“Nom du projet: UNI WATCH”

RNCP Niveau 5 (BAC +2)

Sommaire

1. Introduction 3

2. Résumé du Projet 3

3. Cahier des Charges 3-6

4. Spécifications Techniques 7

5. Technologies du Frontend 7

6. Technologies du Backend 7

7. Sécurité du site 8

8. Compétences du référentiel couvertes par le projet 9

9. Partie Frontend && Partie Backend 9-39

10. Réalisations Personnelles 40

11. Jeu d'essai 41

12. Veille Technologies 42

13. Problématique..... 43

14. Conclusion 44

Remerciements !

Je tiens à exprimer ma profonde gratitude envers tous les enseignants et l'ensemble du personnel de l'organisme de formation. Grâce à leur engagement, leur disponibilité et leu professionnalisme, j'ai pu acquérir les compétences nécessaires pour réaliser mes projets. Leur accompagnement tout au long de mon parcours a été précieux, et je les remercie sincèrement pour leur soutien.

Nom du Projet : Uni-Watch (Site E-commerce de Montres en Ligne)

Contexte :

Le projet consiste à développer un site e-commerce spécialisé dans la vente de montres en ligne. L'objectif est de proposer une large sélection de montres de luxe et classiques, tout en garantissant une expérience utilisateur optimale pour faciliter les achats en ligne.

Résumé du Projet

Dans le cadre de ce projet, j'ai conçu un site e-commerce dédié à la vente de montres en ligne. L'idée principale était de créer une plateforme facile à utiliser, offrant une expérience de shopping fluide et agréable. Le site présente une sélection de montres variées, allant de modèles simples à des pièces plus élégantes, adaptées à différents styles et budgets. L'objectif était de simplifier l'achat en ligne tout en garantissant un service de qualité, une présentation claire des produits et une navigation simple pour les utilisateurs.

Objectifs du projet

- Créer une boutique en ligne dédiée à la vente de montres.
- Faciliter l'achat grâce à un site simple, rapide et intuitif.
- Offrir une expérience utilisateur fluide sur mobile, tablette et ordinateur.

Public Cible :

Hommes à la recherche de montres élégantes, modernes et accessibles, alliant style et fonctionnalité.

Les besoins

Besoins Fonctionnels

Gestion des Utilisateurs :

1. Inscription et Connexion

- Création de compte utilisateur avec le prénom, e-mail et mot de passe.
- Connexion via prénom et mot de passe.

2. Déconnexion

- Les utilisateurs peuvent se déconnecter facilement de leur session.

3. Gestion des Erreurs de Connexion

- Les utilisateurs doivent recevoir un message d'erreur si les informations de connexion sont incorrectes (e-mail ou mot de passe).

4.Gestion des Rôles Utilisateurs:

- **Rôle "User" (Utilisateur Normal):**

Peut:

- Naviguer sur le site, consulter les produits, ajouter des articles au panier, passer des commandes.

Ne peut pas:

- Ajouter, modifier ou supprimer des produits.
- Modifier les informations d'autres utilisateurs
- Accéder à des fonctionnalités réservées aux administrateurs.

- **Rôle "Admin" (Admin):**

  Peut

  - Gérer les utilisateurs (ajouter et supprimer des membres).
  - Modifier le rôle des utilisateurs (passer un utilisateur en admin et inversement).
  - Gérer les produits (ajouter, modifier, supprimer des produits).

  Ne peut pas:

  - Avoir un accès illimité aux informations privées d'un utilisateur (ex. mot de passe).
  - Modifier les autres paramètres de compte des utilisateurs(prénom, email).

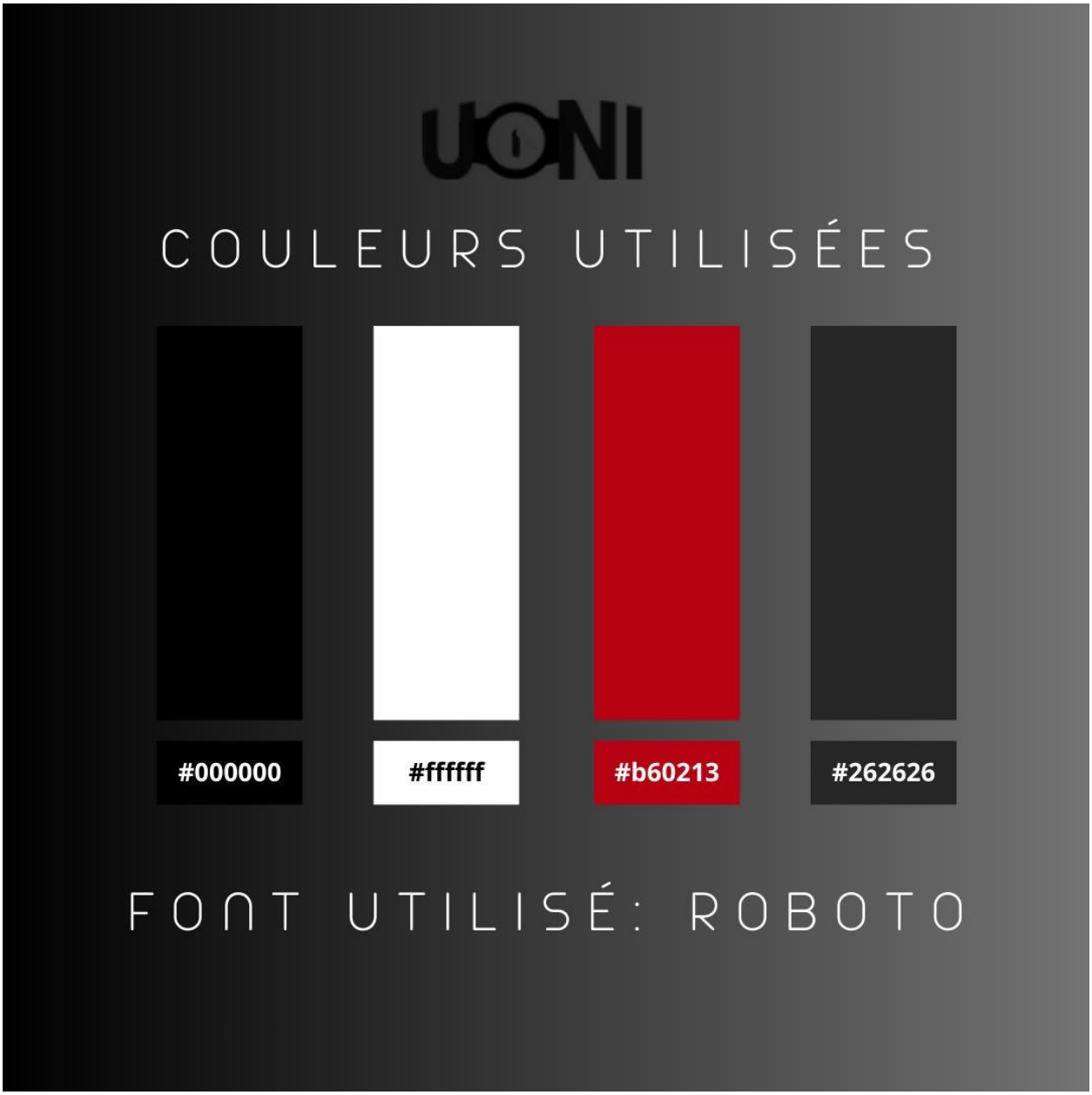
Besoins Techniques

1. **Technologies utilisées** : HTML, CSS, JavaScript, Ajax, PHP (POO), MySQL.
2. **Base de données** : MySQL.
3. **Sécurité des mots de passe** : Les mots de passe sont hachés pour renforcer leur protection.
4. **Responsive Design** : Interface adaptée aux smartphones et tablettes.

Les Contraintes

1. ***Temps:*** *Le projet doit être complété dans un délai de 6 mois à compter du début d développement.*
2. ***Technologiques:*** *Le développement sera réalisé avec les technologies suivantes : (PHP, HTML, CSS, JavaScript, MySQL).*
3. *Utilisation de* ***sécurisation des mots de passe*** *(hachage des mots de passe) pou assurer la protection des données personnelles des utilisateurs.*
4. ***Compatibilité:*** *Le site doit être* ***responsive*** *et s'adaptera correctement à différen appareils tels que les* ***télehpne portable***, ***tablettes*** *et* ***ordinateurs de bureau***.

La Charte Graphique



Spécifications Techniques

1.Technologies Utilisées

Le site de vente de montres utilise un mix de technologies front-end et back-end pour garantir une expérience utilisateur simple et sécurisée.

- **Front-End :**

  HTML5 : Structure des pages web.

  CSS3 : Mise en page et design responsive.

  JavaScript : Dynamisation du site et interactions utilisateur.

  Ajax : Requêtes asynchrones pour éviter le rechargement complet des pages et améliorer la rapidité d'affichage.

- **Back-End :**

  PHP (POO) : Gère les fonctionnalités du serveur et la logique métier en utilisant la programmation orientée objet.

  MySQL : Base de données relationnelle pour stocker toutes les informations du site (produits, utilisateurs, commandes, etc.).

- **Autres outils :**

  Figma : Outil de maquettage utilisé pour concevoir l'interface du site.

  - **MySQL:** Gestion de la base de données.
  - **Apache :** Serveur web pour exécuter les scripts PHP.
  - **VS Code :** Éditeurs de code pour le développement.

Sécurité du site

- **Hachage des mots de passe** : Tous les mots de passe des utilisateurs sont haché à l'aide de la fonction `password_hash()` de PHP, qui utilise l'algorithme **bcrypt** pour garantir une protection maximale des mots de passe. Cela permet de sécuriser les informations des utilisateurs, même en cas de fuite de données.

- **Validation et filtrage des entrées utilisateur** : Toutes les données envoyées pa les utilisateurs sont vérifiées et filtrées pour éviter les attaques comme l'injection SQL et le **Cross-Site Scripting (XSS)**. Pour ça, on utilise la fonction **`htmlspecialchars()`** qui transforme les caractères spéciaux en entités HTML, empêchant ainsi l'exécution de scripts malveillants envoyés par les utilisateurs.

- **Vérification de l'accès administrateur** : La fonction `checkAdminAccess()` sert à vérifier si l'utilisateur connecté a le rôle d'administrateur. Elle vérifie d'abord si un rôle est défini dans la session et s'assure que la valeur du rôle est **1** (indiquant un administrateur). Si l'utilisateur a un rôle de **0** (utilisateur normal) ou si aucune session n'est définie, il est redirigé vers la page d'accueil. Cela garantit que seules les personnes ayant les privilèges d'administrateur peuvent accéder à certaines pages ou fonctionnalités.

- **Sécurisation de l'accès à la page de paiement** : Pour éviter tout accès non autorisé, seuls les utilisateurs connectés peuvent accéder à la page de paiement. Si quelqu'un tente d'y accéder sans être connecté, il est automatiquement redirigé vers la page de connexion. Cela garantit qu'aucune personne non authentifiée ne puisse démarrer une transaction, assurant ainsi la sécurité des informations personnelles et financières des utilisateurs.

