Équipe 204

Projet d’évolution d’un logiciel
Plan de tests logiciels

Version 1.0

Historique des révisions

Date	Version	Description	Auteur
2021-04-10	1.0	Début du document	Samuel Poulin Maxime Fecteau Antoine Morcel Guillaume Beausoleil Étienne Plante

Table des matières

Plan de tests logiciels

1. Introduction

Ce document permet de présenter la stratégie utilisée afin de tester notre système sans rentrer dans les détails d'implémentation de ces tests. Il sera tout d'abord question de faire la liste des exigences à tester. Ensuite, nous entrerons plus en détail sur la stratégie de test utilisée pour chacune de ces exigences. Finalement, il sera question de lister les ressources responsables des tests et d'établir les jalons de projet reliés à la discipline de test.

2. Exigences à tester

Exigences	Tests associés
3.1 Lobby	Tests de fonctions, tests d'interface usager, tests unitaires
3.2.1 Classique	Tests d'interface usager
3.2.2 Solo	Tests de fonctions, tests d'interface usager, tests unitaires
3.2.3 Sprint coopératif	Tests de fonctions, tests d'interface usager, tests unitaires
3.3 Création de paire mot-image sur le client lourd - 3.3.1 - 3.3.2 Manuelle 1 - 3.3.3 Manuelle 2 - 3.3.4 Assistée 1	Tests de fonctions, tests d'interface usager, tests unitaires
3.6 Page de connexion - 3.6.1 - 3.6.2 - 3.6.3 - 3.6.4 - 3.6.5 - 3.6.6	Tests de fonctions, tests d'interface usager
3.7 Système de clavardage	Tests de fonctions, tests d'interface usager, tests unitaires
3.8 Page de choix de partie	Tests de fonctions, tests d'interface usager, tests unitaires
3.10 Menu Principal	Tests d'interface usager
3.11 Profil Utilisateur	Tests de fonctions, tests d'interface usager, tests unitaires

3.12 Liste d'ami	Tests de fonctions, tests d'interface usager, tests unitaires
3.13 Tutoriel	Tests d'interface usager
3.4 Personnalité des joueurs virtuels	Tests de fonctions, tests d'interface usager
4.2.1 Disponibilité du serveur	Tests de stress
4.3.1 Temps de réponse du serveur	Tests de performance, test de stress
4.3.2 Performance du dessin	Tests de fonctions, test de stress
4.7.2 Système d'assistance	Tests d'interface usager

3. Stratégie de test

3.1. Types de test

3.1.1. Tests de fonction

3.1.1.1 Suite de tests de fonctionnalités

Objectif de test:	Tester certaines fonctionnalités critiques du serveur à travers une suite de tests qui fait partie du pipeline d'intégration.
Technique:	Tests de fonctionnalités qui font partie de la suite de tests du serveur. Ces tests sont des tests en boîte noire qui utilisent les fonctionnalités du serveur comme un client qui n'a aucune idée de ce que fait exactement le serveur.
Critère de complétion:	Tests de fonctionnalités sélectionnés doivent tous passer.
Considérations spéciales:	Ces tests font partie de la suite de tests qui est roulé à chaque "merge request", c'est donc un effort sur toute la durée du projet de mettre à jour et s'assurer que ces tests de fonctionnalités passent.

3.1.1.2 Tests à la main utilisant les clients

Objectif de test:	Tester certaines fonctionnalités critiques du serveur à travers l'utilisation des clients.
Technique:	Tester plusieurs scénarios de cas d'utilisation de l'application avec l'application elle-même, pour déceler des défauts ou des interactions à modifier.
Critère de complétion:	La session d'utilisation se termine sans problèmes ou la session rencontre un problèmes, dans ce cas, on règle le problème et on répète le test.
Considérations spéciales:	

3.1.2. Tests d'interface usager

Objectif de test:	Permet de s'assurer de la facilité de compréhension et d'utilisation de l'interface usager pour des personnes n'ayant jamais utilisé l'application.

