**Cahier de Charges**

« Réalisation de l’application web, mobile et bureautique Paris Sportifs Super Bowl »

**Table des Matières**

1. Introduction (Voir Page 3)
2. Public ciblé (Voir Page 3)
3. Disponibilité actuelle des ressources (Voir Page 3)
4. Analyse des besoins (Voir Page 3, 4)
5. Technologies choisies (Voir Page 4)
6. Gestion des risques (Voir Page 5)
7. Tests (Voir Page 5, 6)
8. Livrables (Voir Page 6)
9. Planning (Voir Page 7)
10. **Introduction**

Le Super Bowl est un événement majeur en Amérique du Nord, attirant l'attention non seulement des amateurs de sport, mais également des passionnés de musique grâce à ses performances de haut niveau. Cette année, notre mission est de développer une application web, mobile et de bureau permettant au public de participer à des paris sportifs lors de cet événement emblématique.

**2. Public ciblé**

Les applications sont destinées aux :

* passionnés de sport, notamment ceux qui suivent le football américain et qui souhaitent participer activement en plaçant des paris sur le Super Bowl.
* personnes moins intéressées par le football américain en général, mais qui sont attirées par l'enthousiasme de cette journée spéciale et qui souhaitent tout de même participer en plaçant des paris occasionnels.

**3. Disponibilité actuelle des ressources**

Nous avons déjà de l'expérience dans la création d'applications pour divers événements, mais nous n'avons jamais eu l'opportunité de travailler pour un client d'aussi grande envergure. Nous sommes passionnés par le sport, ce qui nous rend d'autant plus motivés pour ce projet. Le Super Bowl nous fournira les données relatives aux différents matchs.

**4. Analyse des besoins**

Le projet se divise en trois composantes principales, chaque application devant répondre à des besoins spécifiques :

***1. Application Web :***

* Fonctionnalité de paris sportifs avec une interface d'administration pour la configuration des paris par les gestionnaires.
* Interface utilisateur permettant de parier des montants sur les équipes sélectionnées, avec transaction financière réalisée physiquement au stade.
* Capacité pour les utilisateurs de consulter les matchs à venir et de placer des paris en amont.
* Interdiction de parier une fois qu’un match commence.

***2. Application Mobile :***

* Affichage personnalisé des matchs : seuls les matchs sur lesquels l'utilisateur a parié via l'application web sont visibles.
* Présentation des matchs passés en format grisé et mise en avant des matchs à venir.
* Les matchs en cours disposent un accès direct aux commentaires en temps réel et au score actualisé après chaque rafraîchissement
* Affichage détaillé pour l'utilisateur du montant parié et de l'équipe choisie.

***3. Application Bureautique :***

* Interface pour le démarrage et la gestion du suivi du match.
* Module de saisie de commentaires en direct et de mise à jour du score à chaque événement significatif durant le match.
* Fonctionnalité de clôture du match pour terminer le suivi et figer les résultats.

Ces applications doivent être conçues pour fonctionner de manière intégrée, avec un haut niveau de sécurité pour les transactions et une expérience utilisateur fluide et intuitive.

**5. Technologies choisies**

**Site Web :**

*Framework Backend* : Flask sera utilisé pour sa balance optimale entre fonctionnalités et simplicité, convenant parfaitement à un site web de complexité modérée.

*Langage de Programmation* : Python, pour sa polyvalence et sa puissance, servira de base pour l'ensemble du projet.

*Frontend* : HTML5 et CSS3 seront utilisés en combinaison avec le framework Bootstrap pour un développement rapide et réactif.

*Déploiement* : Fly.io a été choisi pour le déploiement du site web, en raison de sa gratuité et de sa facilité d'utilisation.

**Application Mobile :**

*Framework* : KivyMD sera utilisé en conjonction avec Python pour développer l'application mobile, favorisant l'uniformité avec le reste du projet.

*Environnement de Développement* : Visual Studio Code sera l'éditeur de code de choix pour le développement mobile.

**Application Bureautique :**

*Librairie Graphique* : Tkinter, en combinaison avec Python, fournira une méthode simple et portable pour le développement d'une application bureautique avec une interface graphique utilisateur.

**Base de Données :**

*Système de Gestion de Base de Données* : MySQL, pour sa vitesse, sa facilité d'utilisation, et ses fonctionnalités de sécurité, sera le système de gestion de base de données utilisé.

1. **Gestion des Risques**

La gestion des risques est intégrée tout au long du cycle de développement pour anticiper, atténuer et gérer les risques :

**Risque de Performance :**

*Problème :* Performances sous haute charge, notamment pendant le Super Bowl.

*Mitigation :* Des tests de charge seront effectués régulièrement, et une architecture évolutive sera mise en place.

