# Ecole Marocaine des Sciences de l'Ingénieur (Page Officielle)

# Rapport Technique - Optique Saadi

**Remerciements :**

**Mme Nouhaila Moussammi, Professeure en JavaScript à l'EMSI 3iir**

Je tiens à adresser mes plus sincères remerciements à Mme Nouhaila Moussammi, professeure en JavaScript à l'EMSI 3iir, pour son enseignement exceptionnel et son engagement constant tout au long de l'année. Grâce à vous, j'ai non seulement acquis des connaissances techniques solides en JavaScript, mais aussi une véritable passion pour le développement web et la programmation.

Votre approche pédagogique a toujours été claire, structurée et adaptée à chacun de vos étudiants. Vous avez su rendre des concepts parfois complexes faciles à comprendre, en les illustrant avec des exemples concrets et des exercices pratiques. Votre capacité à expliquer les notions fondamentales de manière simple et accessible a été un véritable atout pour moi, et j’ai énormément apprécié la patience et l'attention que vous accordez à chaque étudiant, quel que soit son niveau.

Ce qui m'a le plus impressionné chez vous, c'est votre dévouement à votre matière et à vos étudiants. Vous avez toujours été disponible pour répondre à nos questions, qu'elles soient posées pendant les cours, lors des heures de bureau ou même après les heures de classe. Vous avez su créer une atmosphère d'apprentissage collaborative, où chacun de nous se sentait encouragé à poser des questions, à échanger des idées et à développer ses compétences.

Au-delà des compétences techniques en JavaScript que vous nous avez transmises, vous nous avez également appris à être créatifs et à adopter une approche de résolution de problèmes logique et efficace. Vous avez su, avec une grande finesse, équilibrer théorie et pratique, ce qui a permis d'ancrer durablement les connaissances et de nous rendre plus confiants dans nos capacités de développeurs.

Votre soutien et votre encouragement m'ont permis de surmonter des défis complexes, de m'améliorer sans cesse et d’atteindre un niveau que je n’aurais pas cru possible au début de l'année. Vous avez été une véritable source d'inspiration pour moi et pour mes camarades. Votre passion pour votre métier a été contagieuse et a motivé chacun de nous à donner le meilleur de nous-mêmes.

Je vous suis profondément reconnaissant(e) pour votre pédagogie, votre professionnalisme et votre engagement à nous voir progresser. Vous avez non seulement contribué à notre développement académique, mais aussi à notre épanouissement personnel en tant qu'apprenants. Grâce à vous, je me sens prêt(e) à aborder les défis futurs en tant que développeur web avec confiance et assurance.

Merci encore pour tout ce que vous avez fait pour nous. Votre impact positif sur ma formation et ma carrière est inestimable, et je vous en serai toujours reconnaissant(e).

**Définition des technologies utilisées:**

**Visual Studio Code:**

Visual Studio Code (VS Code) est un éditeur de code source léger mais puissant, développé par Microsoft. Il est compatible avec de nombreuses plateformes (Windows, macOS, Linux) et prend en charge plusieurs langages de programmation. Ses principales fonctionnalités incluent :

* Autocomplétion intelligente grâce à IntelliSense.
* Debugger intégré.
* Extensions pour ajouter des fonctionnalités (comme la prise en charge de frameworks ou outils spécifiques).
* Intégration avec Git pour la gestion de versions.

**HTML (HyperText Markup Language):**

HTML est le langage standard utilisé pour structurer et présenter le contenu sur le web. Il permet de créer la structure de base des pages web en utilisant des balises. Ces balises définissent des éléments tels que des titres, des paragraphes, des liens, des images, et bien plus encore.

**CSS (Cascading Style Sheets):**

CSS est un langage de style utilisé pour décrire l'apparence et la mise en page d'un document HTML. Il permet de contrôler des aspects tels que les couleurs, les polices, les marges, et l'agencement des éléments, rendant ainsi les pages web plus attrayantes et professionnelles.

**JavaScript:**

JavaScript est un langage de programmation côté client qui permet d'ajouter de l'interactivité et des fonctionnalités dynamiques aux sites web. Il est utilisé pour :

* La manipulation du DOM (Document Object Model) pour modifier le contenu et la structure des pages web en temps réel.
* La gestion des événements, comme les clics ou les saisies clavier.
* La communication avec des serveurs via AJAX ou des API REST.

***Cahier de charge:***

Le projet consiste à développer un panneau d'administration moderne et fonctionnel pour la boutique en ligne **Optique Saadi**. Ce panneau permettra de gérer les données liées aux produits, clients, fournisseurs et stocks. Accessible via une authentification sécurisée (nom d'utilisateur et mot de passe), l'interface proposera des fonctionnalités essentielles : l’ajout, la modification, et la suppression de produits, clients et fournisseurs, ainsi que la gestion des stocks et l'affichage d'un état global des quantités disponibles. De plus, un tableau de bord avec des statistiques visuelles sera intégré pour offrir une vue d’ensemble des données sous forme de graphiques interactifs.

L'application, entièrement basée sur des technologies front-end (HTML5, CSS3, et JavaScript), fonctionnera sans serveur, en utilisant **localStorage** pour stocker localement les données. La bibliothèque **Chart.js** sera utilisée pour les graphiques. L’interface sera conçue pour être simple, intuitive et responsive, permettant une utilisation fluide sur différents supports, tels que les ordinateurs et les tablettes.

Le projet devra respecter plusieurs contraintes : compatibilité avec les navigateurs modernes, persistance des données via localStorage, et retour utilisateur clair (messages d’erreur, confirmations, etc.). Les livrables attendus incluent une interface complète, le code source, une documentation explicative et un cahier de test pour valider les fonctionnalités principales. Le développement se déroulera en quatre phases : conception, développement, test/validation, et livraison finale, sur une durée estimée de quatre semaines.

Ce projet vise à répondre aux besoins opérationnels d’Optique Saadi, en offrant une solution simple et efficace pour la gestion des activités administratives, tout en garantissant une expérience utilisateur optimisée. Toutefois, il est à noter que l'utilisation de localStorage présente certaines limitations, notamment en matière de sécurité et de scalabilité, ce qui limite son usage à des environnements simples.

