Projet: Tournoi CMS

(Ce document liste les différentes fonctionnalités à implémenter sur l'application web à créer, les exigences qu'ils faut afin de valider chaque fonctionnalités, le temps accorder au développement de chaque fonctionnalité et le prix de chaque fonctionnalité)

Thème: Gaming

Organiser des compétitions (tournoi) et recueilli des statistiques

Entité

- **Tournoi**
    - Une affiche du tournoi
    - Le nom du tournoi
    - La date de lancement du tournoi
    - Date de fin du tournoi
    - Nombre de participants voulu
    - Participants du tournoi
    - Frais de participation
    - Le type de tournoi (single elimination) pour un début
    - cashprize
    - Description (contiendra une description de l'événement, le déroulé et les règles du tournoi)
    -
- **Utilisateur**
    - l'id de l'utilisateur
    - Pseudo de gamer
    - Pays d'origine
    - Date de naissance
    - Numero de telephone
    - Mots de passe
    - Tournoi auxquels ils participent
        - Tournoi gagner
        - Tournoi perdu
        - Tournoi en cours

- **Participation**
    - l'identifiant de la participation
    - l'identifiant du tournoi
    - l'identifiant de l'utilisateur qui participe au tournoi
    - la date de participation

- **Manche**
  - L'identifiant de la manche
  - L'identifiant du tournoi
  - L'identifiant du participant 1
  - L'identifiant du participant 2
  - Score participant 1
  - Score participant 2
  - (Vainqueur de la manche, est celui qui a le score le plus élevé)
  - Timestamp (date de création, date de modification)

- **Stats**
  - L'identifiant de la stat
  - L'identifiant du participant
  - [Tableau]
    - L'identifiant du tournoi
    - Status (gagné ou perdu)
    - Score final

- **Tickets**
  - L'id du ticket
  - L'id du participant
  - La date d'émission
  - L'identifiant du tournoi

Action :

- Je veux m'inscrire en entrant mon pseudo de gamer, mon numéro de téléphone et un mots de passe, email, pays, date de naissance

Implémentation :

FRONTEND

- Affichage des champs pour entrer les informations de l'utilisateur
- Affichage d'un texte par exemple "Inscription" pour notifier l'utilisateur qu'il est sur la page d'inscription
- Affichage d'un champs pour la "confirmation de mots de passe"
- Affichage d'un bouton "S'inscrire"
- Empêcher l'utilisateur de soumettre le formulaire d'inscription avec des champs vides (affichage des messages d'erreur, en indiquant les champs avec erreurs en utilisant un toast)
- Afficher un toast d'erreur si le format de l'une des valeurs est incorrect (un toast qui ne précise pas le type d'erreur , mais un message d'erreur comme
  - Formats incorrects
- Rendre responsive la page

- Affichage d'un texte "Tu as déjà un compte? Connecte toi juste ici" pour rediriger l'utilisateur vers la page de connexion
- Connecter la page au backend
- S'accorder sur un design à utiliser
    - recherche d'inspiration
    - tester différente chose
    - etc

Durée : 1j/homme (08h de travail)

Prix: 32 €

BACKEND

- Créer une route pour l'inscription /api/auth/register
- Créer un controller pour la route
- Créer un schéma pour valider les données envoyées au corps de la requête de la route
- Vérifier si le numéro de téléphone existe déjà
    - si oui, retourner un status code d'erreur
- Sinon hasher le mot de passe de l'utilisateur
- Stocker dans une collection User , le numéro de téléphone de l'utilisateur, son mots de pass hasher et ses informations personnelles
- Générer un code de vérification de 05 chiffres à envoyer à l'utilisateur
- Stocker ce code de vérification dans une collection OTP contenant le numéro de téléphone de l'utilisateur
- Envoyer ce code sur le numéro de téléphone de l'utilisateur suivi d'un message basique
- Auto supprimer ce code de confirmation après 05 mn
- Retourner un status code de succès lorsque l'inscription réussi
- Mettre en place des tests unitaires pour tester la fonction de manière autonome

Durée : ½ j/homme (04h de travail)

Difficulté: Facile

Prix: 16 €

Action :

Je veux me connecter en entrant mon numéro de téléphone et mots de passe

Implémentation :

FRONTEND

- Affichage des Champs pour Entrer les Informations de Connexion (Numéro de téléphone et mot de passe)
- Affichage d'un Texte pour Notifier l'Utilisateur qu'il est sur la Page de Connexion
- Affichage d'un Bouton "Se Connecter" (Pour soumettre le formulaire)
- Validation des Champs (Pour empêcher l'utilisateur de soumettre le formulaire avec des champs vides en affichant des messages d'erreur indiquant les champs avec erreurs)

