Équipe 102

ColorImage
Plan de tests logiciels

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2021-10-04	1.0	Rédaction initial du plan de tests	Équipe 102
2021-11-15	2.0	Mise à jour du plan de tests pour la remise	Équipe 102

Table des matières

Plan de tests logiciels

1. Introduction

Ce document de test a pour objectif d'évaluer le bon fonctionnement de notre application, *ColorImage.* D'abord, nous allons déterminer les exigences à tester, ensuite nous allons décrire la stratégie qui va être appliquée, en s'assurant de spécifier les ressources utilisées. Enfin, nous allons conclure avec les principaux jalons relatifs à la discipline des tests tout au long de la réalisation du projet.

2. Exigences à tester

Exigences	**Tests associés**
Profil d'utilisateur	Tests de fonction, Tests d'interface usager, Tests d'intégrité des données
Galerie de dessins	Tests de fonction, Tests d'interface usager, Tests d'intégrité des données, Tests de volume, Tests de sécurité et de contrôle d'accès
Dessin collaboratif	Tests de fonction, Tests d'interface usager, Tests d'intégrité des données, Tests de volume
Édition collaborative	Tests de fonction, Tests d'interface usager, Tests de performance, Tests de charge
Équipe de collaboration	Tests de fonction, Tests d'interface usager, Tests de sécurité et de contrôle d'accès
Clavardage	Tests de fonction, Tests d'interface usager
Apparence	Tests de fonction, Tests d'interface usager
Avatar	Tests de fonction, Tests d'interface usager, Tests d'intégrité des données
Utilisabilité	Tests d'interface usager, Tests d'échec/récupération
Performance	Tests de performance, Tests de charge, Tests de stress, Tests de volume

3. Stratégie de test

3.1. Types de test

3.1.1. Tests de fonction

Objectif de test:	Exécuter les différents scénarios de la fonctionnalité décrits par les diagrammes de cas d'utilisation et s'assurer que son résultat reflète ce qui est attendu dans les requis du SRS et des vues de cas d'utilisation.
Technique:	Choisir un scénario spécifique d'une fonctionnalité, exécuter le scénario et vérifier que le résultat est celui attendu en le comparant au requis du SRS. Répéter la procédure jusqu'à ce que tous les scénarios de la fonctionnalité soient couverts.
Critère de complétion:	Le scénario du fonctionnement exécuté est conforme au requis du SRS.
Considérations spéciales:	

3.1.2. Tests d'interface usager

Objectif de test:	Tester les différents boutons, modales, et autres composantes de l'interface avec lesquelles l'utilisateur peut interagir et s'assurer que l'interface répondent correctement aux interactions de l'utilisateur. S'assurer également que l'utilisation est fluide et intuitive.
Technique:	Naviguer à travers l'application en essayant les différentes fonctionnalités implémentées. Vérifier que les différents boutons, modales, et autres composantes avec lesquelles l'utilisateur peut interagir agissent comme indiqué dans le document de requis SRS et les diagrammes de cas d'utilisation.
Critère de complétion:	Les différentes fonctionnalités de l'interface usager fonctionnent correctement et permettent à l'utilisateur de naviguer l'application sans avoir de problèmes d'accès à un service.
Considérations spéciales:	

3.1.3. Tests d'intégrité des données

Objectif de test:	S'assurer que le serveur n'accepte que les données permises et que les données stockées en base de données ne sont pas corrompues et sont du bon type.
Technique:	Faire des requêtes HTTPS et des requêtes à partir du socket contenant des données valides et invalides au serveur à l'aide de l'outil Postman. Valider que le serveur gère les données invalides et qu'aucune données corrompues se retrouve dans la base de données.
Critère de complétion:	Les requêtes invalides retournent une erreur, et ne sont pas sauvegardées en base de données.
Considérations spéciales:	

3.1.4. Tests de performance

Objectif de test:	Mesurer les temps de réponses et les taux de transactions pour les requêtes entre le serveur et le client, entre le serveur et la base de données MongoDB, et entre le serveur et Firebase.
Technique:	Tester et mesurer les latences pour les différentes fonctionnalités qui nécessitent des communications entre le serveur et le client, entre le serveur et la base de données MongoDB, et entre le serveur et Firebase. S'assurer que les mesures obtenues respectent les limites décrites dans les requis du SRS.
Critère de complétion:	Les mesures de latences obtenues doivent respecter les limites décrites dans le document de requis SRS.
Considérations spéciales:	

