsx`

عملکرد:

- نمایش 6 دسته‌بندی اصلی:
  - موتور و قطعات
  - بدنه و شیشه
  - برق و الکترونیک
  - تعلیق و ترمز
  - سیستم خنک‌کننده

    - لوازم جانبی
- هر دسته شامل:
    - آیکون
    - نام
    - تعداد محصولات
    - لینک به صفحه دسته‌بندی

ویژگی‌ها:

- Hover effects
- Responsive grid (2 columns mobile, 3 desktop)
- Framer Motion با Animation

Featured Products 2.1.4

فایل: `components/home/featured-products.tsx`

عملکرد:

- نمایش 4 محصول پرفروش
- هر محصول شامل:
    - تصویر
    - نام
    - قیمت و قیمت اصلی
    - امتیاز و تعداد نظرات
    - دکمه افزودن به سبد
    - دکمه علاقه‌مندی

نکته: در حال حاضر از داده‌های Mock استفاده می‌کند.

2.2 صفحه محصولات (Products Page)

فایل: `app/products/page.tsx`

Product Filters 2.2.1

فایل: `components/product/product-filters.tsx`

عملکرد: فیلترهای پیشرفته شامل:

1. **فیلتر قیمت:**
   - Range Slider
   - محدوده: 0 تا 50,000,000 تومان
   - نمایش قیمت‌ها با فرمت فارسی
2. **فیلتر برند:**
   - Checkbox list
   - برندهای موجود: Brembo, Mann, Castrol, NGK, Bosch, Valeo
   - امکان انتخاب چندگانه
3. **فیلتر دسته‌بندی:**
   - Checkbox list
   - دسته‌بندی‌های مختلف
   - امکان انتخاب چندگانه
4. **فیلتر وضعیت موجودی:**
   - موجود در انبار
   - پیش‌فروش

ویژگی‌ها:

- دکمه "پاک کردن" برای reset فیلترها
- State management محلی با useState
- Responsive design

Product Grid/List 2.2.2

فایل: `components/product/product-grid.tsx`

عملکرد:

- نمایش محصولات در دو حالت:
  - **Grid View**: نمایش کارت‌های محصول
  - **List View**: نمایش لیستی محصولات
- قابلیت تغییر view با دکمه‌های toggle

ویژگی‌های هر محصول:

- تصویر
- نام
- قیمت و قیمت اصلی (در صورت تخفیف)
- امتیاز و تعداد نظرات
- دکمه افزودن به سبد
- دکمه علاقه‌مندی
- Badge تخفیف (در صورت وجود)

:Pagination

- نمایش صفحه‌بندی در پایین
- دکمه‌های قبلی/بعدی
- نمایش شماره صفحات

نکته: در حال حاضر از داده‌های Mock استفاده می‌کند (12 محصول).

2.3 صفحه جزئیات محصول (Product Detail)

فایل: `app/products/[id]/page.tsx` و `components/product/product-detail.tsx`

Image Gallery 2.3.1

عملکرد:

- نمایش تصویر اصلی بزرگ
- Thumbnail gallery در پایین
- امکان تغییر تصویر با کلیک روی thumbnail
- Highlight تصویر انتخاب شده

Product Information 2.3.2

اطلاعات نمایش داده شده:

- نام محصول
- قیمت و قیمت اصلی
- امتیاز و تعداد نظرات
- وضعیت موجودی

- برند
- دسته‌بندی
- Breadcrumb navigation

Quantity Selector 2.3.3

- دکمه‌های + و - برای تغییر تعداد
- حداقل 1 عدد

Action Buttons 2.3.4

- **افزودن به سبد:** اضافه کردن به Zustand store
- **علاقه‌مندی:** (آماده برای پیاده‌سازی)
- **اشتراک‌گذاری:** (آماده برای پیاده‌سازی)

Tabs 2.3.5

تب‌های اطلاعات:

1. **توضیحات:** توضیحات کامل محصول
2. **مشخصات فنی:** جدول مشخصات
3. **نظرات:** (آماده برای پیاده‌سازی)

نکته: در حال حاضر از داده‌های Mock استفاده می‌کند.

