Équipe 207

Fais-moi un dessin
Plan de tests logiciels

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2021-03-09	1.0	Rédaction initiale de l'introduction	Marc-Olivier Riopel Julien Witty
2021-04-18	2.0	Révision du plan et correction	Marc-Olivier Riopel Julien Witty

Table des matières

Plan de tests logiciels

1. Introduction

Le but du plan de tests est principalement de décrire les différents aspects de notre application que nous voulons tester. Ces aspects peuvent être des cas d'utilisation, des exigences fonctionnelles, des exigences non-fonctionnelles, etc. Le plan de tests nous permet également de décrire les différents types de tests qui sont utilisés pour valider notre application ainsi que les outils et ressources nécessaires pour y arriver. Finalement, ce plan nous permet de décrire les différents jalons associés aux tests pour pouvoir segmenter les différentes étapes des tests.

2. Exigences à tester

Exigences	**Tests associés**
3.1.1.1 Clavardage - Intégration	- Tests de fonction - Tests d'interface usager
3.1.1.2 Clavardage - Canaux de discussion	Tests de fonction Tests d'interface usager Tests intégrité de données Tests de performance Tests de stress Tests de volume Tests de sécurité
Profil utilisateur (Exigences de 3.1.2.1 à 3.1.2.12)	Tests de fonction Tests d'interface usager Tests d'intégrité de données Tests de sécurité
Historique (Exigences de 3.1.2.16 à 3.1.2.23)	Tests de fonction Tests d'interface usager
Création de parties (Exigences de 3.1.3.1 à 3.1.3.9)	Tests de fonction Tests d'interface usager Tests de stress Tests de charge Tests de volume
Salle d'attente des parties (Exigences de 3.1.3.12 à 3.1.3.17)	Tests de fonction Tests d'interface usager Tests de stress
Déroulement de la partie classique (Exigences de 3.1.3.17.1 à 3.1.3.17.19)	Tests de fonction Tests d'interface usager Tests de stress Tests de performance Tests de charge Tests de volume Tests intégrité de données

Outils de dessin (Exigences de 3.1.3.17.20 à 3.1.3.17.30)	Tests de fonction Tests d'interface usager
3.1.3.18 - Sprint solo	Tests de fonction Tests d'interface usager Tests de stress Tests de performance Tests de charge Tests intégrité de données
3.1.3.19 - Sprint coopératif	Tests de fonction Tests d'interface usager Tests de stress Tests de performance Tests de charge Tests intégrité de données
3.1.4 - Création de paires mot-image	Tests de fonction Tests intégrité de données Tests interface usager Tests de volume
3.1.5 - Personnalité des joueurs virtuels	Tests de fonction
3.1.6 - Effets visuels et sonores	Tests de fonction Tests d'interface usager
3.1.7 - Tutoriel	Tests de fonction Tests d'interface usager
3.1.8 - Tableau de classement	Tests d'interface usager Tests de fonction
3.1.9 - Suggestion de mots	Tests d'interface usager Tests de fonction
3.1.10 Téléversement d'un dessin d'une partie vers le joueur virtuel	Tests de fonction Tests intégrité de données Tests interface usager Tests de volume
3.1.11 Partie privée avec identifiant de la partie	Tests de fonction Tests d'interface usager Tests de stress
3.1.12 Vote sur dessins des joueurs virtuels	Tests de fonction Tests intégrité de données Tests interface usager
3.1.14 - Fil d'image	Tests interface usager
4.1 - Utilisabilité	Tests interface usager
4.3 - Performance	Tests de performance Tests de charge

4.6 - Sécurité	-Tests de sécurité

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 cas d'utilisation soient implémentés adéquatement et qu'ils soient en accord avec les exigences fonctionnelles du système.
Technique:	Pour valider les tests de fonction, il faut vérifier le comportement de chaque cas d'utilisation pour les différentes exigences afin de couvrir toutes les branches du programme. De cette manière, on doit effectuer des tests manuels de cas d'utilisation (Manual Use Case Testing).
Critère de complétion:	Pour atteindre une complétion de ce type de tests, il faut que les résultats démontrent que les différentes fonctionnalités du système ont le comportement attendu.
Considérations spéciales:	Pas de considération spéciale.

3.1.2. Tests d'interface usager

Objectif de test:	L'objectif de ces tests est de vérifier que l'utilisateur est en mesure de naviguer et d'utiliser l'application de manière adéquate avec l'interface présentée.
Technique:	Pour valider les tests d'interface usager, il est possible d'effectuer des tests d'acceptabilité des utilisateurs (User Acceptability Testing). Grâce à cette technique, il est possible de tester si notre interface est bel et bien appropriée pour les utilisateurs. Il est également possible d'effectuer des tests manuels pour s'assurer que notre interface fonctionne comme prévu. De plus, il a été décidé de faire des tests exploratoires (Exploratory testing). Les développeurs de l'équipe utilisent l'application sans contrainte dans le but de trouver des bogues critiques.
Critère de complétion:	L'utilisateur doit être en mesure d'accomplir les tâches qui lui sont demandées. Pour valider les tests manuels, nous pouvons vérifier que notre interface à tester convient aux exigences du projet fixées au début de celui-ci. Pour atteindre la réussite des tests exploratoires, il faut être en mesure d'utiliser l'application sans rencontrer de problèmes qui nuisent à notre expérience.
Considérations spéciales:	Le test d'acceptabilité doit tenir du groupe cible visé par l'application.

