CHONG Jong Hoa

GAILLARD Maylis

M4-Dev-G1

TP2

|  |  |  |
| --- | --- | --- |
| Caractéristiques |  |  |
| Fonctionnalité | Cahier des charges (Maquette) | L’application doit correspondre au Cahier des charges, c’est-à-dire elle doit contenir une page d’accueil, une page de connexion, une page d’inscription comme c’est stipulé dans ce document. |
| Coverage des tests | 90% de taux couverture du front End |
| Résolution des bugs | Bugs trouvés : 10  Bugs résolus : 10 |
| Performance d'Efficacité | Test de charge légère | Le temps moyen de réponse reste inférieur à une limite prédéfinie pour chaque niveau de charge.  Aucune erreur majeure n'est signalée pendant les tests de charge prolongée. |
| Test de vitesse de chargement normal | Le temps de chargement de la page de connexion dans des conditions normales ne dépasse pas une limite prédéfinie.  Le temps de chargement reste raisonnablement bas même sous une charge accrue.  Les optimisations appliquées améliorent significativement le temps de chargement de la page. |
| Test de performance | Le test de performance a défini que le CPU de la machine de l’utilisateur doit être inférieur à 50%.  Ce test a défini que la consommation de la RAM ne doit pas excéder 200 Mo.  La consommation du GPU ne doit pas excéder 200 Mo dans le cadre de cette application. |
| Fiabilité | Sécurité aux attaques XSS | On utilise les recommandations de React pour éviter les injections XSS. Tel que la Validation et filtrage des entrées utilisateur. La vérification se fait par une bibliothèque de validation validator.js. |
| Sécurité aux injections SQL | On utilise des ORM au niveau du back-end qui limite les injections SQL. On a mis en place un système d’autorisation d’accès à la base de données. |
| Test de la gestion des erreurs de connexion | L’application gère correctement les erreurs de connexion et de création de compte en affichant des messages d’erreur appropriés. |
| Compatibilité | Responsive | L'application s’affiche correctement en version mobile à partir de 321 px, 800 px pour les tablettes, 1200 px pour les ordinateurs portables. |
| Compatibilité au niveau des navigateurs | Les tests exploratoires ont été effectués sur les navigateurs suivant :   * Google Chrome * Fire Fox * Microsoft Edge * Opéra * Safari   Ils démontrent que toutes les fonctionnalités de l'application sont accessibles et fonctionnent correctement dans chaque navigateur testé.  Le rendu visuel et l'agencement des éléments de l'interface utilisateur sont cohérents dans tous les navigateurs. |
| Compatibilité avec les systèmes d’exploitation | La compatibilité a été testé sur les systèmes d’exploitation suivant :   * Windows * Mac OS * Linux   Toutes les fonctionnalités de l'application sont accessibles et fonctionnent correctement sur les différents systèmes d’exploitation cité précédemment.  L'interface utilisateur est cohérente et conforme aux directives de conception respectives pour chaque système d'exploitation. |
| Facilité d'utilisation | Guide d’utilisateur | L'application est décrite pour que l’utilisateur puisse comprendre son fonctionnement étape par étape. |
| Assistance | Mise en place de la page de contact, cette page est accessible dans le footer. Un chatbot suit la navigation de l’utilisateur dans le corps de l’application, un service téléphonique pour le sav et un mail à la disposition de l’utilisateur en cas de besoin sont placé dans le footer de l’application. |
| FAQ | Mise en place d’une page avec des questions générales sur l’application pour aider l’utilisateur en cas de problème ou de questionnement. Cette page est accessible dans le footer de l’application. |
| Satisfaction | Espace de commentaires | Recueillir des commentaires qualitatifs sur ce qui a contribué à la satisfaction de l’utilisateur ou à son insatisfaction. La récupération des commentaires se fait par le biais de l’outil Medallia. Cette plateforme de gestion de l'expérience client permet de recueillir les commentaires des clients à travers différents canaux, y compris les sondages, les évaluations et les commentaires ouverts. |
| Temps de chargement | L'application a un temps de chargement pour l’inscription qui est inférieur à 3 secondes. On a analysé l’application en utilisant l’outil Google PageSpeed Insights. |
| Notations | Demander aux utilisateurs de noter leur satisfaction globale à l’égard de l’application sur une échelle de 1 à 5. |
| Conformité | Confidentialité | Limiter le nombre de requête pour l’accès aux données. Informer les clients de leurs droits d'accès, de rectification, d'opposition et de suppression des informations collectées. Veiller à la sécurité des systèmes d'informations en chiffrant toutes les données sensibles qui sont stockées dans un repos, en utilisant des algorithmes de chiffrement robustes. On utilise pour la sécurisation des données de l’application, l’algorithme de chiffrement AES (Advanced Encryption Standard). Il est un des algorithmes les plus largement utilisé. Il est considéré comme très sécurisé et efficace. |