***Page acceuil:***

## Introduction:

Ce document détaille la conception et la réalisation du site web pour 'Optique Saadi'. Le site est moderne, sans PHP, et utilise des technologies front-end telles que HTML, CSS, et JavaScript pour fournir une expérience utilisateur fluide. L'API et JSON sont utilisés pour la gestion dynamique des données.

## Structure du Site:

Le site est organisé en quatre sections principales accessibles via une barre de navigation : Accueil, Nos Produits, Panier, et Contact.

### 1. Accueil:

La page d'accueil contient une bannière moderne et une introduction à l'opticien.

### 2. Nos Produits:

Affiche une liste de produits récupérés dynamiquement via une API JSON.

### 3. Panier:

Permet de visualiser et gérer les produits ajoutés au panier.

### 4. Contact:

Inclut un formulaire pour que les utilisateurs puissent envoyer un message à l'équipe d'Optique Saadi.

## Design et Styles:

Le site utilise un design réactif basé sur Flexbox et Grid pour garantir une expérience optimale sur tous les appareils. Les couleurs principales reflètent la modernité et la confiance, tout en restant sobres pour un public professionnel.

Exemple de styles principaux :

- Police principale : Arial, sans-serif  
 - Couleurs : Bleu (#007BFF) et Blanc (#FFFFFF)  
 - Espacement uniforme pour une meilleure lisibilité

## Fonctionnalités Interactives:

Le site comprend des fonctionnalités interactives alimentées par JavaScript :

- Chargement dynamique des produits depuis une API JSON  
 - Gestion du panier localement avec mise à jour en temps réel  
 - Validation de formulaire avec des retours visuels

## Technologies Utilisées:

Les technologies utilisées pour ce projet incluent :

- HTML5 : Structure et contenu du site  
 - CSS3 : Design, mise en page et animations  
 - JavaScript : Fonctionnalités interactives  
 - JSON : Gestion et manipulation des données

## Conclusion:

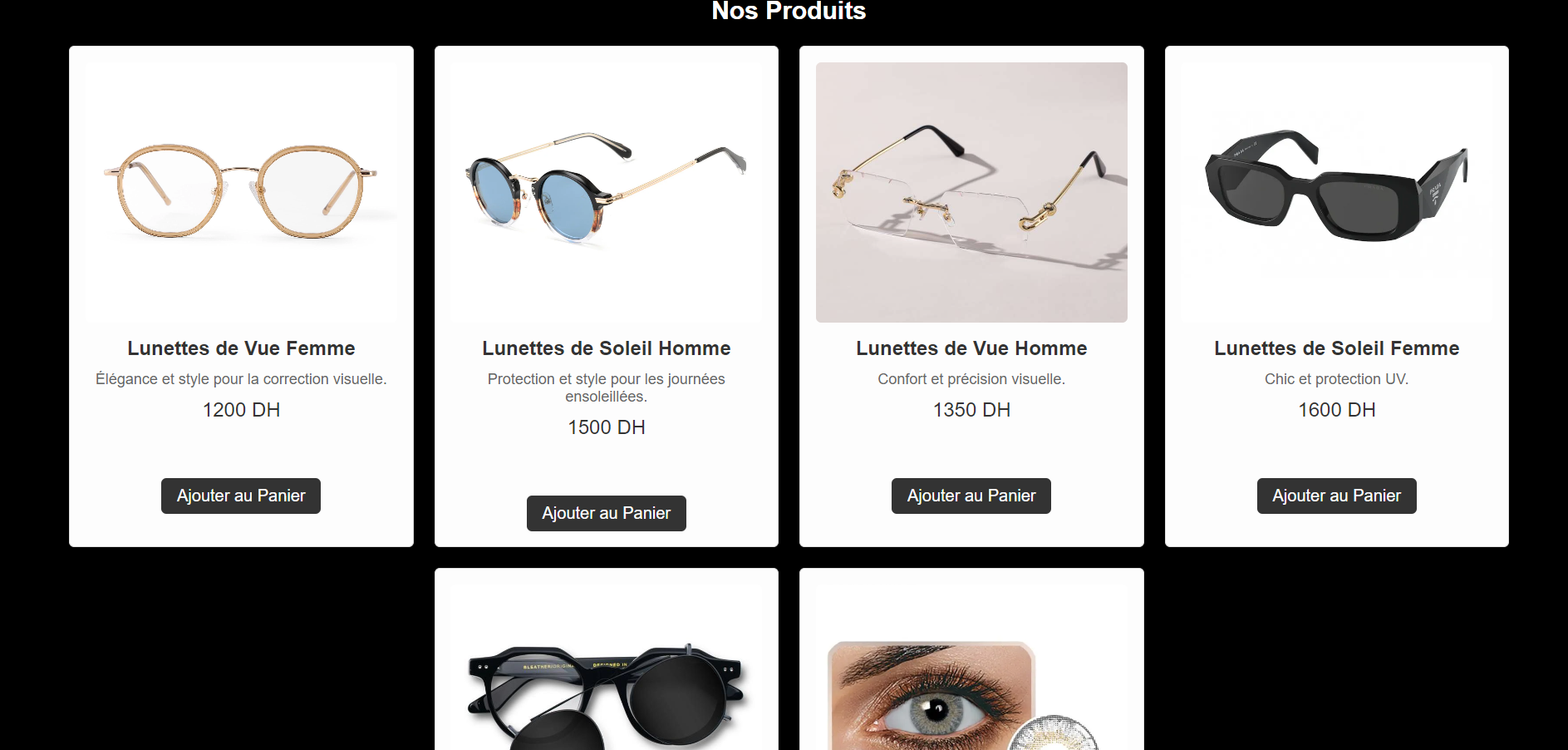
Le site web d'Optique Saadi est un exemple de conception moderne et fonctionnelle, répondant aux besoins d'un opticien souhaitant mettre en valeur ses produits et interagir efficacement avec ses clients.