3.1.5. Tests de charge

Objectif de test:	S'assurer du bon fonctionnement de l'application quand un grand nombre de personnes en font usage en même temps. S'assurer que le bon fonctionnement de l'application est maintenu pour les limites de charges de travail précisées dans le document de requis SRS.
Technique:	Reproduire les scénarios de cas d'utilisation limites décrits dans le document de requis SRS. Vérifier que les scénarios limites sont traités de façon raisonnable comme décrit dans le document de requis SRS.
Critère de complétion:	Il n'y a pas de latence significative entre les actions et l'application ne plante pas. Le bon fonctionnement est garanti selon les limites de charges de travail établies dans le document de requis SRS.
Considérations spéciales:	

3.1.6. Tests de stress

Objectif de test:	S'assurer que les différents fonctionnalités qui accèdent à une même ressource partagée soit géré de façon adéquat selon le document de requis SRS tels que : la concurrence lorsque plus de deux utilisateurs modifient les attributs d'un dessin, d'une même équipe collaborative, ou bien d'un même canal de discussion, etc. S'assurer que lorsqu'un utilisateur tente de répéter une action plusieurs fois à répétitions pendant des intervalles de temps très minime, l'application continue son bon fonctionnement et gère les erreurs de concurrence en affichant un message d'erreur.
Technique:	Tester les fonctionnalités qui accèdent à une même ressource à l'aide de deux utilisateurs en parallèle. S'assurer par la suite que l'ordre des exécutions ont bien été respectées. Spammer les boutons, les modales et les composantes interactives tels que les trait de dessins, les messages envoyés à un canal et s'assurer que les erreurs de concurrences sont gérées de façon appropriée.

Critère de complétion:	S'assurer que les requêtes envoyées au serveur suivent l'ordre de réception. Par exemple, si un utilisateur 1 dessine un trait en même temps qu'un utilisateur 2, mais qu'il est quelques millisecondes en avance. Le trait de l'utilisateur 1 doit être géré avant la requête de l'utilisateur 2.
Considérations spéciales:	

3.1.7. Tests de volume

Objectif de test:	Vérifier que charger et envoyer une grosse quantité de données ne va pas affecter la performance du logiciel et nuire à l'expérience utilisateur.
Technique:	Tester toutes les fonctionnalités qui n'ont pas de limites définies. Il y a, par exemple, le nombre de dessins dans la galerie, le nombre de traits dans un dessin et plusieurs autres. Chacune de ces fonctionnalités sera testée en ajoutant une grande quantité de données afin de s'assurer que le bon fonctionnement est maintenu selon les seuils définis dans le document de requis SRS.
Critère de complétion:	L'application reste relativement fluide et ne plante pas soudainement à cause des grosses quantités de données (grand volume de données). Lorsqu'on s'approche du seuil défini dans le document de requis SRS, le bon fonctionnement doit être maintenu de façon acceptable.
Considérations spéciales:	

3.1.8. Tests de sécurité et de contrôle d'accès

Objectif de test:	Vérifier que le serveur rejette bien les requêtes des utilisateurs ne disposant pas de l'accès. Vérifier aussi que la connexion au socket est impossible sans le token d'accès accordé.
Technique:	Faire des requêtes contenant des token de sécurité valides, invalides et expirés. Tenter de se connecter au socket sans avoir le token d'accès.
Critère de complétion:	Le serveur ne devrait accepter que les requêtes http des utilisateurs disposant du token de sécurité valide et non expiré. Le serveur ne doit pas permettre à un utilisateur non connecté de se connecter au socket.
Considérations spéciales:	

3.1.9. Tests d'échec/récupération

Objectif de test:	Vérifier que l'application est capable de récupérer d'une défaillance matérielle, logicielle ou réseau et s'assurer que ces défaillances ne compromettent pas l'intégrité des données dans la base de données.
Technique:	Tester pour des scénarios de défaillance matérielle et logicielle tel que l'ordinateur se ferme soudainement et s'assurer que le serveur puisse détecter cela et agir en conséquence. Pour l'exemple précédent, le serveur déconnecte l'utilisateur de l'application et l'application du client lourd retourne à la page de connexion.

