

FRUIT AND VEGETABLE MARKET SHOP

Auteur :

- Chaymae Belamkadem
- Ouahiba Yamlahi Alami
- Hiba Jadid



Sommaire

1	INTRODUCTION	-----	p.02
2	CHAPITRE I – PAGE D'ACCUEIL (HOME PAGE & FOOTER & HEADER & PAGE 404)	-----	p.03
3	CHAPITRE II : COMPTE (REGISTER & AUTHENTIFICATION & ACCOUNT INFO)	-----	p.08
4	CHAPITRE III: PAGE DE LISTAGE DES PRODUITS (PLP) & PDP & OTHER ITEMS	-----	p.11
5	CHAPITRE IV : PROCESSUS DE COMMANDE UTILISATEUR	-----	p.15
6	CHAPITRE V: PAGE "CONTACT US" - ANALYSE ET IMPLÉMENTATION	-----	p.19
7	CHAPITRE VI: GESTION DES ERREURS SUR LA PAGE "CONTACT US"	-----	p.23
8	CHAPITRE VII: PAGE DE CONFIRMATION – SUCCÈS DE L'ENVOI US"	-----	p.24
9	ANNEXE : UTILISATION DE MONGODB DANS LE PROJET	-----	p.25
10	CONCLUSION GÉNÉRALE		



Introduction

Ce projet est une application web e-commerce Fruit & Vegetable Shop , c'est une plateforme e-commerce moderne dédiée à la vente en ligne de fruits et légumes frais, naturels et biologiques. Il a été conçu pour offrir aux utilisateurs une expérience d'achat fluide, agréable et intuitive, tout en mettant en valeur la fraîcheur et la qualité des produits proposés.

Dès la page d'accueil, l'utilisateur est accueilli par un slider promotionnel mettant en avant les meilleures offres, comme des réductions allant jusqu'à 50 % sur la première commande. Le design est épuré et professionnel, avec des visuels de haute qualité qui inspirent confiance et soulignent l'aspect naturel des produits.

Objectif du site

L'objectif principal du site est de faciliter l'accès aux produits frais et sains pour les consommateurs, en leur offrant une interface conviviale et des offres attractives. Il répond également aux nouvelles habitudes de consommation, favorisant les circuits courts et les achats responsables.



Chapitre I – Page d'accueil

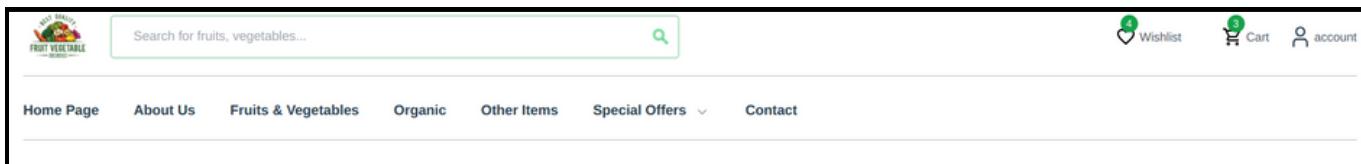
I- Objectif de la Home Page

La page d'accueil est conçue pour attirer l'attention des visiteurs dès leur première visite. Elle remplit plusieurs fonctions essentielles :

- Présenter l'identité et la mission du site.
- Mettre en avant les promotions et produits phares.
- Guider efficacement l'utilisateur vers l'action (achat, inscription, navigation).

II- Composition & Fonctionnalités

1. Barre de navigation (Header)



Située en haut de la page, elle contient :

- Barre de recherche pour trouver rapidement des produits par mot-clé (ex : "pomme", "carotte bio").
- Liens de menu vers les pages principales : Home Page, About Us, Fruits & Vegetables, Organic, Contact.
- Icônes d'accès rapide :
 - **Localisation**
 - **Wishlist**
 - **Panier d'achat (Cart)** avec le nombre d'articles en temps réel
 - **Connexion / Inscription** via le bouton "Account"

Avant toute chose, on a importé tous les éléments nécessaires à notre page :

- **React et useState** : pour gérer l'état si besoin

```
import React, {useState} from 'react';|
```

- **React-slick** : pour le carrousel

```
import Slider from "react-slick";|
```

- **slick-carousel** : on l'a utilisé pour afficher une série d'images, de textes ou de contenus dans un espace limité, généralement avec un effet de défilement automatique ou manuel. On l'a intégré sur notre page d'accueil pour présenter des promotions, des produits. Il est un diaporama interactif qui permet à l'utilisateur de faire défiler des éléments, souvent horizontalement, à l'aide de flèches ou d'indicateurs.

```
import "slick-carousel/slick/slick.css";
import "slick-carousel/slick/slick-theme.css";
```

- **Les images des slides** sont importées localement pour les afficher dans le carrousel. C'est important pour garantir la performance et éviter les appels externes.

```
import Slide1 from '../assets/images/home-slide1.png'
import Slide2 from '../assets/images/home-slide2.png'
import Slide3 from '../assets/images/slider3.png'
import Slide4 from '../assets/images/slider4.png'
import Slide5 from '../assets/images/slider5.png';
import Slide6 from '../assets/images/slider6.png';
import Bg from '../assets/images/bg.webp';
```

- Le formulaire de **newsletter** est importé depuis un composant externe (**NewsletterForm**) avec des hooks personnalisés pour gérer l'état de l'email et la soumission , Ce formulaire permet à l'utilisateur de s'inscrire à une newsletter quotidienne directement depuis le slide.
- Pour l'adaptation automatique à tous les types d'écrans on a utilisé du **Tailwind CSS** .

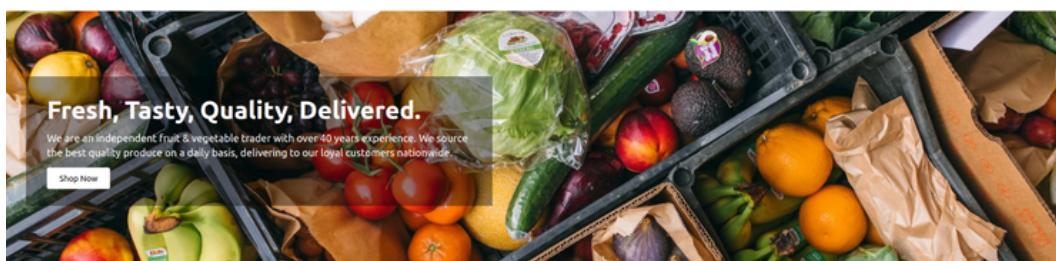
```
import { useNewsletter, NewsletterForm } from './Newsletter';
```

2. Contenu principal de la page d'accueil (Main):

La section centrale de la page d'accueil joue un rôle crucial dans l'expérience utilisateur. Elle est conçue pour capter l'attention, informer et inciter à l'action. Cette partie se compose de plusieurs blocs bien distincts, chacun ayant une fonction spécifique :

1.1- Bannière promotionnelle (Hero section)

En haut du contenu principal, une grande image colorée et appétissante présente des fruits et légumes frais. Un message fort ("Fresh, Tasty, Quality, Delivered.") met en avant la qualité et la fraîcheur des produits, accompagné d'un bouton "Shop Now" qui incite les utilisateurs à commencer leurs achats immédiatement.