Compétences du référentiel couvertes par le projet

Partie Frontend && Partie Backend

CSS et Media Queries :

Pour garantir que le site soit **responsive** et s'adapte à tous les types d'appareils, j'ai utilis
CSS3 et les **media queries**. Les media queries permettent d'appliquer des styles spécifiques en fonction de la taille de l'écran. Par exemple, sur un mobile, les images sont réduites et les éléments sont réorganisés pour une meilleure lisibilité.

J'ai testé le site sur plusieurs tailles d'écran en utilisant les outils de développement intégrés dans le navigateur et en redimensionnant la fenêtre pour m'assurer qu'il fonctionne bien sur toutes les plateformes.

Voici un aperçu du site sur différentes tailles d'écran :

```css
/* header section start  */
.header-section {
    width: 100%;
    height: auto;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    padding: 15px 0;
    background: #fff;
}

.list {
    position: relative;
}

.list ul {
    list-style: none;
    display: flex;
    align-items: center;
    justify-content: center;
    position: relative;
}

.list ul.active {
    right: 0;
}
```

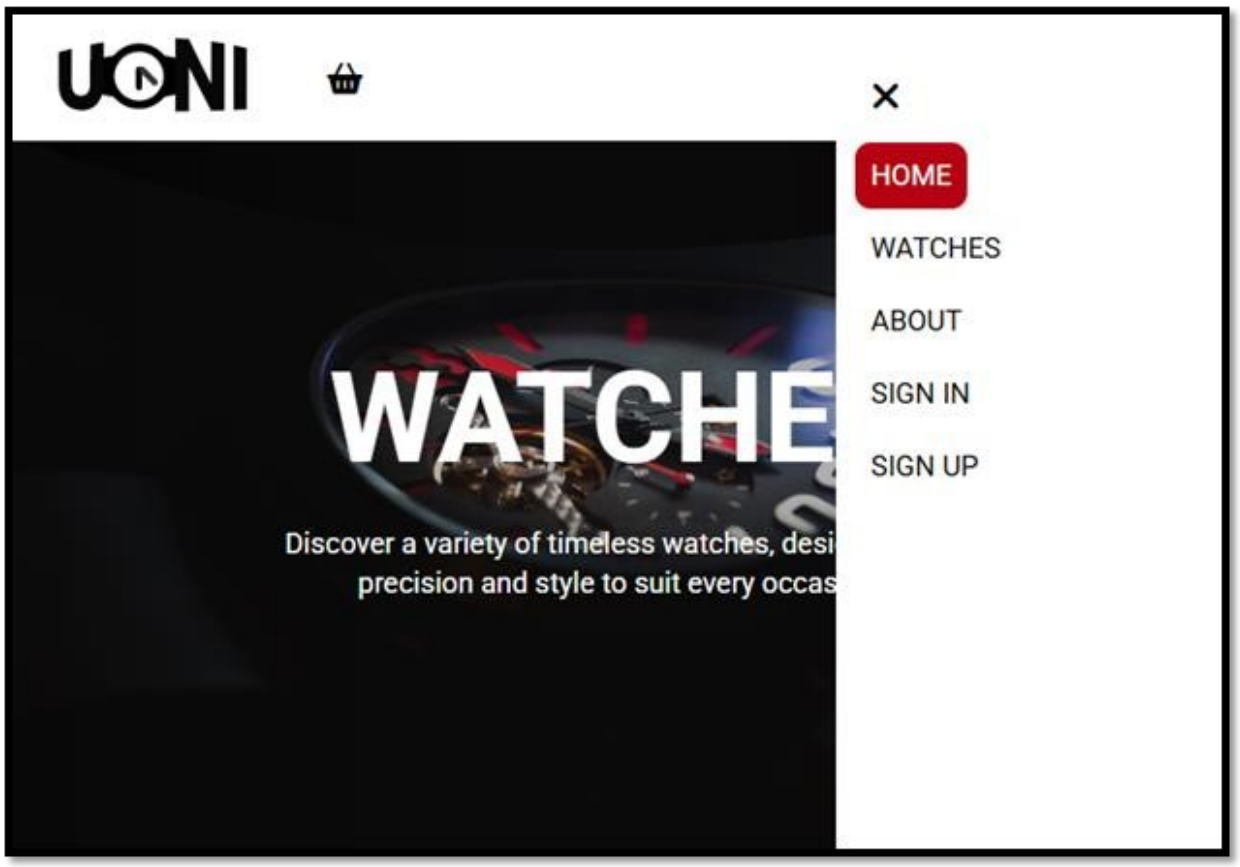
```css
/* large tablet  */
@media (max-width: 990px) {
    /* header section  */
    .header-section {
        height: 80px;
        background: #fff;
        padding: 10px;
        align-items: flex-start;
    }

    /*navbar for tablet and mobile device
    .list ul {
        flex-direction: column;
        align-items: flex-start;
        justify-content: flex-start;
        height: 100vh;
        width: 250px;
        position: fixed;
        top: 0;
        right: -250px;
        background-color: #fff;
        transition: all 0.4s linear;
        padding-top: 20px;
        padding-left: 5px;
        z-index: 999;
    }
```

JavaScript et Interactivité

JavaScript a été utilisé pour ajouter des fonctionnalités interactives et améliorer l'expérience utilisateur sur le site.Exemple :

- **Menu hamburger** : En JavaScript, j'ai géré l'ouverture et la fermeture du menu sur mobile pour une navigation fluide.

```
//open and close the navbar in the tablet and mobile device
document.querySelector('div#open').onclick = () => {
    document.querySelector('ul#nav').classList = 'active'
}
document.querySelector('li#close').onclick = () => {
    document.querySelector('ul#nav').className = ''
}
```



PHP et Gestion du Site

PHP est utilisé pour gérer les interactions côté serveur, traiter les données, gérer les utilisateurs, et garantir la sécurité et la performance du site dans son ensemble.

Inscription d'un Utilisateur

Lorsqu'un utilisateur s'inscrit, ses informations sont envoyées au serveur pour être sauvegardées dans la base de données. Cette méthode insère les données d'inscription dans la base de données via une requête SQL.

```php
public function saveSign_UpForm(Sign_Up $sign_up){
    try {
        $sql = "INSERT INTO user (username, email, password, role)
        VALUES (:username, :email, :password, :role)";
        $stmt = $this->connection->prepare($sql);

        $username = $sign_up->getUsername();
        $email = $sign_up->getEmail();
        $password = $sign_up->getPassword();
        $role = $sign_up->getRole();

        $stmt->bindParam(':username', $username);
        $stmt->bindParam(':email', $email);
        $stmt->bindParam(':password', $password);
        $stmt->bindParam(':role', $role);

        $result = $stmt->execute();

        if ($result) {
            return $this->connection->lastInsertId();
        }

        return false;
    } catch (PDOException $e) {
        error_log("Database error: " . $e->getMessage());
        echo "Something went wrong. Please try again.";
        return false;
    }
}
```

Validation du formulaire d'inscription :

vérifie si le nom d'utilisateur et l'email sont valides, non vides et uniques, et renvoie des messages d'erreur le cas échéant.

```php
$signUpRepo = new Sign_UpRepository();
if ($signUpRepo->checkIfUsernameExists($_POST['username'])) {
    $errors['username'] = "Username already exists. Please choose another one.";
}

if (empty($_POST['email'])) {
    $errors['email'] = "Email is required.";
} elseif (!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
    $errors['email'] = "Invalid email format.";
} elseif ($signUpRepo->checkIfEmailExists($_POST['email'])) {
    $errors['email'] = "Email already exists. Please use a different one.";
}
```

Traitement du formulaire d'inscription :

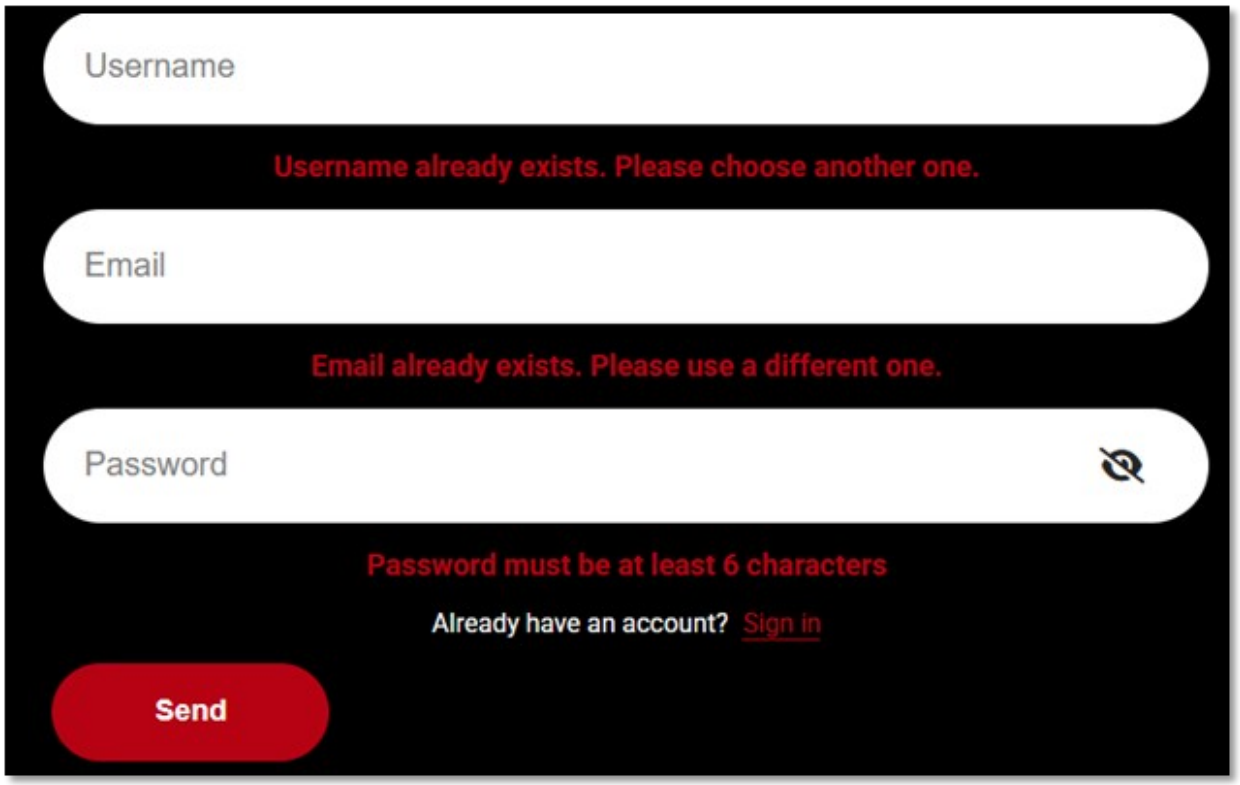
gère la soumission, valide les données et enregistre l'utilisateur en cas de succès.

```php
public function sign_UpForm() {

    $formHandler = new Sign_UpHandleRequest();
    $formHandler->handleSignUpRequest($this->sign_up);
    $errors = [];

    if ($formHandler->isSubmitted()) {
        if ($formHandler->isValid()) {
            $this->saveFormData($this->sign_up);
            redirection('/uni-watch/home/index');
            exit();
        } else {
            $errors = $formHandler->getErrorsForm();
        }
    }

    return $this->render('sign-up.html.php', ['errors' => $errors]);
}
```

Traitement de l'inscription : Vérifie les données saisies et affiche les erreurs éventuell

```
<input type="text" class="sign-up-username" name="username" placeholder="Username">
<p class="error-message username-error">
    <?= !empty($errors['username']) ? $errors['username'] : ''; ?>
</p>/.error-message.username-error


<input type="email" class="sign-up-email" name="email" placeholder="Email">
<p class="error-message email-error">
    <?= !empty($errors['email']) ? $errors['email'] : ''; ?>
</p>/.error-message.email-error


<div class="sign-up-password">
    <input type="password" class="sign-up-password-input" name="password" placeholder="Password">
    <i class="fa-solid fa-eye-slash" id="eye"></i>
</div>/.sign-up-password
<p class="error-message password-error">
    <?= !empty($errors['password']) ? $errors['password'] : ''; ?>
</p>/.error-message.password-error
```

Erreurs de formulaire : Montre les messages d'erreur affichés lorsque les informations soumises ne sont pas valides, permettant à l'utilisateur de corriger les erreurs.