**Resultat:**

**Accueil:**



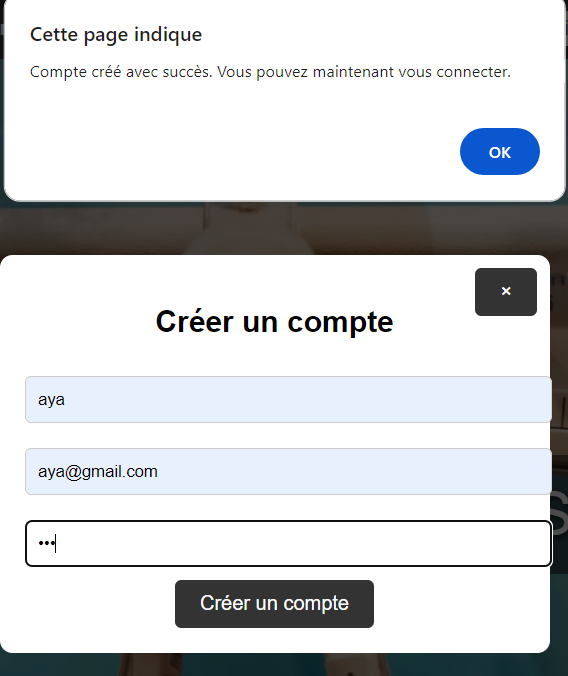
**Nos produits:**

****

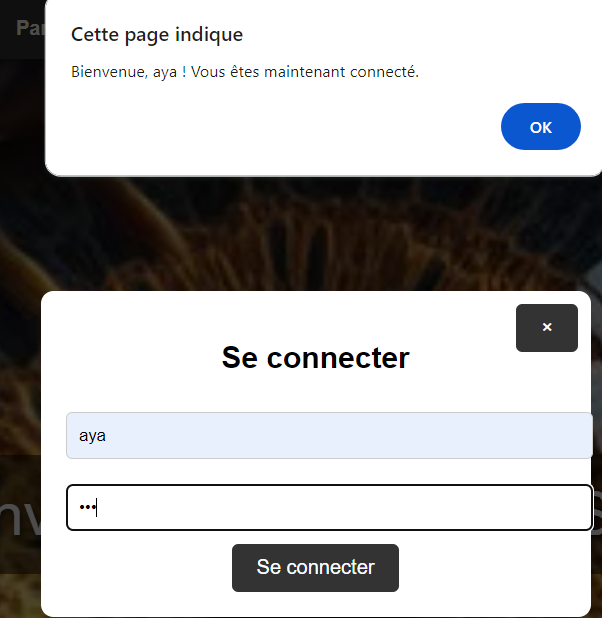
**Contact:**

****

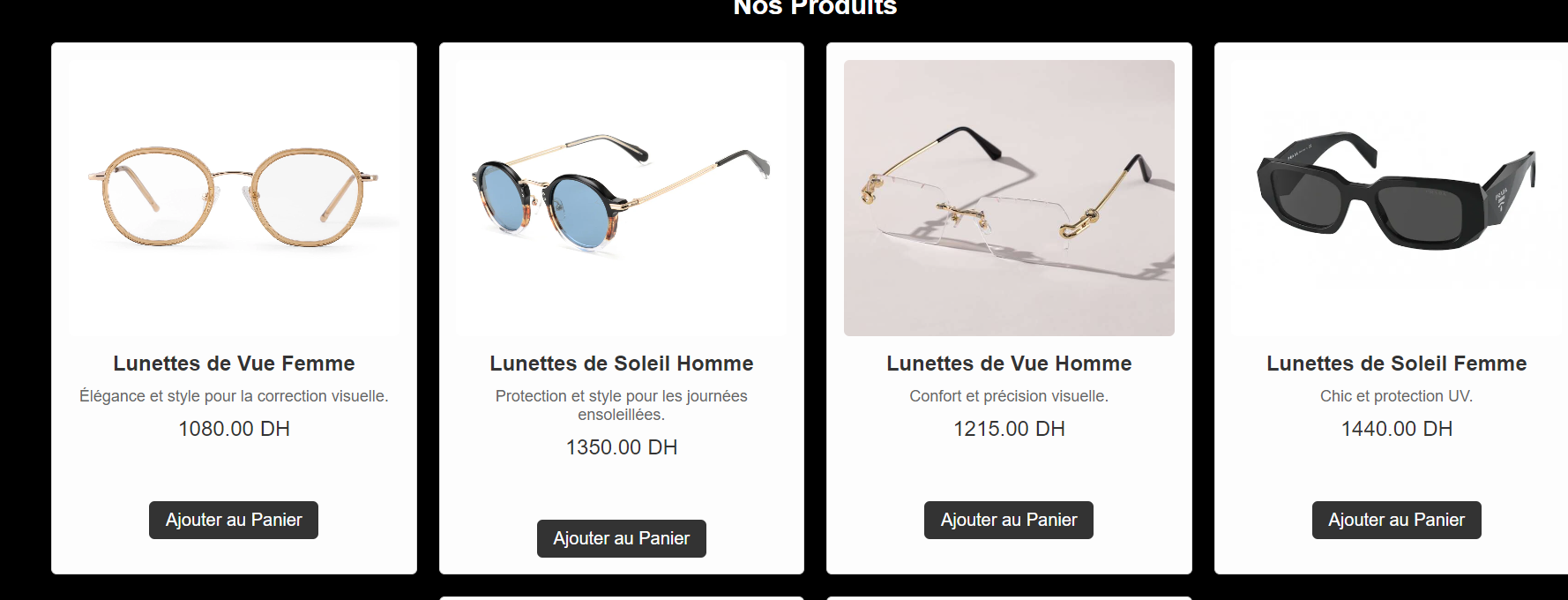
**La creation de compte:**

****

**Connexion:**

****

**Les produits apres la connexion avec la reduction:**

****

***Page panier:***

#### ****Objectif général du code:****

Le code fourni permet de gérer un panier d'achats en ligne pour un site de commerce électronique fictif nommé "Optique Saadi". Il affiche les articles du panier, permet à l'utilisateur de les supprimer et d'afficher le total de la commande. Si le panier est validé, un formulaire de paiement apparaît.

