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 exécuterons 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, une première analyse de risque peut être établie pour identifier les risques qui pourraient affecter la qualité et la fiabilité du programme.

1. Risque de défaillance des tests unitaires : Les tests unitaires peuvent ne pas couvrir toutes les parties importantes du code, ce qui pourrait entraîner des défauts non détectés dans la logique du programme.

2. Risque de conflits d'intégration : L'ajout de nouvelles fonctionnalités peut potentiellement interférer avec le code existant, entraînant des conflits d'intégration et des bogues inattendus.

3. Risque de non-conformité aux spécifications : Les tests fonctionnels doivent garantir que chaque fonctionnalité répond aux spécifications définies. Cependant, il existe un risque que des fonctionnalités ne se comportent pas comme prévu, ce qui pourrait conduire à des cas d'utilisation incorrects ou à des résultats imprévus.

4. Risque de manque de couverture des tests : Bien que nous ayons identifié les tests unitaires, d'intégration et fonctionnels, il existe un risque que toutes les parties du programme ne soient pas testées, laissant ainsi des vulnérabilités non découvertes.

5. Risque de dépendances externes : Si le programme dépend de bibliothèques tierces ou de services externes, il existe un risque de dysfonctionnement ou de changement de comportement de ces dépendances, ce qui pourrait entraîner des problèmes dans notre programme.

6. Risque de performance : En fonction de la taille des données manipulées et de la complexité des algorithmes utilisés, il existe un risque de performances insatisfaisantes, ce qui pourrait affecter l'expérience utilisateur ou la capacité du programme à gérer de grandes quantités de données.

7. Risque de sécurité : Les fonctionnalités de création, gestion et suivi des tournois pourraient présenter des vulnérabilités de sécurité, telles que l'accès non autorisé aux données des utilisateurs ou la manipulation des résultats des tournois, ce qui nécessite une attention particulière lors des tests.

En résumé, ces risques doivent être pris en compte lors de la planification et de l'exécution des tests afin de minimiser les chances de défauts et de garantir la qualité du programme. Des mesures appropriées, telles qu'une couverture de test exhaustive, une analyse statique du code, et des tests de sécurité, peuvent être mises en place pour atténuer ces risques.