Connexion d'un Utilisateur

Description: Lorsqu'un utilisateur se connecte, ses identifiants (le nom d'utilisateur et le mot de passe), sont envoyés au serveur pour une authentification à partir des données stockées dans la base de données.

La méthode getUserByUsername est conçue pour récupérer les informations d'un utilisateur spécifique en fonction de son nom d'utilisateur.

```php
class Sign_InRepository extends BaseRepository {
    public function getUserByUsername($username) {
        try {
            $sql = "SELECT * FROM user WHERE username = :username";
            $stmt = $this->connection->prepare($sql);
            $stmt->bindParam(':username', $username);
            $stmt->execute();

            $user = $stmt->fetch();

            return $user ? $user : null;
        } catch (PDOException $e) {
            error_log("Database error: " . $e->getMessage());
            return false;
        }
    }

}
```

La méthode handleSignInRequest vérifie la soumission, valide les champs username et password, et utilise Sign_InRepository pour récupérer l’utilisateur.

Si les identifiants sont corrects, elle retourne l’utilisateur ; sinon, elle signale une erreur e renvoie null.

```php
public function handleSignInRequest() {
    if ($this->isSubmitted()) {
        $errors = [];

        if (empty($_POST['username'])) {
            $errors['username'] = "Username is required.";
        }

        if (empty($_POST['password'])) {
            $errors['password'] = "Password is required.";
        }

        if (empty($errors)) {
            $signInRepo = new Sign_InRepository();

            $user = $signInRepo->getUserByUsername($_POST['username']);

            if ($user && password_verify($_POST['password'], $user['password'])) {
                return $user;
            } else {
                $errors['general'] = "Invalid username or password.";
            }
        }

        $this->setErrorsForm($errors);
    }
    return null;
}
```

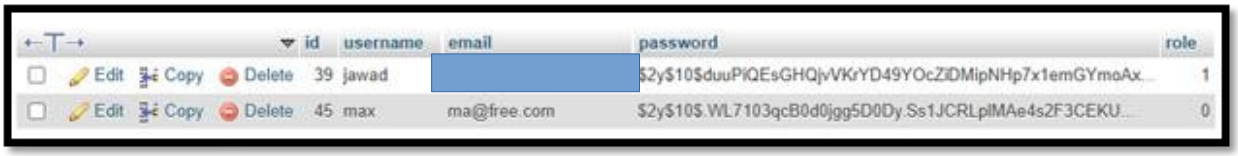
La méthode signInForm crée un objet Sign_InHandleRequest pour gérer la connexion.

Si le formulaire est soumis et valide, elle enregistre les données utilisateur dans la sessio
et redirige vers l'accueil.
Sinon, elle récupère les erreurs et rend le formulaire de connexion avec ces erreurs.

```php
public function signInForm() {
    $formHandler = new Sign_InHandleRequest();
    $formHandler->handleSignInRequest();
    $errors = [];

    if ($formHandler->isSubmitted()) {
        if ($formHandler->isValid()) {
            $user = $formHandler->handleSignInRequest();

            $_SESSION['user_id'] = $user['id'];
            $_SESSION['username'] = $user['username'];
            $_SESSION['role'] = $user['role'];

            redirection('/uni-watch/home/index');
            exit();
        } else {
            $errors = $formHandler->getErrorsForm();
        }
    }

    return $this->render('sign-in.html.php', ['errors' => $errors]);
}
```

Après qu'un utilisateur se connecte ou s'inscrit, l'en-tête affiche son nom à côté de l'icône de profil. S'il est , son nom apparaît suivi de « (admin) », il a un accès aux pages de gestion des produits et des utilisateurs, et une option de déconnexion(logout) es ajoutée.

```php
<?php if (isset($_SESSION['username'])): ?>
    <li class="list-item admin">
        <i class="fa-solid fa-user"></i>
        <?= htmlspecialchars($_SESSION['username']); ?>
        <?php if (isset($_SESSION['role']) && $_SESSION['role']): ?>
            <span>
                (admin) <i class="fa-solid fa-angle-down" id="angle-icon"></i>
            </span>
        <?php endif; ?>
        <div id="admin-links">
            <a href="/uni-watch/admin_add_product/addProduct">Product Management</a>
            <a href="/uni-watch/admin_add_user/addUser_form">User Management</a>
        </div>/#admin-links
    </li>/.list-item.admin
<?php endif; ?>
```



Ajouter un produit au panier

Lorsqu'un utilisateur ajoute un produit, le panier se met à jour instantanément grâce à **AJAX**, sans nécessiter le rechargement de la page.

Processus :
L'ID du produit et la quantité sont envoyés au serveur via **Fetch API avec la méthode G**

```
// add product to the cartContainer(box)
const params = new URLSearchParams(window.location.search);
const productId = params.get("id");
const addToCartBtn = document.querySelector('.add-to-cart-btn');
const productQuantity = document.querySelector('.title .product-quantity');
const quantityInput = document.querySelector('#detail-quantity');
const cartContainer = document.querySelector('.product-detail-container');
const totalPriceElement = document.querySelector('.total-price-container .total-price');

addToCartBtn.onclick = () => {
    const quantity = quantityInput.value;

    fetch(`/uni-watch/basket/productDetail?productId=${productId}&quantity=${quantity}`)
        .then(response => response.json())
```

Gestion du panier avec PHP : Le code vérifie si un panier existe en session. Si ce n'est pas le cas, il initialise un panier vide. Lorsqu'un produit est ajouté, il est stocké en session avec ses détails, et le prix total est mis à jour. Si le produit est déjà dans le panier, une erreur est renvoyée.

Retour des données du panier : Le code renvoie une réponse JSON contenant le statut l'opération (`success`), le nombre total de produits dans le panier (`cartCount`), les détails du produit ajouté (`product`), et le prix total du panier (`totalPrice`).

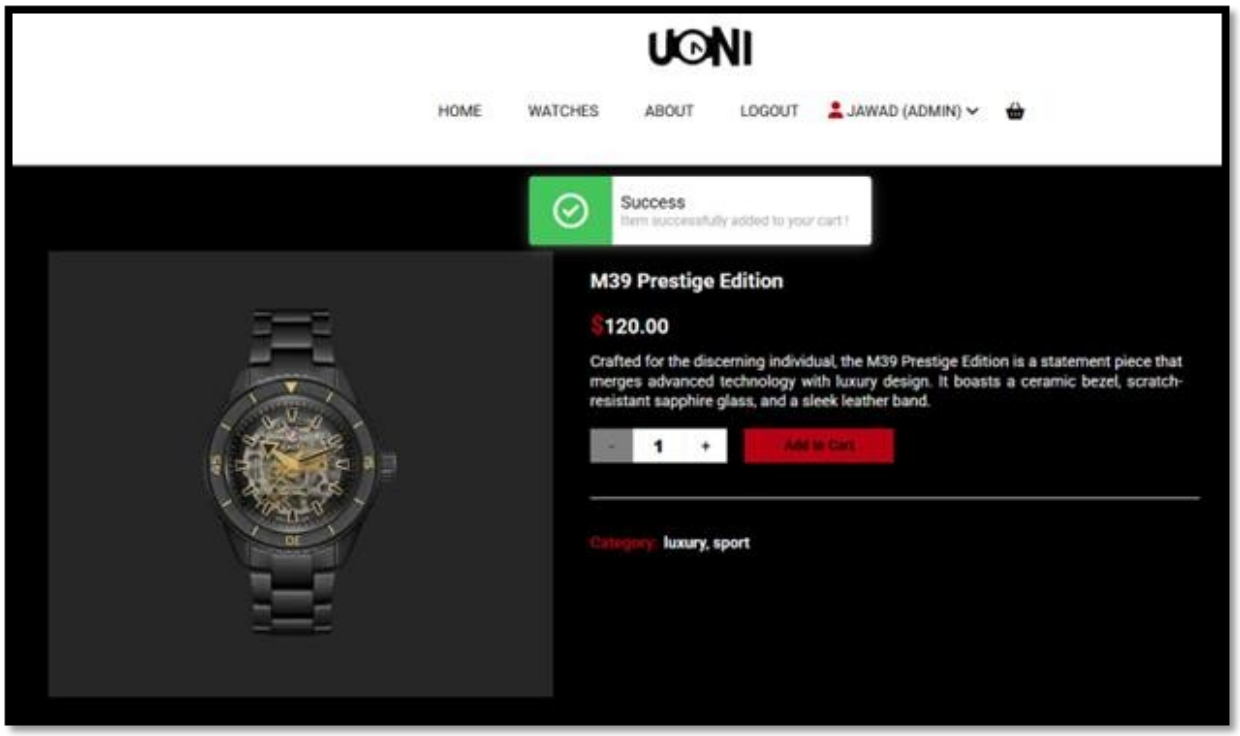
```php
if (!isset($_SESSION['cart'])) {
    $_SESSION['cart'] = [];
    $_SESSION['totalPrice'] = 0;
}

if (isset($_SESSION['cart'][$productId])) {
    echo json_encode(['status' => 'error']);
    exit;
} else {
    $_SESSION['cart'][$productId] = [
        'id' => $product['id'],
        'title' => $product['title'],
        'brand' => $product['brand'],
        'category' => $product['category'],
        'description' => $product['description'],
        'image_path' => $product['image_path'],
        'price' => $product['price'],
        'stock' => $product['stock'],
        'quantity' => $productQuantity
    ];

    $_SESSION['totalPrice'] += $product['price'] * $productQuantity;

    echo json_encode([
        'status' => 'success',
        'cartCount' => count($_SESSION['cart']),
        'product' => $_SESSION['cart'][$productId],
        'totalPrice' => $_SESSION['totalPrice']
    ]);
```