2.4 سبد خرید (Shopping Cart)

فایل: `app/cart/page.tsx` و `components/cart/cart-content.tsx`

Cart State Management 2.4.1

فایل: `store/cart-store.ts`

Zustand Store شامل:

- `items` : آرایه آیتم‌های سبد
- `addItem` : افزودن محصول به سبد
- `removeItem` : حذف محصول
- `updateQuantity` : تغییر تعداد
- `clearCart` : پاک کردن سبد

- `getTotal` : محاسبه جمع کل
- `getItemCount` : تعداد کل آیتم‌ها

Persistence:

- استفاده از middleware `persist`
- ذخیره در `localStorage` با key `cart-storage`
- بازیابی خودکار هنگام بارگذاری صفحه

Cart UI 2.4.2

عملکرد:

- نمایش لیست آیتم‌های سبد
- برای هر آیتم:
  - تصویر
  - نام
  - قیمت واحد
  - کنترل تعداد (+/-)
  - دکمه حذف
  - قیمت کل (قیمت × تعداد)

خلاصه سفارش:

- جمع کل
- هزینه ارسال:
  - رایگان برای سفارش بالای 500,000 تومان
  - 50,000 تومان برای سفارش‌های کمتر
- مبلغ نهایی
- دکمه "ادامه خرید" (لینک به checkout - آماده برای پیاده‌سازی)
- دکمه "پاک کردن سبد"

Empty State:

- نمایش پیام و آیکون در صورت خالی بودن سبد
- دکمه هدایت به صفحه محصولات

State Management 2.5

Zustand Cart Store 2.5.1

مزایا:

- سادگی استفاده
- Performance بالا
- TypeScript support
- Persistence با localStorage

ساختار:

```
} interface CartItem
    ;id: string
    ;name: string
    ;price: number
    ;image: string
    ;quantity: number
                    {

} interface CartStore
    ;[]items: CartItem
    ;addItem: (item) => void
    ;removeItem: (id) => void
;updateQuantity: (id, quantity) => void
    ;clearCart: () => void
    ;getTotal: () => number
;getItemCount: () => number
                    {
```

React Query 2.5.2

فایل: `components/providers.tsx`

پیکربندی:

- `staleTime` : 1 دقیقه
- `refetchOnWindowFocus` : false

استفاده:

- آماده برای data fetching از API
- Caching خودکار
- Background refetching

2.6 انیمیشن‌ها

Framer Motion برای:

- Page transitions
- Scroll-triggered animations
- Hover effects
- Component entrance animations

مثال‌ها:

- Hero section fade-in
- Category cards stagger animation
- Product cards hover effects

Responsive Design 2.7

:Breakpoints

- Mobile: < 640px
- Tablet: 640px - 1024px
- Desktop: > 1024px

استراتژی:

- Mobile-first approach
- استفاده از Tailwind responsive classes
- Adaptive layouts
- Touch-friendly UI

---

3. تحلیل SEO

3.1 وضعیت فعلی SEO

3.1.1 Metadata

فایل: `app/layout.tsx`

Metadata موجود:

```
} = export const metadata: Metadata
  title: "ساد - فروشگاه قطعات خودرو وارداتی",
  description: "فروشگاه آنلاین قطعات خودرو وارداتی با بهترین کیفیت و قیمت
  keywords: ["قطعات خودرو", "قطعات وارداتی", "خودرو", "فروشگاه آنلاین"],
;{
```

مشکلات:

- ❌ Metadata فقط در root layout تعریف شده
- ❌ هیچ metadata برای صفحات محصولات وجود ندارد
- ❌ Open Graph tags وجود ندارد
- ❌ Twitter Card tags وجود ندارد
- ❌ Canonical URLs تعریف نشده
- ❌ Robots meta tags وجود ندارد

3.1.2 Structured Data (Schema.org)

وضعیت: ❌ هیچ structured data وجود ندارد

مشکلات:

- عدم وجود Product schema
- عدم وجود Organization schema
- عدم وجود BreadcrumbList schema
- عدم وجود Review schema

Sitemap 3.1.3

وضعیت: ❌ Sitemap وجود ندارد

Robots.txt 3.1.4

وضعیت: ❌ robots.txt وجود ندارد

3.1.5 بهینه‌سازی تصاویر

وضعیت: ⚠️ نسبی

مزایا:

- ✅ استفاده از Next.js Image component
- ✅ Lazy loading
- ✅ Responsive images

مشکلات:

- ❌ Alt text برای همه تصاویر وجود ندارد
- ❌ استفاده از placeholder images
- ❌ عدم بهینه‌سازی اندازه تصاویر

Core Web Vitals 3.1.6

مشکلات احتمالی:

- عدم استفاده از font-display: swap (در حال بررسی)
- عدم بهینه‌سازی bundle size
- عدم استفاده از code splitting بهینه

3.2 پیشنهادات بهبود SEO

Dynamic Metadata 3.2.1

برای صفحات محصولات:

```tsx
// app/products/[id]/page.tsx

export async function generateMetadata({ params }): Promise<Metadata> {
  const product = await getProduct(params.id);

  return {
    title: `${product.name} - ساد`,
    description: product.description,
    openGraph: {
      title: product.name,
      description: product.description,
      images: [product.image],
      type: 'product',
    },
    twitter: {
      card: 'summary_large_image',
      title: product.name,
      description: product.description,
      images: [product.image],
    },
  };
}
```

برای صفحه لیست محصولات:

```tsx
// app/products/page.tsx

export const metadata: Metadata = {
  title: "محصولات - ساد",
  description: "بیش از 50,000 قطعه خودرو از برندهای معتبر",
  openGraph: {
    title: "محصولات - ساد",
    description: "بیش از 50,000 قطعه خودرو از برندهای معتبر",
    type: 'website',
  },
};
```

Schema.org Structured Data 3.2.2

:Product Schema

```
}
"/context": "https://schema.org@",
"type": "Product@",
"name": "لنت ترمز جلو برند Brembo",
"image": "https://example.com/image.jpg",
"description": "لنت ترمز جلو برند Brembo با کیفیت بالا",
"brand": {
"type": "Brand@",
"name": "Brembo"
},{
"offers": {
"type": "Offer@",
"url": "https://example.com/products/1",
"priceCurrency": "IRR",
"price": "1250000",
"availability": "https://schema.org/InStock"
},{
"aggregateRating": {
"type": "AggregateRating@",
"ratingValue": "4.8",
"reviewCount": "124"
}
{
```

:Organization Schema

```
}
,"/context": "https://schema.org@"
,"type": "Organization@"
,"سادد" :"name"
,"url": "https://saded.ir"
,"logo": "https://saded.ir/logo.png"
} :"contactPoint"
,"type": "ContactPoint@"
,"telephone": "+98-21-12345678"
,"contactType": "customer service"
"email": "info@saded.ir"
{
{
```

:BreadcrumbList Schema

```
}
,"/context": "https://schema.org@"
,"type": "BreadcrumbList@"
] :"itemListElement"
}
,"type": "ListItem@"
,position": 1"
,"خانه" :"name"
"item": "https://saded.ir"
,{
}
,"type": "ListItem@"
,position": 2"
,"محصولات" :"name"
"item": "https://saded.ir/products"
{
[
{
```

Sitemap 3.2.3 پویا

ایجاد: `app/sitemap.ts`

```
'import { MetadataRoute } from 'next

} export default function sitemap(): MetadataRoute.Sitemap
'const baseUrl = 'https://saded.ir

Static pages //
] = const staticPages
}
,url: baseUrl
,()lastModified: new Date
,changeFrequency: 'daily' as const
,priority: 1
,{
}
,`url: `${baseUrl}/products
,()lastModified: new Date
,changeFrequency: 'daily' as const
,priority: 0.9
,{
[

Dynamic product pages //
()const products = await getAllProducts
}) <= const productPages = products.map((product)
,`url: `${baseUrl}/products/${product.id}
,lastModified: product.updatedAt
,changeFrequency: 'weekly' as const
,priority: 0.8
(({

return [...staticPages, ...productPages]
{
```

Robots.txt 3.2.4

ایجاد: `app/robots.ts`

```
'import { MetadataRoute } from 'next

} export default function robots(): MetadataRoute.Robots
                                                } return
                                              } :rules
                                        ,'*' :userAgent
                                            ,'/' :allow
                                ,disallow: ['/api/', '/admin/']
                                                      ,{
                          ,'sitemap: 'https://saded.ir/sitemap.xml
                                                        {
                                                          {
```

3.2.5 بهبود تصاویر

اقدامات:

1. اضافه کردن alt text مناسب برای همه تصاویر
2. استفاده از تصاویر واقعی به جای placeholder
3. بهینه‌سازی اندازه و فرمت تصاویر (WebP)
4. استفاده از srcset برای responsive images

مثال:

```
                                                Image>
                                          src={product.image}
{`{product.brand}$ قطعه خودرو - {product.name}$`}=alt
                                              width={600}
                                             height={600}
                                    priority={isMainImage}
                                        "placeholder="blur
                                                      </
```

