Équipe 102

Fais-moi un dessin
Plan de tests logiciels

Version 1.0

Historique des révisions

Date	Version	Description	Auteur
2020-11-07	1.0	Version initiale du plan de test	Vincent L'Ecuyer-Simard

Table des matières

Plan de tests logiciels

1 Introduction

Le présent document a pour objectif de présenter les tests logiciels planifiés dans le but de vérifier et de valider le logiciel final qui sera remis aux clients.

Dans un premier temps, une liste sommaire des exigences qui feront l'objet de tests ainsi que les types de tests auxquels elles seront soumises seront présentés. Par la suite, les stratégies de tests pour chacun des types de tests et les outils nécessaires à la réalisation des tests seront présentées. Ensuite les ressources humaines et les ressources matérielles nécessaires à la réalisation des tests seront exposées. Finalement, les jalons importants en lien avec la discipline de test seront explorés.

2 Exigences à tester

Exigences	**Tests associés**
3.1 Navigation dans le menu principal	Test de fonction, Test d'interface usager
3.2.2 Gestion des groupes de joueurs	Test de fonction, Test d'interface usager
3.2.3 Début d'une partie	Test de fonction, Test d'interface usager
3.3 Clavardage - intégration	Test de fonction, Test d'interface usager
3.4.1 Gestion des canaux de discussion	Test de fonction, Test d'interface usager
3.4.2 Canaux de discussion dédiés	Test de fonction, Test d'interface usager
3.5.1 Création d'un profil utilisateur	Test de fonction, Test d'interface usager
3.5.2 Connexion à un profil utilisateur	Test de fonction, Test d'interface usager
3.5.3 Consultation de ses propres informations de base	Test de fonction, Test d'interface usager
3.5.5 Consultation de l'historique	Test de fonction, Test d'interface usager
3.5.6 Consultation des statistiques	Test de fonction, Test d'interface usager
3.5.7 Thèmes sombre et clair	Test de fonction, Test d'interface usager
3.6.2 Éléments propres aux dessinateurs	Test de fonction, Test d'interface usager
3.6.3 Éléments propres aux devineurs	Test de fonction, Test d'interface usager
3.6.4 Mode mêlée générale	Test de fonction, Test d'interface usager
3.6.5 Mode sprint solo	Test de fonction, Test d'interface usager
3.6.6 Mode sprint coopératif	Test de fonction, Test d'interface usager
3.7 Création des paires mot-image	Test de fonction, Test d'interface usager
3.10 Tutoriel	Test de fonction, Test d'interface usager

Exigences	Tests associés
4.1 Utilisabilité	Test de fonction, Test de performance
4.2.4- 4.2.8 Persistance des données	Test d'échec/récupération
4.3 Performance	Test de performance, Test de charge
4.6 Sécurité	Test 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:	L'objectif des tests de fonction est de vérifier que les différentes exigences décrites dans le SRS ont été implémentées et que les cas d'utilisation peuvent être complétés par les utilisateurs.
Technique:	Avant de commencer les tests de fonction, un tableur Excel devra être créé. Celui-ci aura pour rôle d'assurer le suivi de ces tests. 2 feuilles devront être créées, une pour chacun des clients. Chaque ligne du tableur devra représenter les exigences atomiques visées par un test de fonction selon la section 2 du présent document. Avant le début d'une séance de test, le testeur devra indiquer son nom, la date et le dernier commit de la branche git utilisée pour le test au sommet d'une colonne qui répertoriera les résultats de la séance en cours. Par la suite, il devra vérifier si chaque exigence est implémentée de façon cohérente avec le SRS et, le cas échéant, avec le cas d'utilisation lié à l'exigence. Il devra noter le résultat de la vérification dans le tableur Excel dans la colonne attitrée à la séance en cours.
Critère de complétion:	Un test sera jugé comme réussi pour une exigence si le résultat est cohérent avec la description de celle-ci dans le SRS et, le cas échéant, avec le cas d'utilisation lié à l'exigence.
Considérations spéciales:	Les cas de tests devraient être réussis du premier coup. En cas de résultats variables dus à un bogue, le test devra être noté comme ayant échoué et des commentaires devront décrire les circonstances entraînant la présence du bogue pour faciliter le débogage subséquent.

3.1.2 Tests d'interface usager

Objectif de test :	L'objectif de ces tests est de s'assurer que l'interface usager fournit à l'utilisateur un accès et une navigation appropriés. Cela signifie que l'utilisateur doit être en mesure d'accéder aisément à la fonctionnalité de son choix, mais également que la fonctionnalité elle-même soit conviviale à utiliser.