3.1.3. Tests d'intégrité des données

Objectif de test:	L'objectif de ces tests est de vérifier que les données stockées par le système sont accessibles et manipulées de la manière espérée.
Technique:	Pour effectuer ce type de tests, il faut faire des requêtes au serveur et ensuite valider les changements dans la base de données pour vérifier les changements apportés aux données par les requêtes.
Critère de complétion:	Pour valider les résultats de ce type de tests, il faut que les données soient modifiées comme prévu par la définition de la requête (Définition décrite dans le protocole de communication, par exemple).
Considérations spéciales:	Pas de considération spéciale.

3.1.4. Tests de performance

Objectif de test:	Le test de performance a pour but de vérifier que les temps de traitement pour les différentes requêtes de l'application ne nuisent pas à l'expérience utilisateur..
Technique:	Mesurer le temps pour qu'une requête ou une action soit complétée lorsque l'application est utilisée normalement.
Critère de complétion:	Le temps de réponse doit respecter les temps préétablis dans le document d'exigences. L'application doit garder la fluidité pour toutes les opérations effectuées.
Considérations spéciales:	Pour les tests concernant l'application mobile, il est important de considérer que ceux-ci sont effectués sur un émulateur de tablette Android et non directement sur une tablette. Cet écart peut créer des différences dans les résultats de performances.

3.1.5. Tests de charge

Objectif de test:	L'objectif des tests de charge est de s'assurer que notre système est capable de soutenir les charges de travail comme prévu dans le document d'exigences.
Technique:	Pour tester la charge, il a été décidé de connecter un certain nombre de clients tel que décrit dans le document d'exigences.
Critère de complétion:	Pour valider les résultats des tests, il faut que le système soit en mesure de gérer de manière adéquate les charges de travail établies dans les tests sans qu'il y ait de panne.

Considérations spéciales:	Pas de considération spéciale.

3.1.6. Tests de stress

Objectif de test:	Le but du test est de vérifier si l'application est capable de considérer des conditions extrêmes d'utilisation des ressources.
Technique:	Faire plusieurs requêtes en même temps pour différentes opérations clés exigeantes de l'application.
Critère de complétion:	Le système doit être capable de supporter les différents tests sans défauts.
Considérations spéciales:	Pas de considération spéciale.

3.1.7. Tests de volume

Objectif de test:	L'objectif de ces tests est de voir si l'application est en mesure de supporter de grosses quantités de données sans problème.
Technique:	Pour vérifier ce type de tests, on peut suivre la technique de tests manuels en essayant d'envoyer des dessins de très grande taille et d'envoyer de longs messages dans les canaux de discussion.
Critère de complétion:	Le serveur doit être capable de gérer les données tel qu'exigé par le document des exigences.
Considérations spéciales:	Pas de considération spéciale.

3.1.8. Tests de sécurité et de contrôle d'accès

Objectif de test:	L'objectif de ces tests est de s'assurer que l'application est protégée contre des attaques d'utilisateurs malveillants
Technique:	Pour vérifier ce type de tests, on peut essayer d'intercepter le trafic réseau pour voir si les informations sensibles sont bel et bien encryptées. Le contrôle d'accès est testé s'il est possible de se connecter à l'application avec de mauvais identifiants de connexion.

Critère de complétion:	Pour valider les résultats de ces tests, il faut s'assurer qu'on ne peut pas intercepter des informations sensibles sur le trafic réseau et qu'on ne peut pas avoir accès à des sections restreintes de l'application sans s'authentifier convenablement au préalable.
Considérations spéciales:	Pas de considération spéciale.

3.2. Outils

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

Type de test	Outil
Test de fonction	- Client léger - Client lourd - Postman (Pour le REST API) - Hoppscotch (Pour Socket.io)
Tests d'interface usager	- Client léger - Client lourd
Tests d'intégrité des données	- Client léger - Client lourd - Postman (Pour le REST API) - Hoppscotch (Pour Socket.io)
Tests de performance	- Client léger - Client lourd - Profiler (Android) - Inspecteur (Desktop) - CloudWatch (AWS)
Tests de stress	- Client léger - Client lourd - Profiler (Android Studio) - Inspecteur (Google chrome) - CloudWatch (AWS)

