

Site de rencontre LoveLink

partie1

I. Contexte

Vous êtes stagiaire dans l'entreprise LoveLink qui est en phase de création d'un site de rencontre qui vise les jeunes adultes de 18 à 35 ans à la recherche de relations amoureuses ou amicales. Le site s'adresse également aux personnes cherchant à élargir leur cercle social.

Vous intégrez l'équipe de développement qui a déjà commencé la réalisation du site avec les technologies suivantes : HTML, CSS (avec le framework TailwindCSS), MySQL, PHP et PDO pour interagir avec la base de données.

II. Installation du projet

1. Lancer gitBash (installer git si ce n'est pas déjà fait)
2. Se positionner dans votre répertoire web : `cd "c:\wamp64\www\"`
(Il faut mettre des guillemets dans le chemin car gitBash est un terminal linux. En linux les chemins sont avec des slashes et non des anti-slash /c/wamp64/www)
3. Lancer la commande `git clone` qui permet de récupérer les sources en local depuis un repository distant :
`git clone https://github.com/StephanieANDRES/siteRencontre.git`
4. Lancer VScode
5. Dans VSCode, faites File>>Open Folder>> et aller chercher le repository c:\wamp64\www\siteRencontre
6. Copier le contenu du fichier `sql.sql` et le coller dans phpMyAdmin :
<http://localhost/phpmyadmin/index.php>
7. Lancer le site : <http://siteRencontre/>

III. Analyse du code source

Vous venez de récupérer le code source du site internet.

Vous venez de lancer la page d'accueil.

Maintenant votre premier réflexe est de regarder comment est écrit le code.

1. Ouvrir la page `connexion.php`.
2. Expliquer le rôle des attributs HTML suivant : `placeholder`, `required`

.....
.....

3. Expliquer les extraits de code suivants :

Extrait de code	Explications
-----------------	--------------

<pre>if (\$_SERVER['REQUEST_METHOD'] == 'POST') { ... }</pre>	
<p>Que peut contenir la variable \$user ?</p> <pre>\$user = \$stmt->fetch();</pre>	
<pre>if (\$user && password_verify(\$mot_de_passe, \$user['mot_de_passe'])) { \$_SESSION['user_id'] = \$user['id']; header('Location: index.php'); }</pre>	

4. Le formulaire de cette page est-il sécurisé ?
 - a. Si NON, quels sont les risques encourus ?
 - b. Si NON, que devrions nous ajouter pour qu'il le soit ?
.....
 - c. Si NON, mettre en place la mesure de sécurité adéquate.
5. Ouvrir la page `profil.php`
6. Expliquer à quoi sert `htmlspecialchars()` et comment il contribue à sécuriser les données qui vont être affichées dans la page.
.....
.....
7. Dans quelle page est réalisée la connexion à la base de données ?
.....
8. Dans quelle(s) page(s) est réalisée la fermeture de la connexion à la base de données ? Est-ce un problème ?
.....
.....

IV. ♥ Connexion au site de rencontre

1. Lancer le site : <http://siteRencontre/>
2. Essayez de vous connecter avec les identifiants stockés en base de données.
Une erreur devrait apparaître.

Vous allez devoir investiguer pour comprendre d'où vient cette erreur.

Connexion

Se connecter

Identifiants incorrects

© 2024 - Site de Rencontre

3. Vérifier en base de données. Dans quelle table regardez-vous ?

.....

4. Vérifier dans le code source. Dans quelle page regardez-vous ?

.....

5. D'après votre analyse, d'où provient cette erreur ?

.....

.....

*Note : Nous allons **fixer** l'erreur dans la section suivante.*

V. Mise à jour des mots de passe en BDD

Stocker des mots de passe **en clair** expose les utilisateurs à des risques élevés en cas de piratage, car leurs informations d'identification sont directement accessibles.