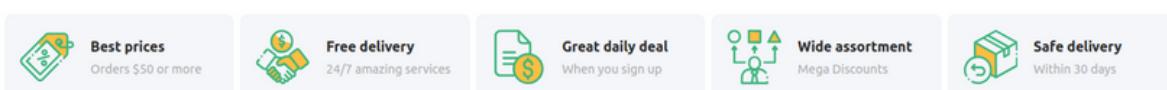
1.2- Avantages du service

Juste en dessous, une section met en avant les atouts majeurs du site :

- Meilleurs prix
- Livraison gratuite
- Offres quotidiennes
- Large assortiment
- Livraison sécurisée

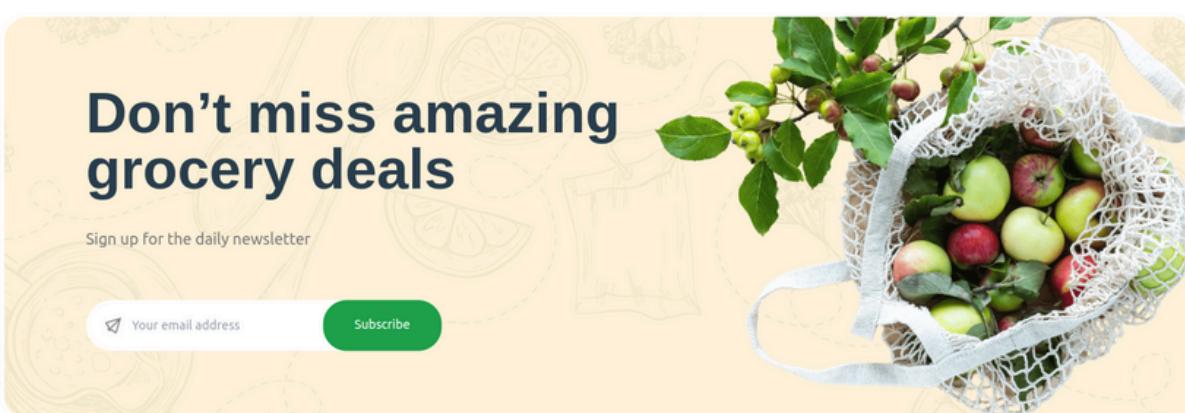
Cette partie rassure l'utilisateur et valorise le service proposé.

Discover the Benefits of Shopping With Us



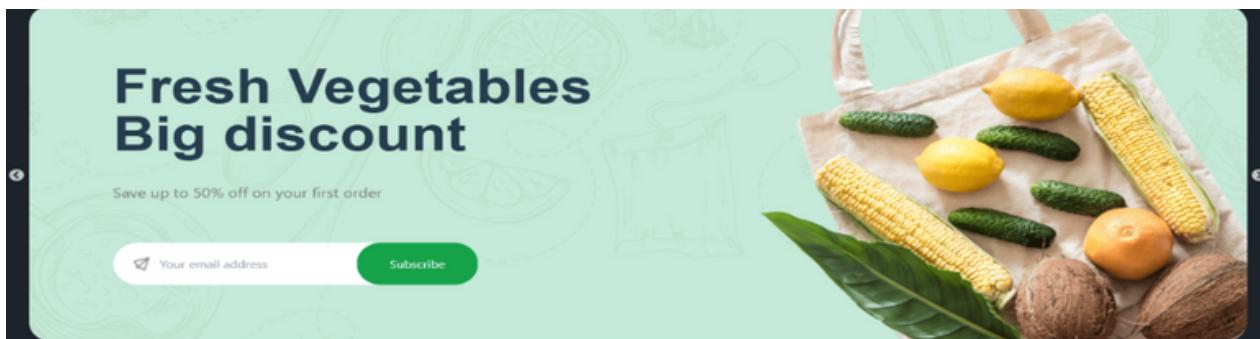
1.3- Inscription à la newsletter

Un encart clair et attrayant invite les utilisateurs à ne pas manquer les bonnes affaires en s'abonnant à la newsletter. Il comprend un champ pour l'email et un bouton "Subscribe", facilitant l'engagement.



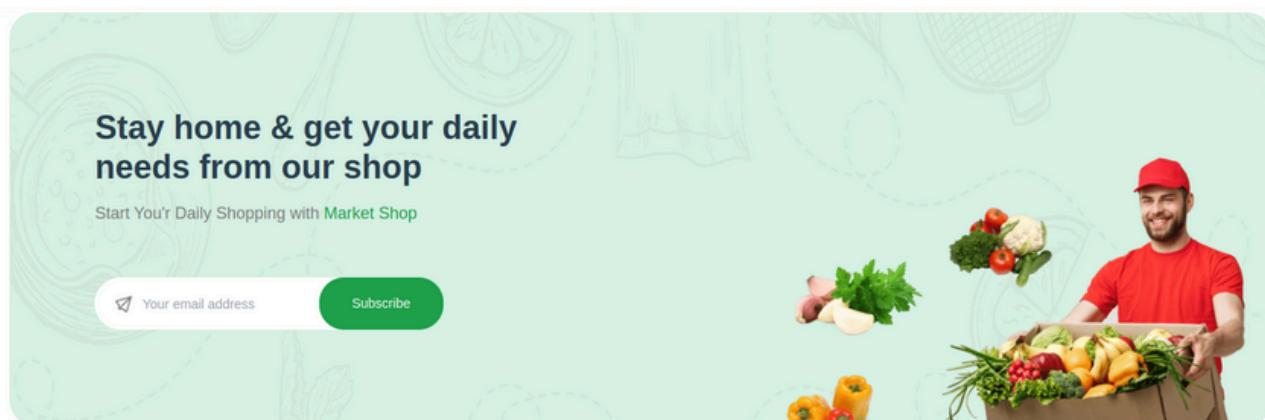
1.4- Carrousel de promotions

Une galerie dynamique de cartes présente des offres saisonnières ou thématiques (fruits frais, produits d'hiver, etc.), rendant le contenu vivant et interactif. Cela encourage la navigation et l'exploration des produits



1.5- Message de fidélisation

Enfin, une dernière section met en avant la simplicité de faire ses courses en ligne et renforce la relation client avec un message encourageant à rester chez soi tout en recevant ses produits frais à domicile.



3. Pied de page (Footer) :



FRESH FOOD

📍 Address: Fes Morocco
📞 Call Us: (+212) 600-000-000
✉️ Email: marketShop@gmail.com
🕒 Hours: 10:00 - 18:00, Mon - Sat

Categories

- Fruits & Vegetables
- Organic
- Special Offers

Customer Service

- Contact Us
- FAQ
- Returns & Refunds
- Shipping Information

Company

- About Us
- Privacy Policy
- Terms & Conditions
- Support Center

Download app on

From App Store or Google Play




Secured Payment Gateways






© 2025 FreshMarket. All rights reserved.

Follow Us





Up to 15% discount on your first subscribe

Le footer de la page d'accueil est conçu de manière structurée et informative. Il fournit aux utilisateurs toutes les informations essentielles pour renforcer leur confiance et faciliter la navigation ou le contact. Il se divise en plusieurs sections :

1. Logo et Informations de contact

Sur la gauche, on retrouve le logo coloré du site "Fresh Food", symbolisant fraîcheur et nature. Juste à côté, plusieurs informations pratiques sont listées :

- Adresse : Fès, Maroc
- Téléphone : numéro de contact local
- E-mail : pour les demandes générales
- Horaires : ouverture du lundi au samedi de 10h à 18h

2. Catégories de produits

Une colonne dédiée aux principales catégories de produits disponibles sur le site, notamment :

- Fruits & Légumes
- Produits bio (Organic)
- Offres spéciales
- Autres articles (Other Items)

3. Service client

Cette partie regroupe des liens utiles pour accompagner l'utilisateur :

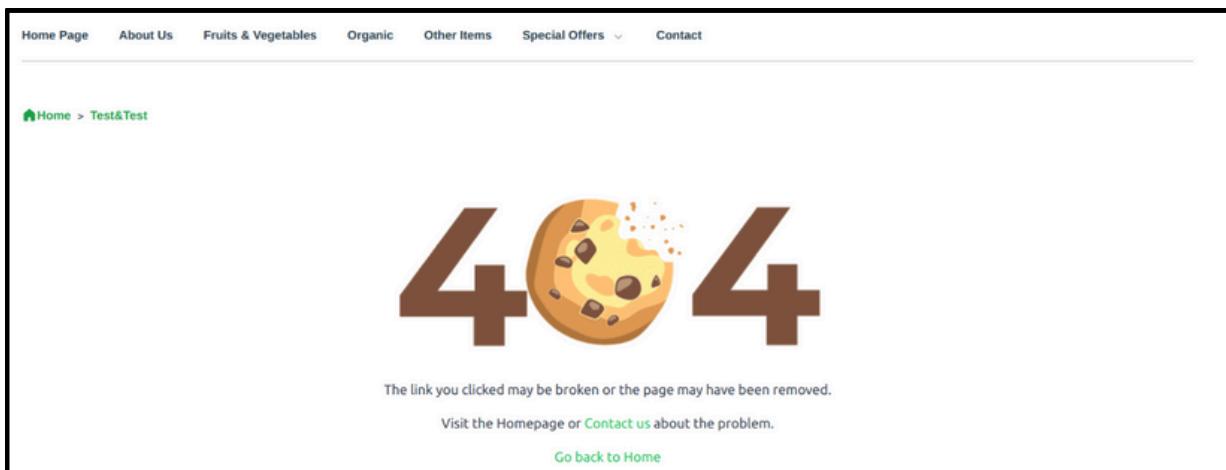
- Contact
- FAQ
- Retours et remboursements
- Informations sur la livraison

4. Informations sur l'entreprise

Des liens vers les pages importantes de l'entreprise, permettant de mieux connaître la politique de la boutique :

- À propos
- Politique de confidentialité
- Conditions générales
- Centre de support

4. Gestion des erreurs – Page 404



L'expérience utilisateur est aussi fortement influencée par la manière dont le site gère les erreurs. Ainsi, une page personnalisée d'erreur 404 a été intégrée afin d'informer l'utilisateur lorsqu'il tente d'accéder à une ressource inexistante ou supprimée.

