Équipe 106

Projet 3 : Fais-moi un dessin
Plan de tests logiciels

Version 1.1

Historique des révisions

Date	Version	Description	Auteur
2020-04-01	1.0	Rédaction de l'introduction	Amar Ghaly
2020-04-02	1.1	Rédaction des exigences à tester et de la stratégie de test	Amar Ghaly

Table des matières

Plan de tests logiciels

1. Introduction

Les tests représentent une étape importante d'un produit logiciel puisqu'ils vérifient le fonctionnement du logiciel tout en maintenant une qualité de code acceptable. Ce document a donc pour but d'illustrer le plan de tests logiciels du projet: Fais-moi un dessin. Dans ce plan de tests, nous illustrerons les cas de tests que nous avons jugés pertinents. Ainsi, nous débuterons en énumérant les exigences à tester. Ensuite, nous définirons la stratégie de tests et les ressources nécessaires. Finalement, nous détaillerons les jalons importants de cette partie du projet.

2. Exigences à tester

Exigences	**Tests associés**
3.1.1 Accueil	Tests unitaires et tests de fonctions
3.1.3 Clavardage – Canaux de discussion	Tests unitaires et tests de fonctions
3.1.4 Profil utilisateur et historique	Tests d'intégrité des données
3.1.8. Mode de jeu - Mêlée générale	Tests d'interface utilisateur, tests de performance
3.1.9. Mode de jeu - Sprint Solo	Tests d'interface utilisateur, tests de performance
3.1.10. Mode de jeu - Sprint Coopératif	Tests d'interface utilisateur et tests de performance
3.1.11. Mode de jeu - Un contre un	Tests d'interface utilisateur et tests de performance
3.1.12. Création d'un jeu - Manuelle	Tests de fonctions et tests d'intégrité des données
3.1.13. Création d'un jeu - Assistée I	Tests de fonctions et tests d'intégrité des données
3.1.16 Classement	Tests unitaires et tests d'intégrité des données

3. Stratégie de test

3.1. Types de test

3.1.1. Tests de fonction

Objectif de test:	L'objectif de ces tests est de vérifier que les fonctions retournent les valeurs attendues en fonction d'une entrée donnée. Les cas de tests doivent être cohérents avec les cas d'utilisation énoncés dans le SRS.
Technique:	Le testeur rédige des cas de tests puis fait des appels de fonctions avec des entrées valides et invalides. Il vérifie que la sortie de la fonction est appropriée à l'entrée. Il est important de ne pas négliger les cas d'erreurs.
Critère de complétion:	Le critère de complétion est d'avoir une couverture supérieure à 80% ainsi que d'obtenir les résultats attendus : fonction qui retourne la bonne valeur, la mauvaise valeur ou une erreur.
Considérations spéciales:	*Puisqu'il s'agit de tests en boîte noire, le testeur ne doit pas considérer l'intérieur de la fonction, c'est-à-dire qu'il est inutile de consulter le code source.*

3.1.2. Tests d'interface usager

Objectif de test:	L'objectif de ces tests est de vérifier que la navigation sur les interfaces du logiciel est appropriée. Cela signifie que l'utilisateur est correctement dirigé d'une page à l'autre, et ce, sur le client lourd et léger. De plus, ces tests permettront de vérifier la cohérence entre nos deux interfaces (pages, boutons, messages d'erreurs, titres semblables).
Technique:	Pour effectuer des tests d'interface usager, le testeur doit se mettre à la place de l'utilisateur. Il doit rédiger des listes de manipulations à effectuer de manière subséquente. Cela correspond aux chemins que l'utilisateur doit emprunter s'il veut faire une action précise avec le logiciel (ex. : changement de pseudonyme).
Critère de complétion:	Le critère de complétion correspond à degré de cohérence entre les deux interfaces (client léger et client lourd) ainsi que chaque action possède le changement d'état voulu de l'application.
Considérations spéciales:	*Idéalement, les testeurs seraient les personnes les moins familières avec les interfaces de l'application logiciel.*

3.1.3. Tests d'intégrité des données

Objectif de test:	L'objectif de ces tests est de vérifier que les données contenues dans la base de données demeurent intégrales.