Technique :	Avant de commencer les tests d'interface usager, un tableur Excel devra être créé. Celui-ci aura pour rôle d'assurer le suivi de ces tests. 2 feuilles devront être créées, une pour chacun des clients. Étant donné la profonde connaissance de l'interface par les développeurs, ces tests devront être effectués avec la participation de potentiels utilisateurs n'ayant pas de connaissances préalables à propos de ce logiciel. Le testeur devra d'abord répertorier dans le tableur Excel différentes actions à compléter par l'utilisateur (une action par ligne). Ces actions devront être relativement simples, mais suffisamment complexes pour permettre d'évaluer la capacité de l'interface à guider l'utilisateur. Ces actions devront avoir un point de départ et un point de fin clairement identifiable. De plus, lorsque le chemin entre le point de départ et le point de fin comporte des choix, l'action devra être non ambiguë. Un exemple d'une telle action pourrait être : "À partir du menu principal, créer un groupe de joueur nommé xyz dans le mode sprint solo en difficulté easy et ajoutez-y un joueur virtuel du nom de xyz". L'union des actions à compléter devra permettre de tester toutes les exigences atomiques visées par un test d'interface usager selon la section 2 du présent document. Une fois cette liste prête, le testeur devra préparer une colonne dans le tableur Excel en y indiquant son nom, le nom de l'utilisateur, la date et le dernier commit de la branche utilisée pour effectuer les tests. Par la suite, il devra rencontrer virtuellement l'utilisateur qui partagera son écran pour que le testeur voie les actions de l'utilisateur pendant le test ou encore le rencontrer en personne si les mesures sanitaires le permettent. Avant de commencer le premier test, le testeur devra demander à l'utilisateur d'éviter le plus possible de poser des questions directement au testeur en cours de test. Il demandera plutôt à l'utilisateur d'exprimer à voix haute tous les pensées et questionnements qui lui viennent pendant les tests. Le testeur notera les réflexions pertinentes durant les tests. Lorsque possible, le testeur devra tenter de faire concorder le point de fin d'une action au point de départ de l'action suivante. Lorsque cela n'est pas possible, il devrait plutôt guider l'utilisateur verbalement vers le prochain point de départ. Lorsque l'utilisateur se trouve au point de départ d'une action, le testeur lui transmettra alors textuellement l'action à accomplir. Par la suite, l'utilisateur tentera d'accomplir celle-ci et le testeur devra noter toute action (erreur, hésitation) ou réflexion (questionnement) dénotant un défaut de l'interface.
Critère de complétion :	Un test sera jugé comme réussi si l'utilisateur parvient à accomplir l'action sans erreur, hésitation ou questionnement important.
Considérations spéciales :	La nécessité de l'intervention du testeur pendant la complétion d'une action entraîne automatiquement un échec du test. La définition d'une erreur, hésitation ou questionnement non important est laissée à la discrétion du testeur, mais ne devrait pas entraîner plus de quelques secondes de retard par rapport à une complétion parfaite. En cas d'échec ou de réussite non parfaite, le testeur devra questionner l'utilisateur afin d'explorer les raisons de celles-ci et les noter dans le tableur Excel.

3.1.3 Tests de performance

Objectif de test:	L'objectif de ces tests sera de vérifier si différentes exigences non-fonctionnelles ont été atteintes dans des conditions minimales d'utilisation du logiciel.
Technique:	Avant de commencer les tests de performance, un tableur Excel devra être créé. Celui-ci aura pour rôle d'assurer le suivi de ces tests. 2 feuilles devront être créées, une pour chacun des clients. Chaque ligne du tableur devra représenter les exigences atomiques visées par un test de performance selon la section 2 du présent document. Avant le début d'une séance de test, le testeur devra indiquer son nom, la date et le dernier commit de la branche git utilisée pour le test au sommet d'une colonne qui répertoriera les résultats de la séance en cours. Les exigences non-fonctionnelles visées par des tests de performance sont toutes associées à une valeur quantitative à rencontrer. Ainsi, cette valeur devra être ajoutée dans le tableur Excel dans la colonne suivant celle des exigences. Avant le début du test, le testeur devra s'assurer qu'aucun autre utilisateur n'utilise le serveur pour la durée de la séance de test et que le serveur est prêt à recevoir des requêtes. Il devra également s'assurer que son poste de travail n'a pas d'autres applications voraces actives. Par la suite, il devra tester les exigences une après l'autre, en s'assurant de ne pas solliciter le serveur ou l'application au-delà du minimum requis pour exécuter le test en question. Le testeur devra noter les différents résultats quantitatifs dans le tableur Excel.
Critère de complétion:	Un test sera jugé comme réussi si le résultat quantitatif observé est dans l'intervalle espéré selon la description de l'exigence non-fonctionnelle.
Considérations spéciales:	