Fonctionnalité :

- Affichage d'un message clair : « The link you clicked may be broken or the page may have been removed. »
- Icône graphique originale représentant le chiffre **404** avec un cookie croqué au centre pour ajouter une touche ludique.
- Liens de navigation rapides :
 - **Retour à la page d'accueil**
 - **Contactez-nous pour signaler le problème.**

Chapitre II : Compte

I- Page d'inscription - Register

The screenshot shows a registration form titled "Create a user account". It has two main sections: "Personal information" and "Login details". The "Personal information" section contains fields for "First Name *" and "Last Name *". The "Login details" section contains fields for "Password *" and "Confirm Password *". A green "Register" button is at the bottom. The URL in the browser bar is "Home > Register".

L'inscription permet à un nouvel utilisateur de créer un compte en saisissant ses informations personnelles (prénom, nom, email, mot de passe). Une fois le formulaire soumis, les données sont envoyées à une API sécurisée pour créer un nouvel utilisateur dans la base de données.

L'interface utilisateur affiche :

- Un titre clair : **Create a user account**
- Deux sections principales :
 - **Personal information** : Prénom, nom, email
 - **Login details** : Mot de passe et confirmation
- Un bouton vert **Register** pour soumettre le formulaire
- Un message de succès ou d'erreur affiché dynamiquement après soumission

Technologies utilisées :

- **useState** : Gestion des champs du formulaire et des états de chargement/erreur
- **useEffect** : Gestion d'effets secondaires (ex : minuterie pour cacher un message)
- **axios** : Requêtes HTTP via une instance préconfigurée api (axiosConfig.js)*
- **Router React** : Navigation après enregistrement (navigate("/login"))
- **useNavigate** : Utilisé pour rediriger l'utilisateur après une inscription réussie.

```
import React, { useEffect, useState } from 'react';
import { useNavigate } from 'react-router-dom';
import api from "../axiosConfig";
```

Stockent les données saisies dans le formulaire :

- **loading** : état booléen pour gérer le bouton pendant la requête.
- **status** : contient un message de feedback (succès ou erreur).

```
const Register = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [loading, setLoading] = useState(false);
  const [status, setStatus] = useState(null);
  const navigate = useNavigate();
```

- **handleRegister** : Empêche le rechargeement de la page et Lance l'état de chargement. rechargeement de la page et Lance l'état de chargement.

```
const handleRegister = async (e) => {
  e.preventDefault();
  setLoading(true);
  setStatus(null);
```

- **Appelle l'API** de création de compte avec les données saisies via axios.

```
try {
  const endpoint = "/auth/register";
  const res = await api.post(endpoint, { email, password, firstName, lastName });
```

- Affiche un message de succès, puis redirige vers la page de connexion après **1 seconde**.

```
} catch (err) {
  setStatus({
    type: "error",
    message: err.response?.data?.message || "Something went wrong. Please try again.",
  });
}
```

- En cas d'échec, affiche un message d'erreur dynamique.

II- Page de Connexion – Authentification des utilisateurs :

La page de connexion permet aux utilisateurs de s'authentifier pour accéder à des fonctionnalités personnalisées telles que l'accès à leur compte, le suivi de commandes, la gestion de la wishlist ou encore la consultation de leur historique d'achats.

Présentation de l'interface

La page est divisée en deux sections principales :

- **Création de compte (gauche)** : Elle invite les nouveaux utilisateurs à créer un compte pour bénéficier de services personnalisés.
- **Connexion client (droite)** : Elle contient un formulaire avec les champs « **Email Address** » et « **Password** », ainsi qu'un bouton **Log in**.

Home Page About Us Fruits & Vegetables Organic Other Items Special Offers ▾ Contact

 Home > Login

Log in or create a user account

New account

If you create a user account in our store, you will be guided through the ordering process more quickly, can save multiple shipping addresses, track your previous order history and much more.

[Create an account](#)

Customer Login

If you have a user account with us, please log in.

Email Address *

Password *

[Log in](#)

Un système de feedback utilisateur pourrait être ajouté (ex. message d'erreur ou de succès).

- **useState** : contrôlé Les champs email et password .
- **axios** : Appel API sécurisé via une instance configurée (api.post).

```
try {
    const endpoint = "/auth/login";
    const res = await api.post(endpoint, { email, password });
}
```

- **token** : Le token reçu est stocké dans localStorage pour la persistance de session

```
setToken(res.data.token);
localStorage.setItem("token", res.data.token);
console.log("success")
```

- **account-info** : navigation après une connexion réussie, redirection automatique vers account-info /.

```
setTimeout(() => navigate("/account-info"), 1000);
```

III- Page de profil – Account Information :

La page Account Info sert de tableau de bord personnel à l'utilisateur connecté. Elle permet d'afficher ses informations personnelles, et potentiellement d'accéder à des sections comme ses commandes, ses paramètres ou encore sa déconnexion.

La page est organisée en trois parties principales :

- **Sidebar latérale** : Affichée à gauche, elle contient une navigation verticale avec un design en accordéon.
- **Contenu dynamique** : Affiche les informations de l'utilisateur récupérées depuis l'API.
- **Un bouton de déconnexion (Logout)** est aussi présent pour permettre à l'utilisateur de fermer sa session de manière sécurisée.

Détail des mécanismes techniques

- **useNavigate** : Hook fourni par react-router-dom pour naviguer entre les routes (ex : rediriger vers /login).
- **FiLogOut** : Icône de déconnexion importée depuis `react-icons` pour styliser un bouton de déconnexion visuellement.


```
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import { FiLogOut } from 'react-icons/fi';
```

Chapitre III : Page de listage des produits (PLP) & PDP & Other Items

The screenshot shows a product listing page for 'Fruits & Vegetables'. At the top, there's a search bar with placeholder text 'Search for fruits, vegetables...', a magnifying glass icon, and user icons for Wishlist, Cart, and account. Below the search bar is a navigation menu with links: Home Page, About Us, Fruits & Vegetables (which is the active category), Organic, Other Items, Special Offers, and Contact.

Below the navigation, a breadcrumb trail shows 'Home > Fruits&Vegetables'. The main title 'Fruits And Vegetables' is displayed in green. To the left, a sidebar titled 'Filters' contains three sections: 'Price Range' (with input fields for 0 and 3000, and a green 'Apply Price Filter' button), 'Product Type' (with a dropdown set to 'All' and a green 'Apply Type Filter' button), and 'Sort By' (with options 'Price: Low to High' and 'Price: High to Low'). A red 'Reset Filters' button is at the bottom of the sidebar.

The main content area displays a grid of 20 product cards, each featuring an image, the product name, a brief description, price, and 'Add to Cart' and 'Remove from Wishlist' buttons.