**Risque Sécuritaire :**

*Problème :* Vulnérabilités de sécurité pouvant mener à des fuites de données.

*Mitigation :* Des audits de sécurité seront réalisés, et les meilleures pratiques de sécurisation des applications seront appliquées.

**Risque de Conformité :**

*Problème :* Respect des réglementations sur les jeux et paris sportifs.

*Mitigation :* Consultation avec des experts légaux pour assurer la conformité réglementaire dès les premières étapes du développement.

Le suivi continu et l'évaluation des risques nous permettront d'identifier rapidement les problèmes et de réagir avec des plans d'action appropriés pour assurer la réussite du projet.

1. **Tests**

Afin d'assurer que les applications web, mobile et bureautique répondent à des standards élevés de qualité et de performance, notre stratégie de test se divise en trois phases principales :

**Tests Unitaires :**

Objectif : Valider la plus petite partie testable du code écrit (fonctions, méthodes, classes).

Outils : Unittest.

Processus : Les développeurs rédigeront et exécuteront des tests unitaires pour chaque nouveau morceau de code avant toute intégration, s'assurant ainsi que chaque fonctionnalité fonctionne de manière isolée.

**Tests d'Intégration :**

Objectif : Tester les modules de l'application ensemble pour identifier tout problème dans les interactions entre différents composants du système.

Outils : pytest sera utilisé pour créer des tests d'intégration qui simuleront les interactions entre différents composants de l'application (comme les appels API, la base de données, et les services externes).

Processus : Après les tests unitaires, les composants seront combinés et testés dans un environnement de test qui imite de près la production pour s'assurer que l'ensemble fonctionne comme attendu.

**Tests de Charge avec Locust :**

Objectif : Évaluer la capacité de l'application à gérer un trafic utilisateur élevé et à maintenir la performance sous charge. Cela inclut la validation de la stabilité et la réactivité du système lorsqu'il est soumis à des volumes élevés de requêtes simultanées.

Outils : Locust, un outil de test de charge open-source écrit en Python, sera utilisé pour simuler des utilisateurs concurrents accédant à l'application et effectuant des actions spécifiques, comme visualiser des matchs.

Processus : Les tests de charge avec Locust évalueront la performance sous haute demande en simulant des utilisateurs virtuels qui effectuent des actions telles que la visualisation des matchs. La configuration ajustera le nombre et la fréquence des utilisateurs pour imiter le trafic réel. Les tests seront surveillés pour les temps de réponse et les erreurs, et les résultats analyseront la robustesse de l'application. Les insights permettront d'optimiser le système pour de meilleures performances lors d'événements à fort trafic.

**Tests Fonctionnels :**

* En complément des tests automatisés, les tests fonctionnels sont essentiels pour valider l'expérience utilisateur globale. Ces tests impliquent des opérateurs humains exécutant un ensemble détaillé de scénarios utilisateur pour s'assurer que le flux de travail de l'application est intuitif, efficace et sans erreur.
* Les tests fonctionnels seront planifiés et exécutés selon des cas de test précis, couvrant des aspects tels que les fonctionnalités de connexion, la navigation dans l'application, la validation des entrées utilisateur et les réponses du système aux actions de l'utilisateur.

Déroulement des Tests :

Les tests seront réalisés en continu tout au long du cycle de développement. L'intégration continue sera mise en œuvre pour automatiser les tests et s'assurer que les régressions sont rapidement identifiées et corrigées. Les tests d'acceptation utilisateur seront également réalisés pour confirmer que l'application répond aux attentes des utilisateurs finaux.

1. **Livrables**

**Code Source :** Le lien vers le dépôt GitHub contenant tout le code source du projet.

**Applications Déployées :** Les liens vers les versions déployées des applications web et mobile.

**Gestion de Projet :** Le lien vers l'outil de gestion de projet utilisé pour le suivi de l'avancement.

**Documentation Technique :** Une documentation complète incluant les spécifications, les manuels d'utilisation, et les guides de déploiement.

1. **Planning**

Le projet débutera le 2 août et se terminera le 21 novembre. Nous commencerons par la réalisation des documents de planification et de conception, qui comprendront les 'Réflexions initiales technologiques', le 'Modèle conceptuel de données', les 'Maquettes', les 'Diagrammes', et les 'Chartes graphiques', du 2 août au 17 août.

La base de données MySQL devra être créée du 22 août au 25 août.

Application Web : La création du site web et ses tests devront être réalisés du 28 août au 20 octobre.

Application Bureautique : La création de l’application bureautique et ses tests devront être réalisés du 20 octobre au 8 novembre.

Application Mobile : La création de l’application mobile et ses tests devront être réalisés du 8 novembre au 21 novembre.

Déploiement : Le déploiement des plateformes devra être effectué du 17 novembre au 21 novembre 2023.