3.2.6 بهبود Core Web Vitals

اقدامات:

1. **LCP (Largest Contentful Paint):**
   - استفاده از prop `priority` برای تصاویر مهم
   - Preload فونت‌ها
   - بهینه‌سازی CSS
2. **FID (First Input Delay):**
   - کاهش JavaScript bundle size
   - Code splitting
   - استفاده از dynamic imports
3. **CLS (Cumulative Layout Shift):**
   - تعیین width و height برای تصاویر
   - Reserve space برای dynamic content

3.2.7 بهبود URL Structure

پیشنهاد:

- استفاده از slug به جای ID در URL
- مثال: `/products/لنت-ترمز-جلو-برند-brembo` به جای `products/1/`

3.2.8 Meta Tags اضافی

برای بهبود SEO:

```
} = export const metadata: Metadata
        existing metadata ... //
                    } :alternates
 ,'canonical: 'https://saded.ir
                                ,{
                        } :robots
                    ,index: true
                   ,follow: true
                    } :googleBot
                  ,index: true
                 ,follow: true
      ,max-video-preview': -1'
 ,'max-image-preview': 'large'
            ,max-snippet': -1'
                              ,{
                                ,{
                    } :verification
,'google: 'google-site-verification-code
                                ,{
                                  ;{
```

3.3 اولویت‌بندی بهبود SEO

اولویت بالا:

1. ✅ Dynamic metadata برای صفحات محصولات
2. ✅ Schema.org structured data
3. ✅ Sitemap پویا
4. ✅ Robots.txt

اولویت متوسط: 5. ⚠️ Open Graph و Twitter Cards 6. ⚠️ بهبود alt text تصاویر 7. ⚠️ Canonical URLs

اولویت پایین: 8. ⚠️ بهبود URL structure با slug 9. Core Web Vitals

---

4. راهنمای توسعه

4.1 نحوه افزودن ویژگی‌های جدید

4.1.1 افزودن صفحه جدید

مراحل:

1. ایجاد فایل در `app/[route]/page.tsx`
2. اضافه کردن metadata
3. ایجاد کامپوننت‌های مربوطه در `/components`
4. اضافه کردن route به navigation (در صورت نیاز)

مثال - افزودن صفحه "درباره ما":

```
// app/about/page.tsx
import { Header } from "@/components/layout/header";
import { Footer } from "@/components/layout/footer";

export const metadata = {
  title: "درباره ما - ساد",
  description: "درباره فروشگاه ساد و تیم ما",
};

export default function AboutPage() {
  return (
    <div className="flex min-h-screen flex-col">
      <Header />
      <main className="flex-1 container py-8">
        <h1>درباره ما</h1>
        {/* محتوا */}
      </main>
      <Footer />
    </div>
  );
}
```

4.1.2 افزودن کامپوننت جدید

ساختار پیشنهادی:

1. ایجاد فایل در `components/[category]/[component-name].tsx`
2. استفاده از TypeScript interfaces
3. استفاده از Tailwind برای styling
4. Responsive design
5. Accessibility (ARIA labels)

مثال:

```tsx
// components/ui/custom-button.tsx
"use client";

import { Button } from "@/components/ui/button";
import { cn } from "@/lib/utils";

interface CustomButtonProps {
  children: React.ReactNode;
  variant?: "default" | "outline";
  className?: string;
  onClick?: () => void;
}

export function CustomButton({
  children,
  variant = "default",
  className,
  onClick,
}: CustomButtonProps) {
  return (
    <Button
      variant={variant}
      className={cn("custom-styles", className)}
      onClick={onClick}
    >
      {children}
    </Button>
  );
}
```

4.2 اتصال به Backend API

4.2.1 ساختار API Routes

پیشنهاد ساختار:

```
/app
└── /api
    ├── /products
    │   ├── route.ts          # GET /api/products
    │   └── /[id]
    │       └── route.ts      # GET /api/products/[id]
    ├── /cart
    │   └── route.ts          # POST /api/cart
    └── /auth
        └── route.ts          # POST /api/auth
```

