Gestion des objets perdus sur un campus universitaire.

On veut implémenter une version minimale d'une application web qui nous permettra de trouver des objet que sont dans une base de donnes, on utilisera SQLite, Bootstrap et les templates HTML et d'autres système dans notre code. Les spécifications retenues sont les suivantes.

1. ***Modèle de données.***

On utilise la librairie cookie-session pour gérer des sessions par le biais du stockage d'un identifiant unique pour chaque session dans un cookie sur la machine. On ne stocke que l'identifiant de session dans le cookie, le reste des données de session.

a) ***Tables***

I. Une table **Etablissement**, clé primaire auto- incrémentée colonne **id**, une colonne **Name**, et une colonne **adresse** la classe établissement cette table est remplies en amont

II. Une table **User** pour gérer les utilisateur de notre site avec une clé primaire auto- incrémentée colonne **id**, une colonne image et une référencer **Etablissemet_id** de la table Etablissement

III. Une table **OBJET**, clé primaire auto- incrémentée colonne **id**, une colonne avec le nom de l'objet, colonne avec la photo de l'objet ,une colonne **Etablissemet_id** pour le lieu l'établissement où l'objet a été trouver, **id_user** !

IV. Une Table **PassWord** avec une clé primaire auto- incrémentée colonne **id**, une colonne **id_user** en référence à l'utilisateur et une colonne **password.**

V. Une Table **administrateur** avec une clé primaire auto- incrémentée colonne id, un **username**, **password**

b) ***Fonction d'accès***

I. **login**(username, password) : retourne la clé id associée à l'utilisateur dans la table user si l'utilisateur existe et si le mot de passe correspond à celui enregistré; retourne-1 sinon.

II. **Add_admi**(username, password) :ajoute un administrateur à la base de donné

III. **user_exists**(username) : retourne True si l'utilisateur existe et False sinon.

IV. **new_user**(username, password) : si l'utilisateur existe déjà retourne-1; sinon ajoute un utilisateur et retourne la clé id associée dans la table user !

V. **add_objet**(name,) : qui permet d'ajout un objet dans la base de donné !

VI. **delete_objet**(idobj) : qui supprimer l'objet de la base de donné

VII. **liste_objet**(idobj) : qui retourné les liste des objet perdu par établissement où celui-ci a été trouver

VIII. **detail_objet**(idobj) : qui donne les détaille de l'objet trouver !

IX. **update_objet**(name,url_imag,idEt) : qui mettra à jour notre les information de l'objet.

X. **Super_utilisateur(password)** :qui vérifie si l'utilisateur est un administrateur !

2. ***Vues***

(a) Une en-tête **header** utilisé sur toutes les pages, avec notamment une barre de navigation. Si l'utilisateur est connecté à son compte personnel, la barre de navigation affiche un lien vers le profil de l'utilisateur, un lien pour faire une nouvelle demande d'ami et un lien pour se déconnecter de son compte. Si l'utilisateur n'est pas connecté, la barre de navigation affiche un lien pour se connecter à un compte existant et un lien pour créer un nouveau compte

(b) Un bas de page **footer**, utilisé sur toute les pages

(c) Une page d'accueil **accueille**, affichant un message de bienvenue et une image
(d) Une page **login**, contenant un formulaire pour se connecter à l'aide de son nom d'utilisateur et de son mot de passe.
(e) Une page **new_user,** contenant un formulaire pour enregistrer les information du user
(f)
(g) Une page **creat_objet** qui contiendra un formulaire pour enregistrer les information de l'objet
(h) Une page **index** qui est la page principale du site qui contiendra quelque image objet perdue.
(i) Une page **détails** qui contiens tout détaille de l'objet choisit
(j) Une page **liste** qui contiendra la liste des objet perdu par établissement et un lien retour ver la page principal.
(k) Une page **update_objet** qui contiendra un formulaire pour mettre à jour les informations d'un objet

Une page **administrateur** qui te permet de créer comme un administrateur

3. ***Route et logique de contrôle***

(a) Route **GET /** qui affiche la page accueil
(b) Sur les routes **GET /login, GET /new_user** on affiche respectivement les pages de connexion et de création de compte
(c) La route **POST /login** essaye de connecter l'utilisateur à partir de l'identifiant et du mot de passe fournis et redirige l'utilisateur vers la page index si la connexion réussit (c'est à dire si l'identifiant et le mot de passe correspondent à un utilisateur enregistré sur le site et t'affiche certain bouton en plus pour un administrateur) et vers la route **GET /login** sinon
(d) La route **POST /new_user** essaye de créer un nouveau compte avec l'identifiant et le mot de passe fournis et connecte et redirige l'utilisateur vers la route **GET/login** (c'est à dire si le nom d'utilisateur n'est pas déjà pris) et vers la route **GET /new_user** sinon, avec le message « *User already exists* ».
(e) La route **GET /admis** qui te redirige vers la page **administrateur**
(f) La route **GET /logout** déconnecte l'utilisateur et le redirige vers la route **GET /**
(g) La route **GET /liste** qui dirige l'utilisateur vers la page liste.
(h) La route **GET /Détail/<id>** qui dirige l'utilisateur vers page détail
(i) La route **GET /remove_obj/<id>** qui permettra de supprimer un objet de la liste
(j) La route **GET/ remove_user/<id>** qui permettra de supprimer un utilisateur