

# Rapport WE4A - The Social Network

POURCINE Mattéo, BONNET Rémi

May 9, 2024



Figure 1: Logo de notre réseau social

---

## Contents

<b>1 Avant-propos / Informations utiles</b>	<b>3</b>
1.1 Compte administrateur : . . . . .	3
1.2 Compte utilisateur . . . . .	3
1.3 Base de données . . . . .	3
<b>2 Organisation de notre travail</b>	<b>3</b>
<b>3 Utilisateurs connectés/déconnectés</b>	<b>4</b>
3.1 Utilisateurs déconnectés . . . . .	4
3.2 Utilisateurs connectés . . . . .	5
3.3 Côté backend . . . . .	5
<b>4 S'inscrire / se connecter</b>	<b>5</b>
4.1 Se connecter . . . . .	5
4.1.1 Vérification des données . . . . .	5
4.2 S'inscrire . . . . .	7
<b>5 Découvrir des posts/utilisateurs</b>	<b>7</b>

---

# 1 Avant-propos / Informations utiles

Avant de commencer vous trouverez ici toutes les informations importantes pour pouvoir tester et voir toutes les fonctionnalités de notre réseau social.

## 1.1 Compte administrateur :

Ce compte est l'unique compte "admin". Si celui-ci viendrait à disparaître. Vous pouvez allez dans la base de données, prendre n'importe quel utilisateur et "UPDATE" la colonne admin avec un booléen égal à 1. ("UPDATE user SET admin = 1 WHERE id = 40" par exemple).

Email: admin@tzu.com

Mot de passe : admintzu1234

## 1.2 Compte utilisateur

Pour vous permettre de tester les fonctionnalités "signin - signup". Vous pouvez soit créer un compte en cliquant sur "Not Register" dans le popup signin ( celui-ci s'affiche quand on clique sur l'icône "utilisateur" dans la navbar). Soit vous identifiez directement par ce même popup après avoir créé un compte ou en utilisant les identifiants ci-dessous.

Email: glenmorton5555@gmail.com

Mot de passe: 1234

**Vous trouverez en annexe les mails/mots de passe de tous les utilisateurs inscrit.**

## 1.3 Base de données

La base de données se trouve à la base du dossier du projet **./socialnetworkBONNETPOURCINE.sql**. Pour pouvoir vous connectez, utilisez votre logiciel de gestion de serveur web local (XAMMP fonctionne très bien). Créez une nouvelle base de données et importez celle fournit dans notre projet.

Il ne vous reste plus qu'à changer les informations dans le fichier **./model/database.php**.

```
$db = new PDO("mysql:host=localhost;dbname=socialnetwork", "root", "");  
$con = mysqli_connect("localhost", "root", "", "socialnetwork");
```

Figure 2: Connexion à la base de données

Remplacez "socialnetwork" par le nom que vous avez choisi lors de la création de la base de données.

Bien, maintenant que toutes ces informations sont entre vos mains, le site devrait être prêt pour quelques tests.

Dans les prochaines parties, nous vous expliquerons en détails le fonctionnement de toutes les features.

# 2 Organisation de notre travail

Le travail demandé étant conséquent. Nous avons dû nous organiser pour prévenir à d'éventuelles difficultés. Notre projet s'organise autour de 7 dossiers. Nous avons essayé d'être le plus clair

---

possible concernant le nommage des fichiers pour éviter toutes ambiguïtés.

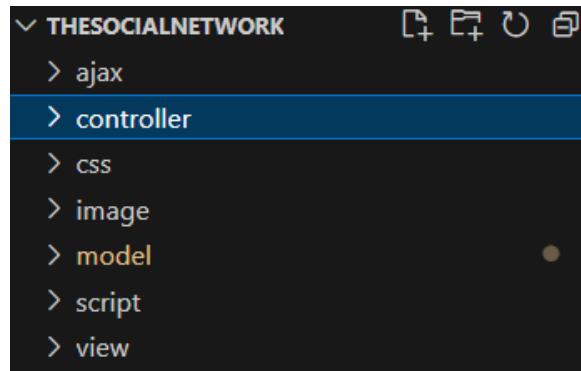


Figure 3: Arborescence du projet

Les noms étant assez explicites, nous allons expliquer leur rôle de manière non-exhaustive.

