Plan de test

1. Introduction

L'Objectif de ce plan de test sera de valider la partie fonctionnalité.
En termes de périmètre, la couverture s'étendra à la création, la gestion, l'exécution et le suivi des résultats des tournois.

Notre scope comprend des fonctionnalités essentielles à les générations de tournois :

- La création des équipes à partir de joueurs
- Les créations de tournois

Quant à notre out of scope, il comprend la fonctionnalité qui permet de charger des fichiers afin de générer des équipes ou de charger des équipes prêtes à concourir.

Dans le but de nous assurer que notre programme répond aux fonctionnalités attendues, nous allons mettre en place différents types de test :

- Test Unitaires : Dans le but de tester les parties importantes de lié à la logique de notre code
- Test d'intégration : Ici notre souhait est de nous assurer que l'ajout de nouvelles fonctionnalités ne viennent pas casser le code existant
- Tests Fonctionnels : Pour vérifier chaque fonctionnalité contre ses spécifications.
- etc.

2. Environnement de Test

Le projet étant dans une phase de prototype. Nous exposerons les tests sur nos propres machines. Il n'est pas nécessaire d'automatiser les tests aujourd'hui ou de prévoir des environnements de déploiement

3. Analyse de risque

En nous basant sur le scope défini, nous avons réalisé une première analyse de risque.

3.1 Risques liés au fichier CSV

3.1.1 Risque : Format de fichier CSV incorrect

- Probabilité : Moyenne
- Impact : Élevé

Stratégie d'atténuation : Établir une spécification claire pour le format du fichier CSV, et mettre en place des vérifications rigoureuses pour détecter et gérer les erreurs de format.

3.1.2 Risque : Manque de données dans le fichier CSV

- Probabilité : Faible
- Impact : Moyen

Stratégie d'atténuation : Fournir des jeux de données de test complets et diversifiés pour garantir une couverture adéquate des scénarios possibles.

3.2 Risques liés à la génération d'équipe

3.2.1 Risque : Inéquité dans la composition des équipes générées

- Probabilité : Moyenne
- Impact : Élevé

Stratégie d'atténuation : Définir des critères clairs pour la génération d'équipes, et effectuer des ajustements périodiques pour assurer l'équité.

3.2.2 Risque : Difficultés dans la gestion des joueurs dans le système

- Probabilité : Faible
- Impact : Moyen

Stratégie d'atténuation : Mettre en place des mécanismes de gestion des joueurs robustes, et effectuer des tests de stress pour évaluer la capacité du système à gérer un grand nombre de joueurs.

3.3 Risques liés à la génération du tournoi

3.3.1 Risque : Erreurs dans la sélection des équipes pour le tournoi

- Probabilité : Moyenne
- Impact : Élevé

Stratégie d'atténuation : Impliquer les parties prenantes dans la définition des critères de sélection des équipes, et effectuer des revues régulières pour éviter les erreurs.

3.3.2 Risque : Problèmes de cohérence dans la structure du tournoi

- Probabilité : Faible
- Impact : Moyen

Stratégie d'atténuation : Établir des procédures de vérification rigoureuses pour la structure du tournoi, et automatiser autant que possible ces processus.

3.4 Risques liés à l'environnement de test

3.4.1 Risque : Incompatibilité des machines de test

- Probabilité : Faible
- Impact : Moyen

Stratégie d'atténuation :* Documenter les configurations des machines de test, et effectuer des tests préliminaires pour s'assurer de la compatibilité avant le début des tests.

3.4.2 Risque : Inconsistance des données de test sur les machines de test

- Probabilité : Moyenne
- Impact : Élevé

Stratégie d'atténuation : Mettre en place des procédures de nettoyage automatisées et des mécanismes de restauration des données pour maintenir la cohérence des environnements de test.

4. Réalisation des tests

4.1 Fichier CSV

4.1.1 Objectif

Vérifier que les fichiers CSV ne généreront pas d'anomalie dans notre système.

4.1.2 Pré-requis

Le fichier CSV reçu doit respecter un certain modèle défini avec des données valide.

4.1.3 Procédure

Le test se déroulera en plusieurs étape:

- Chargement du fichier CSV et des données
- Vérification que les données reçu par le fichier soit valident
- Transformation des données en données utilisable dans notre projet

4.1.4 Critères de réussite

Pour que le test soit valide, il doit valider plusieurs critères pour qu'il soit réussi.

- Aucune erreur ne doit arriver pendant l'exécution du test
- Les données doivent être correctement créées depuis le fichier

4.2 Génération d'équipe

4.2.1 Objectif

S'assurer que le programme génère correctement les équipes à partir des joueurs.

4.2.2 Pré-requis

Une liste de joueur doit être présent.

4.2.3 Procédure

Le test se déroulera en plusieurs étape:
- Récupération de la liste de joueur
- Lancement du processus de génération d'équipe
- Vérification des équipes en se basant sur les critères de réussite

4.2.4 Critères de réussite

Pour que le test soit valide, il doit valider plusieurs critères pour qu'il soit réussi.
- Le test s'est exécuté sans erreur
- Les équipes générées ne peuvent contenir plusieurs fois le même joueur
- Les équipes générées doivent contenir le même nombre de joueur, ce nombre est définie préalablement

4.3 Génération du tournoi

4.3.1 Objectif

Vérifier que le programme peut générer un tournoi avec plusieurs équipes en entrée.

4.3.2 Pré-requis

Pour générer le tournoi, une liste d'équipes est requise.

4.3.3 Procédure

Le test se déroulera en plusieurs étape:
- Récupération d'une liste d'équipe
- Lancement du programme de génération du tournoi
- Vérification du contenu du tournoi ainsi créer, ce tournoi doit valider les critères de réussite

4.3.4 Critères de réussite

Pour que le test soit valide, il doit valider plusieurs critères pour qu'il soit réussi.
- Aucune erreur ne doit arriver pendant l'exécution du test
- Le tournoi doit contenir toutes les équipes entrées

5. Planning des tests

5.1 Test unitaires

Les tests unitaires seront réalisés tout au long de l'avancement du projet, ils évolueront en même temps que le projet. Chaque test répondra à un composant logique du code.

5.2 Test d'intégration

Les tests d'intégration seront effectués à chaque nouvelles fonctionnalités ajoutées, pour repérer des potentiels problèmes et les résoudres le plus rapidement possible pour garantir le bon fonctionnement du programme.

5.3 Test fonctionnels

Exécuter les tests définis pour chaque fonctionnalité spécifique.