Équipe 107

DrawHub
Plan de tests logiciels

Version 1.0

Historique des révisions

Date	Version	Description	Auteurs
2020-12-02	1.0	Rédaction initiale complétée par l'équipe de DrawHub.	Bruno Curzi-Laliberté William Glazer-Cavanagh Thomas Houtart Antoine Martin Simon Robatto-Simard Vincent Tessier

Table des matières

Plan de tests logiciels

1. Introduction

Ce document présente une vue d'ensemble de la procédure de tests adoptée le long du projet. Il sera question d'expliciter les divers groupes d'exigences établies dans le SRS et d'énoncer les types de tests qui ont été exécutés afin de tester chaque groupe d'exigences. Finalement, les responsabilités des membres de l'équipe ainsi que les jalons respectifs à la discipline des tests seront explicités.

2. Exigences à tester

Chaque exigence fonctionnelle a, pour chaque client, eu des tests d'intégrations manuels lors du développement ainsi qu'à chaque jalon du projet. Ces tests ne sont pas mentionnés à chaque ligne dans le tableau mais seront explicités dans les résultats des tests.

Exigences fonctionnelles		
Exigence	**Tests associés**	
	Clients	**Serveur**
3.1 - Clavardage	Tests de fonction Tests d'interface usager Tests d'intégration automatique Tests unitaires	Tests d'intégration automatique Tests unitaires
3.2 - Accès à un compte existant	Tests de fonction Tests d'interface usager Tests d'intégration automatique	Tests d'intégration automatique Tests unitaires Tests de sécurité et de contrôle d'accès
3.3 - Création d'un compte	Tests de fonction Tests d'interface usager Tests d'intégration automatique	Tests d'intégration automatique Tests unitaires Tests de sécurité et de contrôle d'accès
3.8 - Création d'une partie	Tests de fonction Tests d'interface usager	Tests d'intégration automatique
3.9 - Lobby	Tests de fonction Tests d'interface usager	Tests d'intégration automatique
3.10 - Mode spectateur	---	Tests d'intégration automatique
3.11 - Exigences générales des modes de jeu	Tests de fonction	Tests d'intégration automatique
3.12 - Mêlée générale	Tests de fonction Tests d'interface usager	Tests d'intégration automatique
3.15 - Bataille royale	Tests de fonction Tests d'interface usager	Tests d'intégration automatique
3.24 - Interface de dessin	Tests de fonction Tests d'interface usager Tests unitaires	---

Exigences non fonctionnelles		
Exigence	**Tests associés**	
	Clients	**Serveur**
4.1 - Utilisabilité	Tests d'interface usager	---
4.2 - Fiabilité	---	Tests d'intégration à long terme en laissant le serveur actif sur Heroku
4.4 - Maintenabilité	Tests de qualité de code	Tests de qualité de code
4.5 - Contraintes de conception	Tests de qualité de code	Tests de qualité de code
4.6 - Sécurité	---	Tests d'intégration automatique Tests de sécurité et de contrôle d'accès

3. Stratégie de test

3.1. Types de test

3.1.1. Tests de fonction

Objectif de test:	Assurer le bon fonctionnement des éléments de base du logiciel, s'assurer qu'au moment où la fonction est écrite, elle fonctionne.
Technique:	Tests en boîte noire.
Critère de complétion:	Les cas de tests identifiés, notamment pour les cas extrêmes, fonctionnent.
Considérations spéciales:	Aucune.

3.1.2. Tests unitaires

Objectif de test:	Valider le fonctionnement d'une méthode ou d'une fonction.
Technique:	Boîte blanche pour la couverture des branches.
Critère de complétion:	Le test automatique passe et fait partie d'un ensemble de tests qui valident plusieurs scénarios.
Considérations spéciales:	Idéalement, la couverture des branches pour un test ne devrait pas être inférieure à 80%. Mais on a accepté des couvertures inférieures dans le cas où le test d'une branche s'avérait trop difficile à implémenter.

3.1.3. Tests d'interface usager

Objectif de test:	Vérifier la cohérence de l'interface utilisateur afin d'assurer une compréhension facile et une utilisation cohérente des éléments de l'interface (boutons, icônes, titres, etc.)
Technique:	Obtenir la rétroaction d'un coéquipier ou d'un tiers n'ayant pas implémenté le client.
Critère de complétion:	Un utilisateur doit être capable de découvrir les fonctionnalités de l'application sans trop poser de questions. Un utilisateur doit déduire facilement les options offertes sur chaque page.
Considérations spéciales:	La rétroaction des coéquipiers est utile car ils amènent un point de vue professionnel et des critiques constructives par rapport aux problèmes de l'interface, alors que les points de vue des tiers montrent comment l'application est perçue par un usager cible. C'est pour ça qu'on a choisi d'inclure les deux.