Cela peut également entraîner une perte de confiance des utilisateurs envers la plateforme et des conséquences légales pour l'entreprise.

Il est donc crucial de procéder au hachage des mots de passe pour protéger la vie privée des utilisateurs et sécuriser leurs comptes.

1. Expliquer pourquoi il est préférable de hasher les mots de passe plutôt que de les chiffrer ?

.....

Pour sécuriser nos mots de passe en clair dans la base de données, **nous avons 2 possibilités** :

2. soit **jouer un script** SQL directement dans phpMyAdmin. Cette méthode n'est pas recommandée car elle est moins sécurisée que les algorithmes spécifiques de PHP.

Pour votre culture, voici la commande SQL que nous aurions pu passer **(NE PAS LA JOUER)**:

```
//Hash en utilisant l'algorithme SHA-2
UPDATE utilisateurs
SET mot_de_passe = SHA2(mot_de_passe, 256);
```

3. soit créer un script PHP qui se connecte à la BDD et hashé tous les mots de passe. C'est ce que nous allons faire dans cette section.

Pour cela,

4. Créer un répertoire **admin** à la racine du site de rencontre.
Ce sera notre répertoire d'administration "backoffice".
5. Créer un fichier **update_passwords.php** qui met à jour le mot de passe de chaque utilisateur. Voici les instructions à coder en PHP :
 - Inclure le fichier **db.php** permettant de se connecter à la bdd
 - Récupérer tous les **id** et les **mots de passe** des utilisateurs
 - Pour chaque utilisateur
 - hasher le mot de passe avec la commande php :
`password_hash($mot_de_passe_en_clair, PASSWORD_DEFAULT);`
 - Mettre à jour le mot de passe pour l'utilisateur en BDD
 - Afficher un message qui spécifie que tous les mots de passe ont été hashés avec succès.
6. Lancer le script http://siteRencontre/admin/update_passwords.php

Résultats attendus :

Le script affiche un message OK

← → ↺ initiationphp/siteRencontre/admin/update_passwords.php

Mots de passe mis à jour avec succès !

En BDD, les mots de passe sont hashés

pseudo	email	mot_de_passe
jean_01	jean01@example.com	\$2y\$10\$aG9HweRGbPK3pLsEQscQ.YwDCE3OIWTFIX9EztzMzr...
marie_02	marie02@example.com	\$2y\$10\$Lc6y.U2QZITxLZQxiJMri.T5OntxYqQUZ6EZJiZiNOQ...

7. **Renommer** la page **hash_passwords.php** en **hash_passwords.php.old** pour ne pas rejouer le script par inadvertance.

VI. La messagerie

1. Se logger avec jean01@example.com : mdp1
2. Envoyer un message à Marie_02
3. Se logger avec marie02@example.com : mdp2
4. Envoyer un message à Jean
5. Se logger à nouveau avec Jean et aller dans la messagerie.

Messagerie

marie_02

Votre message

Envoyer

Vos messages

Moi: hello

Autre: bonjour qui êtes vous ?

© 2024 - Site de Rencontre

Que remarquez-vous ?

.....

6. Contrôler les informations en BDD.
 - a. Quelle table regardez-vous ?
 - b. Est-ce que l'on est capable de savoir QUI à envoyé un message à QUI dans cette conception (bdd) ?
7. Modifications nécessaires pour afficher le nom de l'interlocuteur au lieu de "Autre" :
 - a. Dans la page `messagerie.php`, modifier la requête qui permet de récupérer tous les messages en ajoutant une jointure vers la table utilisateurs (ainsi nous pourrons avoir le nom de l'expéditeur du message).
 - b. Au moment de l'affichage, remplacer le "Autre" par le pseudo de l'expéditeur du message (issu de la requête que nous venons de modifier).
8. Se logger avec Marie
9. Aller dans la messagerie

Résultat attendu :

Messagerie

jean_01

Votre message

Envoyer

Vos messages

jean_01: hello

Moi: bonjour qui êtes vous ?