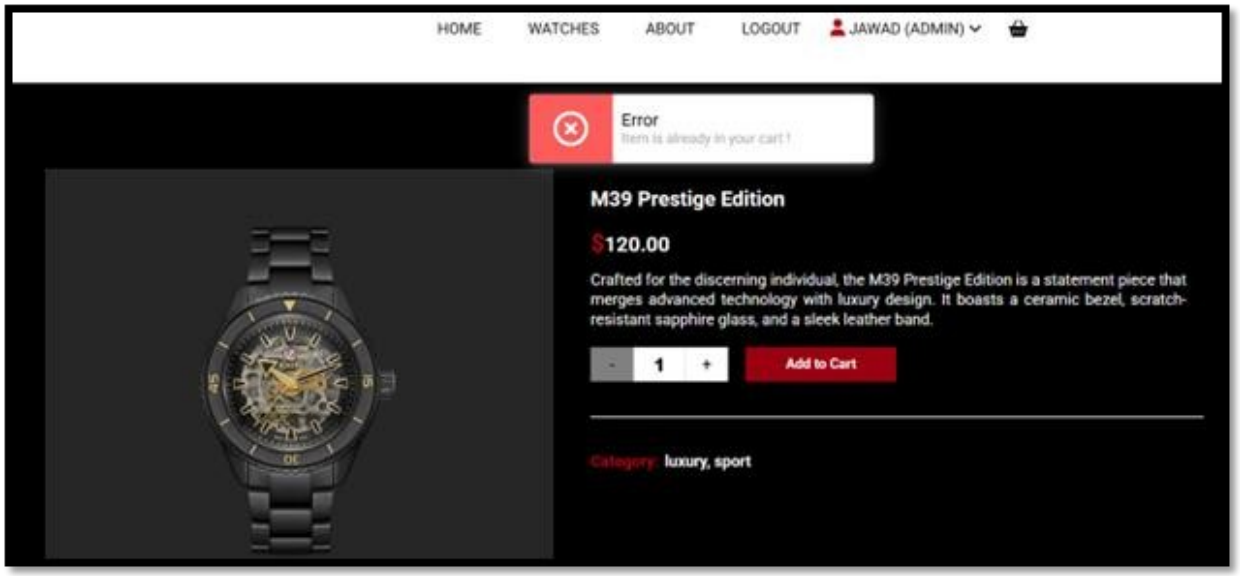
Si l’ajout réussit, un nouveau bloc produit est créé dynamiquement avec le nom, le prix, la quantité, l’image et l’icône de suppression.
Le total d’articles et le prix total en temps réel.

```
productQuantity.innerHTML = `${data.cartCount} <span>Product(s)</span>`;

const productBox = document.createElement('div');
productBox.classList.add('product-box');
productBox.innerHTML = `
    <div class="delete-icon-container">
        <i class="fa-regular fa-trash-can" data-id= "${data.product.id}"></i>
    </div>
    <div class="product-details-container">
        <p class="product-name">${data.product.title}</p>
        <div class="product-price">
            <p class="price">$${data.product.price}</p>
            <p class="product-quantity"><i class="fa-solid fa-xmark"></i>${data.product.quantity}</p>
        </div>
    </div>
    <div class="product-img-container">
        <img src="/uni-watch/public/assets/images/watches/${data.product.image_path}" alt="${data.product.title}">
    </div>
`;

cartContainer.append(productBox);

if (data.cartCount > 0 && data.totalPrice > 0) {
    totalPriceElement.textContent = `Total Price: $${data.totalPrice.toFixed(2)}`;
} else {
    totalPriceElement.textContent = "Your Basket Is Empty!";
}
```

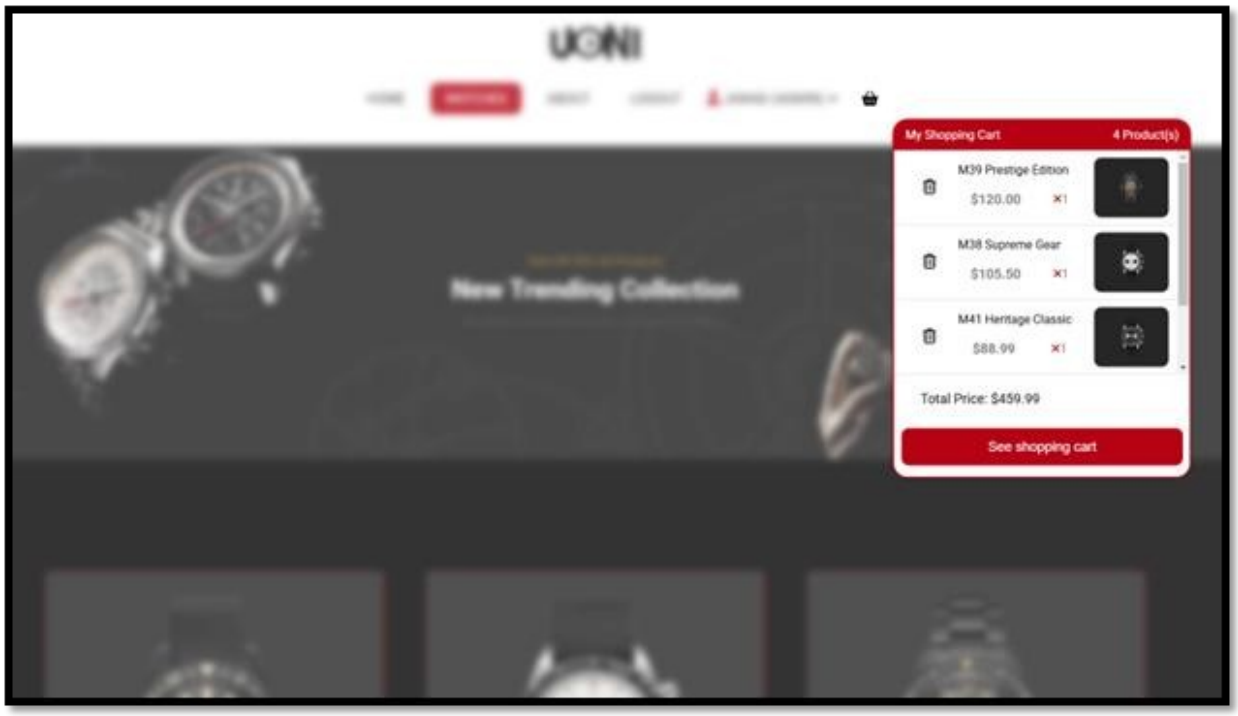
```
<div class="product-box">
    <div class="delete-icon-container">
        <i class="fa-regular fa-trash-can" data-id="<?= htmlspecialchars($item['id']); ?>"></i>
    </div>/.delete-icon-container
    <div class="product-details-container">
        <p class="product-name"><?= htmlspecialchars($item['title']); ?></p>
        <div class="product-price">
            <p class="price">$<?= htmlspecialchars($item['price']); ?></p>
            <p class="product-quantity"><i class="fa-solid fa-xmark"></i><?= $item['quantity']; ?></p>
        </div>/.product-price
    </div>/.product-details-container
    <div class="product-img-container">
        <img src="/uni-watch/public/assets/images/watches/<?= htmlspecialchars($item['image_path']); ?>" al
    </div>/.product-img-container
</div>/.product-box
```

Ajout au panier : Après clic sur "Add to Cart", un message de succès s'affiche si le produ
est ajouté, sinon un message d'erreur informe que l'article est déjà dans le panier.





Lorsqu’un utilisateur clique sur l’icône du panier, une boîte s’affiche, montrant la liste des produits ajoutés avec leur prix, leur quantité et le prix total. Une icône de suppression est également présente pour permettre à l’utilisateur de retirer un produit du panier.



Suppression d’un Produit du Panier:

JavaScript :

- Lorsqu'un utilisateur clique sur l'icône de suppression (`fa-trash-can`), l'ID du produit est récupéré.
- Une requête **fetch** est envoyée au serveur pour supprimer le produit.
- Si la suppression réussit, un message de succès s'affiche, l'élément est retiré du DOM et le prix total est mis à jour.

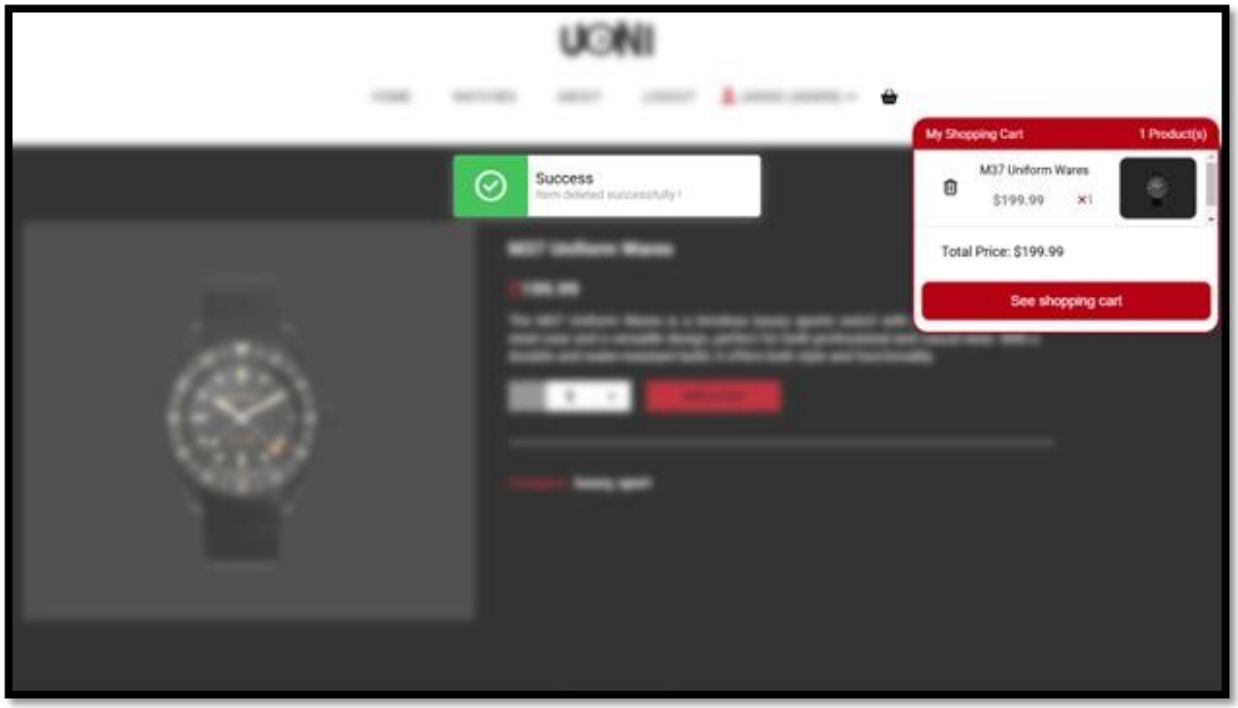
PHP :