| Audit de code | L’audit du code de l’application est satisfaisant. Il révèle qu’elle est conforme aux principales normes de sécurité OWASP Top 10. Cette application respecte ces normes par le biais des intégrations suivantes :  Validation des entrées et échappement des sorties :  Toutes les entrées utilisateur sont correctement validées et échappées pour prévenir les attaques d'injection (injection SQL, XSS, etc.). On utilise des ORM au niveau du back-end qui limite les injections SQL. Tel que la validation et filtrage des entrées utilisateur. La vérification se fait par une bibliothèque de validation validator.js.  Gestion appropriée des sessions :  Les identifiants de session sont générés de manière aléatoire par des tokens JWT, stockés de manière sécurisée et expirés après une période d'inactivité.  Contrôles d'accès appropriés :  L'accès aux ressources sensibles est restreint en fonction des rôles et des autorisations de l'utilisateur, et les contrôles d'accès sont mis en œuvre à la fois au niveau de l'interface utilisateur et au niveau de la logique métier.  Authentification forte :  L'authentification est mise en œuvre de manière sécurisée, en utilisant des méthodes telles que le hachage des mots de passe, la salaison et la gestion adéquate des mécanismes d'authentification multi-facteurs (MFA).  Protection contre les attaques CSRF :  Des jetons CSRF sont utilisés pour protéger les formulaires contre les attaques de falsification de requêtes inter-sites (CSRF).  Sécurité des communications :  Les communications entre l'application et les utilisateurs sont sécurisés en utilisant des protocoles de cryptage appropriés comme HTTPS.  Gestion des erreurs sécurisée :  Les erreurs de l'application ne révèlent pas d'informations sensibles et sont gérées de manière appropriée pour empêcher les attaquants de tirer profit des messages d'erreur.  Sécurité de la configuration :  Les configurations de l'application et des composants tiers sont correctement sécurisées, et les paramètres sensibles sont stockés de manière sécurisée et non exposés dans le code source ou les fichiers de configuration.  Mises à jour et correctifs réguliers :  L'application est régulièrement mise à jour avec les derniers correctifs de sécurité pour toutes les bibliothèques, Framework et composants tiers utilisés qui compose l’application.  Tests de sécurité réguliers :  Des tests de sécurité réguliers, y compris des tests de pénétration et des audits de sécurité du code source, sont effectués pour identifier et corriger les vulnérabilités potentielles.  Toutes ses pratiques misent en place respectes les normes OWASP Top 10. |
| Charte de qualité de développement. | Le codage de l’application respecte les conventions de nommages du framework React Js. Les variables et les constantes sont camelCase, où le premier mot commence en minuscule et les mots suivants commencent par une majuscule. Par exemple : **myVariable**, **userName**, **isLoggedIn**.  La gestion des erreurs respects les recommandations du framework React js. Elle est effectuée par le biais des **Composants ErrorBoundary** : les composants ErrorBoundary encapsulent des parties l’application et gèrent les erreurs qui se produisent à l'intérieur de ces parties. Un ErrorBoundary est un composant de haut niveau qui utilise **componentDidCatch()** pour intercepter les erreurs et afficher un message d'erreur à l'utilisateur.  Le code de l’application est maintenable en raison de l’utilisation de composants réutilisable qui se concentrent principalement sur l'interface utilisateur. Les logiques répétitives sont dans des hooks personnalisés. Cette application est codée également avec des hooks et des contextes fournis par React pour simplifier la logique et réduire la complexité.  D’autre part les hooks permettent de simplifier le test des composants fonctionnels.  Cette méthode de développement permet de pouvoir maintenir l’application dans de futur développement.  La logique métier est séparé du reste du code pour une meilleure maintenabilité et permettre de travailler sur l'ajout de nouvelles fonctionnalités ou la modification des fonctionnalités existantes sans impacter le reste de l'application.  La documentation de l’application est claire et concise. Elle détail les fonctionnalités des dépendances, des composants et des hooks.  Cette documentation permet de comprendre le fonctionnement de chaque partie de l’application et de pouvoir effectuer un travail collaboratif.  Des commentaires sont mis en place dans le code de l’application pour faciliter le travail avec l’équipe de développement.  La revue de code se fait une à deux fois par semaine lorsqu’il n’y a pas de nouvelle fonctionnalité ou mise à jour majeur. Ce procédé permet de détecter plus rapidement les bugs éventuels. La mise en place de nouvelle fonctionnalité ou mise à jour majeure entraine une revue de code pour débuguer le plus rapidement possible le code si des bugs ont été trouvés.  L’application est testée par des tests unitaires. Ces tests sont automatisés et intégré avant le développement et au moment du déploiement de l’application. Cette pratique permet de détecter d'éventuel beug le plus rapidement possible.  Les tests mis en place dans cette application sont indépendants des autres.  Ils testent une seule fonctionnalité ou méthode à la fois. Cela facilite l'identification et la résolution des problèmes lorsqu'un test échoue.  On utilise le framework Jest pour les tests unitaires et d’intégration continu.  Enzyme est mis en place pour tester les composants de l’application. |