Partie 1 : Technologies utilisées et justification des choix

1. **HTML** :
    - **Rôle** : HTML est utilisé pour structurer la page web.
    - **Justification du choix** : HTML est la norme de base pour créer des pages web. Il permet de définir la structure du contenu, comme les titres, les paragraphes, les images, etc. Il est essentiel pour l'accessibilité et pour le référencement SEO.
    - **Alternative** : D'autres formats tels que **JSP** ou **ASP.NET** pourraient être utilisés, mais **HTML** est plus largement compatible et standardisé.
2. **CSS** :
    - **Rôle** : CSS est utilisé pour le style et la mise en forme des éléments HTML (polices, couleurs, marges, etc.).
    - **Justification du choix** : CSS est la technologie standard pour le design des pages web. Il permet de séparer la présentation visuelle du contenu, améliorant ainsi la maintenabilité du code et la performance en chargeant les styles une seule fois.
    - **Alternative** : Des frameworks comme **SASS** ou **Bootstrap** auraient pu être utilisés pour améliorer la gestion des styles, mais CSS est suffisant pour des sites plus simples.
3. **JavaScript (JS)** :
    - **Rôle** : JavaScript est utilisé pour rendre le menu interactif et d'autres éléments dynamiques.
    - **Justification du choix** : JavaScript permet de rendre les interfaces utilisateur plus interactives sans nécessiter de rechargement complet de la page. C'est la seule technologie acceptée de manière native sur tous les navigateurs pour la manipulation côté client.
    - **Alternative** : Une technologie comme **jQuery** pourrait être utilisée pour simplifier certaines interactions, mais avec les améliorations d'ES6, JavaScript pur est suffisant et léger.
4. **PHP (Back-End)** :
    - **Rôle** : PHP gère la logique serveur, les requêtes à la base de données, et la génération dynamique du contenu.
    - **Justification du choix** : PHP est un langage largement utilisé pour le développement de sites web dynamiques. Il est particulièrement adapté à l'intégration avec des bases de données MySQL via **phpMyAdmin**. PHP est également compatible avec la plupart des hébergements web et offre une courbe d'apprentissage raisonnable.
    - **Alternative** : D'autres langages back-end comme **Node.js** ou **Python (Django)** auraient pu être utilisés. Cependant, PHP est plus adapté pour les projets nécessitant une intégration facile avec des bases de données MySQL et est largement supporté par les plateformes d'hébergement.
5. **phpMyAdmin (Base de données)** :
    - **Rôle** : phpMyAdmin est utilisé pour gérer les bases de données MySQL.
    - **Justification du choix** : phpMyAdmin est un outil web puissant et facile d'utilisation pour gérer les bases de données MySQL. Il permet d'effectuer des tâches comme la création de tables, l'insertion de données, et la gestion des utilisateurs sans avoir besoin de taper des commandes SQL.

- **Alternative** : Des outils en ligne de commande comme **MySQL CLI** ou des interfaces comme **Adminer** auraient pu être utilisés, mais phpMyAdmin est plus visuel et accessible.

Partie 2 : Dispositions prises pour la sécurité

1. **Sécurisation des données utilisateurs** :
   - **SQL Injection** : Utilisation de requêtes préparées pour éviter les attaques par injection SQL. Les données envoyées via des formulaires ou des URL sont toujours validées et filtrées avant d'être traitées.
   -
     - **Exemple** :

```
$stmt = $pdo->prepare('SELECT * FROM users WHERE email = :email');
$stmt->bindParam(':email', $email);
$stmt->execute();
```

- **Validation des données** : Mise en place de la validation des données côté serveur (PHP) pour s'assurer que les entrées utilisateur respectent les règles établies (taille, format, etc.). Côté client, **JavaScript** est utilisé pour une validation rapide, mais ne remplace pas la validation côté serveur.

Protection contre les failles XSS (Cross-Site Scripting) :

- **Échappement des sorties** : Toutes les données envoyées par l'utilisateur et affichées sur le site sont échappées avant d'être incluses dans le code HTML, afin d'éviter l'injection de code malveillant.
  - **Exemple** :

```
echo htmlspecialchars($userInput, ENT_QUOTES, 'UTF-8');
```

1. **Sécurisation des sessions** :
   - **Utilisation de sessions sécurisées** pour gérer les connexions utilisateurs. Un identifiant de session unique et sécurisé est généré lors de chaque connexion et régénéré régulièrement pour éviter les attaques de type session hijacking.
   - **Utilisation de HTTPS** pour sécuriser toutes les communications entre le client et le serveur, en chiffrant les données échangées (en particulier les identifiants et mots de passe).
2. **Sécurisation de la base de données** :
   - **Permissions des utilisateurs MySQL** : Chaque utilisateur de la base de données a des droits limités en fonction des besoins. Par exemple, l'utilisateur chargé de gérer les données utilisateurs n'a que des droits de lecture/écriture sur certaines tables, mais pas sur celles contenant des informations sensibles.
   - **Hashage des mots de passe** : Les mots de passe sont **hachés** avant d'être stockés dans la base de données, en utilisant des fonctions comme password_hash() en PHP pour garantir que les mots de passe ne soient jamais stockés en clair.

Partie 3 : Justification des choix techniques et points de vue sur les différentes technologies

1. **Choix de PHP** :
    - **Avantages** : PHP est une technologie mature, facile à déployer sur de nombreux hébergements web à bas coût. Il offre une documentation riche et une grande communauté, ce qui permet une résolution rapide des problèmes. Son intégration avec des bases de données MySQL est fluide et il est bien adapté à des projets de petite à moyenne envergure.
    - **Limites** : PHP peut manquer de certaines fonctionnalités modernes proposées par des langages comme **Node.js** (exécution asynchrone native) ou **Python** (extensibilité avec de nombreux frameworks).
2. **Choix de MySQL et phpMyAdmin** :
    - **Avantages** : MySQL est l'une des bases de données relationnelles les plus utilisées dans le monde. Elle est rapide, fiable, et supporte de grosses charges. phpMyAdmin est un excellent outil pour gérer MySQL de façon intuitive via une interface web. Cela facilite le déploiement et la gestion des données sans avoir à taper des commandes SQL.
    - **Limites** : Pour des projets plus complexes ou nécessitant une scalabilité plus importante, des bases de données comme **PostgreSQL** (qui offre une meilleure gestion des relations et des transactions complexes) pourraient être un meilleur choix.
3. **Point de vue sur JavaScript pour l'interactivité** :
    - **Avantages** : JavaScript est le langage standard pour ajouter de l'interactivité à une page web. Il est supporté par tous les navigateurs et dispose de nombreuses bibliothèques et frameworks qui facilitent son utilisation (comme **React.js** ou **Vue.js**). L'utilisation de JavaScript natif permet de maintenir une légère empreinte et un contrôle total sur le code.
    - **Limites** : Si le projet devient plus complexe, il pourrait être judicieux d'utiliser un framework JavaScript plus robuste pour gérer l'interactivité avec un état global (comme **React** ou **Vue.js**).

Conclusion

En résumé, le choix des technologies HTML, CSS, JavaScript, PHP et MySQL est justifié pour ce type de projet car elles sont largement utilisées et bien supportées, tout en offrant une grande flexibilité et accessibilité. Les mesures de sécurité prises garantissent que l'application web sera résistante aux attaques courantes, tandis que les choix technologiques sont basés sur leur simplicité d'utilisation et leur compatibilité avec les hébergements web courants.