- Vérifie si un `productId` est fourni et existe dans le panier (`$_SESSION['cart']`).
- Déduit son prix du **total** et le supprime du panier.
- Renvoie une réponse JSON avec le **nouveau total** et le **nombre d'articles restan**

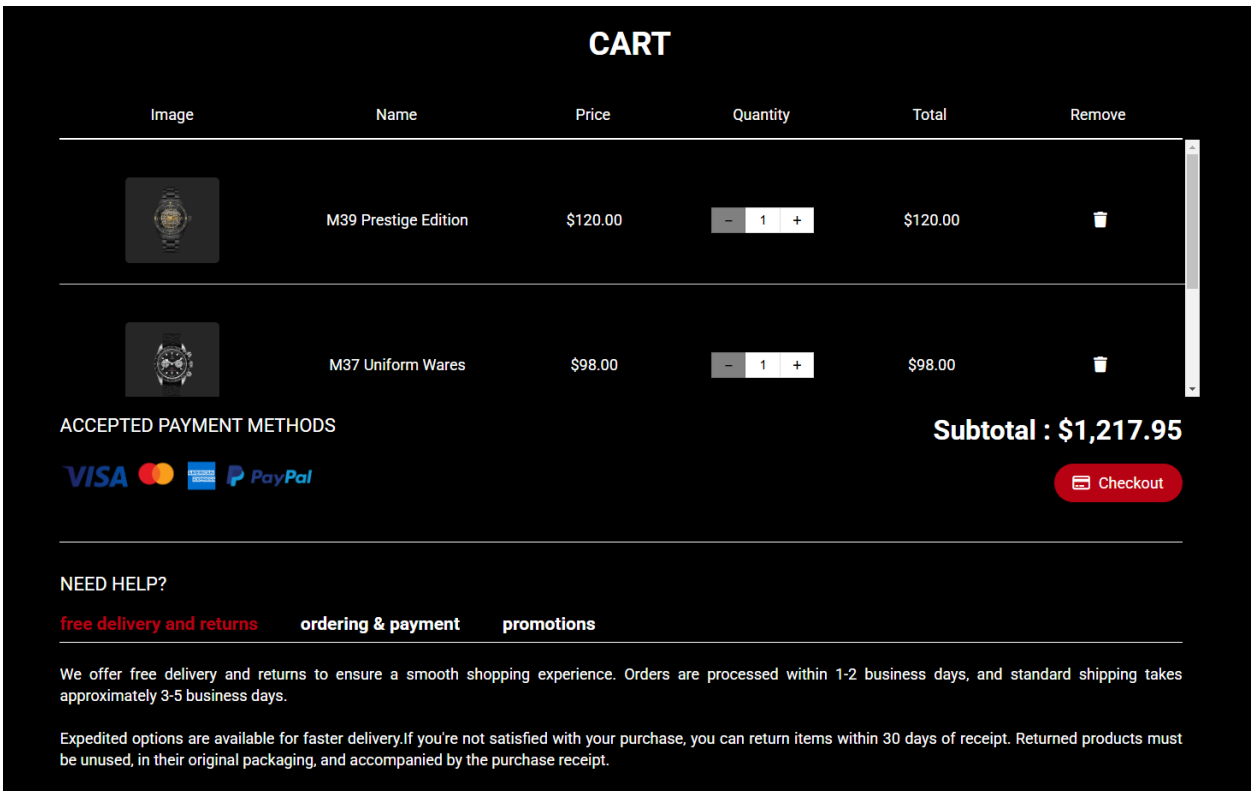
```
// delete product from the cart box
document.querySelector('.product-detail-container').onclick = (evt) => {
    if(evt.target.classList.contains('fa-trash-can')) {

        const item = evt.target
        const productId = item.getAttribute('data-id')

        fetch(`/uni-watch/basket/deleteProduct?productId=${productId}`)
        .then(response => response.json())
        .then(data => {
            if(data.status ==='success') {
                // success box generator function
                successBoxGenerator('Item deleted successfully !');

                item.closest('.product-box').remove()

                updateTotalPrice(data.totalPrice, data.cartCount);
            }
        })
        .catch(error => console.error('Error deleting product:', error));

    }
}
```

```php
// delete a product from the cart box
public function deleteProduct() {
    if (isset($_GET['productId'])) {
        $productId = (int)$_GET['productId'];

        if (isset($_SESSION['cart'][$productId])) {

            $_SESSION['totalPrice'] -= $_SESSION['cart'][$productId]['price'] * $_SESSION['cart'][$productId]['quantity'];
            unset($_SESSION['cart'][$productId]);

            $cartCount = count($_SESSION['cart']);

            echo json_encode([
                'status' => 'success',
                'cartCount' => $cartCount,
                'totalPrice' => $_SESSION['totalPrice']
            ]);
            exit;
```

Si la suppression est réussie, un **message de succès** s’affiche, le produit disparaît immédiatement du panier, et le prix total ainsi que le nombre d’articles sont mis à jour en temps réel.



Affichage et gestion du panier d'achat : Lorsqu’un utilisateur clique sur le bouton "See shopping cart", il est redirigé vers la page du panier. Sur cette page, il peut voir la liste complète des produits , comprenant leur image, leur nom, leur prix unitaire, la quantité sélectionnée et le prix total. Il a également la possibilité de modifier la quantité d chaque produit ou de le supprimer à l’aide de l’icône de suppression. Le sous-total du panier est affiché, ainsi qu’un bouton "Checkout" pour finaliser l’achat. De plus, les méthodes de paiement acceptées sont indiquées, et une section d’aide fournit des informations sur la livraison, le paiement et les retours.



Processus de commande : Lorsqu’un utilisateur clique sur le bouton "Checkout", s'il est connecté et que le panier n’est pas vide il est redirigé vers la page de paiement où il retrouve la liste des produits sélectionnés, ainsi qu’un formulaire pré-rempli qu’il n’a pas besoin de remplir à nouveau. Ensuite, en cliquant sur "Continue To Payment", un message de confirmation s'affiche, indiquant que la commande a bien été créée. L’utilisateur est ensuite redirigé vers la page d'accueil.

Si le panier est vide, il est redirigé vers la page des produits, et s'il n'est pas connecté, ve la page de connexion.

Réinitialisation du panier : Après la confirmation de la commande, `unset($_SESSION['cart']);` est utilisé pour vider le panier de la session, indiquant que la commande a été traitée et que le panier est réinitialisé pour une nouvelle session.

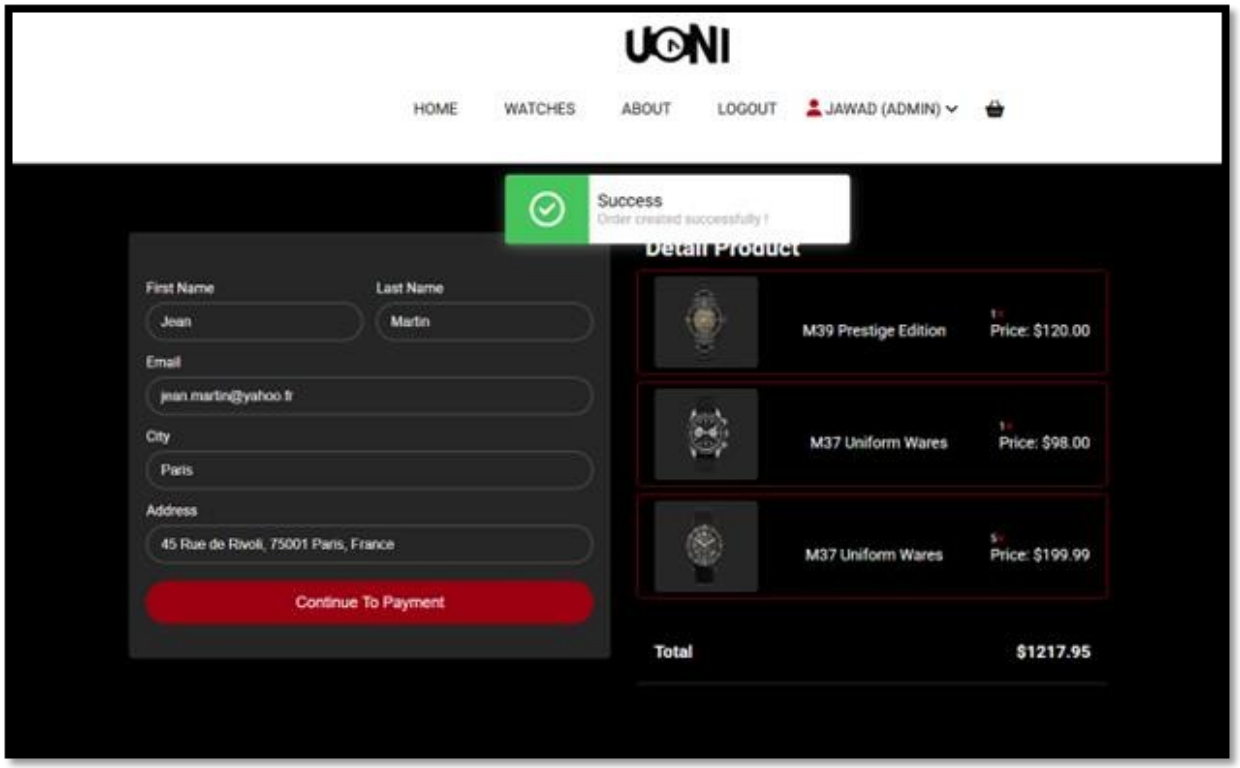
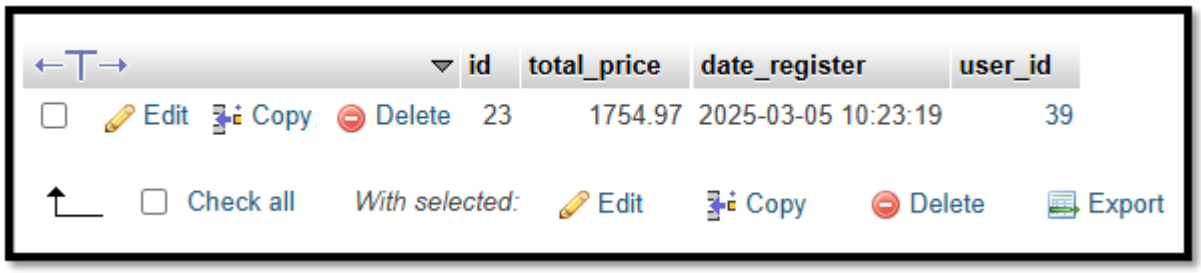