3.1.4. Tests de sécurité et de contrôle d'accès

Objectif de test:	Assurer l'intégrité du système.
Technique:	Tests d'intégration automatiques simulant la connexion d'utilisateurs, sur le serveur.
Critère de complétion:	Les clients ne reçoivent pas d'information ne leur étant pas adressée et n'ont pas le droit de faire toutes les actions/requêtes au serveur.
Considérations spéciales:	Aucune.

3.1.5. Tests d'intégration automatiques

Objectif de test:	Assurer la qualité du logiciel à chaque version. Ces tests très haut niveau servent à vérifier que nos changements sur le code n'affectent pas le comportement attendu. Ces tests sont pratiquement uniquement des tests de régression.
Technique:	Pour le serveur: Imiter des clients SocketIOs et effectuer des actions variées, effectuer les requêtes HTTP associées dans le cadre de la simulation d'interaction client serveur. Utiliser une technique boîte noire pour les interactions client-serveur avec un contrôle dans le cadre des attentes du test sur l'état du système après chaque étape. Pour les clients: Simuler un serveur qui communique de manière prévisible les messages voulus dans le cadre de la simulation.
Critère de complétion:	L'état du serveur ou du client est correct après chaque étape du test. Le client ou le serveur reçoit les communications et y répond.
Considérations spéciales:	Aucune.

3.1.6. Tests d'intégration manuels

Objectif de test:	Assurer la fonctionnalité d'une partie du système
Technique:	Tester l'application via Heroku (le service qui permet d'héberger le serveur) en simulant des situations d'usage commun. Ces tests sont souvent faits en équipes de deux ou de trois.
Critère de complétion:	Les clients n'entrent pas dans un état incohérent et ils ne s'éteignent pas. Le serveur doit rester fonctionnel et dans un bon état. La fonctionnalité testée doit respecter l'exigence qui lui appartient. Le test doit être validé par au moins un membre de l'équipe n'ayant pas participé à son implémentation.
Considérations spéciales:	Lors des tests manuels, on ne vise pas à tester chaque exigence granulaire du SRS, mais bien un ensemble d'exigences qui constituent une fonctionnalité. Ces tests ne sont pas documentés ou effectués formellement. Ils sont

	généralement effectués lors du développement.

3.1.7. Tests de qualité de code

Objectif de test:	Assurer la qualité et la réutilisabilité du code.
Technique:	Rouler les “Linters” établis dans le SRS. Peer Reviews.
Critère de complétion:	Pas d’avertissements ou d’erreurs lors de la compilation (transpilation) du code ni lorsqu’on roule le “Linter” établi dans le SRS.
Considérations spéciales:	Certaines parties du systèmes peuvent utiliser des règles pour surcharger le “Linter” dans des cas où les règles que nous avions choisies ne sont pas applicables.

3.2. Outils

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

Type de test	Outil
Automatiques Android	JUnit
Intégration manuels Android	Android Emulator
Automatiques Angular	Jasmine
Intégration manuels Angular	Electron
Automatiques Serveur	Jest
Intégrations manuels Serveur (indirect)	Heroku

4. Ressources

Les rôles de chaque membre de l'équipe sont similaires en ce qui concerne la discipline de tests. En effet, chaque membre a adopté le rôle de testeur global afin de notifier l'équipe de bogues trouvés dans l'ensemble des applications (client léger, client lourd et serveur).

Membre de l'équipe	Responsabilités
Bruno	Tester le serveur, notamment les tests d'intégration pour permettre d'assurer la qualité du code grâce aux tests de régression. De plus, il teste indirectement le client léger en l'utilisant sans faire partie de son développement.
William	Tester le serveur, notamment la base de données, ainsi que le client lourd, par l'entremise de tests automatiques et manuels dans le cadre de son développement.
Simon	Valider la cohérence et le fonctionnement de l'interface utilisateur du client mobile et l'intégration avec le serveur.
Antoine	Valider la cohérence et le fonctionnement de la logique du client mobile et des connexions avec le serveur.
Thomas	Tester les interactions du client lourd avec le serveur pour assurer la cohérence des données du client.
Vincent	Tester les interfaces du client léger ainsi que les requêtes http.

5. Jalons du projet

Jalon	Effort (h/p)	Date de début	Date de fin
Remise d'appel d'offres	96	26 septembre	2 octobre
Débogage pour la remise finale	400	2 novembre	1 décembre
Écriture du plan de tests et des résultats des tests	20	1 décembre	2 décembre