Technique:	Faire utiliser l'application par un utilisateur non expérimenté et recueillir la rétroaction de son expérience usager à l'aide de questions.
Critère de complétion:	L'obtention de réponses aux questions d'interactions posées aux utilisateurs.
Considérations spéciales:	

3.1.3. Tests d'intégrité des données

Objectif de test:	S'assurer que pour toute donnée présente dans la base de données, cette donnée peut être consultée et utilisée sans causer une erreur de fonctionnement de l'application.
Technique:	Insérer des données dans la base de données comme si un client le faisait à travers le serveur, puis utiliser ces données de façon habituelle
Critère de complétion:	Ne pas déclencher d'erreur lors de l'utilisation des données
Considérations spéciales:	

3.1.4. Tests de performance

Objectif de test:	S'assurer d'avoir un temps de réponse adéquat pour les échanges de données dans l'application ou avec un serveur.
Technique:	Évaluation du temps entre l'envoi d'une requête au serveur et la réponse de celui-ci ou l'évaluation du temps suite à un événement déclenché volontairement dans l'application.
Critère de complétion:	Obtenir une métrique valide à la fin du test qui peut être évaluée et analysée.
Considérations spéciales:	

3.1.5. Tests de charge

Pour notre application, nous avons conclu que les tests de charge ne sont pas assez pertinents pour en inclure dans le plan de tests.

3.1.6. Tests de stress

Objectif de test:	S'assurer que le système peut ingérer l'input concurrent de plusieurs clients pour les différentes fonctionnalités de l'application.
Technique:	Connecter plusieurs clients en même temps sur l'application et les faire interagir en même temps sur les mêmes fonctionnalités.
Critère de complétion:	Le serveur doit compléter les demandes de tous les clients dans un délai raisonnable et sans échec de réponse aux demandes.
Considérations spéciales:	Le serveur de l'application est suffisamment puissant pour plusieurs utilisateurs, mais il n'est pas prêt à avoir des milliers d'utilisateurs au même

	moment.

3.1.7. Tests de volume

Pour notre application, nous avons conclu que les tests de volume ne sont pas assez pertinents pour en inclure dans le plan de tests puisque nous limitons la taille des données gérées par le serveur lorsqu'elles sont applicables.

3.1.8. Tests de sécurité et de contrôle d'accès

Objectif de test:	S'assurer que même si quelqu'un fait des appels au serveur à l'extérieur du client (à travers Postman par exemple) que l'accès aux données est sécurisé et régulé par le serveur.
Technique:	Essayer d'accéder à des données sensibles sans être authentifié et essayer d'accéder à des données non publiques lorsque authentifié.
Critère de complétion:	Aucune donnée non publique devrait être retournée par le serveur et aucune donnée devrait être retournée à un utilisateur non authentifié.
Considérations spéciales:	

3.1.9. Tests d'échec/récupération

Objectif de test:	S'assurer que si le serveur rencontre une erreur au milieu d'une série d'insertion et qu'il trouve des données manquantes à la prochaine consultation, il s'occupe de compléter la transaction partielle.
Technique:	Exécuter des transactions partielles et observer le comportement à la consultation.
Critère de complétion:	La prochaine consultation termine la transaction.
Considérations spéciales:	

3.1.10. Tests de configuration

Pour notre application, nous avons conclu que les tests de configuration ne sont pas pertinents puisque cette dernière s'exécute sous la plateforme Electron, qui gère elle-même l'exécution attendue de l'application web peu importe la configuration matérielle et logicielle.

3.1.11. Tests d'installation

3.1.11.1 Test d'installation de l'application

Objectif de test:	Vérifier que le client lourd peut être installé avec succès sur le système d'exploitation grâce à l'exécutable compilé par electron-builder.