- Affichage d'un Texte "Tu n'as pas de compte ? Inscris-toi ici avec un lien vers la page d'inscription.
- Connexion de la Page au Backend (redirection vers la page principale ou de vérification en cas de succès ou affichage d'un message d'erreur en cas d'échec)
- Afficher un toast d'erreur si le numéro de téléphone l n'existe pas ou que le mots de passe est incorrect (un toast qui ne précise pas le type d'erreur , mais un message d'erreur comme
    - Aucun compte associé à ce numéro ou mot de passe incorrect
- Rendre responsive la page

Durée : ½ j/homme (04 heures de travail)

Difficulté : Facile

Prix: 16 €

BACKEND

- Créer une route pour la connection /api/auth/login
- Créer un controller pour la route
- Créer un schéma pour valider les données envoyées au corps de la requête de la route
- Vérifier si il existe un compte avec le numéro de téléphone de l'utilisateur
- Si non retourner un code et un message d'erreur
- Si oui créer un hash du mots de passe entré par l'utilisateur
- Comparer ce hash avec le hash du mots de passe stockés dans la base de donnée
- Si correct
    - Générer un token de connexion (jwt) à l'utilisateur connecté
- Sinon retourner un message d'erreur
- Mettre en place des tests unitaires pour tester la fonction de manière autonome

Durée : ½ j/homme (04h de travail)

Difficulté: Facile

Prix: 16 €

Action :

Je veux vérifier mon compte en entrant un code de confirmation envoyé à mon numéro de téléphone

Implémentation :

FRONTEND

- Affichage des Champs pour Entrer le Code de Confirmation
- Affichage d'un Texte pour Notifier l'Utilisateur qu'il est sur la Page de Vérification
- Affichage d'un Bouton "Vérifier"
- Validation du champ (Empêcher l'utilisateur de soumettre le formulaire avec le champ vide en affichant un message d'erreur comme: "Le code de confirmation est obligatoire").
- Rendre la page responsive

- Afficher un toast d'erreur si le le code de vérification est incorrect (un toast qui ne précise pas le type d'erreur , mais un message d'erreur comme
    - Le code de vérification est incorrect
- Connexion de la Page au Backend
- Renvoi du Code de Confirmation (Option pour l'utilisateur de demander un nouveau code si nécessaire.)

Durée : ½ j/homme (04h de travail)

Difficulté: Facile

Prix: 16 €

BACKEND

- Créer une route pour la vérification de l'otp /api/otp/verify
- Créer un controller pour la route
- Créer un schéma pour valider les données envoyées au corps de la requête de la route
- Vérifier le code opt envoyé par l'utilisateur avec le code otp stocké dans la base de donnée
- Si correct vérifié l'utilisateur
- Sinon retourner un status code et un message d'erreur
- Créer une route pour renvoyer le code de confirmation à un utilisateur
- Créer un controller pour la route
- Créer un schéma pour valider les données envoyées au corps de la requête de la route
- Si un code opt existe déjà pour cet utilisateur, supprimer l'existant , générer un nouveau otp, le stocker dans la collection OTP et l'envoyer par sms sur le numéro de téléphone de l'utilisateur
- Sinon juste générer un nouveau otp, le stocker dans la collection OTP et l'envoyer par sms sur le numéro de téléphone de l'utilisateur
- Mettre en place des tests unitaires pour tester la fonction de manière autonome
- Fait un choix du service d'envoie d'sms a utilisé, le configuré dans le projet
- Tester les différentes routes
- Ecrire des units tests

Durée : 1j/homme (08h de travail)

Difficulté : Facile

Prix: 32 €

Action :

Je veux pouvoir voir la liste des tournois auxquels je participe (Une fois connecté) en plus des autres tournois disponible

FRONTEND

Implémentation :

- Créer une section pour afficher les tournois auxquels l'utilisateur participe.
- Créer une section pour afficher les autres tournois disponibles.
- Utiliser des cartes ou des listes pour afficher les informations de chaque tournoi (nom, date, type, etc.).

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix: 16 €

BACKEND

- Créer une route /api/tournament/query? pour filtrer les tournois
- Créer un controller pour la route
- Créer un schéma pour valider les données envoyées au corps de la requête de la route
- Mettre un middleware qui vérifie que l'utilisateur est connecté pour accéder à cette route
- Retourner une list de tournoi qui valide la query
- Mettre en place des tests unitaires

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix: 16 €

Action :

Je veux voir le détail de chaque tournoi

Implémentation :

FRONTEND

- Créer une page ou un modal pour afficher les détails complets d'un tournoi (nom, type, dates, règles, participants, etc.).
- Ajouter un bouton ou un lien sur la liste des tournois pour accéder aux détails.(En cas de page)

Durée : ½ j/homme (04 de travail)

