Équipe 111 (Stack Overwork)

Projet d’évolution logiciel
Plan de tests logiciels

Version 1.1

Historique des révisions

Date	Version	Description	Auteur
2021-04-16	1.0	Rédaction des sections 1, 2, 3 et 4	Raphaël Lasalle Théo Turell
2021-04-19	1.1	Ajustements aux sections 3 et 4 Rédaction de la section 5	Raphaël Lasalle Jaafar Kaoussarani

Table des matières

Plan de tests logiciels

1. Introduction

Ce document décrit le processus mis en place pour tester le fonctionnement de l'application *Fais-moi un dessin*. Il décrit en détails les exigences du document de spécification des requis (SRS) qui sont testées, ainsi que les stratégies employées pour tester les exigences en question. Il liste également les ressources utilisées pour les tests ainsi que les jalons du projet concernés.

2. Exigences à tester

Exigences	Tests associé
3.1 Gestion de compte	- Tests de fonction - Tests d'interface usagers - Tests d'intégrité des données - Tests de sécurité et de contrôle d'accès Test de charge
3.2 Clavardage 3.3 Clavardage Client Lourd	- Tests d'interface usagers - Tests d'intégrité des données - Tests de volume
3.5 Profil Privé 3.5 Profil Public 3.7 Historique des partie 3.8 Historique de connexion	- Tests d'interface usagers - Tests d'intégrité
3.9 Interface de dessin	- Tests de fonction - Tests d'interface usagers - Tests de charge - Tests de volume
3.10 Menu et fonctionnalité générales	- Tests d'interface usagers
3.11 Création d'une partie	- Tests d'interface usagers - Tests d'intégrité des données - Tests de performances - Tests de charges - Tests de stress
3.12 Menu de sélection de salles d'attentes	- Tests d'interface usagers - Tests d'intégrité des données - Tests de stress - Tests de volume
3.13 Salle d'attente d'une partie 3.14 Salle d'attente d'une partie "Classique" 3.15 Salle d'attente d'une partie "Chacun pour soi"	- Tests d'intégrité des données

3.16 Déroulement d'une partie 3.17 Mode de jeu "Classique" 3.18 Mode de jeux "Chacun pour soi"	- Tests de fonction - Tests d'interface usager - Tests d'intégrité des données - Tests de performance - Tests de charge - Tests de stress - Tests de volume
3.19 Création d'une paire mot-image	- Tests de fonction - Tests d'interface usager - Tests d'intégrité des données - Tests de volume
3.20 Joueurs Virtuels	- Tests de fonction - Tests de performance - Tests de charge - Tests de stress
3.21 Effets visuels et sonores	- Tests de fonction - Tests d'interface usager
3.22 Tutoriel 3.23 Tutoriel client lourd	- Tests de fonction - Tests d'interface usager
3.25 Écran de fin de partie	- Tests de fonction - Tests d'interface usager - Tests d'intégrité des données - Tests de volume
3.26 Système relationnel	- Tests de fonction - Tests d'interface usager
4.1 Utilisabilité	- Tests d'interface usager - Tests de performances - Tests de configuration
4.3 Performances	- Tests de performance
4.6 Sécurité	- Tests de sécurité et de contrôle d'accès

3. Stratégie de test

3.1. Types de test

3.1.1. Tests de fonctions

Objectif de test:	Vérifier que les logiques à l'intérieur de fonction sont exécutés correctement
Technique:	Des tests unitaires
Critère de complétion:	Plus de 95% des cas de tests sont validés
Considérations spéciales:	N/A

3.1.2. Tests d'interface usager

Objectif de test:	Vérifier que l'interface est intuitive et utilisable par un nouvel utilisateur sans délais ou demandes d'explications excessives.
Technique:	Un nouvel utilisateur face à l'application et reçoivent une série d'instructions sur les actions à effectuer avec l'interface.
Critère de complétion:	Chaque utilisateur est capable de compléter chaque tâche qui lui est assignée avec un délai d'au plus 15 secondes. Aucun des deux utilisateurs n'a de critique majeure à faire sur l'interface et son utilisation.
Considérations spéciales:	La réussite de ce cas de test dépend en partie de la rétroaction des utilisateurs qui testent l'interface, qui ne peut pas être entièrement quantifiée par des métriques.

3.1.3. Tests d'intégrité des données

Objectif de test:	vérifier que les données écrites et lues dans la base de données sont préservées lors de leur transmission et de leur traitement.
Technique:	inscrire un nouvel utilisateur à l'application. vérifier que les statistiques utilisateurs sont mises à jour correctement après une partie.
Critère de complétion:	Les données sont mises à jour correctement dans la base de données et affichées correctement sur le client
Considérations spéciales:	N/A

3.1.4. Tests de performance

Objectif de test:	Vérifier que les métriques de performance du serveur sont dans des limites acceptables
Technique:	vérification des métriques générées par le serveur Heroku : - utilisation de la capacité totale du serveur ("dyno load") ne doit pas dépasser 90% - nombre de requêtes par seconde : ne doit pas dépasser 10 requêtes par seconde - délais maximal de traitement d'une requête : ne doit pas dépasser 5000 ms - utilisation maximale de la mémoire : ne doit pas dépasser 50% de la mémoire totale
Critère de complétion:	les métriques respectent les contraintes posées
Considérations spéciales:	N/A