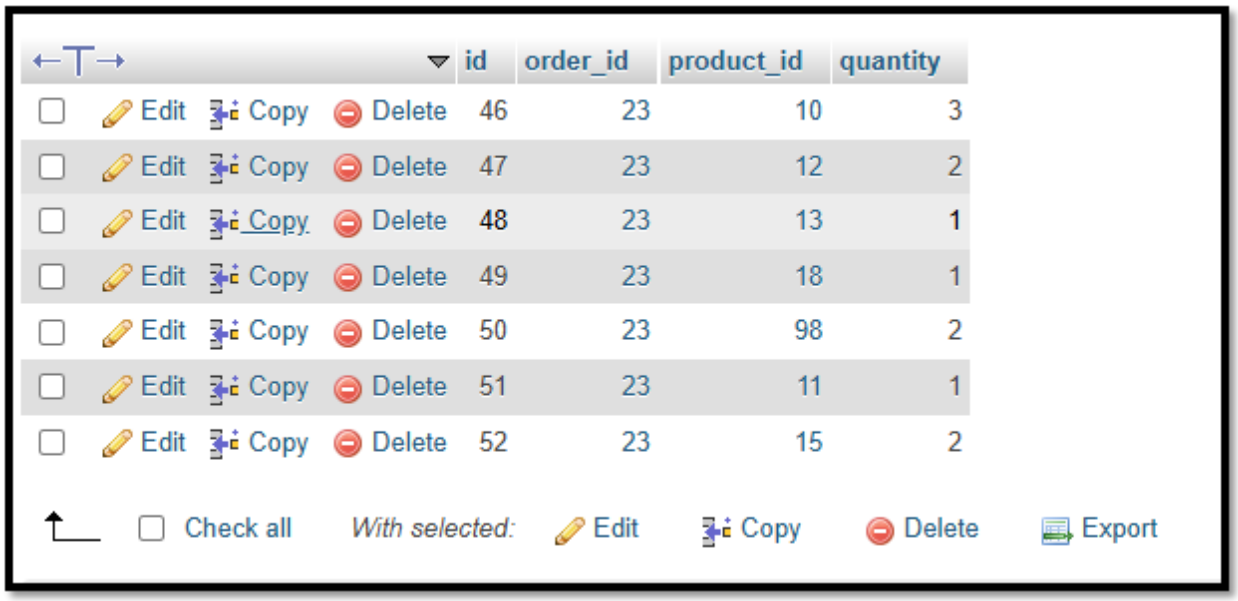
Table "order" : Contient les informations générales de la commande, comme l'**ID de l'utilisateur**, le **prix total** et la **date d'enregistrement**.



	id	total_price	date_register	user_id
Edit Copy Delete	23	1754.97	2025-03-05 10:23:19	39

Check all *With selected:* Edit Copy Delete Export

Table "order_detail" : Stocke les détails des produits commandés, incluant l'**ID de la commande**, l'**ID du produit** et la **quantité achetée**.

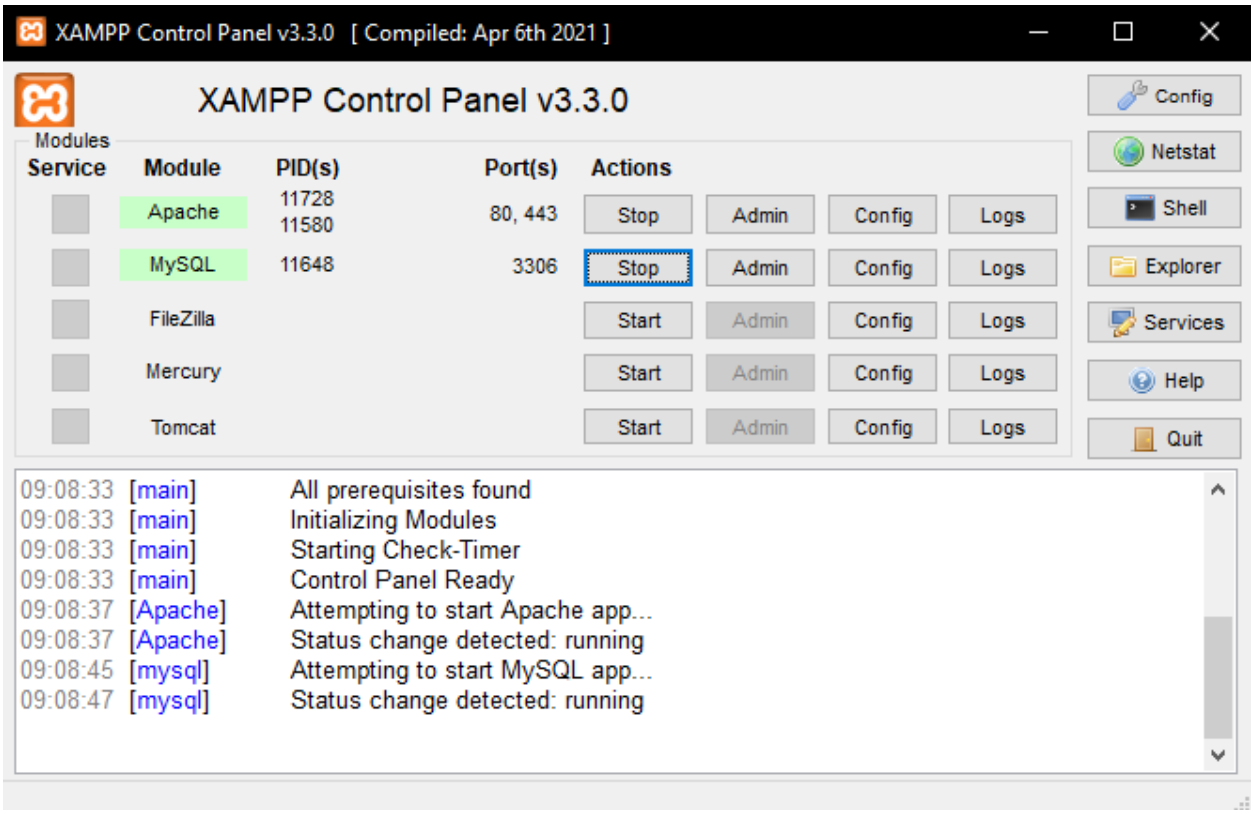


	id	order_id	product_id	quantity
Edit Copy Delete	46	23	10	3
Edit Copy Delete	47	23	12	2
Edit Copy Delete	48	23	13	1
Edit Copy Delete	49	23	18	1
Edit Copy Delete	50	23	98	2
Edit Copy Delete	51	23	11	1
Edit Copy Delete	52	23	15	2

Check all *With selected:* Edit Copy Delete Export

Utilisation de XAMPP pour l'hébergement local et la gestion des données:

XAMPP est un environnement de développement local qui permet d'exécuter Apache (serveur web) et MySQL (serveur de base de données) sur une machine locale. Dans ce projet, XAMPP est utilisé pour héberger l'application web et gérer les données via MySQL. Le démarrage des services Apache et MySQL permet de lancer l'application et d'interagir avec la base de données en local.



Conception et structuration de la base de données (MLD, MCD, MPD) :

Modèle Logique de Données (MLD)

Nom de la base de données : *uni-watch****.***

Les Entités :

1. User

- *id INT PRIMARY KEY AUTO_INCREMENT*
- *username VARCHAR(255) NOT NULL UNIQUE*
- *email VARCHAR(255) NOT NULL UNIQUE*
- *password VARCHAR(255) NOT NULL*
- *role INT(11) NOT NULL DEFAULT 0 — 0 pour utilisateur normal, 1 pour administrateur*
- ***Relation*** *: Un User peut "passer" plusieurs Commandes (relation 1:N).*

2. Product

- *id INT PRIMARY KEY AUTO_INCREMENT*
- *title VARCHAR(255) NOT NULL*
- *brand VARCHAR(255) NOT NULL*
- *category VARCHAR(255) NULL*
- *description TEXT NOT NULL*
- *image_path VARCHAR(255) NOT NULL*
- *price DECIMAL(10,2) NOT NULL*
- *stock INT(11) NOT NULL*
- ***Relation*** *: Peut être contenu dans plusieurs Commandes*

3. Order

- id INT PRIMARY KEY AUTO_INCREMENT
- total_price DECIMAL(10,2) NOT NULL
- date_register TIMESTAMP NOT NULL DEFAULT current_timestamp()
- user_id INT(11) NOT NULL (Foreign Key referencing **user.id**)
- **Relation:** Un **User** peut passer plusieurs **Orders** (relation 1:N), Peut contenir plusieurs Produits.

4. order_detail

- id INT PRIMARY KEY AUTO_INCREMENT
- order_id INT(11) NOT NULL (Foreign Key referencing **order.id**)
- product_id INT(11) NOT NULL (Foreign Key referencing **product.id**)
- quantity INT(11) NOT NULL DEFAULT 1
- **Relation :** Une **Commande** peut contenir plusieurs **Produits**, et un **Produit** peut être présent dans plusieurs **Commandes** (relation N:M via **detail_order**).

MCD (Modèle Conceptuel de Données)

1. Entité : User

- **Attributs** :
  id (Identifiant unique de l'utilisateur)
  username (Nom d'utilisateur, unique)
  email (Adresse e-mail, unique)
  password (Mot de passe haché)
  role (0 pour utilisateur normal, 1 pour administrateur)
- **Relation** :
  Un **User** peut passer plusieurs **Orders**.
  Cardinalité : 1:N (Un utilisateur peut avoir plusieurs commandes, mais chaque commande appartient à un seul utilisateur).

2. **Entité : Product**

- **Attributs** :
  id (Identifiant unique du produit)
  title (Titre du produit)
  brand (Marque du produit)
  category (Catégorie du produit, optionnel)
  description (Description du produit)
  image_path (Nom du fichier de l’image du produit, "watch1.png")
  price (Prix du produit)
  stock (Quantité en stock)
- **Relation** :
  Un **Product** peut être inclus dans plusieurs **Orders** via **Detail_Order**.
  Cardinalité : N:M (Un produit peut être présent dans plusieurs commandes, et une commande peut contenir plusieurs produits).

3. **Entité : Order**

- **Attributs** :
  id (Identifiant unique de la commande)
  total_price (Prix total de la commande)
  date_register (Date et heure d'enregistrement de la commande)
  user_id (Référence à l'ID de l'utilisateur, identifie l'utilisateur ayant passé la commande)
- **Relation** :
  Un **Order** appartient à un **User** et peut contenir plusieurs **Products** via **Detail_Order**.
  Cardinalité : 1:N (Un utilisateur peut passer plusieurs commandes, mais chaque commande appartient à un seul utilisateur).

4. **Entité : order_detail**

- **Attributs** :
  id (Identifiant unique de la ligne de commande)
  order_id (Référence à l'ID de la commande)
  product_id (Référence à l'ID du produit)
  quantity (Quantité du produit dans la commande)
- **Relation** :
  Un **Order_Detail** relie un **Order** à un **Product**.
  Cardinalité : N:M (Une commande peut avoir plusieurs produits et un produit peut être dans plusieurs commandes).