Technique:	Ouvrir l'exécutable compilé et attendre la fin de l'installation.
Critère de complétion:	L'installation ne lance pas d'erreur, elle laisse une entrée dans le menu du système d'exploitation et dans le menu de désinstallation.
Considérations spéciales:	

3.1.11.2 Test de désintallation de l'application

Objectif de test:	Vérifier que le client lourd installé peut être désinstallé à travers l'entrée du système d'exploitation et que toute trace de l'application est effacée dans les menus de celui-ci
Technique:	Désinstaller l'application à travers le menu du système d'exploitation.
Critère de complétion:	Les entrées du système d'exploitation sont enlevées et le dossier contenant l'application est effacé.
Considérations spéciales:	

3.1.12. Tests unitaires

Objectif de test:	S'assurer que les différents services et "composants" de l'application sont fonctionnels de manière individuelle et que ceux-ci ont le comportement voulu.
Technique:	Avoir des fichiers de tests unitaires pour chaque fichiers qu'on souhaite tester.
Critère de complétion:	Avoir un objectif de code coverage qui est atteint avec tous les tests unitaires qui passent.
Considérations spéciales:	Ces tests font partie de la suite de tests qui est roulé à chaque "merge request", c'est donc un effort sur toute la durée du projet de mettre à jour et s'assurer que ces tests unitaires passent.

3.2. Outils

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

Type de test	Outil
Tests de fonctionnalités	Mocha, Chai, Sinon et Polydessin
Tests de sécurité et contrôle d'accès	POSTMAN
Tests de stress	Polydessin
Tests d'interface utilisateur	Polydessin
Tests d'installation	Polydessin et Windows.
Tests de performance	PM2

Tests unitaires	Mocha, Chai, Sinon, Robolectric

4. Ressources

4.1. Équipe de test

Rôle	Membre de l'équipe	Responsabilités
Testeur	Antoine Morcel	Tests de fonctionnalités, Tests de sécurité et contrôle d'accès, Tests de stress, Tests unitaires
Testeur	Samuel Poulin	Tests de fonctionnalités, Tests d'installation, Tests de stress
Testeur	Maxime Fecteau	Tests de fonctionnalités, Tests d'interface usager, Tests de stress, Tests de performance
Testeur	Guillaume Beausoleil	Tests unitaires
Testeur	Étienne Plante	Tests unitaires

4.2. Système

Pour notre discipline de tests, nous possédons plusieurs environnements et configurations pour les différentes parties de l'infrastructure de notre application : le serveur, le client lourd, le client léger et l'environnement du serveur sur Gitlab.

Serveur : Nous possédons pour le serveur des fichiers de tests unitaires qui permettent de tester nos services et fonctionnalités individuellement à l'aide de Mocha, Chai et Sinon. Pour les tests qui requièrent des informations dans la base de données, nous avons un environnement établi avec différentes requêtes HTTP à l'aide de Postman qui nous permettent de tester et vérifier les informations renvoyées par le serveur lors des requêtes.

Client Lourd : Pour le client lourd, notre environnement et configuration est très similaire au serveur. Nous utilisons des fichiers de tests unitaires qui nous permettent de tester nos modules individuellement à l'aide de Jasmine et Karma. Le client lourd peut aussi utiliser l'environnement de requêtes HTTP de Postman pour l'obtention d'informations de la base de données.

Client Léger : Nous possédons des tests unitaires permettant d'assurer le bon fonctionnement de la logique de l'application. L'environnement utilisé pour faire les tests est Robolectric avec Hilt pour l'injection de dépendance et le remplacement des modules pour les tests.

Gitlab : Nous possédons un pipeline qui s'occupe de simuler l'environnement de l'application à chaque fois qu'une PR est passée sur la branche develop. Avant même l'approbation des autres développeurs, le pipeline exécute chaque test dans l'application (simule npm run test) et accepte la PR seulement si ceux-ci passent toutes. Cela permet de s'assurer qu'aucun changement aux fonctionnalités testées ne peut introduire des changements problématiques sur la branche develop.

5. Jalons du projet

Jalon	Effort	Date de début	Date de fin
Tests unitaires et tests de fonctionnalités durant la session	4	2021-02-01	2021-04-19
Période de test de l'application finale avant la remise.	7	2021-04-16	2021-04-19