4.2.2 استفاده از React Query

مثال - Fetching Products:

```
                                                lib/api/products.ts //
} export async function getProducts(filters?: ProductFilters)
            } ,'const response = await fetch('/api/products
                                              ,'method: 'POST
        ,headers: { 'Content-Type': 'application/json' }
                                ,body: JSON.stringify(filters)
                                                          ;({
                                          ;()return response.json
                                                            {

                                  components/product/product-grid.tsx //
            ;'import { useQuery } from '@tanstack/react-query

                                        } ()export function ProductGrid
          })const { data: products, isLoading } = useQuery
                                        ,queryKey: ['products']
                                          ,queryFn: getProducts
                                                          ;({

                                    ;</ if (isLoading) return <Loading

                                                       ) return
                                                        <div>
                                    ) <= products?.map(product}
  </ ProductCard key={product.id} product={product}>
                                                        {((
                                                       <div/>
                                                             ;(
                                                                {
```

Error Handling 4.2.3

مثال:

```
})const { data, error, isLoading } = useQuery
                                  ,queryKey: ['products']
                                    ,queryFn: getProducts
                                               ,retry: 3
                                    } <= onError: (error)
                          ;console.error('Error fetching products:', error)
                                        Show toast notification //
                                                                ,{
                                                                 ;({
```

4.3 پیاده‌سازی احراز هویت

4.3.1 پیکربندی NextAuth

نصب:

```
pnpm add next-auth
```

پیکربندی:

```
app/api/auth/[...nextauth]/route.ts //
;"import NextAuth from "next-auth
;"import CredentialsProvider from "next-auth/providers/credentials

} = export const authOptions
] :providers
})CredentialsProvider
,"name: "Credentials
} :credentials
,email: { label: "Email", type: "email" }
,password: { label: "Password", type: "password" }
,{
} async authorize(credentials)
Validate credentials against database //
;const user = await validateUser(credentials)
} if (user)
;return user
{
;return null
,{
,({
,[
} :pages
,'signIn: '/auth/signin
,{
} :callbacks
} async jwt({ token, user })
} if (user)
;token.id = user.id
{
;return token
,{
} async session({ session, token })
;session.user.id = token.id
;return session
,{
,{
;{
```

```
;export const handler = NextAuth(authOptions)
;export { handler as GET, handler as POST }
```

استفاده در کامپونت‌ها:

```
components/auth/signin-form.tsx //
;"use client"

;"import { signIn } from "next-auth/react

} ()export function SignInForm
} <= const handleSubmit = async (e: React.FormEvent)
;()e.preventDefault
} ,"const result = await signIn("credentials
,email: e.target.email.value
,password: e.target.password.value
,redirect: false
;({

} if (result?.ok)
;router.push("/dashboard")
{
;{

) return
<form onSubmit={handleSubmit}>
{/* Form fields */}
<form/>
;(
{
```

4.3.2 محافظت از صفحات

Middleware:

```
middleware.ts //
;"import { withAuth } from "next-auth/middleware

})export default withAuth
} :pages
,"signIn: "/auth/signin
,{
;({

} = export const config
,matcher: ["/dashboard/:path*", "/checkout/:path*"]
;{
```

4.4 اضافه کردن درگاه پرداخت

4.4.1 پیکربندی زرین‌پال

نصب:

```
pnpm add @zarinpal/zarinpal-checkout
```

استفاده:

```
                                                    lib/payment/zarinpal.ts //
            ;"import ZarinPalCheckout from "@zarinpal/zarinpal-checkout

                                    )const zarinpal = ZarinPalCheckout.create
                                        ,!process.env.ZARINPAL_MERCHANT_ID
                                      "process.env.NODE_ENV === "production
                                                                          ;(

} export async function createPayment(amount: number, orderId: string)
                                 })const result = await zarinpal.PaymentRequest
                                                          ,Amount: amount
      ,`CallbackURL: `${process.env.NEXT_PUBLIC_URL}/payment/verify
                                        ,`{orderId}$ سفارش` :Description
                                          ,"Email: "customer@example.com
                                                  ,"Mobile: "09123456789
                                                                      ;({

                                                      ;return result
                                                                      {
```

4.4.2 صفحه Checkout

ایجاد: `app/checkout/page.tsx`

```
;"use client"

;"import { useCartStore } from "@/store/cart-store
;"import { createPayment } from "@/lib/payment/zarinpal

} ()export default function CheckoutPage
;()const { items, getTotal } = useCartStore

} <= () const handlePayment = async
;()const amount = getTotal
;()const orderId = generateOrderId

;const payment = await createPayment(amount, orderId)

} if (payment.status === 100)
;window.location.href = payment.url
{
;{

) return
<div>
{/* Checkout form */}
<button/>پـرد اخت<button onClick={handlePayment}>
<div/>
;(
{
```

