Pflichtenheft: Webshop

Dieses Pflichtenheft beschreibt die technische Umsetzung eines Webshops mit Produktanzeige, Warenkorb, Zahlungsabwicklung und einem Admin-Panel.

1. Einleitung

Der Webshop soll Kunden ein nahtloses Einkaufserlebnis bieten, Produkte übersichtlich darstellen und gängige Zahlungsmethoden unterstützen. Administratoren benötigen ein intuitives Panel zur Produktverwaltung. Besonderer Wert wird auf ein responsives Design gelegt.

2. Technische Umsetzung

2.1. Architektur

- **Client-Server-Architektur:** Trennung von Frontend (Client) und Backend (Server).
- **API-basiert:** Frontend kommuniziert über RESTful APIs mit dem Backend.

2.2. Technologien

- **Frontend:**
    - **Framework:** React (für interaktive Benutzeroberflächen)
    - **Meta-Framework:** Next.js (für Server-Side Rendering (SSR) und Static Site Generation (SSG) zur Verbesserung von SEO und Performance)
    - **Styling:** Tailwind CSS (für schnelles, responsives und utility-first CSS)
    - **Icon-Bibliothek:** Lucide-React oder Heroicons
- **Backend:**
    - **Laufzeitumgebung:** Node.js
    - **Web-Framework:** Express.js (für RESTful API-Endpunkte)
    - **Authentifizierung/Autorisierung:** JWT (JSON Web Tokens) für sichere API-Zugriffe
- **Datenbank:**
    - **Typ:** MongoDB (NoSQL-Datenbank, flexibel für Produktkataloge, Bestellungen, Benutzerdaten)
    - **ODM (Object Data Modeling):** Mongoose (für Node.js, zur einfacheren Interaktion mit MongoDB)

2.3. Funktionen und Module

2.3.1. Produktanzeige

- **Frontend:**

        - Produktliste mit Paginierung, Filter- und Sortieroptionen.
        - Produktdetailseite mit Bildern, Beschreibung, Preis, Verfügbarkeit und "In den Warenkorb"-Button.
        - Suchfunktion.
- **Backend:**
    - API-Endpunkte zum Abrufen aller Produkte, einzelner Produkte, Filtern und Suchen.

2.3.2. Warenkorb-System

- **Frontend:**
    - Anzeige der ausgewählten Produkte mit Menge und Gesamtpreis.
    - Möglichkeit, Mengen zu ändern oder Produkte aus dem Warenkorb zu entfernen.
    - "Zur Kasse gehen"-Button.
- **Backend:**
    - API-Endpunkte zum Hinzufügen, Aktualisieren und Entfernen von Produkten im Warenkorb (serverseitige Warenkorb-Logik, um Konsistenz zu gewährleisten).

2.3.3. Bezahlung

- **Technologien:**
    - **PayPal Integration:** Offizielle PayPal REST API oder PayPal Checkout SDK.
    - **Kreditkarten Integration:** Stripe API (bietet eine robuste und sichere Lösung für Kreditkartenzahlungen).
- **Ablauf:**
    - Kunde wählt Zahlungsmethode an der Kasse.
    - Frontend leitet die Zahlungsanfrage an das Backend weiter.
    - Backend kommuniziert sicher mit den jeweiligen Zahlungs-Gateways (PayPal/Stripe).
    - Nach erfolgreicher Zahlung wird die Bestellung im Backend als "bezahlt" markiert und eine Bestätigung an den Kunden gesendet.
    - Fehlerbehandlung und Rückmeldung an den Kunden bei fehlgeschlagenen Transaktionen.
- **Sicherheit:** PCI DSS-Konformität (durch Nutzung von Stripe/PayPal, die diese Anforderungen erfüllen). Keine Speicherung sensibler Kreditkartendaten auf eigenen Servern.

2.3.4. Admin-Panel

- **Implementierung:** Separates Frontend-Projekt (ebenfalls React/Next.js/Tailwind CSS) mit dedizierten Routen und Komponenten.

- **Authentifizierung/Autorisierung:**
    - Administratoren melden sich mit spezifischen Admin-Zugangsdaten an.
    - JWT-basierte Authentifizierung.
    - Rollenbasierte Zugriffskontrolle (RBAC), um sicherzustellen, dass nur autorisierte Admins auf bestimmte Funktionen zugreifen können (z.B. Produkte anlegen, Bestellungen verwalten).
- **Funktionen:**
    - Produktverwaltung (Anlegen, Bearbeiten, Löschen von Produkten, Hochladen von Bildern).
    - Bestellverwaltung (Anzeigen, Status ändern).
    - Benutzerverwaltung (optional).
- **Backend:**
    - Geschützte API-Endpunkte für Admin-Operationen, die eine Admin-Rolle erfordern.

2.4. Responsives Design (Benutzeroberfläche)

- **Ansatz:** Mobile-First-Design mit Tailwind CSS.
- **Techniken:**
    - **Fluid Grids:** Verwendung von Flexbox und CSS Grid für flexible Layouts.
    - **Media Queries:** Tailwind's responsive Breakpoints (`sm:`, `md:`, `lg:`, `xl:`) für Anpassungen von Layout, Schriftgrößen, Abständen etc.
    - **Flexible Bilder:** `max-w-full`, `height-auto` für Bilder, um Skalierung zu gewährleisten.
    - **Touch-Optimierung:** Ausreichend große Touch-Targets für Buttons und Links auf Mobilgeräten.
    - **Navigation:** Responsive Navigationsmenüs (z.B. Hamburger-Menü auf Mobilgeräten).

2.5. Sicherheit

- **Authentifizierung/Autorisierung:** JWT, RBAC.
- **Datenbank:** Schutz vor SQL-Injection (nicht relevant für NoSQL, aber allgemeine Validierung), Schutz vor NoSQL-Injection.
- **API-Sicherheit:** HTTPS/SSL für alle Kommunikationen, Rate Limiting, CORS-Konfiguration.
- **Eingabevalidierung:** Serverseitige Validierung aller Benutzereingaben.
- **Passwort-Hashing:** Sicheres Hashing von Passwörtern (z.B. bcrypt).

- **Sicherheits-Header:** Implementierung von HTTP Security Headers (z.B. CSP, X-Content-Type-Options).

2.6. Teststrategie

- **Unit Tests:** Für Frontend-Komponenten (z.B. Jest, React Testing Library) und Backend-Logik (z.B. Jest, Mocha).
- **Integration Tests:** Für API-Endpunkte und Datenbankinteraktionen.
- **End-to-End Tests:** Mit Tools wie Cypress oder Playwright, um den gesamten Benutzerfluss zu testen (Produktauswahl, Warenkorb, Checkout, Admin-Panel).
- **Performance Tests:** Lasttests, um die Skalierbarkeit des Systems zu prüfen.
- **Sicherheitstests:** Penetrationstests, Schwachstellen-Scans.
- **Cross-Browser/Cross-Device Tests:** Manuelle und automatisierte Tests auf verschiedenen Browsern und Geräten.