- **Dossier View :**

Ce dossier regroupe tous les fichiers qui affichent un modules sur le site.

- **Dossier script :**

Celui-ci regroupe tous les **scripts** javascript servant à rendre le site plus agréables pour l'utilisateur. On y retrouve par exemple les fichiers servant à envoyer des requêtes AJAX.

- **Dossier Model :**

Ce dossier regroupe tous les fichiers servant aux traitements des données dans la BDD.

- **Dossier Image :**

Celui-ci stocke les images des posts, des avatars des utilisateurs et des images des commentaires. Les images ne sont pas stockées dans la BDD mais bien en local.

- **Dossier Ajax :**

Ce dossier traite toutes les requêtes **AJAX** côté BDD.

Concernant le **versionning** de notre code. Nous avons choisi **Git**. Vous pouvez trouver le lien de notre répertoire **Github** sur tous les pieds de pages.

### 3 Utilisateurs connectés/déconnectés

Le but de cette partie est d'expliquer dans les grandes lignes la différence entre les droits accordés aux utilisateurs connectés et ceux non connectés.

#### 3.1 Utilisateurs déconnectés

Un utilisateur qui arriverait sur le site pour la première fois pourrait entre autres:

1. Rechercher dans la barre de recherche un **utilisateur** ou un **post** mais sans interagir avec ceux-ci. Impossibilité de **liker** et **commenter** un post et impossibilité de **suivre** ni même de voir les **followers** ou les **following** d'un utilisateur.

- 
2. Les posts marqués comme "**sensible**" ne peuvent pas être défloutés. Un badge précise qu'il faut avoir un compte et être connecté pour pouvoir faire cette action sur les **posts concernés**.
  3. Les **commentaires** de chaque post peuvent être **consultés** mais pas likés.

### 3.2 Utilisateurs connectés

Les utilisateurs connectés ont accès aux fonctionnalités de la liste **3.1**. En plus de cela, sur sa page, un utilisateur peut changer ses informations personnelles et consulter les statistiques de son compte.

### 3.3 Côté backend

Côté backend, on le verra plus en détails dans l'explication de chaque feature. Mais le principal pour la connexion d'un utilisateur est géré via **des variables de session**.

C'est-à-dire qu'à la connexion ou l'inscription, certaines données sont stockées dans une variable **SESSION**. Comme par exemple `$_SESSION["id_user"]`. C'est cette variable qui permet de donner les accès vus en **3.1** et **3.2**.

La variable `$_SESSION["admin"]` permet de donner les accès à l'**administrateur** du réseau social. Il peut flouter et supprimer un post jugé trop violent. Il peut également envoyer un **avertissement** et **bannir** un utilisateur.

## 4 S'inscrire / se connecter

Le centre névralgique de notre site est la **navbar** située dans le fichier `./view/navbar.php`. Celle-ci s'affiche sur toutes les pages de notre site sauf pour les pages `./model/undelete_page.php` et `./model/unblur_page.php` uniquement disponibles pour l'administrateur.

### 4.1 Se connecter

Le backend et le frontend de l'inscription se font dans les fichiers suivant :

- `./view/popup_signin.php` (définit les "**divs**" nécessaires au pop-up d'inscription).
- `./view/form_signin.html` (définit le formulaire pour l'inscription).
- `./model/signin.php` (vérifie les données des "**input**" rentrées par les utilisateurs: email + mot de passe)
- `./css/signin.css` (fichier css pour le frontend du signin).

#### 4.1.1 Vérification des données

Dans cette partie nous allons expliquer en détails le fichier `./model/signin.php`.

L'utilisateur arrive sur le site et clique sur l'icône utilisateur de la navbar. Un popup apparaît (figure 3).

Il rentre ses informations et clique sur "join". Les données sont envoyés grâce à la méthode **POST**. Côté backend, on vérifie si la variable `$_POST` n'est pas vide avec la fonction `empty()`. Ensuite on **sécurise** les inputs pour s'abstenir des injections SQL et autres fantaisies.