3.1.5. Tests de charge

Objectif de test:	Vérifier que le serveur est capable d'accommoder une charge élevée.
Technique:	Deux développeurs font connecter chacun de leur côté 8 comptes utilisateurs en succession rapide.
Critère de complétion:	Le serveur accepte tous les comptes sans panne ou dégradation visible de performance (ex : latence trop élevée, dénis de service)
Considérations spéciales:	N/A

3.1.6. Tests de stress

Objectif de test:	Vérifier que le serveur est capable d'orchestrer plusieurs parties en simultanée sans baisse de performance perceptible.
Technique:	5 parties en mode "chacun pour soi" comportant 8 joueurs chacune sont lancées en simultané. 5 parties en mode "classique" comportant chacune 2 utilisateurs et 2 joueurs virtuels sont lancées en simultané. (les 5 parties "chacun pour soi" sont exécutées séparément de celles en mode classique).
Critère de complétion:	Chaque partie se déroule sans baisse de performance visible (délais dans l'envoi de données, effet de saccade sur les dessins…) Le serveur reste opérationnel pendant toute la durée des parties.
Considérations spéciales:	N/A

3.1.7. Tests de volume

Objectif de test:	Vérifier que le serveur est capable de traiter et d'envoyer une grande quantité de données
Technique:	Reproduire le test du **3.1.6** et s'assurer que les 5 parties finissent au même moment, c'est -à -dire arrivent en même temps sur leur écran de fin de partie.
Critère de complétion:	Chaque écran de fin de partie s'affiche avec les données correctes sans délais excessif (au plus 5 secondes).
Considérations spéciales:	N/A

3.1.8. Tests de sécurité et de contrôle d'accès

Objectif de test:	Vérifier qu'il n'est pas possible de se connecter à plusieurs clients sur un même compte utilisateurs. Vérifier qu'il n'est pas possible d'accéder aux outils développeurs sur la version finale de l'application
Technique:	plusieurs développeurs essaient de se connecter en simultané à un seul compte utilisateur. Un développeur essaie d'ouvrir les outils développeurs dans l'application.

Critère de complétion:	Un seul développeur parvient à se connecter au compte cible. Les outils développeurs ne sont pas accessibles.
Considérations spéciales:	N/A

3.1.9. Tests de configuration

Objectif de test:	Vérifier que l'application est fonctionnelle sur plusieurs machines et systèmes d'opérations différents.
Technique:	l'application est exécutée sur windows, sur un ordinateur de bureau (desktop) et sur un ordinateur portable.
Critère de complétion:	l'application fonctionne sans baisse de performance majeure sur chaque machine et système d'opération.
Considérations spéciales:	N/A

3.2. Outils

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

Type de test	Outil
Tests de fonction	Karma, Jasmine
Tests d'interface usager	Manuel
Tests d'intégrité des données	Manuel
Tests de performance	Heroku
Tests de charge	Heroku
Tests de stress	Heroku
Tests de volume	Manuel, Heroku
Tests de sécurité et de contrôle d'accès	Manuel
Tests de configuration	Manuel

4. Ressources

4.1. Équipe de test

Rôle	Membres de l'équipe	Responsabilités
Analyseur	Raphaël Lasalle Jaafar Kaoussarani	Observer et interpréter les donnés donner par Heroku et mettre dans le rapport toutes les observations
Testeur	Raphaël Lasalle Théo Turell	Effectue toutes les manipulations qui requiert une intervention humaine

4.2. Système

Nous avons fait le choix de ne tester que le client lourd pour les tests de fonction pour plusieurs raisons. Notamment, le fait de ne pas avoir à passer par un émulateur et que le client lourd a eu ses fonctionnalités fini avant le client léger, les membres de l'équipe s'occupant du client lourd furent les membre s'occupant des tests.

Pour la réalisation des tests, nous utiliserons le même cadriciel utilisé dans le projet LOG2990. Ce choix s'impose tant pour la familiarité que nous avons avec, mais aussi par le fait que nous avons créé notre projet sur une base provenant de LOG2990. Dans cette optique de familiarité, les tests seront exécutés sous Windows lorsque cela sera adéquat. De plus, en considérant le fait que qu'Electron utilise l'engin Chromium, nous testerons seulement dans un environnement simulant une fenêtre Chrome.

Le serveur étant hébergé sous Heroku, le service nous aide à obtenir des informations sur l'état du serveur. Nous les utiliserons pour compléter les résultats de tests.

5. Jalons des tests

Jalon	Effort (Heure-personne)	Date de début	Date de fin
1) Mise en place des environnements d'exécution des tests automatisés	10	19-02-2021	20-02-2021
2) Implémentation et exécution des tests manuels et automatisés	35	12-04-2021	19-04-2021