#### ****Structure du code:****

1. **HTML Structure**
   * **En-tête (<header>)** :
     + Contient le nom de l'entreprise ("Optique Saadi") et un menu de navigation avec des liens vers les pages d'accueil, des produits, le panier et la page de contact.
   * **Corps principal (<main>)** :
     + Affiche un titre indiquant "Votre Panier".
     + Un conteneur #cart-items qui est vide au départ mais qui contiendra la liste des articles du panier au fur et à mesure que l'utilisateur les ajoutera.
     + Un total qui calcule la somme des prix des articles.
     + Un bouton pour passer à la caisse, qui déclenche la fonction checkout() qui valide le panier et montre un formulaire de paiement.
   * **Formulaire de paiement (#payment-form)** :
     + Ce formulaire est masqué au départ et devient visible lorsque l'utilisateur valide son panier. Il permet de saisir les informations de paiement telles que le nom, l'email, l'adresse et le numéro de la carte bancaire.
2. **CSS (Styles)**
   * Les styles sont inclus dans le même fichier, et ils sont utilisés pour rendre la page plus esthétique et fonctionnelle.
   * Le fond de la page est d'un gris clair (#f8f9fa), avec des conteneurs blancs pour les articles du panier et le formulaire de paiement, créant un contraste agréable.
   * Des boutons sont stylisés avec des transitions pour les rendre plus interactifs.
   * Le design global suit une approche moderne et simple, avec une attention particulière aux éléments de navigation et à l'ergonomie du panier.
3. **JavaScript (Gestion du panier et interaction avec le localStorage)**
   * Le JavaScript est utilisé pour dynamiser la page en manipulant le panier et ses articles stockés dans le **localStorage** du navigateur.

**Voici un détail des fonctionnalités JavaScript :**

* + **Initialisation du panier (avec JSON et localStorage)** : Le code commence par récupérer les données du panier stockées dans **localStorage** sous forme de chaîne JSON :
  + const cartData = JSON.parse(localStorage.getItem('cart')) || [];

Si aucune donnée n'existe dans **localStorage**, le panier est initialisé comme un tableau vide ([]).

* + **Affichage des articles du panier (displayCartItems())** :
    - La fonction displayCartItems() est responsable de l'affichage dynamique des articles présents dans le panier. Elle crée un élément HTML pour chaque article, montre son nom et son prix, et permet de supprimer l'article via un bouton "Supprimer".
    - Le total est mis à jour en fonction des prix des articles ajoutés dans le panier.
    - Si le panier est vide, un message indiquant "Votre panier est vide" est affiché.
  + if (cartData.length === 0) {
  + cartItemsContainer.innerHTML = `<p class="empty-message">Votre panier est vide.</p>`;
  + totalContainer.innerHTML = '';
  + return;
  + }
  + **Suppression d'un article (removeFromCart(index))** :
    - La fonction removeFromCart() supprime un article du panier en utilisant son index dans le tableau **cartData**.
    - Après suppression, le panier est sauvegardé à nouveau dans **localStorage** :
  + localStorage.setItem('cart', JSON.stringify(cartData));

Cela permet de maintenir la persistance des données même après le rafraîchissement de la page.

* + **Validation du panier et affichage du formulaire de paiement (checkout())** :
    - Lorsque l'utilisateur clique sur "Passer à la Caisse", la fonction checkout() est appelée. Si le panier est vide, un message d'alerte apparaît.
    - Si le panier contient des articles, un message de remerciement est affiché, le panier est vidé de **localStorage**, et le formulaire de paiement devient visible.
  + localStorage.removeItem('cart');
  + displayCartItems();
  + paymentForm.style.display = 'block';

1. **Utilisation de JSON et localStorage** Le code repose fortement sur l'utilisation de **JSON** pour sérialiser et désérialiser les données du panier dans le **localStorage** :
   * Lorsqu'un article est ajouté ou supprimé, l'état du panier est mis à jour et converti en chaîne JSON avec JSON.stringify(cartData) avant d'être stocké dans **localStorage**.
   * Lorsque le panier est récupéré, il est reconverti en un objet JavaScript avec JSON.parse(localStorage.getItem('cart')).

Cela permet de rendre l'application fonctionnelle même si l'utilisateur quitte la page et y revient plus tard, car les données du panier sont conservées dans le navigateur.

#### ****Points importants:****

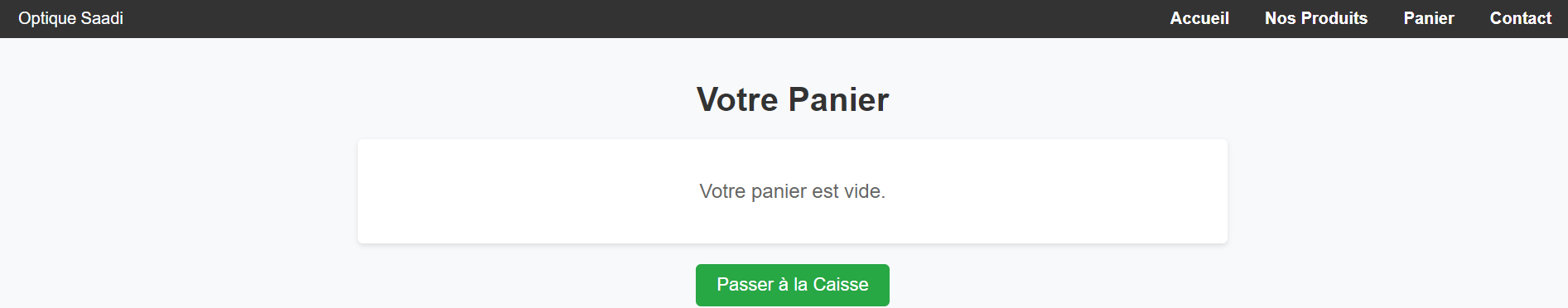
1. **Persistance des données** :
   * **localStorage** permet de stocker les données localement dans le navigateur de l'utilisateur sans nécessiter un serveur. Cela rend le panier persistant même après un rafraîchissement de la page ou une fermeture du navigateur.
2. **Interaction dynamique** :
   * Le panier est mis à jour dynamiquement en fonction des actions de l'utilisateur (ajout, suppression d'articles).
   * Les prix sont recalculés en temps réel et affichés à l'utilisateur.
3. **Formulaire de paiement** :
   * Bien que le formulaire de paiement soit montré à l'utilisateur, il n'est pas fonctionnel dans ce code. En pratique, il devrait être relié à une API de paiement pour traiter les informations bancaires de manière sécurisée.