Product	Description	Price	Action Buttons
Apple	Fresh and crispy	150 dhs/Kg	Add to Cart, Remove from Wishlist
Banana	Rich in potassium	75 dhs/Kg	Add to Cart, Remove from Wishlist
Orange	Packed with vitamin C	200 dhs/Kg	Add to Cart, Remove from Wishlist
Carrot	Healthy and crunchy	100 dhs/Kg	Add to Cart, Add to Wishlist
Broccoli	Nutrient-rich greens	175 dhs/Kg	Add to Cart, Add to Wishlist
Grapes	Sweet and juicy	250 dhs/Kg	Add to Cart, Add to Wishlist
Strawberry	Delicious red berries	300 dhs/Kg	Add to Cart, Remove from Wishlist
Lettuce	Crisp and fresh	120 dhs/Kg	Add to Cart, Add to Wishlist
Tomato	Versatile and flavorful	180 dhs/Kg	Add to Cart, Add to Wishlist
Cucumber	Cool and hydrating	130 dhs/Kg	Add to Cart, Add to Wishlist
Mango	Sweet and tropical	220 dhs/Kg	Add to Cart, Add to Wishlist
Pineapple	Tropical and tangy	280 dhs/Kg	Add to Cart, Add to Wishlist
Peach	Soft and juicy	190 dhs/Kg	Add to Cart, Add to Wishlist
Watermelon	Refreshing summer fruit	300 dhs/Kg	Add to Cart, Add to Wishlist
Cherry	Small and sweet	260 dhs/Kg	Add to Cart, Add to Wishlist
Spinach	Leafy and nutritious	90 dhs/Kg	Add to Cart, Add to Wishlist

I- Page de listage des produits (PLP)

La page Fruits & Vegetables constitue une partie essentielle du site e-commerce. Elle offre aux utilisateurs une expérience de navigation claire, fluide et interactive pour découvrir, filtrer et sélectionner des produits frais en fonction de leurs besoins. Elle combine à la fois des éléments fonctionnels (filtres, tri, wishlist) et esthétiques (interface moderne, couleurs attrayantes, composants dynamiques).

Structure et composants de la page

La page est divisée en deux grandes sections principales :

1. Barre latérale de filtres (Sidebar)

Située à gauche, cette barre permet aux utilisateurs de personnaliser leur recherche grâce à :

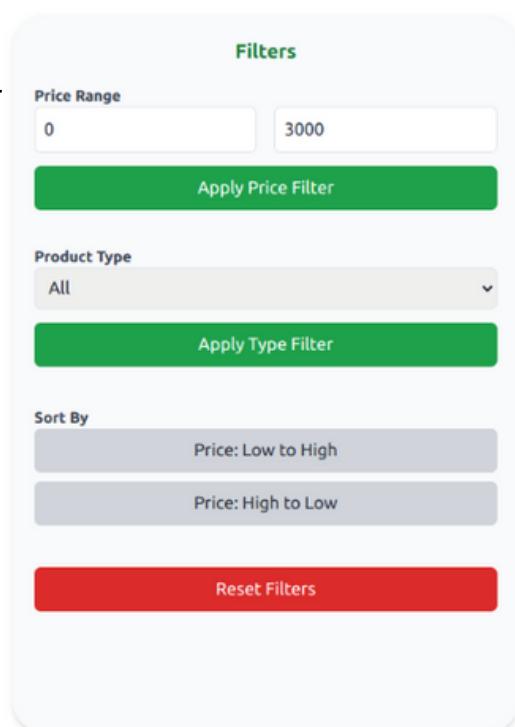
- Filtrage par prix:** Deux champs permettent d'indiquer un intervalle de prix minimum et maximum, puis d'appliquer le filtre.
- Filtrage par type de produit:** Un menu déroulant permet de choisir entre "Fruit", "Vegetable" ou autres catégories définies.
- useState :** pour suivre les filtres (prix, type) et les produits triés.
- useEffect :** pour actualiser la liste triée si les données changent.

```
import React, { useContext, useEffect, useState } from 'react';
import ProductItem from './ProductItem';
import { itemContext } from '../context/ItemContext';
import FilterSection from './FilterSection';
```

- Tri des produits :** Deux boutons désactivés pour l'instant sont prévus pour trier les produits du moins cher au plus cher, ou inversement.

```
const [sortedProducts, setSortedProducts] = useState([...products]);
const [minPrice, setMinPrice] = useState(0);
const [maxPrice, setMaxPrice] = useState(3000);
```

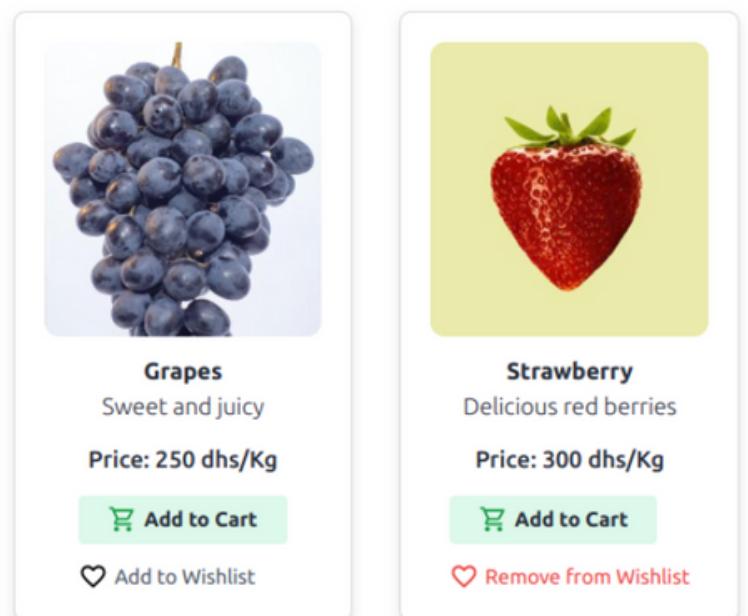
- Bouton de réinitialisation :** Permet de supprimer tous les filtres et réafficher l'ensemble des produits.



2. Liste des produits

À droite, les produits sont affichés sous forme de cartes bien structurées contenant :

- Une image du produit**
- Le nom**
- Une courte description**
- Le prix (exprimé en Dhs/Kg)**
- Deux boutons d'action :**
 - Add to Cart
 - Add to Wishlist



II- Product Detail Page (PDP)

The screenshot shows a Product Detail Page (PDP) for an Orange. At the top, there's a navigation bar with links to Home Page, About Us, Fruits & Vegetables, Organic, Other menu, Special Offers, and Contact. Below the navigation, the breadcrumb navigation shows Home > Product > 67c75710160dbanan7db6eb1.

Product Details:

- Image:** A large image of an orange, with one slice cut open to show the segments.
- Name:** Orange
- Rating:** ★★★☆☆ (26 reviews)
- Price:** 20 dhs (10% off)
- Description:** Packed with vitamin C. Lorem ipsum dolor sit amet consectetur adipisicing elit...
- Quantity:** 1
- Add to Cart:** A green button.
- Wishlist:** A button with a counter showing 2.

Similar Products:

- Carrot:** 8 dhs
- Broccoli:** 25 dhs
- Apple:** 15 dhs

Categories:

- Fruits & Vegetables (12)
- Dairy Products (8)
- Bakery (6)
- Meat & Fish (10)
- Drinks (15)
- Organic Products (7)

New Products:

- Fresh Organic Pineapple:** \$2.99
- Homemade Whole Bread:** \$3.00
- Natural Farm Yogurt:** \$1.20

Leave a Review:

Your Rating:

Your Comment:

Share your thoughts about this product...

Submit

Cette page produit propose une interface claire et intuitive, avec les éléments suivants :

- **Section principale du produit :**
- **Affiche l'image du produit** (orange), le nom, le prix (avec réduction), les avis des utilisateurs, la disponibilité en stock, et des détails comme le type, la méthode de conservation et la ferme d'origine.
- **Boutons d'action :**
- Permet à l'utilisateur de choisir la quantité, d'ajouter le produit au panier ou à la liste de souhaits.
- **Carrousel de produits similaires :**
- Suggère d'autres produits (carotte, brocoli, pomme) pour favoriser les ventes croisées.
- **Barre latérale (Sidebar) :**
 - Liste des catégories avec le nombre de produits.
 - Section Nouveaux Produits présentant les derniers ajouts avec leurs prix.
- **Zone d'évaluation :**
 - Les utilisateurs peuvent laisser une note et un commentaire sur le produit.