4.5 بهبود Performance

4.5.1 Code Splitting

:Dynamic Imports

```
// Lazy load heavy components
import dynamic from 'next/dynamic';

const ProductDetail = dynamic(() => import('@/components/product/product-detail'), {
  loading: () => <Loading />,
  ssr: false, // اگر نیاز به SSR نیست
});
```

4.5.2 بهینه‌سازی تصاویر

استفاده از Next.js Image:

```
import Image from 'next/image';

<Image
  src={product.image}
  alt={product.name}
  width={600}
  height={600}
  priority={isAboveFold}
  placeholder="blur"
  blurDataURL={blurDataUrl}
/>
```

4.5.3 Bundle Analysis

نصب:

```
pnpm add @next/bundle-analyzer
```

پیکربندی: `next.config.ts`

```
})const withBundleAnalyzer = require('@next/bundle-analyzer')
                    ,'enabled: process.env.ANALYZE === 'true
                                                        ;({

                                        })module.exports = withBundleAnalyzer
                                                    existing config ... //
                                                        ;({
```

4.6 اضافه کردن تست‌ها

4.6.1 Unit Tests

مثال - تست کامپوننت Button:

```
                                tests__/components/button.test.tsx__ //
          ;'import { render, screen } from '@testing-library/react
                  ;'import { Button } from '@/components/ui/button

                                          } <= () ,'describe('Button
                                      } <= () ,'it('renders correctly
                                ;render(<Button>Click me</Button>)
      ;()expect(screen.getByText('Click me')).toBeInTheDocument
                                                            ;({

                                    } <= () ,'it('handles click events
                                      ;()const handleClick = jest.fn
     ;render(<Button onClick={handleClick}>Click me</Button>)

                                ;()screen.getByText('Click me').click
                    ;expect(handleClick).toHaveBeenCalledTimes(1)
                                                            ;({
                                                              ;({
```

4.6.2 Integration Tests

مثال - تست Cart Store:

```
                                    tests__/store/cart-store.test.ts__ //
                        ;'import { useCartStore } from '@/store/cart-store

                                              } <= () ,'describe('Cart Store
                                                    } <= ())beforeEach
                                  ;()useCartStore.getState().clearCart
                                                                ;({

                                          } <= () ,'it('adds item to cart
                      ;()const { addItem, items } = useCartStore.getState

                                                             })addItem
                                                           ,'id: '1
                                                  ,'name: 'Test Product
                                                           ,price: 1000
                                                  ,'image: '/test.jpg
                                                                ;({

                                        ;expect(items).toHaveLength(1)
                                ;expect(items[0].name).toBe('Test Product')
                                                                ;({

                                    } <= () ,'it('calculates total correctly
                  ;()const { addItem, getTotal } = useCartStore.getState

ddItem({ id: '1', name: 'Product 1', price: 1000, image: '/1.jpg' })
ddItem({ id: '2', name: 'Product 2', price: 2000, image: '/2.jpg' })

                                        ;expect(getTotal()).toBe(3000)
                                                                ;({
                                                                  ;({
```

E2E Tests 4.6.3

استفاده از Playwright:

```
e2e/cart.spec.ts //
;'import { test, expect } from '@playwright/test

} <= test('add to cart flow', async ({ page })
;await page.goto('/products')

Click on first product //
;await page.click('[data-testid="product-card"]:first-child')

Add to cart //
;('("افزودن به سبد")button:has-text')await page.click

Go to cart //
;await page.click('[href="/cart"]')

Verify item in cart //
;()await expect(page.locator('[data-testid="cart-item"]')).toBeVisible
;({
```

Best Practices 4.7

Code Organization 4.7.1

- **Separation of Concerns:** جدا کردن logic از UI
- **Reusable Components:** استفاده مجدد از کامپوننت‌ها
- **Custom Hooks:** استخراج logic تکراری به hooks
- **Type Safety:** استفاده کامل از TypeScript

Error Handling 4.7.2

```
                                                lib/error-handler.ts //
} export function handleError(error: unknown)
                                  } if (error instanceof Error)
                                ;console.error(error.message)
                                   Show toast notification //
                                ;toast.error(error.message)
                                                        } else {
                              ;console.error('Unknown error:', error)
                                          ;('خطایی رخ داد')toast.error
                                                                {
                                                                  {
```