#### ****Conclusion:****

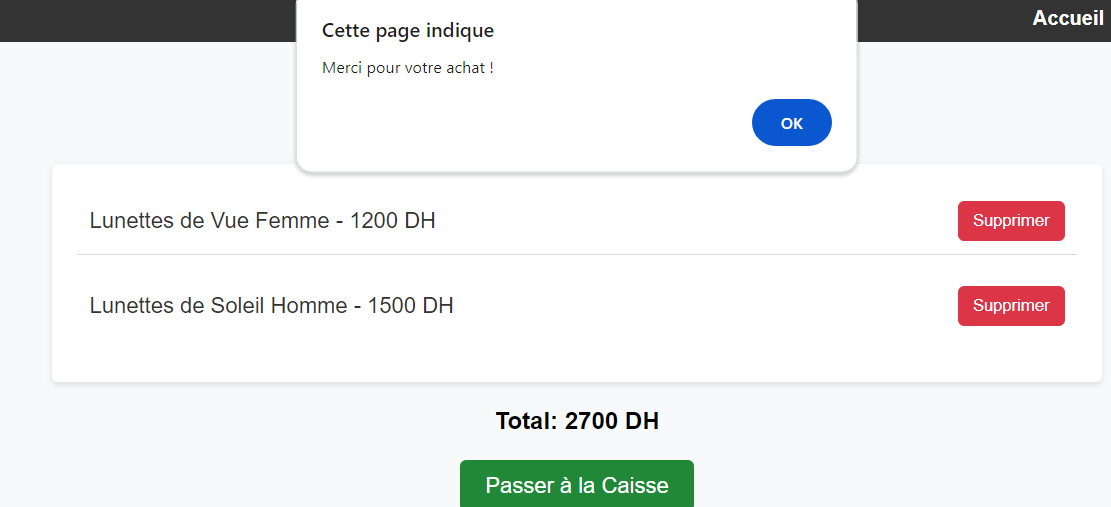
Le code HTML, CSS et JavaScript que vous avez fourni offre une solution simple mais fonctionnelle pour un panier d'achat en ligne. Il utilise efficacement **JSON** et **localStorage** pour gérer la persistance des articles dans le panier. Cependant, le formulaire de paiement nécessite une intégration avec un système de paiement réel pour le rendre complet et sécurisé.

**Resultat:**

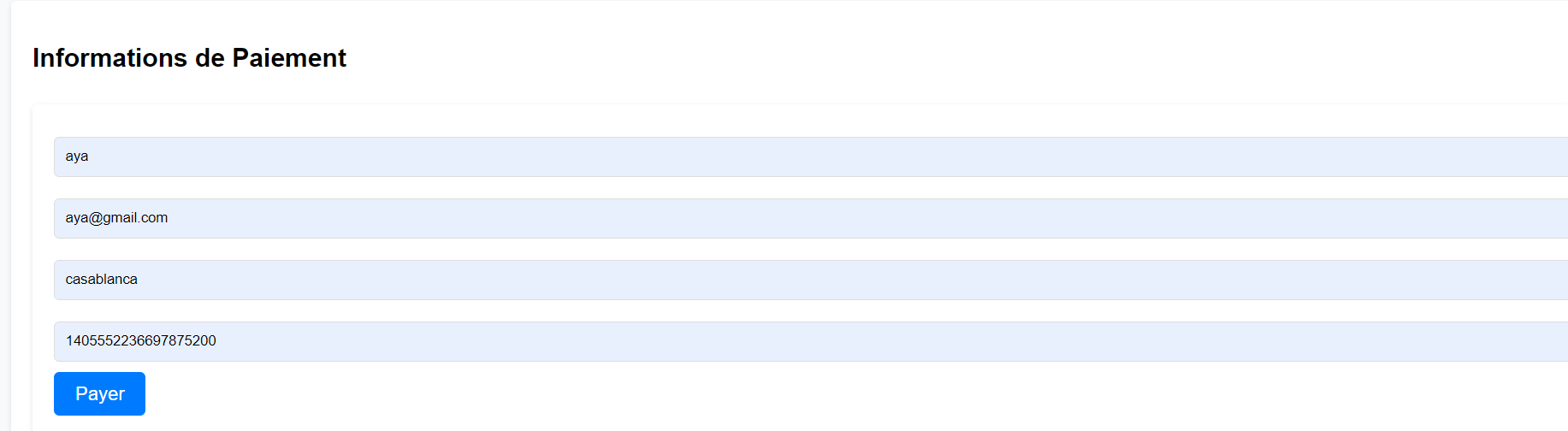
**Panier vide:**

****

**Les produits dans le panier:**

****

**Paiement:**

****

***Page admin:***

### ****1. Structure Générale du Code****

Le code contient une combinaison de HTML, CSS et JavaScript, créant une interface pour un panneau d'administration destiné à gérer les produits, les clients, les fournisseurs et le stock d'un magasin. La structure du code suit une organisation logique avec une séparation claire des préoccupations :

1. **HTML** : La structure de la page et les éléments du formulaire de connexion, de gestion des produits, des clients, des fournisseurs et du stock.
2. **CSS** : Le style de la page, y compris les mises en forme du formulaire, des boutons, des sections et des éléments interactifs.
3. **JavaScript** : La logique du côté client pour gérer les actions dynamiques comme l'ajout, la modification, la suppression d'éléments, ainsi que la gestion du stockage local

### ****2. HTML : Structure et Contenu****

#### ****a. Page de Connexion (Login)****

* **Structure** : La première section de la page consiste en un formulaire de connexion qui demande le nom d'utilisateur et le mot de passe.
* **Formulaire** : Deux champs de saisie (username, password) sont inclus pour la connexion, accompagnés d'un bouton de soumission. Un message d'erreur est affiché en cas de mauvaise identification.