III- La Page Other Items :

Item	Price	Quantity	Action
Black Bomber Cheese	199.00 dhs	- 1 +	<button>Add to Cart</button>
Double XI eggs (West Mids & Shropshire Only)	5.00 dhs	- 1 +	<button>Add to Cart</button>
Spicy fig & plum Chutney	99.00 dhs	- 1 +	<button>Add to Cart</button>
Miller's Damsel Buttermilk Wafers	125.00 dhs	- 1 +	<button>Add to Cart</button>

La page other items est dédiée à l'affichage de produits complémentaires dans le cadre d'un site e-commerce de fruits et légumes. Ces produits peuvent inclure des articles comme des jus de fruits, paniers bio, ou accessoires liés à la cuisine. Cette page joue un rôle essentiel dans l'expérience utilisateur en proposant des achats additionnels et en enrichissant l'offre du site.

Fonctionnalités principales

- Visualisation de chaque produit** avec son image, son nom et son prix.
- Contrôle de quantité** par produit avec des boutons + / -.
- Ajout au panier** avec prise en compte de la quantité choisie.
- Redirection vers la page** de détails lorsqu'un produit est cliqué.

Technologies et logique utilisées :

Outil / Librairie	Utilisation principale
React	Création de composants fonctionnels
useContext	Partage du panier via <code>itemContext</code>
useState	Suivi des quantités d'articles par produit
React Router	Navigation vers la page de détails
Tailwind CSS	Mise en page réactive et design des composants

Chapitre IV : Processus de Commande Utilisateur

Le processus de commande sur notre site e-commerce suit une logique simple, fluide et intuitive. Il se compose de trois grandes étapes :

1. Ajout au Panier

Objectif :

Permettre aux utilisateurs d'ajouter des produits qu'ils souhaitent acheter.

Fonctionnement :

- Depuis la page catalogue, l'utilisateur clique sur "**Ajouter au panier**".
- L'action déclenche la fonction `addToCart(product)` du contexte global (`itemContext`).
- Si le produit existe déjà : sa quantité est incrémentée.
- Sinon : il est ajouté avec une quantité par défaut de 1.
- Le panier est mis à jour en temps réel.

1-1. Ajouter au panier (Add to Cart) :

Product	Unit Price	Quantity	Subtotal	Actions
Banana	75 dhs	- 1 +	75 dhs	Remove
Strawberry	300 dhs	- 1 +	300 dhs	Remove
Cherry	260 dhs	- 2 +	520 dhs	Remove

Cart Summary

Subtotal: 895 dhs

Estimate Shipping and Tax
Shipping and additional costs are calculated based on values you have entered.

Choose a country:

Enter your zip/postal code:

Total to Pay: 885 dhs

[Proceed To CheckOut](#)

Apply Coupon

Using A Promo Code?

[Apply Coupon](#)

```
const handleAddToCart = (product) => {
  addCart(product)
}
```

Technologies et logique utilisées :

Élément	Technologie / Outil	Rôle
React Context API	itemContext	Stockage global du panier
useContext + useState	React Hooks	Accès et mise à jour du panier
Fonction addToCart(product)	Fonction personnalisée	Ajoute un produit au panier
Stockage temporaire	State local ou localStorage (optionnel)	Persistance sur la session

Fonctionnement :

- L'utilisateur clique sur le bouton “Ajouter au panier” sur un produit affiché.
- Le composant ProductItem appelle la fonction addToCart(product) fournie par itemContext.
- La fonction vérifie si le produit est déjà présent dans le panier :
 - Si oui : elle incrémente la quantité.
 - Si non : elle l'ajoute avec une quantité de 1.

Le panier est mis à jour dynamiquement dans le contexte partagé.

1-2. Coupons de Réduction et Taxes

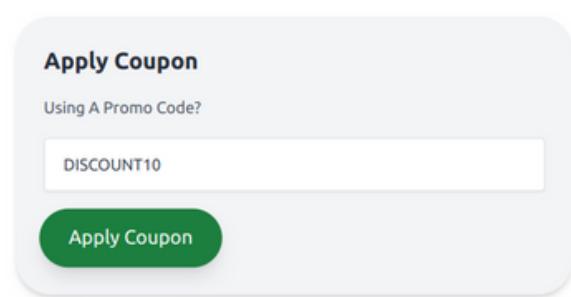
Pendant le processus de finalisation de commande, l'utilisateur peut également :

Application de Code Promo

L'utilisateur peut saisir un code promotionnel, par exemple DISCOUNT10, dans un champ prévu à cet effet avant de valider sa commande.

Fonctionnalité :

- Vérifie si le code est valide (par exemple : DISCOUNT10 accorde -10%).
- Applique la réduction sur le **total** ou **sous-total**.
- Actualise automatiquement le montant affiché dans le résumé du panier.



Visualiser les taxes ajoutées selon la localisation :

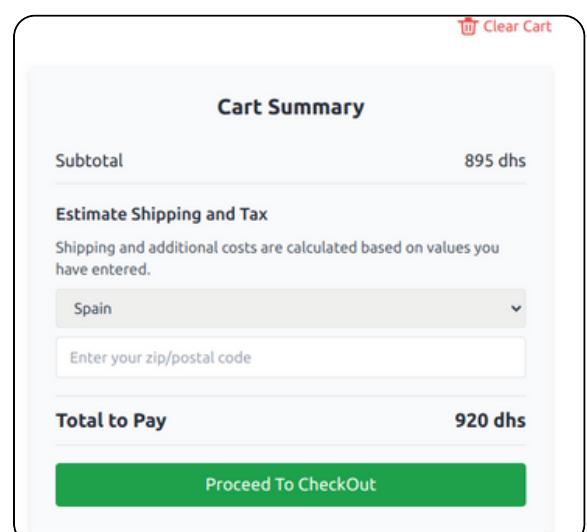
Afficher de façon transparente les taxes ajoutées à la commande en fonction de la localisation de l'utilisateur.

Fonctionnement :

- Les taxes sont calculées dynamiquement après que l'utilisateur remplit le formulaire d'adresse.
- Une règle de base est appliquée (par exemple : **TVA 30% si pays = France**).
- Le montant total affiché dans le panier est mis à jour automatiquement.

```
let newShippingCost = 0;

switch (country) {
  case "Morocco":
    newShippingCost = 20; // Exemple : frais de port pour le Maroc
    break;
  case "France":
    newShippingCost = 30; // Exemple : frais de port pour la France
    break;
  case "Spain":
    newShippingCost = 25; // Exemple : frais de port pour l'Espagne
    break;
}
```



1-3. Checkout – Finalisation de la Commande

The screenshot shows a checkout page with the following sections:

- Billing Address:** Fields for First Name, Last Name, Username, Email, Shipping Address, and Address 2 (Optional).
- Shipping Address:** Fields for Country (Germany), State (Berlin (Germany)), and Zip.
- Payment:** A radio button selected for Credit Card, and fields for Name on Card and Credit Card Number.
- Cart Summary:** Subtotal 650 dhs, Estimate Shipping and Tax (disabled), Total to Pay 650 dhs, and a Cart Edit button.
- Place Order:** A large green button at the bottom.

L'utilisateur est dirigé vers une page "Checkout" où il remplit :

- Nom, prénom, email
- Adresse de livraison (et secondaire facultative)
- Pays, état, code postal
- Mode de paiement (pré-rempli par défaut)

Technologies :

- React + useState : pour gérer tous les champs de saisie.
- useNavigate (React Router) : pour rediriger après validation.
- API Express côté serveur pour recevoir les données (/api/checkout).