Technique:	Pour effectuer les tests d'intégrité des données, le testeur doit accéder directement à la base de données et la modifier. Il peut ajouter des données valides et invalides. Ensuite, le testeur peut laisser un temps s'écouler entre la modification de données et leur analyse. Au moment de l'analyse, le testeur devra s'assurer que les données sont fiables et crédibles. Par exemple, une donnée n'ayant pas été modifiée est demeurée la même tout au long de son cycle de vie.
Critère de complétion:	Le critère de complétion consiste à assurer que les accès à la base de données n'influencent pas la fiabilité et la crédibilité des données qu'elle contient.
Considérations spéciales:	*Ces tests doivent s'effectuer uniquement via la plateforme en ligne de MongoDB, c'est-à-dire sans faire usage des interfaces des clients légers et lourds.*

3.1.4. Tests de performance

Objectif de test:	L'objectif de ces tests est de vérifier que les temps de réponse et les taux de transaction de l'application logicielle sont conformes aux exigences contenues dans le SRS.
Technique:	Pour ce faire, le testeur peut rédiger des scripts constitués d'échanges avec le serveur. À travers ces scripts, il est possible de mesurer le temps de réponse. Ensuite, en servant d'outils tels qu'Excel ou simplement les statistiques de MongoDB, il peut déterminer le temps de réponse moyen à chaque requête.
Critère de complétion:	Le critère de complétion doit vérifier que le temps de réponse moyen et les taux de transaction sont inférieurs ou égaux aux exigences du SRS.
Considérations spéciales:	---

3.1.5. Tests unitaires

Objectif de test:	L'objectif de ces tests est de vérifier que les fonctions retournent les valeurs attendues en fonction d'une entrée donnée.
Technique:	Le testeur rédige des cas de tests en se servant d'un graphe de flot de contrôle pour s'assurer de couvrir toutes les branches. Ensuite, il fait des appels de fonctions appropriées à chaque branche. Il vérifie que la sortie de la fonction est adéquate.
Critère de complétion:	Le critère de complétion est d'avoir une couverture de branche supérieure à 80%.
Considérations spéciales:	*Puisqu'il s'agit de tests en boîte blanche, le testeur peut considérer l'intérieur de la fonction.*

3.2. Outils

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

Type de tests	Outil
3.1.1. Tests de fonctions	Samsung Galaxy Tab E avec Android 9, PC avec Windows 10
3.1.2. Tests d'interface utilisateur	Samsung Galaxy Tab E avec Android 9, PC avec Windows 10
3.1.3 Tests d'intégrité des données	MongoDB
3.1.4 Tests de performance	Serveur NodeJs, MongoDB et/ou Excel
3.1.5 Tests unitaires	Android studio en Kotlin avec les cadriciels Mockito et JUnit, Visual Studio en C# avec le cadriciel xUnit

4. Ressources

4.1. Équipe de test

Rôle	Membre de l'équipe	Responsabilités
Concepteur et testeur des tests unitaires client léger	Amar Ghaly	Rédiger les cas de tests, le graphe de flot de contrôle et les tests.
Concepteur et testeur des tests unitaires client lourds	Karima Salem	Rédiger les cas de tests, le graphe de flot de contrôle et les tests.
Concepteur et testeur des tests d'interface utilisateur	Émilio Gagnon	Déterminer les manipulations menant à une action spécifique. Vérifier que la navigation entre les pages des clients légers et lourds est cohérente et conforme aux exigences.

Concepteur et testeur des tests de fonctions du client léger	Nicole Joyal	Rédiger les cas d'utilisations en fonction des exigences fonctionnelles du SRS. Tester les fonctions en mode boîte noire.
Concepteur et testeur des tests d'intégrité des données	Zakari Rahal	Ajouter ou modifier des données contenues dans la base de données. Analyser la fiabilité et la crédibilité des données dans leur cycle de vie.
Concepteur et testeur des tests de performance	Hubert Michaud-Dufour	Rédiger les scripts de commandes du serveur. Mesurer les temps de réponse et les taux de transaction. Déterminer le temps de réponse moyen et les taux de transaction moyens.

4.2. Système

Afin de parvenir aux objectifs contenus dans ce plan de tests, il faut que chaque ressource humaine possède les ressources système nécessaires. Tout d'abord, l'exécution des tests doit s'effectuer sur un minimum de 2 tablettes et 2 ordinateurs. L'environnement des tablettes doit être de version minimale Android 4.1.2 et celui des ordinateurs, Windows 10. Ensuite, chaque outil matériel doit posséder un minimum de 1GB de mémoire vive. En ce qui concerne les tests unitaires, les cadriciels pour tests unitaires du client léger doivent être Mockito et JUnit et celui du client lourd xUnit.

5. Jalons du projet

Jalon	Effort	Date de début	Date de fin
Planification des cas de tests	3	2020-04-02	2020-04-02
Conception des tests de fonction	2	2020-04-03	2020-04-04
Conception des tests d'interface usager	2	2020-04-03	2020-04-04
Conception des tests unitaires	2	2020-04-04	2020-04-06
Conception des tests d'intégrité de données	2	2020-04-04	2020-04-05
Conception des tests de performance	2	2020-04-05	2020-04-06
Écriture des tests	5	2020-04-06	2020-04-10