<div class="login-container">

<h2>Se connecter</h2>

<form id="loginForm">

<input type="text" id="username" placeholder="Nom d'utilisateur" required>

<input type="password" id="password" placeholder="Mot de passe" required>

<button type="submit">Se connecter</button>

</form>

<div class="error-message" id="error-message"></div>

</div>

#### ****b. Section Admin (Dashboard)****

Cette section est cachée par défaut et devient visible après une connexion réussie. Elle contient plusieurs sous-sections pour gérer différents aspects de l'administration :

* **Statistiques** : Un graphique dynamique (créé avec Chart.js) montre des informations relatives aux produits en stock.
* **Ajout de Produits** : Un formulaire permettant d'ajouter des produits avec des champs comme le nom, le prix, les couleurs disponibles et une image.
* **Gestion des Clients, Fournisseurs et Stock** : Formulaires et listes pour ajouter, afficher et supprimer des clients, des fournisseurs et des produits en stock.

<section id="adminSection" style="display: none;">

<!-- Section Statistiques -->

<div class="stats-container">

<canvas id="statsChart" width="150" height="200"></canvas>

</div>

<!-- Formulaires pour gérer produits, clients, etc. -->

</section>

### ****3. CSS : Mise en Forme et Esthétique****

Le CSS est utilisé pour assurer la présentation des éléments, avec des couleurs et une mise en page responsive. Voici un résumé des styles appliqués :

#### ****a. Styles Globaux****

* **Polices et Couleurs** : La police Roboto est utilisée, et une palette de couleurs comprenant des tons de bleu et de blanc est appliquée pour une interface moderne et professionnelle.
* **Disposition** : Un fond dégradé est utilisé pour la page de connexion, tandis que chaque section du tableau de bord est séparée par des boîtes ombrées pour un effet de profondeur.

body {

font-family: 'Roboto', sans-serif;

background-color: #f7f7f7;

margin: 0;

padding: 0;

color: #333;

}

#### ****b. Formulaires et Boutons****

Les formulaires sont stylisés avec des bords arrondis et une transition douce sur les champs de saisie et les boutons. Le bouton de soumission change de couleur au survol, ce qui améliore l'interaction avec l'utilisateur.

button {

width: 100%;

padding: 14px;

background-color: #4d75e1;

color: white;

border: none;

font-size: 1.1rem;

cursor: pointer;

border-radius: 5px;

}

button:hover {

background-color: #365cb2;

}

#### ****c. Disposition Responsive****

Des requêtes média sont utilisées pour ajuster la mise en page sur des écrans plus petits (moins de 768px de large), garantissant que l'interface reste fluide et accessible sur mobile.

@media (max-width: 768px) {

section {

margin: 10px;

padding: 15px;

}

}

### ****4. JavaScript : Logique Dynamique****

Le JavaScript implémente les fonctionnalités essentielles du panneau d'administration, telles que l'ajout, la modification et la suppression de produits, de clients, de fournisseurs et de stock, ainsi que l'affichage dynamique des statistiques.

#### ****a. Connexion et Accès Admin****

Lorsqu'un utilisateur entre ses identifiants (nom d'utilisateur et mot de passe), le formulaire est validé. Si les informations sont correctes, la section admin devient visible. Sinon, un message d'erreur est affiché.

document.getElementById('loginForm').addEventListener('submit', (event) => {

event.preventDefault();

const username = document.getElementById('username').value;

const password = document.getElementById('password').value;

if (username === "admin" && password === "admin123") {

document.querySelector('.login-container').style.display = 'none';

document.getElementById('adminSection').style.display = 'block';

} else {

document.getElementById('error-message').textContent = "Identifiants incorrects.";

}

});

#### ****b. Gestion des Produits, Clients et Stock****

Les fonctions addProduct, addClient, addSupplier, et manageStock ajoutent des données aux tableaux respectifs. Ces données sont ensuite stockées dans le localStorage pour permettre une persistance entre les sessions. L'affichage des produits, des clients et des fournisseurs est mis à jour dynamiquement chaque fois qu'un élément est ajouté ou modifié.

Exemple pour l'ajout d'un produit :

function addProduct() {

const name = document.getElementById('productName').value;

const price = document.getElementById('productPrice').value;

const colors = document.getElementById('productColors').value.split(',');

const image = document.getElementById('productImage').files[0];

if (!name || !price || !colors.length || !image) {

alert('Veuillez remplir tous les champs');

return;

}

const reader = new FileReader();

reader.onloadend = () => {

const newProduct = { name, price, colors, image: reader.result };

products.push(newProduct);

localStorage.setItem('products', JSON.stringify(products));

renderProducts();

};

reader.readAsDataURL(image);

}

#### ****c. Statistiques avec Chart.js****

Le graphique de statistiques utilise la bibliothèque Chart.js pour afficher des informations sur le stock de produits. Les données sont extraites du localStorage, puis affichées dans un graphique en barres.

function renderStats() {

const ctx = document.getElementById('statsChart').getContext('2d');

new Chart(ctx, {

type: 'bar',

data: {

labels: Object.keys(stock), // Produits

datasets: [{

label: 'Quantité en stock',

data: Object.values(stock), // Quantités

backgroundColor: '#007bff',

borderColor: '#0052cc',

borderWidth: 1

}]

}

});