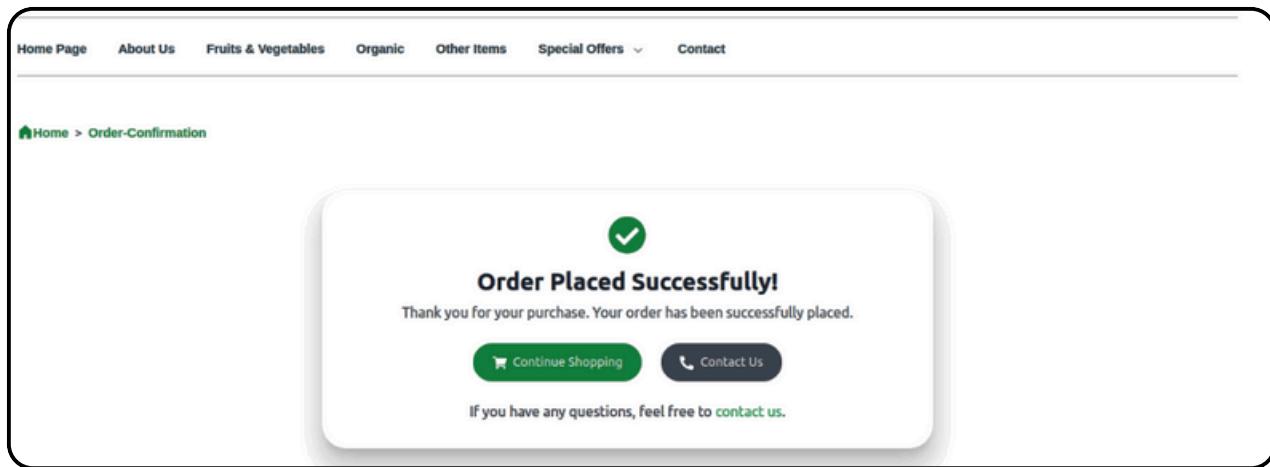
```
import React, { useState } from 'react';
import { Link, useLocation, useNavigate } from 'react-router-dom';

const Checkout = () => {
  const location = useLocation();
  const { totalPrice, totalWithShipping } = location.state || { totalPrice: 0, totalWithShipping: 0 };

  const [customerEmail, setCustomerEmail] = useState('');
  const [firstName, setFirstName] = useState('');
  const [lastName, setLastName] = useState('');
  const [username, setUsername] = useState('');
  const [shippingAddress, setShippingAddress] = useState('');
  const [address2, setAddress2] = useState('');
  const [country, setCountry] = useState('');
  const [state, setState] = useState('');
  const [zip, setZip] = useState('');
  const [paymentMethod, setPaymentMethod] = useState('credit');
  const [ccName, setCcName] = useState('');
  const [ccNumber, setCcNumber] = useState('');

  const navigate = useNavigate();
```

1-4. Confirmation de Commande (Order Confirmation)



Après la validation du panier et la finalisation du paiement, l'utilisateur est automatiquement redirigé vers une page de confirmation. Cette page assure une communication claire et rassurante en affichant les éléments suivants :

- ✓ Message de succès : "Order Placed Successfully!" accompagné d'une icône visuelle pour renforcer le retour positif.
- 🛒 Bouton vert "Continue Shopping" : permet à l'utilisateur de revenir facilement au catalogue et de poursuivre ses achats.
- 📞 Bouton "Contact Us" : pour offrir un accès rapide au service client en cas de question ou de problème.
- 💬 Message personnalisé : "Thank you for your purchase..." pour une touche plus humaine et engageante.

👉 Pour mettre en œuvre cette interface de manière fluide, moderne et réactive, plusieurs technologies et principes d'expérience utilisateur (UX) ont été mobilisés.

Cette page améliore l'expérience utilisateur grâce à :

- ⌚ Une navigation fluide et intuitive, facilitant le parcours post-achat
- 🛍️ Un affichage clair des éléments de commande, avec des actions accessibles rapidement
- 🎯 Des boutons visibles et efficaces, permettant de continuer les achats ou contacter le support
- 🖼️ Une présentation soignée et engageante, intégrant un carrousel d'éléments visuels et un effet de zoom pour renforcer l'impact visuel (si tu veux garder ça selon le design)

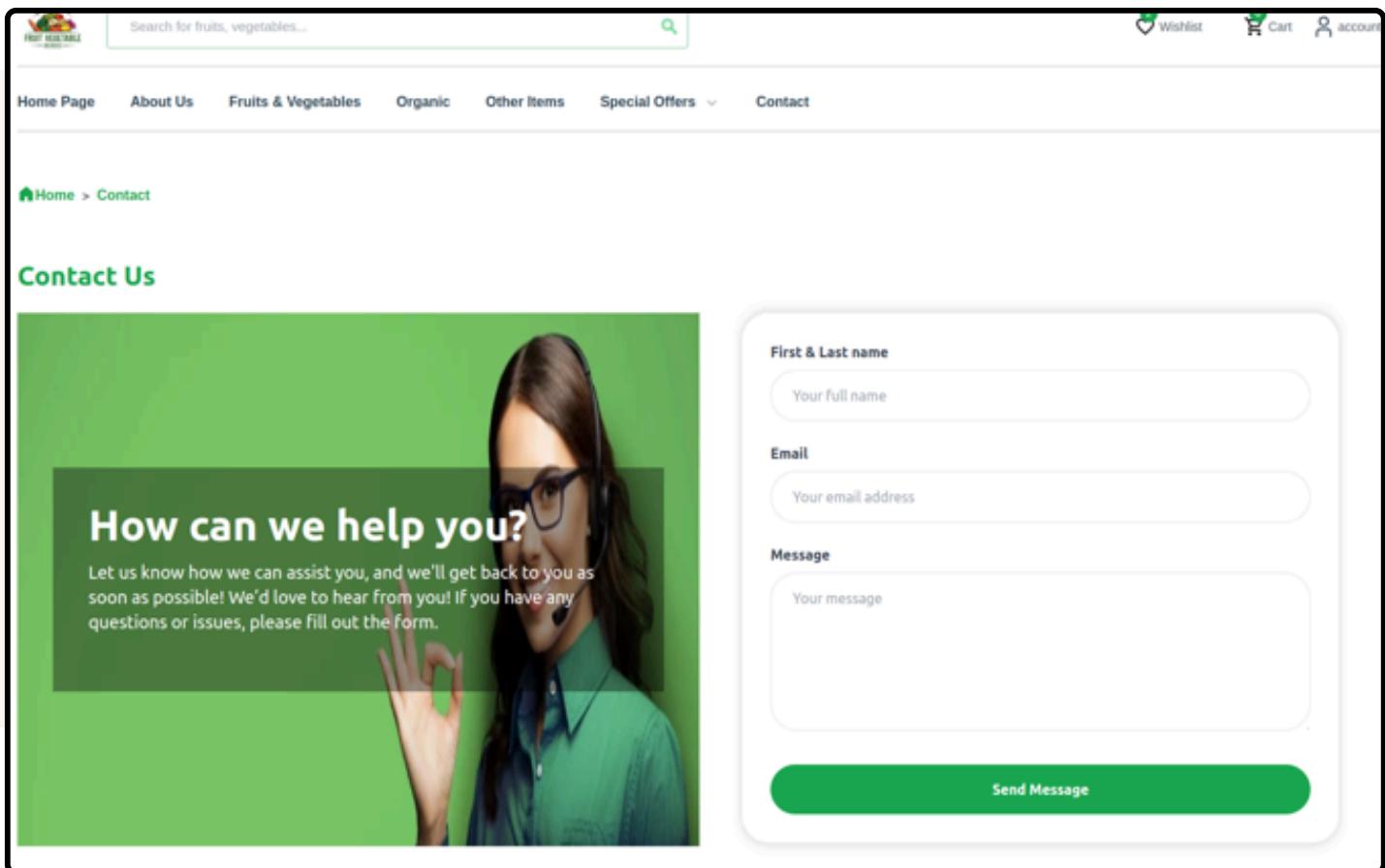
Technologies utilisées :

Élément	Outil / Technologie	Fonction
Redirection	React Router (<code>useNavigate</code>)	Redirige automatiquement l'utilisateur après la commande
Gestion d'état	<code>useState</code> , <code>useContext</code>	Nettoyage du panier, affichage dynamique selon l'état de la commande
Icônes & Boutons	React Icons , Tailwind CSS	Ajout d'éléments visuels réactifs et stylisés
Styles visuels	Tailwind CSS	Structure moderne, responsive et épurée de la page de confirmation

Chapitre V: Page "Contact Us"

- Analyse et Implémentation

Ce chapitre présente la page "**Contact Us**" de l'application web, conçue pour offrir aux utilisateurs un moyen simple et efficace de poser des questions ou de signaler un problème. Développée avec React, cette page intègre des fonctionnalités modernes pour la gestion dynamique des formulaires, assurant une expérience utilisateur fluide et réactive.



I. Structure de la Page

La page "Contact Us" est divisée en deux sections principales :

1. Section de contact

- Contient un formulaire avec les champs suivants :
 - Nom
 - Email
 - Message
- Un bouton d'envoi permet de transmettre les informations saisies.

2. Section visuelle

- Présente une image de fond avec un texte d'accroche superposé, apportant une touche esthétique et engageante à la page.