On vérifie également que l'email est bien au format attendu même si le "type" de l'input dans le

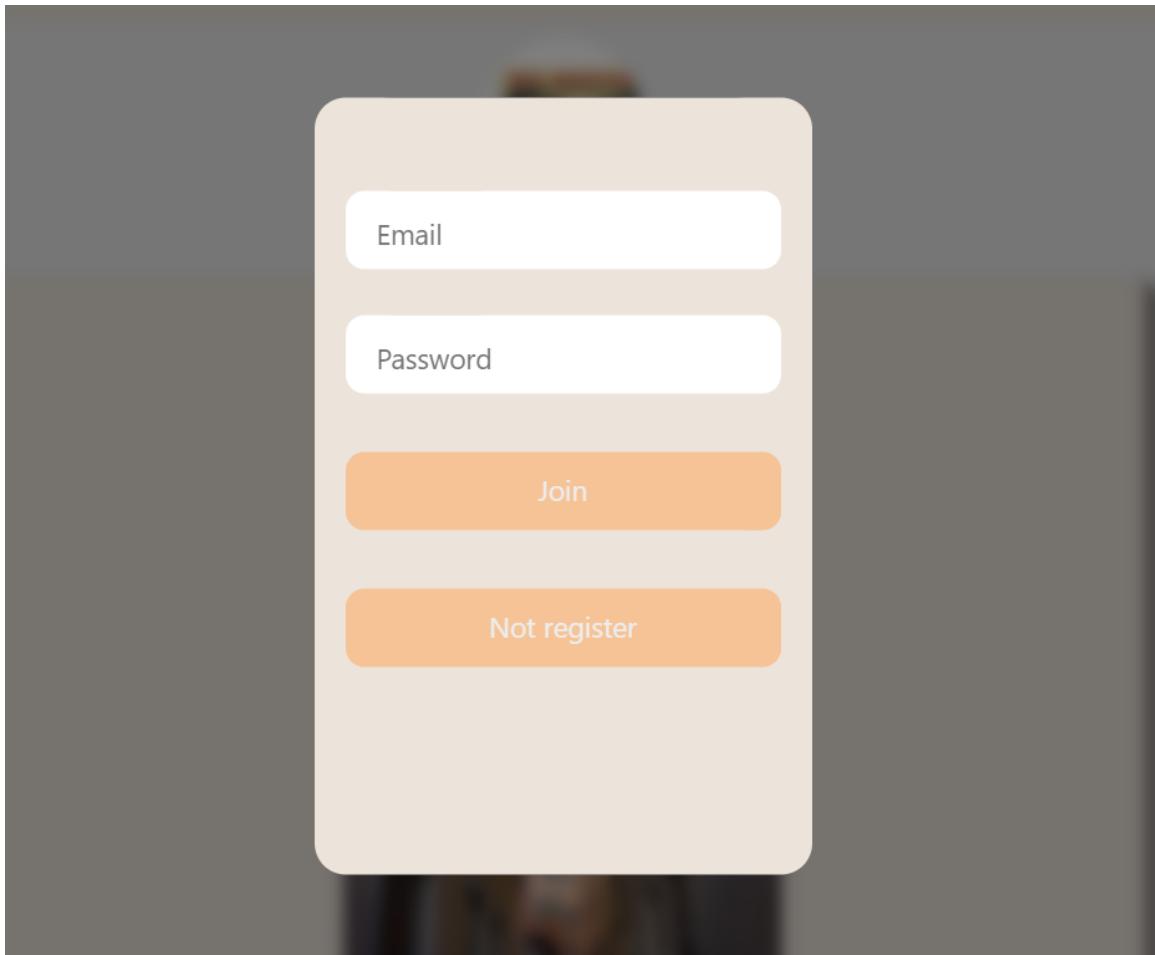


Figure 4: Pop-up de connexion

formulaire est "email". Si ce n'est pas le cas on envoie une **alert** javascript pour prévenir l'utilisateur.