3.1.4 Tests de charge

Objectif de test:	L'objectif de ces tests sera de vérifier si différentes exigences non-fonctionnelles ont été atteintes dans des conditions précises d'utilisation du logiciel, soit celles correspondant à 2 parties en mode mêlée générale en cours.
Technique:	Avant de commencer les tests de charge, un tableur Excel devra être créé. Celui-ci aura pour rôle d'assurer le suivi de ces tests. 2 feuilles devront être créées, une pour chacun des clients. Chaque ligne du tableur devra représenter les exigences atomiques visées par un test de charge selon la section 2 du présent document. Avant le début d'une séance de test, le testeur devra indiquer son nom, la date et le dernier commit de la branche git utilisée pour le test au sommet d'une colonne qui répertoriera les résultats de la séance en cours. Les exigences non-fonctionnelles visées par des tests de charge sont toutes associées à une valeur quantitative à rencontrer. Ainsi, cette valeur devra être ajoutée dans le tableur Excel dans la colonne suivant celle des exigences elles-mêmes. Avant le début de chaque test, le testeur devra s'assurer qu'il y a deux parties de 4 joueurs humains en mode mêlée générale qui sont en cours. Cette situation

	permet de valider hors de tout doute que les exigences non-fonctionnelles sont atteintes lorsqu'une seule partie est en cours. Il devra également s'assurer que son poste de travail n'a pas d'autres applications voraces actives. Par la suite, il devra tester les exigences une après l'autre dans ces conditions d'utilisation du serveur. Le testeur devra noter les différents résultats quantitatifs dans le tableur Excel.
Critère de complétion:	Un test sera jugé comme réussi si le résultat quantitatif observé est dans l'intervalle espéré selon la description de l'exigence non-fonctionnelle .
Considérations spéciales:	

3.1.5 Tests de sécurité et de contrôle d'accès

Objectif de test:	L'objectif de ce test sera de vérifier que différentes exigences non-fonctionnelles en lien avec la sécurité ont été implémentées.
Technique:	Avant de commencer les tests de sécurité et de contrôle d'accès, un tableur Excel devra être créé. Celui-ci aura pour rôle d'assurer le suivi de ces tests. 2 feuilles devront être créées, une pour chacun des clients. Chaque ligne du tableur devra représenter les exigences atomiques visées par un test de sécurité et de contrôle d'accès selon la section 2 du présent document. Avant le début d'une séance de test, le testeur devra indiquer son nom, la date et le dernier commit de la branche git utilisée pour le test au sommet d'une colonne qui répertoriera les résultats de la séance en cours. Le testeur doit ensuite parcourir ces exigences une à une et vérifier si elles ont bien été implémentées.
Critère de complétion:	Un test sera jugé comme réussi pour une exigence si le résultat est cohérent avec la description de celle-ci dans le SRS.
Considérations spéciales:	Le testeur doit effectuer ces tests en boîte noire. Cela signifie qu'il ne doit pas, par exemple, chercher à voir dans le code source si le mot de passe est bel et bien crypté avant d'être stocké dans la base de données. Il doit plutôt créer un compte et aller vérifier dans la base de données que le mot de passe qui se trouve attaché à ce nouveau compte a été crypté.

3.1.6 Tests d'échec/récupération

Objectif de test:	L'objectif de ce test sera de vérifier que différentes exigences non-fonctionnelles en lien avec la fiabilité du système ont été implémentées.
Technique:	Avant de commencer les tests d'échec/récupération, un tableur Excel devra être créé. Celui-ci aura pour rôle d'assurer le suivi de ces tests. 2 feuilles devront être créées, une pour chacun des clients. Chaque ligne du tableur devra représenter les exigences atomiques visées par un test d'échec/récupération selon la section 2 du présent document. Avant le début d'une séance de test, le testeur devra indiquer son nom, la date et le dernier commit de la branche git utilisée pour le test au sommet d'une colonne qui répertoriera les résultats de la séance en cours. Plusieurs exigences non-fonctionnelles en lien avec la fiabilité du système impliquent la persistance de certaines données à la suite de la défaillance du

	serveur. Ainsi, le testeur devra d'abord répertorier les données accessibles pour chacune des exigences dans le tableur. Ensuite, il doit simuler une défaillance du serveur en demandant un redémarrage de celui-ci. Finalement, il doit vérifier, après le redémarrage, que toutes les données visées pas les exigences sont toujours accessibles en notant celles-ci dans le tableur et en vérifiant la concordance entre les données pré et post redémarrage du serveur.
Critère de complétion:	Un test sera jugé comme réussi si les données répertoriées sont toujours accessibles à la suite du redémarrage du serveur.
Considérations spéciales:	Il n'est pas attendu que le client se reconnecte automatiquement au serveur une fois que celui-ci sera à nouveau fonctionnel. Ainsi, il est important que le testeur attende le redémarrage du serveur avant de démarrer une nouvelle session sur un client.