Diagramme des relations :

- **User** (1) — (N) **Order**
  Un utilisateur peut passer plusieurs commandes, mais chaque commande appartient à un seul utilisateur.
- **Order** (N) — (N) **Product** (via **detail_order**)
  Une commande peut contenir plusieurs produits, et un produit peut être présent dans plusieurs commandes. Cela se fait via la table intermédiaire **detail_order**.
- **Product** (N) — (N) **Order** (via **detail_order**)
  Un produit peut être présent dans plusieurs commandes, et une commande peut avoir plusieurs produits, grâce à la table **detail_order**.
- **Contact_Message** : Entité indépendante, sans relation directe avec les autres entités.

"Modèle Physique de Données (MPD) :

```
CREATE TABLE user (
id INT(11) NOT NULL AUTO_INCREMENT, -- Identifiant unique de l'utilisateur
username VARCHAR(255) NOT NULL, -- Nom d'utilisateur (unique)
email VARCHAR(255) NOT NULL, -- Email de l'utilisateur (unique)
password VARCHAR(255) NOT NULL, -- Mot de passe de l'utilisateur
role INT(11) NOT NULL DEFAULT 0, -- 0 pour utilisateur normal, 1 pour administrateur
PRIMARY KEY (id), -- Clé primaire sur l'identifiant de l'utilisateur
UNIQUE KEY username_UNIQUE (username), -- Unique constraint sur le nom d'utilisateur
UNIQUE KEY email_UNIQUE (email) -- Unique constraint sur l'email
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
CREATE TABLE product (
id INT(11) NOT NULL AUTO_INCREMENT, -- Identifiant unique du produit
title VARCHAR(255) NOT NULL, -- Titre du produit
brand VARCHAR(255) NOT NULL, -- Marque du produit
category VARCHAR(255) DEFAULT NULL, -- Catégorie du produit (optionnel)
```

```
description TEXT NOT NULL, -- Description du produit

image_path VARCHAR(255) NOT NULL, -- Nom de l'image du produit (ex: "watch1.png")

price DECIMAL(10, 2) NOT NULL, -- Prix du produit

stock INT(11) NOT NULL, -- Quantité en stock

PRIMARY KEY (id) -- Clé primaire sur l'identifiant du produit

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `order` (

id INT(11) NOT NULL AUTO_INCREMENT, -- Identifiant unique de la commande

total_price DECIMAL(10, 2) NOT NULL, -- Prix total de la commande

date_register TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Date et heure de l'enregistrement de la commande

user_id INT(11) NOT NULL, -- Référence à l'ID de l'utilisateur

PRIMARY KEY (id), -- Clé primaire sur l'identifiant de la commande

FOREIGN KEY (user_id) REFERENCES user(id) -- Clé étrangère liant la commande à un utilisateur

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

CREATE TABLE order_detail (id INT(11) NOT NULL AUTO_INCREMENT, -- Identifiant unique de la ligne de commande

order_id INT(11) NOT NULL, -- Référence à l'ID de la commande

product_id INT(11) NOT NULL, -- Référence à l'ID du produit

quantity INT(11) NOT NULL DEFAULT 1, -- Quantité du produit dans la commande

PRIMARY KEY (id), -- Clé primaire sur l'identifiant de la ligne de commande

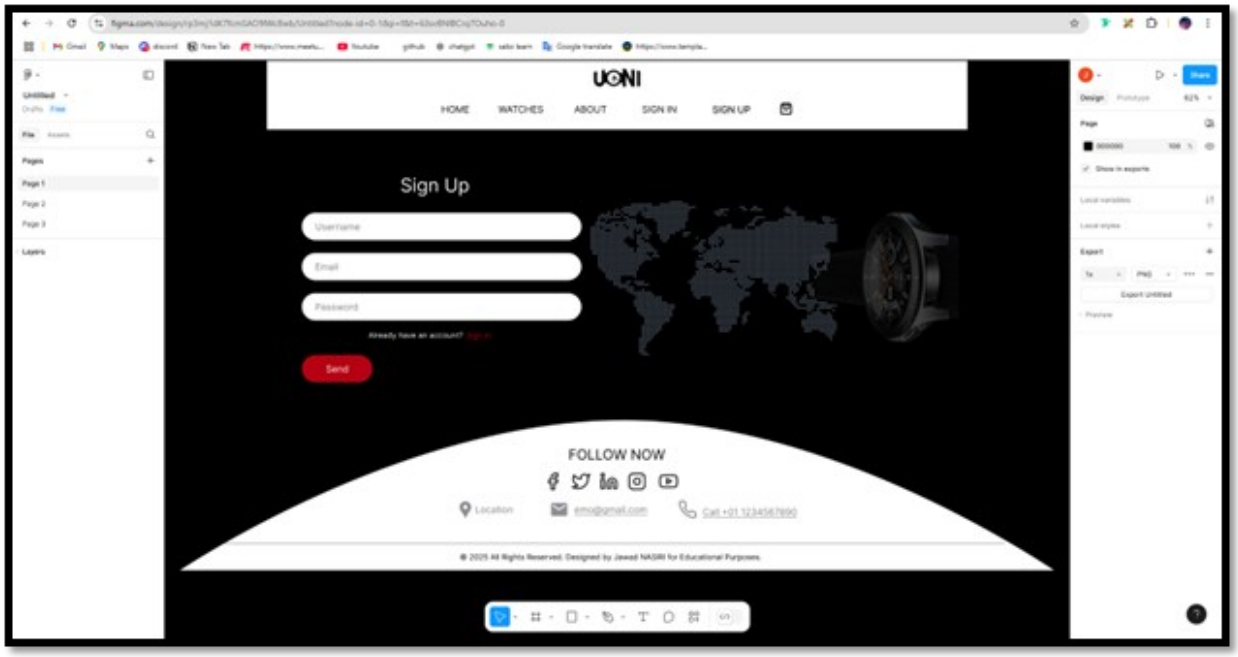
FOREIGN KEY (order_id) REFERENCES `order`(id), -- Clé étrangère liant la ligne de commande à une commande

FOREIGN KEY (product_id) REFERENCES product(id) -- Clé étrangère liant la ligne de commande à un produit

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

Utilisation de Figma pour la conception d'interfaces interactives :

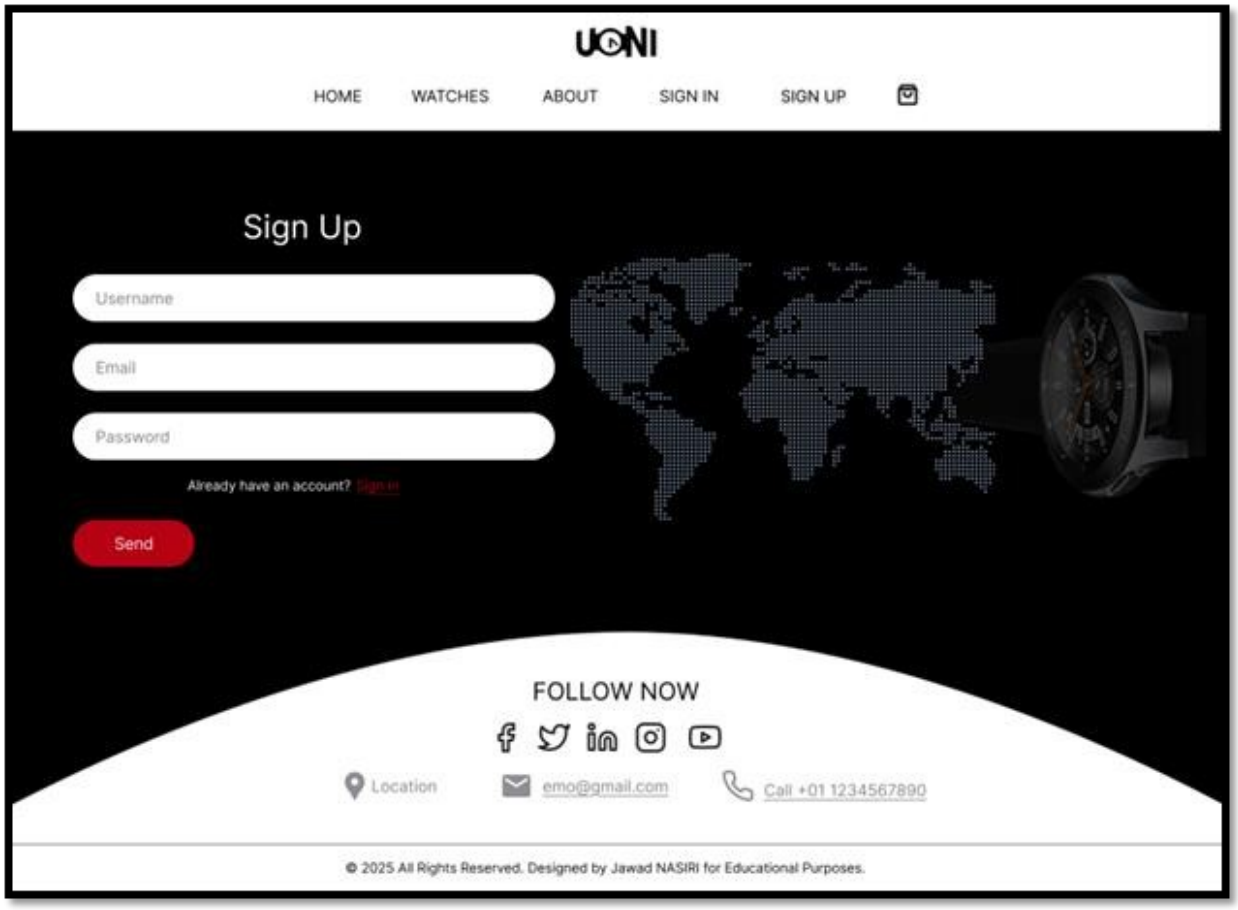
Figma : outil de design utilisé pour créer des maquettes interactives et visuelles de l'interface utilisateur. Il facilite la conception d’interfaces utilisateur.



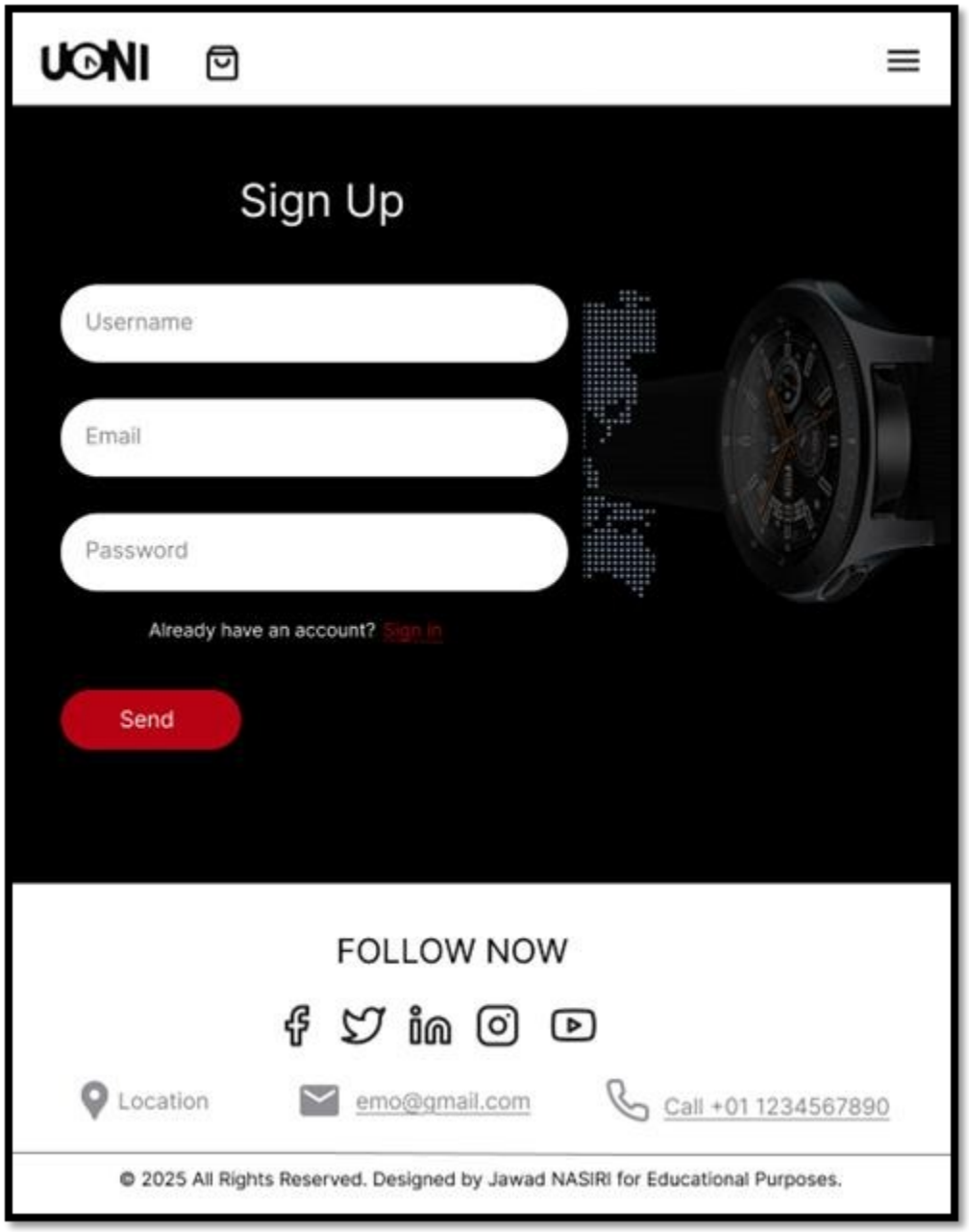
Maquette Figma du Formulaire d'Inscription :

Voici la maquette du formulaire d'inscription réalisée sur Figma, montrant l'interface pour entrer le nom d'utilisateur, l'email et le mot de passe, conçue pour les versions **desktop**, **tablette** et **mobile**, avec un design simple et adapté à chaque format.

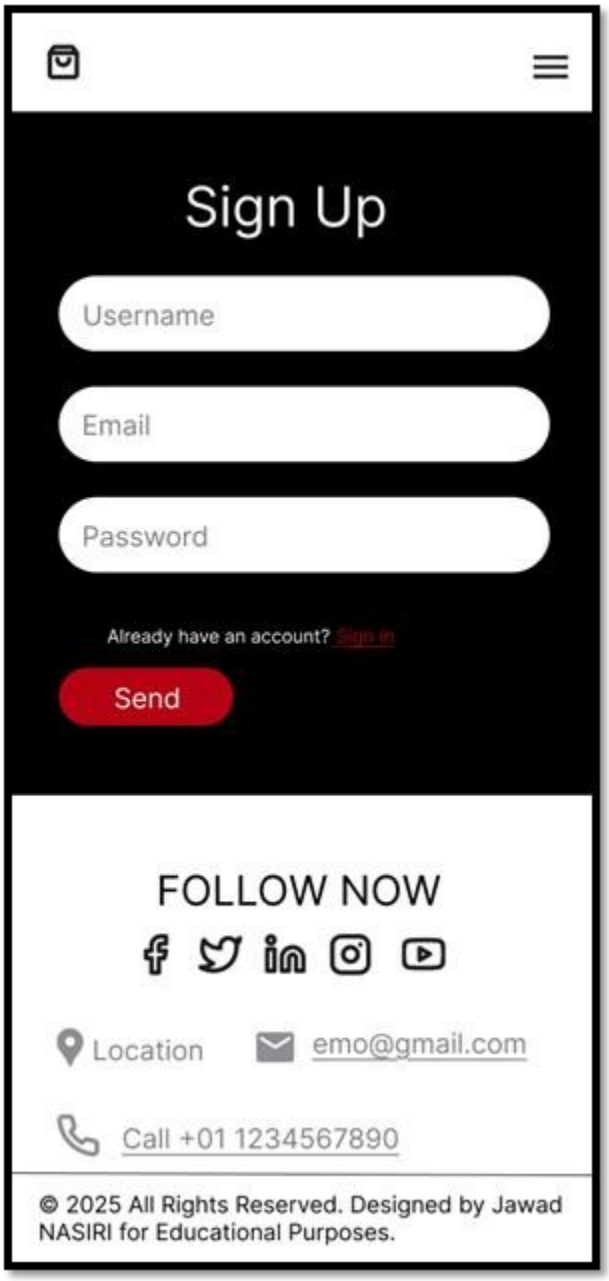
version **desktop :**



version **tablette** **:**



version **Mobile** :



Réalisations Personnelles

1. Dans un projet précédent, j’ai créé un site web de vente de vêtements. J’ai pris en charge la conception et le développement du site en utilisant HTML et CSS pour la structure et le style, JavaScript pour les interactions dynamiques, et PHP (orienté objet) avec MySQL pour gérer la base de données des produits et des utilisateurs. Pour la phase de design, j’ai utilisé Figma afin de réaliser une maquette claire et fonctionnelle. Ce projet m’a permis de renforcer mes compétences en programmation web et en gestion de bases de données, tout en surmontant des défis comme l’intégration fluide entre le front-end et le back-end.

Projet Personnel: Générateur de Mots de Passe

2. J'ai développé une application web permettant de générer des mots de passe sécurisés. Après avoir sélectionné des critères comme la longueur, l’utilisation de majuscules, de minuscules, de chiffres et de symboles, l'utilisateur peut générer un mot de passe aléatoire. L'application permet également de copier le mot de passe en un clic et de visualiser sa force en temps réel. J’ai utilisé HTML et CSS pour l'interface, JavaScript pour gérer les interactions et la génération du mot de passe. Ce projet m’a permis de renforcer mes compétences en développement front-end et en gestion des événements.

Jeu d’essai :

La fonctionnalité la plus intéressante de mon projet est le carrousel de la boutique ("shop carousel"). Ce carrousel affiche des images de montres qui défilent automatiquement sans jamais s’arrêter, et J’ai trouvé cela très intéressant à développer. J’ai utilisé HTML pour structurer les éléments, CSS pour le style, et JavaScript pour le faire tourner. Ce qui rend ce code intéressant, c’est la manière dont j’ai créé un défilement infini en utilisant une *deep copy* (copie profonde) des éléments.

Une *shallow copy* (copie superficielle) aurait juste recopié les références des items, mais avec une *deep copy*, j’ai cloné chaque élément du carrousel (avec cloneNode(true)) pour en faire une vraie copie indépendante. Ensuite, j’ai ajouté ces clones à la liste pour doubler le contenu. Avec JavaScript, je fais défiler le carrousel toutes les 3 secondes en utilisant scrollBy pour avancer d’une largeur d’item. Quand il arrive à la moitié (la fin des originaux), je le ramène au début avec scrollLeft = 0. Comme les clones sont là, on ne voit pas de coupure, et ça tourne en boucle sans arrêt. C’était un vrai défi de comprendre ces notions de copie et de fluidité, et je trouve le résultat visuel hyper satisfaisant.