}

### ****5. Recommandations et Améliorations****

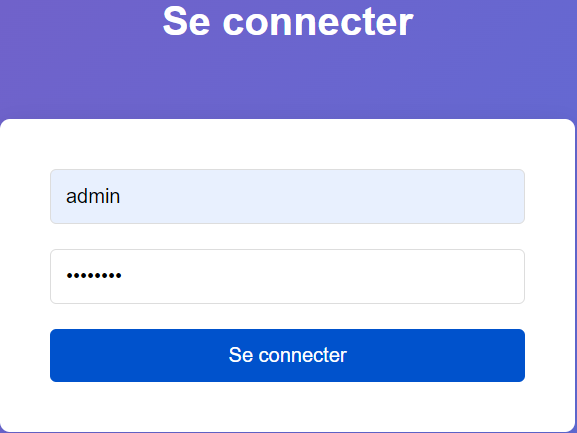
1. **Validation des Formulaires** : Bien que certaines vérifications de validité des champs soient présentes, des validations côté client plus robustes (par exemple, validation d'email pour les clients) pourraient être ajoutées pour améliorer la sécurité et la fiabilité des données.
2. **Sécurité** : L'authentification est gérée de manière basique avec un nom d'utilisateur et un mot de passe en dur dans le code. Il serait préférable d'implémenter un système d'authentification plus sécurisé côté serveur, comme un token JWT, pour renforcer la sécurité.
3. **Accessibilité** : Ajouter des éléments pour améliorer l'accessibilité, tels que des attributs aria et des descriptions supplémentaires pour les éléments visuels, améliorerait l'accessibilité pour les utilisateurs ayant des besoins spécifiques.
4. **Optimisation Mobile** : Bien que des requêtes médias soient utilisées, certains éléments comme les tableaux pourraient nécessiter un ajustement pour être plus conviviaux sur les petits écrans. Par exemple, les éléments de la liste pourraient être présentés sous forme de cartes sur mobile.
5. **Modifications dans la gestion des images** : Actuellement, les images des produits sont lues en tant que Data URL. Cela peut entraîner des problèmes de performance si un grand nombre d'images est chargé. Il peut être utile d'utiliser des URL d'images stockées sur un serveur.

### ****Conclusion****

Votre application de gestion du panneau d'administration pour Optique Saadi est bien structurée et offre une interface simple pour gérer les produits, les clients, les fournisseurs et le stock. Cependant, certaines améliorations peuvent être apportées en termes de sécurité, de validation des formulaires et d'optimisation pour les appareils mobiles.