Difficulté : Facile

Prix : 16 €

BACKEND

- Créer une route qui retourne les détails d'un tournoi par son identifiant /api/tournament/detail/identifiant
- Créer un controller pour cette fonction
- Créer un schéma pour valider les données envoyées au corps de la requête de la route
- Mettre en place des tests unitaires

Durée : 02h de travail

Difficulté : Facile

Prix : 8 €

Action :

- Je veux voir l'avancée du classement des tournois en cours

  Implémentation :

FRONTEND

- Créer une section ou une page pour afficher le classement des tournois en cours
- Ajouter un conteneur ou une section dédiée dans l'interface utilisateur.
- Ajouter un titre clair, par exemple "Classement des Tournois en Cours".
- Faire une requête à l'API pour obtenir le classement des tournois en cours.
- Définir une variable d'état pour stocker les données de classement récupérées de l'API.
- Mettre à jour cette variable d'état avec les données obtenues lors de la réponse de l'API.
- Créer un tableau ou une liste pour afficher les informations de classement.
- Les résultats des classements et manches doivent être affichés en temps réel.
- Inclure des colonnes pour le rang, le nom des participants et leurs scores.
- Appliquer des styles CSS et des classes tailwind pour rendre le tableau ou la liste adaptable à différentes tailles d'écran.
- Utiliser des classes ou des règles de style pour assurer une présentation claire et lisible sur tous les dispositifs

Durée : 1j/homme (08h de travail)

Difficulté : Intermédiaire

Prix : 32 €

BACKEND

- Créer une route /api/tournament/ranking/identifiant-du-tournoi pour retourner le classement d'un tournoi en fonction de son id
- Créer un controller pour la fonction

- Créer un schéma pour valider les données envoyées au corps de la requête de la route
- Récupérer les différentes manches du tournoi afin de définir le classement
- Mettre en place un algorithme de classement en fonction des scores
- Retourner le classement
- Implémenter cette route en websocket
- Ecrire des tests unitaires pour tester la fonctionnalité

Durée : 1j/homme (08h de travail)

Difficulté : Intermédiaire

Prix : 40 €

Action :

Je veux participé à un tournoi en payant ma participation (en utilisant les moyens de paiement locaux)

FRONTEND

Implémentation :

- Créer un formulaire ou un bouton pour s'inscrire au tournoi.
- Déclencher le processus de paiement lors du clic sur le bouton "Participer".
- Intégrer le SDK ou l'API de Fedapay ou Kkiapay pour initier le paiement.
- Récupérer les détails de paiement de l'utilisateur.
- Soumettre les informations de paiement au service de paiement.
- Gérer la réponse du service de paiement.
- Mettre à jour l'inscription de l'utilisateur au tournoi.
- Envoyer une confirmation par email.
- Ajouter des styles CSS et des classes Tailwind pour améliorer l'expérience utilisateur.

Durée : 1j/homme (08h de travail)

Difficulté : Intermédiaire

Prix : 32 €

BACKEND

- Définir une route POST `/api/initiate-payment` pour initier le paiement.
- Vérifier si le tournoi existe.
- Créer un enregistrement de paiement en attente.
- Initier le paiement via l'API de Fedapay ou Kkiapay.
- Retourner l'URL de paiement à l'utilisateur.
- Créer un service pour gérer l'interaction avec Fedapay ou Kkiapay.
- Utiliser Axios pour envoyer des requêtes à l'API de paiement.
- Gérer les erreurs de paiement et retourner les résultats pertinents.
- Définir une route POST `/api/payment-response` pour gérer les réponses de paiement.
- Mettre à jour l'état du paiement dans la base de données.
- Ajoutez l'utilisateur à la liste des participants du tournoi si le paiement est réussi.
- Envoyer un email de confirmation à l'utilisateur.

- Ajouter des gestionnaires d'erreurs pour gérer les erreurs survenues lors du processus de paiement.
- Retourner des messages d'erreur clairs à l'utilisateur.
- Configurer un service pour envoyer des emails de confirmation après un paiement réussi.
- Utiliser Nodemailer pour envoyer le tickets par email à l'utilisateur

Durée : 1j/homme (08h de travail)

Difficulté : Intermédiaire

Prix : 40 €

Action :

Je veux voir mes statistiques

Implémentation :

FRONTEND

- Créer une section ou une page pour afficher les statistiques de l'utilisateur.
- Ajouter un lien ou un bouton dans le menu de navigation pour accéder à la page des statistiques.
- Faire une requête à l'API pour obtenir les statistiques de l'utilisateur.
- Définir une variable d'état pour stocker les données de statistiques récupérées de l'API.
- Mettre à jour cette variable d'état avec les données obtenues lors de la réponse de l'API.
- Créer une section ou un composant pour afficher les statistiques.
- Inclure des sections pour différentes catégories de statistiques.
- Appliquer des styles CSS et des classes Tailwind pour rendre la page attrayante et responsive.