```
const itemsClone = Array.from(shopCarouselItems).map(item => item.cloneNode(true))
itemsClone.forEach(item => shopCarouselContainer.append(item))

window.addEventListener('resize', function(){
    shopCarouselItemWidth = document.querySelector('.shop-carousel-item').offsetWidth + 1
});

setInterval(() => {
    shopCarouselContainer.scrollBy({
        left: shopCarouselItemWidth,
        behavior: 'smooth'
    });

    if(shopCarouselContainer.scrollLeft >= shopCarouselContainer.scrollWidth / 2){
        shopCarouselContainer.scrollLeft = 0
    }

}, 3000)
```

Veille Technologique :

Pour ce projet, j’ai souvent eu besoin de chercher des réponses à mes questions sur le code. Pour le front-end avec HTML, CSS et JavaScript, je suis allé sur Stack Overflow pour trouver des solutions, surtout pour le carrousel qui défile sans arrêt, et j’ai aussi consulté javascript.info pour des explications claires sur JavaScript. Pour le back-end avec PHP (orienté objet) et MySQL, j’ai lu la documentation officielle sur PHP.net, qui m’a bien aidé à gérer la base de données. J’ai également regardé des vidéos YouTube pour apprendre à utiliser Figma pour mes maquettes. En plus, j’ai suivi des blogs tech comme W3Schools ou Medium pour des astuces et des idées. D’ailleurs, j’ai même échangé avec Grok, une IA créée par xAI, qui m’a donné des pistes utiles. Ces ressources m’ont permis de résoudre mes problèmes et d’avancer efficacement dans le projet.

Problématique

J’ai rencontré un problème avec le carrousel et le responsive design. Les ajustements ne s’appliquaient qu’après quelques secondes, ce qui me faisait penser que mon code était erroné. Après plusieurs tests, j’ai compris que le souci venait du timing entre le défilemen et le recalcul de la taille des éléments. Une fois identifié, j’ai adapté mon approche pour garantir une meilleure fluidité.

Avant : Le carrousel ne s’adaptait pas immédiatement, causant un affichage décalé.

Après : Les éléments s’ajustent correctement, offrant un rendu fluide et responsive.

Conclusion : Une plateforme e-commerce sécurisée et conviviale pour la v de montres

En conclusion, le projet Uni-Watch a pour objectif de proposer une expérience d'achat en ligne fluide, simple et sécurisée. Grâce à l’utilisation de technologies modernes, le site offre une navigation intuitive et permet aux utilisateurs de trouver facilement les montres qu'ils recherchent. Les fonctionnalités essentielles sont bien couvertes, permettant une gestion simple des produits et des commandes. Ce projet représente une base solide pou une plateforme e-commerce réussie, alliant performance, sécurité et expérience utilisateur.