II. Méthodes Utilisées

La gestion des interactions et des états sur cette page repose sur les React Hooks, en particulier useState.

1. Gestion des données du formulaire

- o formData : stocke les valeurs saisies par l'utilisateur dans les champs du formulaire.

2. Gestion des messages de statut

- o statusMessage : contient le message de retour (succès ou erreur).
- o statusType : permet d'identifier le type de message (ex. : succès, erreur) pour un affichage conditionnel.

3. Gestion de la modale de confirmation

- o isModalOpen : détermine si la modale de confirmation doit être affichée après l'envoi du formulaire.

```
const Contact = () => {
  const [formData, setFormData] = useState({
    name: '',
    email: '',
    message: ''
  });
  const [statusMessage, setStatusMessage] = useState('');
  const [statusType, setStatusType] = useState('');
  const [isModalOpen, setIsModalOpen] = useState(false);
```

4. Gestion des Événements

Les événements du formulaire sont gérés à l'aide de deux fonctions principales

4-1 handleChange

- o Met à jour l'état formData à chaque modification d'un champ du formulaire.
- o Permet une saisie en temps réel et synchronisée avec l'interface.

4-2 handleSubmit

- o Intercepte la soumission du formulaire.
- o Envoie les données saisies à une API (backend ou service externe).
- o Met à jour les messages de statut (statusMessage, statusType) en fonction de la réponse.

```
const handleChange = (e) => {
  const { name, value } = e.target;
  setFormData((prevState) => ({
    ...prevState,
    [name]: value
  }));
};

const handleSubmit = async (e) => {
  e.preventDefault();
```

5. Communication avec l'API

La communication avec le backend est assurée via la méthode fetch, permettant l'envoi des données du formulaire vers un endpoint défini

5-1. Envoi des données

- Les données saisies sont envoyées en méthode POST au point d'accès suivant :
- `http://localhost:5000/api/contact`

5-2. Gestion des réponses

- En cas de succès : un message de confirmation est affiché à l'utilisateur.
- En cas d'erreur (échec réseau, validation serveur, etc.) : un message d'erreur personnalisé est présenté.

```
try {
  const response = await fetch('http://localhost:5000/api/contact', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(data)
});
```

6. Affichage des Résultats

Après la soumission du formulaire, une modale est utilisée pour informer l'utilisateur du résultat de l'opération :

6-1. Affichage conditionnel d'une modale

- En cas de succès : la modale affiche un message de confirmation (ex. : "Votre message a bien été envoyé.").
- En cas d'erreur : un message d'erreur personnalisé est présenté (ex. : "Une erreur est survenue, veuillez réessayer.").

6-2. Réinitialisation du formulaire

- Si la soumission est réussie, les champs du formulaire sont automatiquement vidés pour permettre un nouvel envoi éventuel.

```
if (response.ok) {
  setStatusType('success');
  setStatusMessage(result.message);
  setFormData({
    name: '',
    email: '',
    message: ''
  });
} else {
  setStatusType('error');
  setStatusMessage(result.message || 'Error sending message.');
```

III- Détails Techniques

1. Validation

- La validation des champs n'est pas encore pleinement implémentée, mais elle pourrait être ajoutée pour améliorer la fiabilité des données :
 - Vérification de la validité de l'adresse email.
 - Contrôle que tous les champs obligatoires sont remplis avant la soumission.

2. Sécurité

- Les données du formulaire sont envoyées au serveur au format JSON, avec l'en-tête HTTP approprié (Content-Type: application/json).
- Les éventuelles erreurs lors de l'envoi sont gérées à l'aide d'un bloc try/catch, évitant les crashes de l'application et permettant un retour utilisateur clair.

3. Accessibilité

- Le formulaire utilise des labels implicites pour les champs. Cependant, l'usage de labels explicites permettrait d'améliorer l'accessibilité, notamment pour les technologies d'assistance.
- La modale de retour permet d'informer visuellement l'utilisateur du succès ou de l'échec de son action, renforçant l'interactivité.

III-Conclusion

La page "Contact Us" propose une interface simple, claire et efficace, facilitant la communication entre les utilisateurs et l'équipe de support. Son développement repose sur des bonnes pratiques de développement React, telles que :

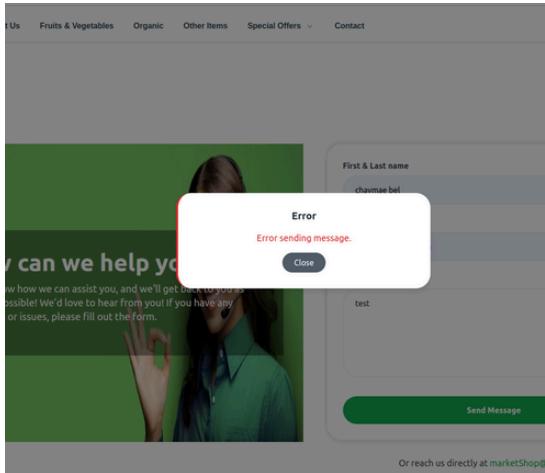
- La gestion d'état avec les hooks React.
- La communication asynchrone avec un backend via fetch.
- Un feedback utilisateur immédiat grâce à une modale conditionnelle.

Des axes d'amélioration restent possibles, notamment :

- L'ajout d'une validation client robuste.
- L'intégration de meilleures pratiques d'accessibilité (labels explicites, navigation clavier, etc.).

Ce chapitre a permis d'examiner en détail la page "**Contact Us**", tant sur le plan technique que fonctionnel, en soulignant son rôle dans la navigation du site et son comportement après soumission.

Chapitre VI: Gestion des Erreurs sur la Page "Contact Us"



Ce chapitre complémentaire analyse spécifiquement les mécanismes de gestion des échecs dans le processus d'envoi de message via le formulaire de contact. Il met en lumière la manière dont l'interface réagit face à une erreur, afin d'assurer une bonne expérience utilisateur même en cas de dysfonctionnement.

```

if (response.ok) {
  setStatusType('success');
  setStatusMessage(result.message);
  setFormData({
    name: '',
    email: '',
    message: ''
  });
} else {
  setStatusType('error');
  setStatusMessage(result.message || 'Error sending message.');
}
setModalOpen(true);
} catch (error) {
  console.error('Error:', error);
  setStatusMessage('Error sending message.');
  setModalOpen(true);
}
};

```

I .MESSAGE D'ERREUR AFFICHÉ

Lorsque l'envoi du message échoue, l'interface affiche une alerte claire .

Cette notification informe l'utilisateur qu'un problème a été rencontré lors de l'envoi. Le bouton "Close" permet de fermer manuellement l'alerte.

II . ANALYSE TECHNIQUE

1. Mécanisme d'Affichage

- Le message d'erreur reprend le même format que celui du succès, mais avec un statut de type "error".
- Il est affiché dans une modale ou une zone superposée au contenu.
- Le bouton Close permet de fermer la notification sans recharger la page.

2. Causes Possibles de l'Erreur

Selon l'analyse du code React :

- Échec de la requête API (réponse HTTP différente de 2xx)
- Erreur réseau (connexion perdue, serveur inaccessible)
- Problème de validation serveur (champ requis manquant, format invalide, etc.)
- Exception non gérée dans le bloc try/catch (ex. : bug dans la logique de traitement)

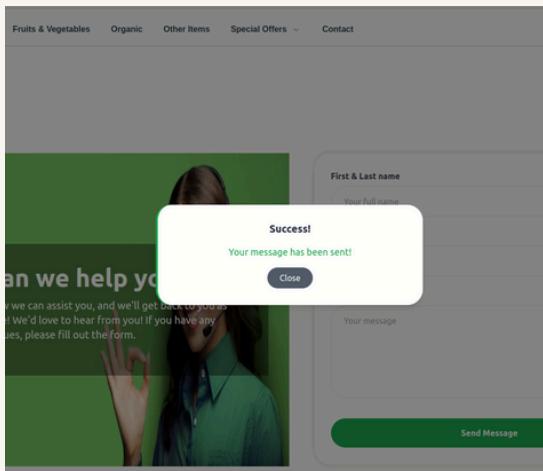
3. Comportement du Système

Lorsque l'erreur est détectée :

- Le système définit statusType à "error"
- Le message affiché est soit :
 - Le message retourné par le serveur
 - Un message générique comme "Error sending message."
- La modale est ouverte automatiquement pour informer l'utilisateur.