© 2024 - Site de Rencontre

10. Envoyer un message à Paul_03.

Messagerie

jean_01

Votre message

Vos messages

jean_01: hello

Moi: bonjour qui êtes vous ?

Moi: Bonjour Paul !

Que s'est-il passé ?

Lorsque l'on envoie des messages à des personnes différentes, tout est mélangé dans la même fenêtre de messagerie.

Ce n'est pas le comportement désiré. Nous souhaitons que seuls les messages entre moi-même et l'utilisateur de la liste déroulante soient affichés dans la fenêtre de messagerie.

11. Quelle est l'URL de la page sur laquelle vous êtes ?
12. L'URL contient-elle des paramètres récupérables en GET ?
13. Revenir sur l'accueil et cliquer sur "Voir le profil" de Paul Lemoine.
14. Quels sont les paramètres de l'URL ?
15. Cliquer sur "Envoyer un message". Quels sont les paramètres de l'URL ?
.....
16. Que pouvez-vous en conclure ?
.....
.....

Nous allons maintenant récupérer le paramètre `destinataire_id` afin de n'afficher que les échanges de message entre *lui* et *moi*.

17. Nous avons remarqué que parfois la page `messagerie.php` est chargée avec des paramètres en `GET`, parfois non (selon si on passe par le menu messagerie ou par le profil d'un utilisateur).

Dans la page `messagerie.php`, tester l'existence de `$_GET["destinataire_id"]` :

```

Si existe ($_GET["destinataire_id"]) alors
    $destinataire_id = $_GET["destinataire_id"];
}sinon
    $destinataire_id = null;
}

```

18. Nous devons maintenant modifier la requête si nous avons un `destinataire_id` qui est renseigné. Sinon nous exécuterons la requête telle qu'elle est définie actuellement.

La nouvelle requête doit sélectionner tous les messages ayant pour expéditeur *moi* ET comme destinataire *lui* OU pour expéditeur *lui* ET pour destinataire *moi*. C'est une requête à 4 paramètres.

Voici les instructions que l'on devrait retrouver dans votre code :

```

Si ($_GET["destinataire_id"]==null) alors
    exécuter la requête de base
}sinon
    exécuter la nouvelle requête
}

```

Résultat attendu :

Dans cet exemple, nous sommes logués avec *Marie* et nous communiquons avec *Paul*. Les seuls messages que nous voyons, ce sont nos échanges avec *Paul*.

Messagerie

jean_01

Votre message

Envoyer

Vos messages

Moi: Bonjour Paul !

paul_03: Bonjour Marie

19. Nous remarquons que, bien que nous communiquons avec Paul, c'est Jean_01 qui est affiché dans la liste déroulante et je dois à chaque fois changer pour communiquer avec Paul.

Pourquoi est-ce Jean_01 qui est affiché ?

20. Nous allons maintenant faire en sorte que si le `destinataire_id` est renseigné, la liste déroulante soit automatiquement affichée avec le bon destinataire.

Pour cela, nous allons utiliser un attribut HTML `selected = "selected"` qui permet de sélectionner une ligne dans la liste. Cet attribut est à positionner sur la balise `<option>`.