	Tester pour des scénarios de défaillance réseau tel que la perte de connexion à internet sur le client lourd et léger et s'assurer que le serveur puisse détecter cela et agir en conséquence. Pour l'exemple précédent, le serveur déconnecte l'utilisateur de l'application qui a perdu connexion et navigue le client vers la page de connexion où celle-ci affichera une erreur mentionnant le problème de connexion.
Critère de complétion:	Les erreurs de défaillances doivent être géré de façon adéquate comme mentionné par le document de requis SRS et s'assurer que l'utilisateur est avisé du problème de défaillance.
Considérations spéciales:	Les défaillances du côté du serveur Heroku ne peuvent être traitées lorsque le service est en panne. Cependant, lorsque le serveur Heroku redevient stable, le serveur initialisera de nouveau les dessins, les canaux de discussion, et les utilisateurs en se fiant aux données retrouvées dans la base de données.

3.2. Outils

Les outils suivants seront utilisés au sein de la discipline de test:

Type de test	Outil
Tests de fonction	Client lourd, Client léger
Tests d'interface usager	Client lourd, Client léger
Tests d'intégrité des données	Client lourd, Client léger, Heroku, MongoDB, Postman
Tests de performance	Client lourd, Client léger, Heroku, MongoDB, Firebase
Tests de charge	Client lourd, Client léger, Heroku
Tests de stress	Client lourd, Client léger, Heroku
Tests de volume	Client lourd, Client léger, Heroku
Tests de sécurité et de contrôle d'accès	Client lourd, Client léger, Postman, MongoDB, Heroku
Tests d'échec/récupération	Client lourd, Client léger, Heroku

4. Ressources

4.1. Équipe de test

Rôle	Membre de l'équipe	Responsabilités
Concepteur de tests pour le client lourd	**Aleksandar Stijelja**	Rédiger les cas de tests pour le client lourd
Testeur pour le client léger	**Andy Lam**	Effectuer les cas de test pour le client léger et noter les résultats
Testeur pour le client lourd	**Anis Zouatene**	Effectuer les cas de test pour le client lourd et noter les résultats
Testeur pour le client serveur	**Ismail Bakkouri**	Effectuer les cas de test pour le serveur et noter les résultats
Concepteur de tests pour le serveur	**Jason Thai**	Rédiger les cas de tests pour le serveur
Concepteur de tests pour le client léger	**Mohammed Ariful Islam**	Rédiger les cas de tests pour le client léger

4.2. Système

Afin d'effectuer les tests pour notre, différents systèmes sont nécessaires. D'abord, pour le client lourd, sera un logiciel construit avec Electron pour ainsi être exécuté . Ensuite, pour le client léger, une tablette Galaxy Tab A possédant un écran de 1920 x 1200 pixels et roulant la version 9 d' Android sera utilisée. Finalement, le serveur est déployé sur Heroku et donc n'a pas besoin de configuration spécifique.

5. Jalons du projet

Jalon	Effort	Date de début	Date de fin
Conception des cas de tests pour client lourd	3	2021-10-04	2021-10-05
Conception des cas de tests pour client léger	3	2021-10-04	2021-10-05
Conception des cas de tests pour le serveur	3	2021-10-06	2021-10-08
Exécution des test sur client lourd	2	2021-10-12	2021-11-21
Exécution des test sur client léger	2	2021-10-12	2021-11-21
Exécution des test sur le serveur	2	2021-10-15	2021-11-21
Réévaluation et modification des cas de test pour client lourd	1	2021-11-31	2021-12-01
Réévaluation et modification des cas de test pour client léger	1	2021-11-31	2021-12-01

Réévaluation et modification des cas de test pour serveur	1	2021-11-31	2021-12-01
Exécution des nouveaux test sur client lourd	2	2021-12-04	2021-12-06
Exécution des nouveaux test sur client léger	2	2021-12-04	2021-12-06
Exécution des nouveaux test sur le serveur	1	2021-12-05	2021-12-06