Chapitre VII: Page de Confirmation

– Succès de l'Envoi Us"



La page de succès — ou plus précisément la fenêtre modale de confirmation — est affichée après l'envoi réussi (ou échoué) du message de contact. Elle joue un rôle clé dans la validation de l'action de l'utilisateur et renforce la confiance dans le bon fonctionnement de l'interface.

```
if (response.ok) {
  setStatusType('success');
  setStatusMessage(result.message);
  setFormData({
    name: '',
    email: '',
    message: ''
  });
} else {
  setStatusType('error');
  setStatusMessage(result.message || 'Error sending message.');
}
setModalOpen(true);
} catch (error) {
  console.error('Error:', error);
  setStatusMessage('Error sending message.');
  setModalOpen(true);
}
```

I

.FONCTIONNEMENT GÉNÉRAL

L'interface repose sur un mécanisme de modale conditionnelle, déclenchée après la tentative d'envoi. Le comportement est piloté par plusieurs variables d'état :

- isModalOpen :
 - Booléen contrôlant la visibilité de la modale.
 - Passé à true après la soumission, qu'elle soit réussie ou non.
- statusType :
 - Chaîne de caractères valant "success" ou "error".
 - Permet d'adapter dynamiquement la couleur, l'icône et le titre de la modale.
- statusMessage :
 - Contenu textuel renvoyé par l'API (message de confirmation ou d'erreur).
 - Affiché directement sous le titre de la modale.
- closeModal :
 - Fonction appelée lors du clic sur le bouton "Close".
 - Met isModalOpen à false, fermant la modale.

II EXPÉRIENCE UTILISATEUR

L'approche par feedback immédiat via une modale est cohérente avec les standards modernes en UX/UI. Elle permet à l'utilisateur de :

- Confirmer que son action a bien été prise en compte.
- Être informé rapidement en cas de problème.
- Ne pas quitter la page ou la rafraîchir inutilement.

Conclusion

La modale de confirmation constitue une composante essentielle de l'interaction sur la page "Contact Us". Elle démontre une bonne maîtrise des états conditionnels avec React, tout en mettant en œuvre un retour visuel immédiat. Quelques ajustements mineurs (notamment sur le formulaire visible et le nettoyage du code) permettraient d'en faire une implémentation parfaitement aboutie.

Annexe : Utilisation de MongoDB dans le projet

Dans le cadre de ce projet, **MongoDB** a été adopté comme base de données NoSQL afin de gérer certaines fonctionnalités nécessitant une structure de données souple, une performance élevée en écriture et une scalabilité adaptée. Trois modules principaux s'appuient sur MongoDB : la newsletter, la page de contact, et la gestion des comptes utilisateurs.

1. Module Newsletter

L'inscription à la newsletter permet aux utilisateurs de soumettre leur adresse email afin de recevoir des promotions et actualités. Lors de la soumission, l'adresse est enregistrée dans une collection MongoDB dédiée.

The screenshot shows the MongoDB Compass interface connected to the 'localhost' host and the 'fruitvegmarke' database. The 'newsletters' collection is selected. Two documents are listed:

```

{
  "_id": ObjectId("67f26e37f03d569346c1d09c"),
  "email": "chaymae@gmail.com",
  "__v": 0
}

{
  "_id": ObjectId("67f26f7ff03d569346c1d09e"),
  "email": "test@gmail.com",
  "__v": 0
}

```

Avantages de MongoDB pour ce module :

- Stockage simple et rapide
- Éviter les doublons avec une contrainte d'unicité sur le champ email
- Facilité de recherche et de filtrage pour les campagnes marketing

2. Module Contact

La page **Contact Us** permet aux utilisateurs d'envoyer un message à l'équipe via un formulaire (nom, adresse email, sujet, message). Chaque message est sauvegardé dans MongoDB pour un traitement ultérieur par le service client.

La structure de document type:

json

```
{
  "name": "Ouahiba Yamlahi",
  "email": "ouahiba.doe@email.com",
  "message": "Je voudrais savoir quand ma commande sera livrée.",
  "createdAt": "2025-04-14T11:15:30Z"
}
```

The screenshot shows the MongoDB Compass interface connected to the 'localhost:27017' host and the 'fruitvegmarke' database. The 'contactforms' collection is selected. Three documents are listed, each representing a user message:

```

{
  "_id": ObjectId("67f27768f03d569346c1d0ac"),
  "name": "chaymae bel",
  "email": "customer@gmail.com",
  "message": "Great website i love it !!\nthank u so much market shop for making my ...",
  "createdAt": "2025-04-06T12:45:28.068+00:00",
  "__v": 0
}

{
  "_id": ObjectId("67f277e4f03d569346c1d0b0"),
  "name": "chaymae bel",
  "email": "chaymae@gmail.com",
  "message": "test test test",
  "createdAt": "2025-04-06T12:47:32.295+00:00",
  "__v": 0
}

{
  "_id": ObjectId("67f27812f03d569346c1d0b2"),
  "name": "chama chama",
  "email": "chaymae2002@gmail.com",
  "message": "chama chama chama",
  "createdAt": "2025-04-06T12:48:18.262+00:00",
  "__v": 0
}

```

Avantages :

- Historique des échanges conservé dans le temps
- Indexation rapide pour faciliter les recherches
- Possibilité d'ajouter des métadonnées (lu/non lu, prioritaire...)

3. Module Comptes Utilisateurs

MongoDB est également utilisé pour stocker les données de base des comptes utilisateurs, telles que les identifiants et les informations de profil.

Exemple de structure de document :

json

```
{
  "firstname": "Chaymae",
  "lastname": "belamkadem",
  "email": "chaymae.dupont@mail.com",
  "passwordHash": "$2a$12$xxxxxxxx"
}
```

Mesures de sécurité mises en place :

- Les mots de passe sont hashés (jamais stockés en clair)
- Utilisation d'un algorithme sécurisé tel que bcrypt
- Vérification d'unicité de l'adresse email lors de l'inscription

4. Bénéfices Généraux de MongoDB:

✓ Modèle NoSQL : idéal pour des structures évolutives et peu rigides

⚡ Haute performance : en écriture notamment, utile pour les formulaires et les logs

📦 Scalabilité horizontale : possibilité de répartir la charge sur plusieurs serveurs

🔧 Flexibilité du schéma : simplifie les évolutions sans migration complexe

Conclusion Générale

Le projet Vegetable Fruit Shop s'inscrit pleinement dans la dynamique actuelle de digitalisation du commerce alimentaire, en proposant une plateforme e-commerce moderne, réactive et ergonomique dédiée à la vente de fruits et légumes. Développée avec React.js pour le front-end et Node.js / Express.js couplé à MongoDB pour le back-end, cette application met l'accent sur l'expérience utilisateur, la performance et la scalabilité.

L'interface se distingue par un design clair et minimaliste, enrichi par Tailwind CSS, ainsi que par une série de fonctionnalités interactives : carrousel dynamique, navigation fluide, filtrage intelligent, wishlist, panier en temps réel et affichage détaillé des produits. Ces éléments contribuent à une expérience d'achat intuitive et engageante.

Sur le plan fonctionnel, le site intègre :

- Un système de coupons de réduction personnalisables (ex. DISCOUNT10)
- Un calcul automatique des taxes basé sur la localisation
- Un formulaire de newsletter pour la fidélisation
- Un mode sombre pour le confort visuel Ces ajouts renforcent le réalisme du processus d'achat et reflètent les pratiques du e-commerce professionnel.

D'un point de vue technique, le projet adopte une architecture claire et modulaire, séparant efficacement les responsabilités entre le front-end et le back-end. Les interactions avec MongoDB sont optimisées pour assurer la persistance des données clés (produits, commandes, préférences utilisateur).

Remerciements

Nous tenons à exprimer notre sincère gratitude à notre professeur, M. Youness Khamlchi, pour son encadrement exceptionnel, ses conseils précieux et sa disponibilité tout au long de ce projet. Grâce à son accompagnement, nous avons pu approfondir notre compréhension des concepts du développement web et mettre en pratique les compétences acquises.