Cahier des charges pour la construction d'un mini catalogue de vente en ligne avec JavaScript, Node.js, Express, HTML, CSSS

1. Introduction

Le but de ce cahier des charges est de définir les spécifications fonctionnelles et techniques pour le développement d'un site Web e-commerce utilisant JavaScript. Le site permettra aux utilisateurs de parcourir, rechercher, acheter des produits et gérer leur compte.

2. Objectifs

- Créer une plateforme e-commerce conviviale et intuitive pour les utilisateurs.
- Permettre aux utilisateurs de parcourir une large gamme de produits, de les ajouter au panier et de finaliser leur achat en toute sécurité.
- Fournir un système de gestion des comptes utilisateur permettant l'inscription, la connexion, la gestion des informations personnelles et le suivi des commandes.
- Intégrer des fonctionnalités de recherche avancées pour aider les utilisateurs à trouver rapidement les produits souhaités.
- Implémenter un processus de paiement sécurisé et fiable.

3. Spécifications fonctionnelles

3.1. Interface utilisateur

- Page d'accueil présentant les produits populaires, les offres spéciales et les catégories de produits.
- Pages de catégorie pour parcourir les produits par type.
- Page de détail du produit avec des informations détaillées, des images et des options de sélection.
- Panier d'achat permettant aux utilisateurs de voir les produits ajoutés et de passer à la caisse.
- Processus de paiement avec des options de paiement sécurisées.
- Pages d'inscription et de connexion pour les utilisateurs.

- Tableau de bord utilisateur pour gérer les informations personnelles, les commandes passées et les informations de paiement.

3.2. Gestion des produits

- Ajout, édition et suppression de produits pour les administrateurs.
- Gestion des catégories et des sous-catégories de produits.
- Options de filtrage et de tri des produits pour les utilisateurs.

3.3. Gestion des utilisateurs

- Formulaire d'inscription avec validation des champs.
- Système de connexion avec gestion des sessions utilisateur.
- Modification des informations personnelles (adresse, mot de passe, etc.).
- Historique des commandes avec suivi des statuts de livraison.

3.4. Fonctionnalités de recherche

- Barre de recherche permettant aux utilisateurs de rechercher des produits par mot-clé.
- Options de recherche avancée avec des filtres par catégorie, prix, etc.

3.5. Processus de paiement

- Intégration de passerelles de paiement sécurisées (ex: PayPal, Stripe).
- Processus de paiement en plusieurs étapes avec confirmation de commande et réception de confirmation par e-mail.

4. Spécifications techniques

4.1. Frontend

- Utilisation de HTML5, CSS3 et JavaScript pour le développement de l'interface utilisateur.
- Framework JavaScript moderne tel que React, Angular ou Vue.js pour la construction des composants frontaux.
- Utilisation de bibliothèques ou de frameworks CSS pour un design réactif et attrayant.

4.2. Backend

- Utilisation de Node.js pour le développement du backend.

- Framework web tel que Express.js pour la gestion des routes et des requêtes HTTP.
- Base de données relationnelle ou non relationnelle pour le stockage des données des utilisateurs, des produits et des commandes (ex: MongoDB, PostgreSQL, MySQL).

4.3. Sécurité

- Cryptage des données sensibles telles que les mots de passe des utilisateurs.
- Utilisation de HTTPS pour sécuriser les communications entre le serveur et le navigateur.
- Validation et échappement des données utilisateur pour éviter les attaques XSS et les injections SQL.

4.4. Performances

- Optimisation des requêtes et de la charge du serveur pour assurer des temps de réponse rapides.
- Mise en cache des données fréquemment consultées pour améliorer les performances.

4.5. Hébergement et déploiement

- Hébergement sur une plateforme cloud fiable et scalable (ex: AWS, Google Cloud, Heroku).
- Utilisation d'outils de déploiement automatisés pour faciliter le déploiement continu.

5. Conclusion

Ce cahier des charges définit les spécifications fonctionnelles et techniques pour le développement d'un site Web e-commerce robuste et convivial utilisant JavaScript. En suivant ces directives, l'objectif est de créer une expérience utilisateur optimale tout en assurant la sécurité et la fiabilité de la plateforme.

1. Structure HTML :

• Créez des fichiers HTML pour différentes pages comme :

- l'accueil
- la liste des produits,
- les détails du produit,
- le panier,
- la caisse, etc.

• Structurez chaque page avec des balises et des éléments HTML appropriés.

2. Style CSS :

• Stylisez vos éléments HTML à l'aide de CSS pour rendre votre site Web visuellement attrayant et convivial.

3. Fonctionnalité JavaScript :

• Implémentez JavaScript pour gérer diverses fonctionnalités telles que :

• Ajout de produits au panier

• Mise à jour du panier

• Supprimer des articles du panier

• Calcul des totaux

• Gestion de l'authentification et de l'autorisation des utilisateurs

• Récupération des données produit à partir d'un serveur backend (à l'aide d'AJAX ou de l'API Fetch)

• Implémentation de la validation de formulaire côté client

• Gestion du processus de paiement

• Gestion des sessions utilisateurs

• Implémentation de la fonctionnalité de recherche

• Mise en œuvre de la pagination pour les listes de produits

• Mise en œuvre de filtres pour les listes de produits (par exemple, par catégorie, gamme de prix)

4. Développement back-end :

serveur principal pour gérer des tâches telles que l'authentification des utilisateurs, le stockage des données produit, la gestion des sessions utilisateur, le traitement des paiements, etc.