Une fois que les données sont prêtes à être exploitées, on effectue une requête à la base de donnée pour voir si l'utilisateur existe. ("SELECT \* FROM user WHERE email = \$email"). Si c'est le cas, on vérifie si le "password" est le même. Celui-ci étant "hash" avec la fonction **password\_hash()**, on utilise une fonction **password\_verify()** pour le comparer au "hash" de la BDD.

Si le mot de passe correspond il ne reste plus qu'à faire une seul vérification.

Un utilisateur pouvant être **banni** un nombre de jour choisit entre 0 et 100 par l'administrateur. On se **doit de vérifier à chaque connexion** la durée restante du "ban" de l'utilisateur. Si celui-ci est concerné, on se doit de le débannir si la durée est arrivée à échéance.

Une fois toutes ces étapes effectuées. On "set" **les variables de session** utiles au bon fonctionnement du site. Puis on redirige l'utilisateur vers sa page personnelle. La navbar s'adapte quand l'utilisateur est connecté.



Figure 5: Navbar après connexion, les notifications seront alors accessibles sur toutes les pages

## 4.2 S'inscrire

S'inscrire s'inscrit dans la même dynamique que la connexion. Des informations sont demandées en plus comme une photo de profile qui **n'est pas obligatoire**, une date d'anniversaire, un nom et un prénom.

Si aucune photo est choisie une couleur est générée aléatoirement. Un macaron avec la première lettre du prénom sera alors afficher à la place de la photo de profile.

La gestion du formulaire se situe sur une page à part entière `./model/signup.php`.

## 5 Découvrir des posts/utilisateurs

La navbar se compose également d'une barre de recherche. Celle-ci permet de rechercher des utilisateurs et des posts en même temps. Deux sections vont alors s'afficher lors de la recherche.

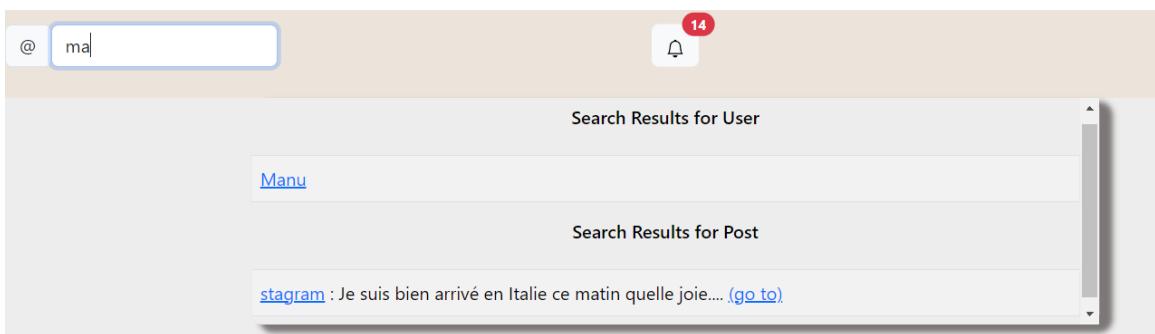


Figure 6: Recherche

Cette recherche est complètement gérée via des **requêtes Ajax**. La requête est créée dans le fichier `./script/ajax_search.js`. Lorsque l'utilisateur **frappe une touche de son clavier**. Une requête est envoyée au fichier `./ajax/livesearch.php`.

Ce fichier fait le traitement dans la BDD pour **rechercher des utilisateurs** dont le pseudo (de manière analogue pour les posts mais cette fois ci sur le "content" du post) est composé des lettres rentrées par l'utilisateur dans la barre de recherche. Si il y a une correspondance, on crée deux tables avec les pseudos et les posts correspondants.

Lorsque la requête Ajax reçoit le résultat du fichier php. Elle change la **propriété** de la div "wrap" pour la rendre **visible** puis elle ajoute à la div "searchresult" contenue dans la div "wrap" les deux tables générées lors du traitement des données de la requête Ajax.

Dans le cas où il n'y a pas d'utilisateurs ni de posts **correspondants**. Le résultat de la requête avertira l'utilisateur qu'il n'y a pas d'utilisateurs ou de posts trouvés.