**Resultat:**

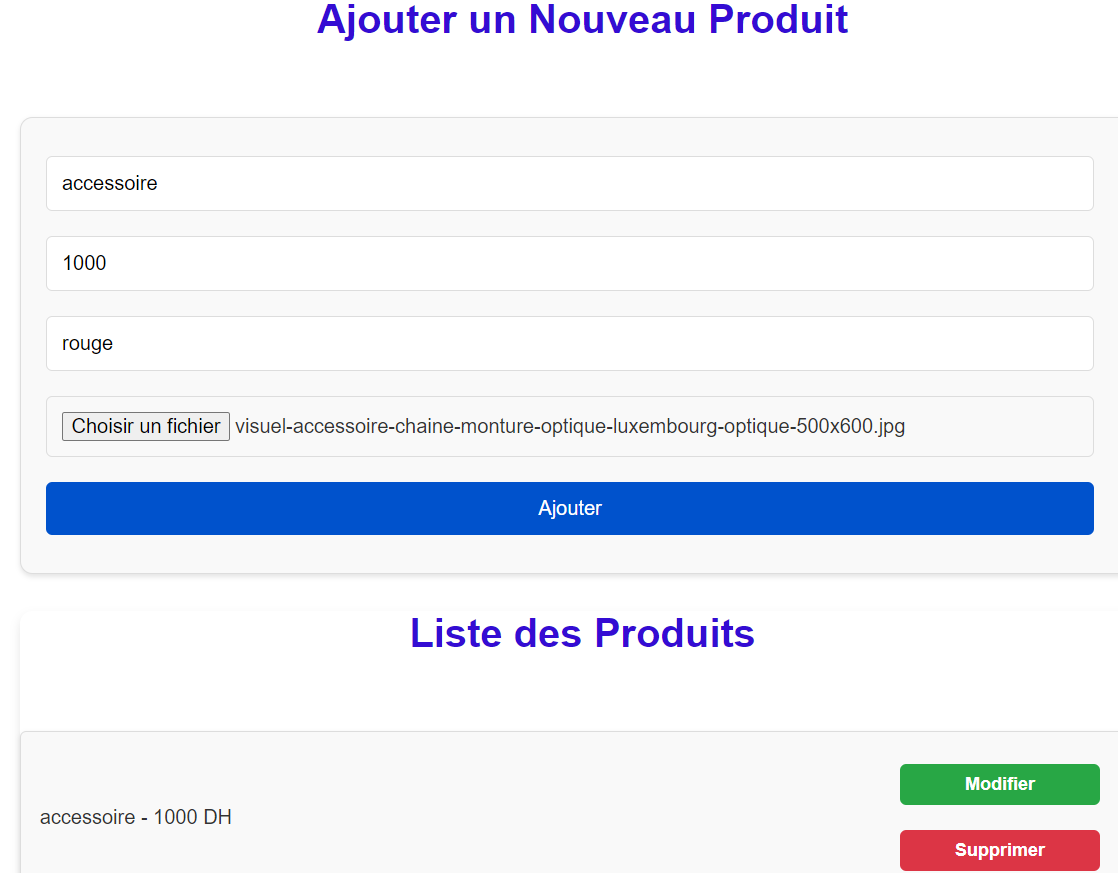
**Connexion:**

****

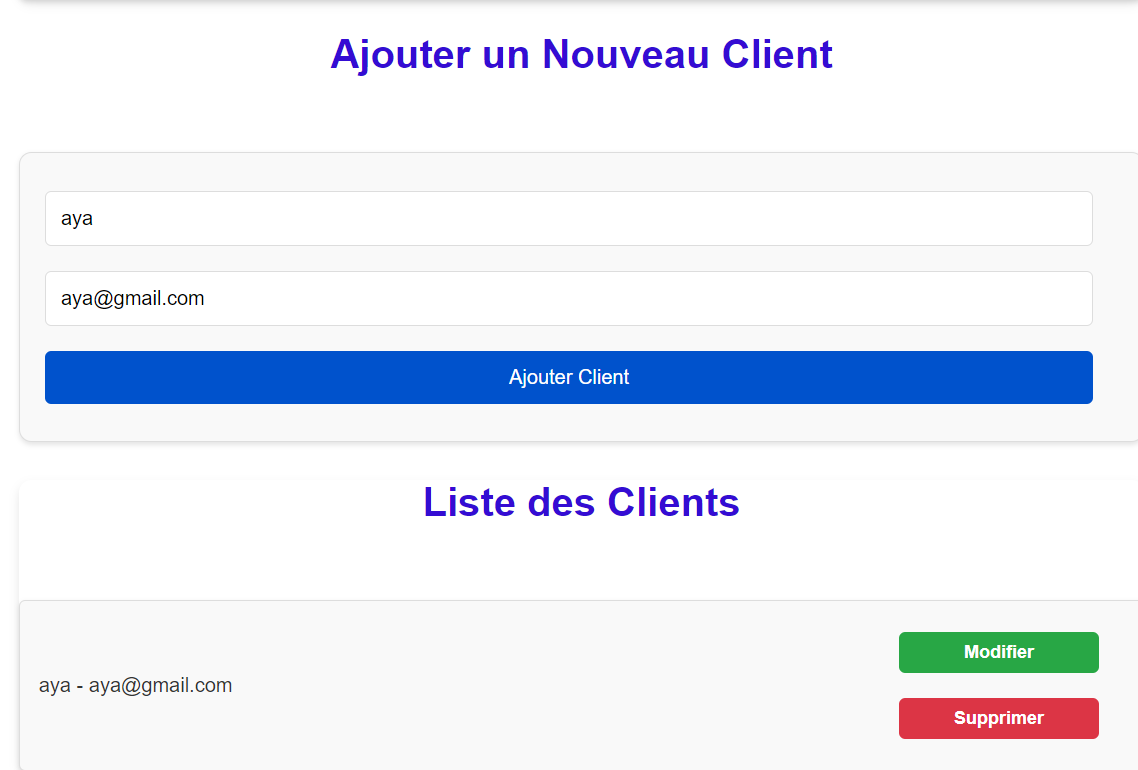
**Dashboard:**

****

**Ajout de produit:**

****

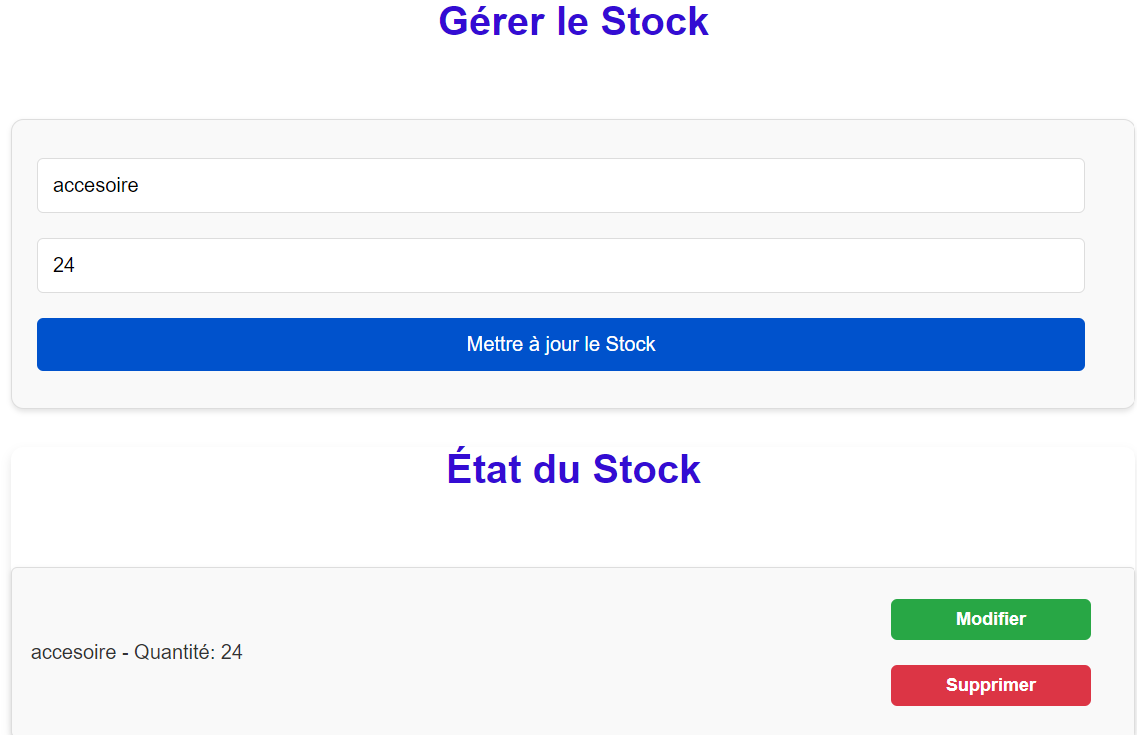
**Ajout un nouveau client :**

****

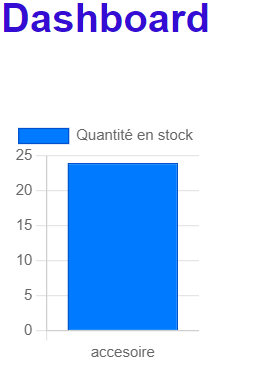
**Ajout de fournisseur:**

****

**Le stock:**

****

**Dashboard après l’ajout:**

****

**Bibiographie:**

[1] Wikipedia : Site officiel pour trouver toutes les définitions. (s. d.). Consulté le 9 janvier 2025, à partir de <https://www.wikipedia.org/>

[2] GitHub : Où le monde développe des logiciels. (s. d.). Consulté le 9 janvier 2025, à partir de <https://github.com/>

[3] Visual Studio : Site officiel pour télécharger et installer Visual Studio. (s. d.). Consulté le 9 janvier 2025, à partir de <https://visualstudio.microsoft.com/>

[4] OpenAI ChatGPT : Modèle de langage AI pour générer des réponses intelligentes. (s. d.). Consulté le 9 janvier 2025, à partir de <https://chat.openai.com/>