• Choisissez une pile technologique backend comme Node.js, Django, Flask, Ruby on Rails, etc., en fonction de votre familiarité et des exigences du projet.

• Implémentez des API RESTful pour interagir avec votre code JavaScript frontend.

5. Gestion de la base de données :

• Choisissez un système de base de données approprié (par exemple, MySQL, PostgreSQL, MongoDB) pour stocker les données utilisateur, les informations sur les produits, les commandes, etc.

• Concevoir et créer les tables et schémas de base de données nécessaires.

6. Intégration avec la passerelle de paiement :

• Mettre en œuvre l'intégration avec une passerelle de paiement (par exemple, Stripe, PayPal) pour gérer le traitement des paiements en toute sécurité.

7. Considérations de sécurité :

• Mettre en œuvre des mesures de sécurité pour protéger les données des utilisateurs, empêcher les failles XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery) et autres vulnérabilités de sécurité.

8. Tests et déploiement :

• Testez minutieusement votre site Web pour vous assurer qu'il fonctionne comme prévu et qu'il est exempt de bugs.

• Déployez votre site Web sur un service ou une plateforme d'hébergement Web.

Ce plan fournit un aperçu général de ce qu'implique la création d'un site Web de commerce électronique fonctionnel à l'aide de JavaScript. En fonction de votre expertise et de vos besoins, vous devrez peut-être approfondir chaque aspect et explorer des technologies et techniques supplémentaires.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ma Boutique en ligne</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Ma Boutique en ligne</h1>
        <button id="cartBtn">Panier</button>
    </header>
    <main id="productContainer">
        <!-- Contenu des produits sera ajouté ici dynamiquement -->
    </main>
    <div id="cartOverlay" class="overlay">
```

```
        <div class="cartContent">
            <span class="closeBtn" onclick="closeCart()">&times;</span>
            <h2>Panier</h2>
            <ul id="cartItems">
                <!-- Contenu du panier sera ajouté ici dynamiquement -->
            </ul>
            <button id="checkoutBtn">Passer la commande</button>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

header {
    background-color: #333;
    color: #fff;
    padding: 20px;
    text-align: center;
```

```
}

h1 {
    margin: 0;
}

button {
    background-color: #007bff;
    color: #fff;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
}

main {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    padding: 20px;
}

.product {
    width: 300px;
    border: 1px solid #ccc;
    margin: 10px;
    padding: 10px;
}

.product img {
    max-width: 100%;
}
```

```
.overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
    display: none;
    justify-content: center;
    align-items: center;
}

.cartContent {
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    max-width: 400px;
}

.closeBtn {
    float: right;
    cursor: pointer;
}

#cartItems {
    list-style: none;
    padding: 0;
}

.cartItem {
```

```
    padding: 10px 0;
    border-bottom: 1px solid #ccc;
}

#checkoutBtn {
    background-color: #28a745;
    color: #fff;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    margin-top: 10px;
    width: 100%;
}

#checkoutBtn:hover {
    background-color: #218838;
}
```

JS

```
// Données des produits (exemple)
const products = [
    { id: 1, name: "Produit 1", price: 20, image: "product1.jpg" },
    { id: 2, name: "Produit 2", price: 30, image: "product2.jpg" },
    { id: 3, name: "Produit 3", price: 25, image: "product3.jpg" }
];

// Afficher les produits
const productContainer = document.getElementById('productContainer');

products.forEach(product => {
    const productCard = document.createElement('div');
```

```
    productCard.classList.add('product');
    productCard.innerHTML = `
        <img src="${product.image}" alt="${product.name}">
        <h3>${product.name}</h3>
        <p>${product.price} €</p>
        <button onclick="addToCart(${product.id})">Ajouter au panier</button>
    `;
    productContainer.appendChild(productCard);
});

// Gestion du panier
const cartBtn = document.getElementById('cartBtn');
const cartOverlay = document.getElementById('cartOverlay');
const checkoutBtn = document.getElementById('checkoutBtn');

cartBtn.addEventListener('click', openCart);
checkoutBtn.addEventListener('click', checkout);

function openCart() {
    cartOverlay.style.display = 'flex';
}

function closeCart() {
    cartOverlay.style.display = 'none';
}

function addToCart(productId) {
    const product = products.find(item => item.id === productId);
    const cartItems = document.getElementById('cartItems');
    const cartItem = document.createElement('li');
    cartItem.classList.add('cartItem');
```

```
    cartItem.innerHTML = `${product.name} - ${product.price} €`;
    cartItems.appendChild(cartItem);
}

function checkout() {
    // Logique de paiement
    alert('Merci pour votre commande!');
    // Vider le panier
    const cartItems = document.getElementById('cartItems');
    cartItems.innerHTML = '';
    closeCart();
}
```