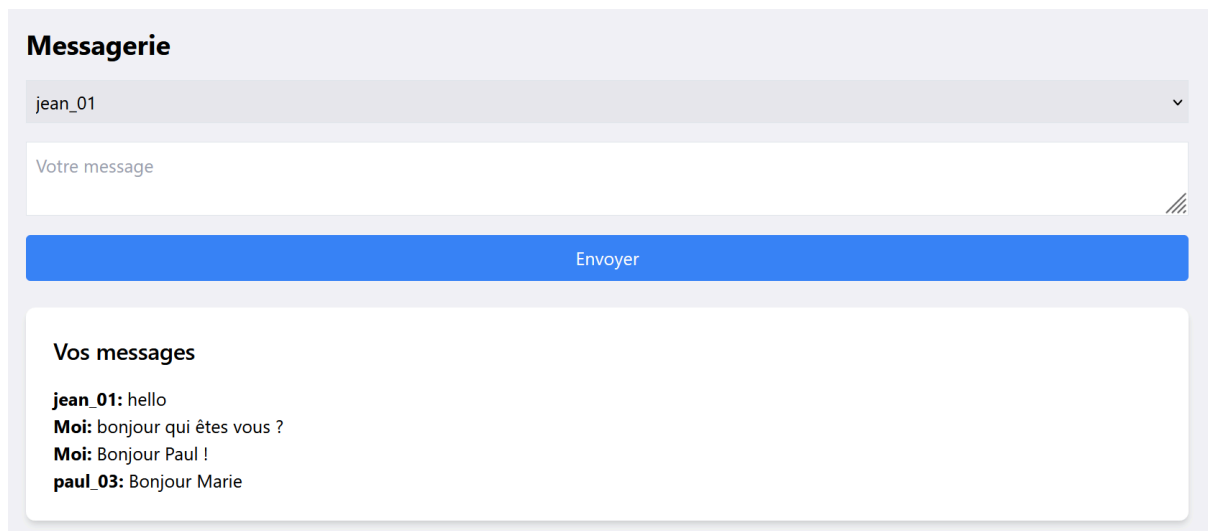
Lorsque nous parcourons la liste des utilisateurs (avec le `while`), nous allons rajouter une condition :

Si l'utilisateur que je suis en train de parcourir correspond à mon interlocuteur (`destinataire_id`) alors j'ajoute l'attribut `selected` sur l'`<option>` de cet utilisateur.

Résultat attendu :

Lorsque je consulte un profil, que je clique sur "envoyer un message", le pseudo du destinataire que j'ai choisi est sélectionné par défaut dans la liste déroulante.

21. Retourner dans la messagerie via le menu (en haut à droite). Remarquez qu'il n'y a pas d'erreur mais nous avons toujours la liste de tous les messages échangés, tous utilisateurs confondus :



C'est normal puisque nous sommes dans le cas où nous n'avons pas de `destinataire_id` dans les paramètres en GET donc nous chargeons tous les messages dont je suis *expéditeur* ou *destinataire*.

Pour le moment nous acceptons ce comportement.

VII. 🦄 Le profil

1. Le concepteur de l'application a fait le choix de stocker l'âge des utilisateurs. Était-ce une bonne idée ? Justifiez.

.....
.....

2. Nous souhaitons remplacer la donnée calculée "âge" par la date de naissance. Avant de modifier la base de données, nous devons prendre en compte le fait qu'il y ait déjà des utilisateurs dans la base de données avec des âges renseignés. Comment pourrions-nous faire pour ne pas perdre l'âge de chaque utilisateur ?

.....
.....

Faire valider votre proposition par le professeur. Et quand vous avez le GO, implémentez-la.

3. Dans la page profil.php, afficher le *pseudo*, l'*âge*, le *genre*, la *préférence*, la *bio*, la *localisation* et le *nombre de jours depuis quand l'utilisateur est inscrit*. Supprimer le nom, prénom et email de l'affichage.

VIII. Pour aller plus loin

1. Rajouter la notion de message **lu/non lu** en base de données.
 - a. Pourquoi pas une colonne booléenne ? Dans le jargon du développeur, nous appelons ça un flag. 0 pour non lu, 1 pour lu.
2. Implémenter la fonctionnalité messages **lu/non lu** dans la page messagerie.
 - a. Quand j'envoie un message par défaut le message est non lu (flag à 0)
 - b. Quand le destinataire lit le message, le flag passe à 1
3. S'il y a des messages non lus, afficher le nombre de messages non-lus sur les profils ayant envoyés des messages sur la page index.php