Loading States 4.7.3

```
                                                components/ui/loading.tsx //
 export function Loading({ size = 'md' }: { size?: 'sm' | 'md' | 'lg' })
                                                                ) return
ed-full border-t-2 border-b-2 border-primary ${sizeClasses[size]}`}>
                                                                      ;(
                                                                         {
```

---

5. شناسایی نقاط ضعف

5.1 مشکلات فعلی

5.1.1 داده‌های Mock

مشکل:

- ❌ تمام داده‌های محصولات Mock هستند
- ❌ هیچ اتصالی به Backend وجود ندارد
- ❌ داده‌ها در کامپوننت‌ها hardcode شده‌اند

تأثیر:

- عدم امکان استفاده واقعی از سیستم
- نیاز به refactoring برای اتصال به API

راه حل:

- ایجاد API Routes در Next.js
- اتصال به Database (PostgreSQL/MongoDB)
- استفاده از React Query برای data fetching

5.1.2 عدم وجود Backend

مشکل:

- ❌ هیچ API endpoint وجود ندارد
- ❌ عدم وجود Database
- ❌ عدم وجود Authentication system

تأثیر:

- سیستم فقط یک prototype است
- عدم امکان استفاده در production

راه حل:

- پیاده‌سازی API Routes
- راه‌اندازی Database
- پیاده‌سازی Authentication

5.1.3 SEO ناقص

مشکلات:

- ❌ Metadata فقط در root layout
- ❌ عدم وجود Structured Data
- ❌ عدم وجود Sitemap
- ❌ عدم وجود robots.txt

تأثیر:

- رتبه‌بندی ضعیف در موتورهای جستجو

- عدم نمایش مناسب در شبکه‌های اجتماعی

راه حل:

- پیاده‌سازی Dynamic Metadata
- اضافه کردن Schema.org
- ایجاد Sitemap و robots.txt

5.1.4 عدم وجود Authentication

مشکل:

- ❌ NextAuth نصب شده اما پیکربندی نشده
- ❌ هیچ صفحه Sign In/Sign Up وجود ندارد
- ❌ عدم محافظت از صفحات

تأثیر:

- عدم امکان مدیریت کاربران
- عدم امکان ذخیره اطلاعات کاربر

راه حل:

- پیکربندی NextAuth
- ایجاد صفحات Authentication
- اضافه کردن Middleware

5.1.5 عدم وجود Payment Gateway

مشکل:

- ❌ هیچ درگاه پرداختی پیاده‌سازی نشده
- ❌ صفحه Checkout وجود ندارد
- ❌ عدم امکان تکمیل خرید

تأثیر:

- سیستم فقط برای نمایش است
- عدم امکان فروش واقعی

راه حل:

- پیاده‌سازی زرین‌پال یا Stripe
- ایجاد صفحه Checkout
- پیاده‌سازی Payment Verification

5.1.6 تست‌های محدود

مشکل:

- ❌ فقط یک تست نمونه وجود دارد
- ❌ عدم وجود Integration Tests
- ❌ عدم وجود E2E Tests

تأثیر:

- عدم اطمینان از صحت عملکرد
- احتمال وجود باگ‌های پنهان

راه حل:

- نوشتن Unit Tests برای کامپوننت‌ها
- اضافه کردن Integration Tests
- پیاده‌سازی E2E Tests با Playwright

5.1.7 مشکلات Performance

مشکلات:

- ⚠️ عدم استفاده بهینه از Code Splitting
- ⚠️ عدم بهینه‌سازی Bundle Size
- ⚠️ عدم استفاده از Image Optimization

راه حل:

- استفاده از Dynamic Imports
- Bundle Analysis و بهینه‌سازی
- استفاده کامل از Next.js Image

5.1.8 مشکلات Accessibility

مشکلات:

- ⚠️ عدم وجود ARIA labels در برخی بخش‌ها
- ⚠️ عدم پشتیبانی کامل از Keyboard Navigation
- ⚠️ عدم تست با Screen Readers

راه حل:

- اضافه کردن ARIA labels
- بهبود Keyboard Navigation
- تست با Screen Readers

5.2 مشکلات معماری

5.2.1 عدم وجود Error Boundaries

مشکل:

- ❌ هیچ Error Boundary وجود ندارد
- ❌ خطاها ممکن است کل صفحه را crash کنند

راه حل:

```tsx
// components/error-boundary.tsx
"use client";

import { Component, ReactNode } from 'react';

interface Props {
  children: ReactNode;
}

interface State {
  hasError: boolean;
}

export class ErrorBoundary extends Component<Props, State> {
  constructor(props: Props) {
    super(props);
    this.state = { hasError: false };
  }

  static getDerivedStateFromError() {
    return { hasError: true };
  }

  componentDidCatch(error: Error, errorInfo: React.ErrorInfo) {
    console.error('Error caught:', error, errorInfo);
  }

  render() {
    if (this.state.hasError) {
      return <ErrorFallback />;
    }

    return this.props.children;
  }
}
```

5.2.2 عدم وجود Loading States

مشكل:

- ⚠️ Loading states در برخی بخش‌ها وجود ندارد
- ⚠️ عدم وجود Skeleton Loaders

راه حل:

- اضافه کردن Loading Components
- استفاده از Skeleton Loaders

5.2.3 عدم وجود Analytics

مشکل:

- ❌ هیچ سیستم Analytics وجود ندارد
- ❌ عدم امکان ردیابی رفتار کاربران

راه حل:

- اضافه کردن Google Analytics
- یا استفاده از Vercel Analytics

5.3 مشکلات امنیتی

5.3.1 عدم وجود Input Validation

مشکل:

- ⚠️ Validation فقط در سمت Client
- ⚠️ عدم وجود Server-side Validation

راه حل:

- استفاده از Zod برای Validation
- اضافه کردن Server-side Validation

5.3.2 عدم وجود Rate Limiting

مشکل:

- ❌ هیچ Rate Limiting وجود ندارد
- ❌ امکان Abuse API

راه حل:

- پیاده‌سازی Rate Limiting در API Routes

5.4 اولویت‌بندی رفع مشکلات

اولویت بالا (Critical):

1. ✅ اتصال به Backend و Database
2. ✅ پیاده‌سازی Authentication
3. ✅ اضافه کردن Payment Gateway
4. ✅ بهبود SEO

اولویت متوسط (Important): 5. ⚠️ اضافه کردن تست‌ها 6. ⚠️ بهبود Performance 7. ⚠️ اضافه کردن Error Handling

اولویت پایین (Nice to have): 8. ⚠️ بهبود Accessibility 9. ⚠️ اضافه کردن Analytics 10. Rate Limiting

---

6. نتیجه‌گیری

6.1 خلاصه

سیستم **ساد** یک فروشگاه آنلاین مدرن برای قطعات خودرو است که با استفاده از تکنولوژی‌های پیشرفته ساخته شده است. این سیستم دارای معماری خوب، UI/UX مناسب و قابلیت‌های اولیه است، اما برای استفاده در production نیاز به تکمیل و بهبود دارد.

6.2 نقاط قوت

- ✅ معماری مدرن با Next.js App Router
- ✅ UI/UX مناسب و Responsive
- ✅ استفاده از TypeScript برای Type Safety
- ✅ State Management مناسب با Zustand
- ✅ آماده برای اتصال به Backend

6.3 نقاط ضعف

- ❌ عدم وجود Backend و Database
- ❌ داده‌های Mock

- ❌ SEO ناقص
- ❌ عدم وجود Authentication
- ❌ عدم وجود Payment Gateway
- ❌ تست‌های محدود

6.4 مسیر پیش رو

برای تبدیل این سیستم به یک فروشگاه آنلاین کامل و قابل استفاده، باید:

1. **Backend Development:**
   - راه‌اندازی Database
   - ایجاد API Routes
   - پیاده‌سازی Authentication
2. **Feature Completion:**
   - اضافه کردن Payment Gateway
   - تکمیل فرآیند خرید
   - پیاده‌سازی پنل کاربری
3. **SEO & Performance:**
   - بهبود SEO
   - بهینه‌سازی Performance
   - بهبود Core Web Vitals
4. **Testing & Quality:**
   - نوشتن تست‌های جامع
   - بهبود Code Quality
   - اضافه کردن Error Handling
5. **Deployment:**
   - آماده‌سازی برای Production
   - راه‌اندازی CI/CD
   - Monitoring و Analytics

پایان مستندات

این مستندات به صورت مداوم به‌روزرسانی می‌شود. برای آخرین نسخه، به مخزن پروژه مراجعه کنید.