	- Jmeter
Test de charge	- Client léger - Client lourd - Profiler (Android Studio) - Inspecteur (Google chrome) - CloudWatch (AWS) - Jmeter
Tests de volume	- Client léger - Client lourd - Tableau de bord Atlas - Postman - CloudWatch (AWS) - Postman (Pour le REST API) - Hoppscotch (Pour Socket.io) - Jmeter
Tests de sécurité et de contrôle d'accès	- Client léger - Client lourd - WireShark

4. Ressources

4.1. Équipe de test

Rôle	Membre de l'équipe	Responsabilités
Testeur mobile	Julien Witty	- Exécuter les tests relatifs à l'application mobile du système - Rédaction de la section du rapport sur les résultats de tests de l'application mobile - Gestion des correctifs des bogues trouvés sur l'application mobile
Testeur serveur	Marc-Olivier Riopel	- Exécuter les tests relatifs au serveur - Rédaction de la section du rapport sur les résultats de tests du serveur - Gestion des correctifs des bogues trouvés sur le serveur
Testeur lourd	Marc-Alain Tétreault	- Exécuter les tests relatifs à l'application du client lourd du système - Rédaction de la section du rapport sur les résultats de tests de l'application du client lourd - Gestion des correctifs des bogues trouvés sur l'application du

4.2. Système

4.2.1 Client léger et Profiler

Le client léger nécessite l'environnement de programmation Android Studio. Il faut créer un émulateur avec les spécifications de la tablette Android. Android Studio s'occupe d'attacher le processus de l'application à développer à l'émulateur. Finalement, pour utiliser le profileur, il faut attacher le profileur à l'émulateur avec l'application.

4.2.2 Client lourd et Inspecteur

Le client lourd est testé sur le navigateur google chrome en plus de mesurer de tester avec la version exécutable d'Electron. Il faut avoir le Node Package Manager d'installé (npm) ainsi que toutes ces dépendances pour pouvoir exécuter l'environnement de programmation.

4.2.3 Postman

Pour pouvoir analyser la communication HTTP entre les clients et le serveur, on doit ouvrir une session avec Postman (En ayant l'agent Postman d'installé sur son ordinateur local) et s'authentifier. Ensuite, il est possible d'envoyer et de recevoir des réponses avec le serveur pour tester les routes du REST API en utilisant les paramètres appropriés pour chaque requête.

4.2.4 Hoppscotch

Pour pouvoir analyser la communication par socket entre les clients et le serveur, on doit ouvrir une session avec Hoppscotch et entrer les paramètres de communication nécessaires pour que le serveur sache à quel utilisateur il répond. Ainsi, il est possible d'envoyer et de recevoir des messages *socket* avec le serveur pour valider les tests.

4.2.5 Tableau de bord Atlas

Pour pouvoir analyser les métriques de la base de données, il faut s'authentifier avec un compte administrateur de la

base de données sur le site Atlas et aller dans la section *Metrics* de la collection que l'on veut observer. De cette manière, on peut voir les transmissions de données qui s'effectuent en temps réel à l'aide de graphiques.

4.2.6 Cloudwatch (AWS)

Pour pouvoir analyser les métriques des instances AWS, il faut s'authentifier avec un compte administrateur de l'instance sur le site de AWS et aller dans la section *Dashboard* de Cloudwatch. Ainsi, il est possible de voir les différents graphiques liés à l'utilisation de chacune des instances.

4.2.7 Wireshark

Pour pouvoir effectuer les tests nécessitant Wireshark, il faut installer l'application sur son ordinateur et ouvrir une session. Avec une session ouverte en mode écoute du réseau, il est possible de détecter les communications réseau entre l'ordinateur et Internet.

4.2.8 Jmeter

Pour pouvoir effectuer les tests nécessitant Jmeter, il faut installer l'application sur son ordinateur. Il est ensuite possible de faire des requêtes pour tester la capacité du serveur à gérer les requêtes de différentes manières.

5. Jalons du projet

Tout au long du projet, les fonctionnalités étaient intégrées à l'ensemble du projet via des merge requests. Ainsi, à chaque fois qu'une personne terminait une fonctionnalité, les autres membres de l'équipe devaient valider la fonctionnalité. De ce fait, cela a permis d'avoir une branche de développement plus stable.

Jalon	Effort	Date de début	Date de fin
Acceptabilité des tests concernant les clavardages, les profils utilisateur, les historiques de partie ainsi que les fonctionnalités relatives à la partie classique.	20h	15 février 2021	9 mars 2021
Acceptabilité des tests concernant les sprints solos, les sprints coop, la création paire mot-image, les effets sonores et visuels ainsi que les personnalités des joueurs virtuels.	15h	9 mars 2021	26 mars 2021
Acceptabilité des tests concernant le tutoriel, le tableau de classement, les suggestions de mots, ainsi que le téléversement des dessins d'une partie.	15h	26 mars 2021	5 avril 2021
Acceptabilité des tests concernant les parties privées, les votes sur les dessins et le fil d'images	10h	5 mars 2021	16 avril 2021
Acceptabilité des tests de l'ensemble du système	6h	16 avril 2021	19 avril 2021