3.2 Outils

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

Type de test	Outil
Test de fonction	Excel, Windows 10, Android Studio
Test d'interface usager	Excel, Windows 10, Android Studio, Utilisateur externe, Discord
Tests de performance	Excel, Windows 10, Android Studio, Chronomètre, Gestionnaire de tâche Windows, Explorateur de fichiers Windows, Liste des applications Android, Heroku Dashboard
Tests de charge	Excel, Windows 10, Android Studio, Chronomètre, Gestionnaire de tâche Windows, Explorateur de fichiers Windows, Liste des applications Android, Heroku Dashboard
Tests de sécurité et de contrôle d'accès	Excel, Windows 10, Android Studio, MongoDb Atlas, Postman
Tests d'échec/récupération	Excel, Windows 10, Android Studio, Heroku Dashboard

4 Ressources

4.1 Équipe de test

Rôle	Membre de l'équipe	Responsabilités
Concepteur de cas de test d'interface usager	Vincent L'Ecuyer-Simard	Concevoir les actions à accomplir par l'utilisateur afin de couvrir toutes les exigences pertinentes. Préparer le document Excel nécessaire à l'exécution des tests et le document de résultats des tests logiciels.
Concepteur des cas de test basés directement sur les exigences	Vincent L'Ecuyer-Simard	Préparer le document Excel nécessaire à l'exécution des tests et le document de résultats des tests logiciels en se basant directement sur les exigences ciblées.

Testeur des cas de test d'interface usager - Client lourd	Vincent L'Ecuyer-Simard Simon Berhanu	Compléter les tests nécessitant un utilisateur en passant à travers les différentes actions sur le client lourd et noter les résultats dans le document Excel et dans le document de résultats des tests logiciels.
Testeur des cas de test d'interface usager - Client léger	Abderrahim Ammour	Compléter les tests nécessitant un utilisateur en passant à travers les différentes actions sur le client léger et noter les résultats dans le document Excel et dans le document de résultats des tests logiciels.
Testeur des cas de test basés directement sur les exigences - Client lourd	Vincent L'Ecuyer-Simard	Compléter les tests basés directement sur les exigences sur le client lourd et noter les résultats dans le document Excel et dans le document de résultats des tests logiciels.
Testeur des cas de test basés directement sur les exigences - Client léger	Hakim Payman	Compléter les tests basés directement sur les exigences sur le client léger et noter les résultats dans le document Excel et dans le document de résultats des tests logiciels.

4.2 Système

Tous les membres impliqués dans la conception ou la réalisation des tests devront avoir accès à internet afin de pouvoir modifier les tableurs Excel et le document de résultats des tests logiciels puisque ceux-ci seront stockés dans notre répertoire d'équipe infonuagique.

Les testeurs de cas de test d'interface usager et les utilisateurs externes devront avoir accès au logiciel Discord et à une connexion internet permettant une communication audio-vidéo de qualité. L'utilisateur impliqué dans les tests pour le client lourd devra avoir accès à un ordinateur Windows 10 respectant les critères décrits dans le document d'architecture logiciel. De plus, il devra avoir accès à un exécutable du client lourd produit en mode release provenant d'une version stable du projet. L'utilisateur impliqué dans les tests pour le client léger devra avoir accès à Android Studio, son émulateur et d'une version stable du projet.

Les testeurs de cas de test basés directement sur les exigences pour le client lourd devront avoir accès à un ordinateur Windows 10 respectant les critères décrits dans le document d'architecture logiciel et à une connexion internet respectant ces mêmes critères. De plus, ils devront avoir accès à un exécutable du client lourd produit en mode release provenant d'une version stable du projet et à un navigateur pour accéder au Heroku Dashboard du serveur et à MongoDb Atlas.

Les testeurs de cas de test basés directement sur les exigences pour le client léger devront avoir accès à Android Studio, son émulateur, une version stable du projet et à une connexion internet respectant les critères du document d'architecture logiciel. Ils devront également avoir accès à un navigateur pour accéder au Heroku Dashboard du serveur et à MongoDb Atlas.

5 Jalons du projet

Jalon	Effort (hrs-pers)	Date de début	Date de fin
Conception des tests d'interface usagers	6	2020-11-12	2020-11-18
Conception des cas de test basés directement sur les exigences	2	2020-11-12	2020-11-18
Exécution des cas de test d'interface usager - Client lourd	5	2020-11-19	2020-11-29
Exécution des cas de test d'interface usager - Client léger	5	2020-11-19	2020-11-29
Exécution des cas de test basés directement sur les exigences - Client lourd	2	2020-11-19	2020-11-29
Exécution des cas de test basés directement sur les exigences - Client léger	2	2020-11-19	2020-11-29
Réexécution des tests préalablement échoués après correctifs (si nécessaire)	2	2020-11-30	2020-12-01