Durée : 1j/homme (08h de travail)

Difficulté : Facile

Prix : 32 €

BACKEND

- Créer une route /api/user/stats pour retourner les statistiques de l'utilisateur connecter
- Mettre un middleware de protection sur cette route
- Créer un controller
- Rechercher l'utilisateur dans la bd et retourner ses stats
- Tester la route en écrivant des tests unitaires

Durée: ½ j/homme (04h de travail)

Difficulté : Facile

Prix : 16 €

Action :

Je visualiser mes informations personnelles

Implémentation :

FRONTEND

- Créer une section ou une page pour afficher les informations personnelles de l'utilisateur.
- Ajouter un lien ou un bouton dans le menu de navigation pour accéder à la page des informations personnelles.
- Faire une requête à l'API pour obtenir les informations personnelles de l'utilisateur.
- Définir une variable d'état pour stocker les informations personnelles récupérées de l'API.
- Mettre à jour cette variable d'état avec les données obtenues lors de la réponse de l'API.
- Créer une section ou un composant pour afficher les informations personnelles.
- Inclure des champs pour différentes catégories d'informations personnelles.
- Appliquer des styles CSS et des classes Tailwind pour rendre la page attrayante et responsive.

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix : 16 €

BACKEND

- Créer un modèle utilisateur dans MongoDB avec les informations personnelles nécessaires.
- Définir une route GET `/api/user/me` pour récupérer les informations personnelles d'un utilisateur par son identifiant.
- Créer un contrôleur pour gérer la requête et retourner les informations personnelles.
- Implémenter une fonction de middleware pour vérifier que l'utilisateur est authentifié avant d'accéder aux informations personnelles.
- Gérer les erreurs et retourner des messages d'erreur clairs en cas de problème.
- Tester la route en écrivant des tests unitaires

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix : 16 €

Action :

Je veux recevoir mon ticket de participation à un tournoi par mail

Implémentation :

FRONTEND

- Générer le ticket de participation avec les détails du tournoi et de l'utilisateur.
- Envoyer une requête à l'API pour créer le ticket et le préparer pour l'envoi par email.
- Stocker les informations du ticket dans une variable d'état.
- Afficher un message de confirmation à l'utilisateur indiquant que le ticket a été envoyé par email.

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix : 16 €

BACKEND

- Créer une route /api/tournament/join pour accepter les participations à un tournoi
- Créer un controller pour la route
- Récupérer l'identifiant du tournoi, l'identifiant de l'utilisateur
- Initialiser le paiement
- Si réussit , lancer la génération de ticket
    - Créer un template html pour le ticket
    - Afficher sur le ticket les informations requis
    - Envoyer le ticket par email à l'utilisateur
- Ecrire des tests unitaires pour s'assurer que la fonctionnalité marche

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix : 16 €

Action :

Je veux pouvoir me déconnecter

Implémentation :

FRONTEND

- Ajouter un bouton ou un lien pour se déconnecter dans le menu de navigation ou le profil.
- Déclencher le processus de déconnexion lors du clic sur le bouton ou le lien.
- Supprimer les informations d'authentification stockées dans le localStorage .
- Rediriger l'utilisateur vers la page de connexion ou d'accueil après la déconnexion.
- Afficher un toast avec un message de confirmation 'Vous êtes déconnectés"

- Appliquer des styles CSS et des classes Tailwind pour rendre le bouton ou le lien attrayant.

Durée : (1h de travail)

Difficulté : Facile

Prix : 2 €

Action :

Je veux pouvoir contacter l'administrateur en cas de problème .

Implémentation :

FRONTEND

- Ajoutez un espace dans le footer pour le formulaire de contact.
- Ajoutez un champ de texte pour l'email et un bouton de soumission.
- Configurez un événement pour gérer l'envoi du formulaire.
- Créez des variables pour stocker et gérer l'email.
- Envoyez l'email à l'API pour l'ajouter à la liste de diffusion ou pour contacter l'administrateur.
- Affichez un message indiquant si l'email a été envoyé avec succès ou s'il y a eu un problème.
- Réinitialisez les champs du formulaire après l'envoi.
- Utilisez des styles pour améliorer l'apparence du formulaire.

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix : 16 €

BACKEND

- Utiliser les services existants pour l'envoie de mail
- Envoyer la plainte de l'utilisateur à l'administrateur
- Ecrire des tests unitaires pour s'assurer que la fonctionnalité marche

Durée : 1h de travail

Difficulté : Facile

Prix : 2 €

Compte administrateur

Action :

Je veux appliquer un CRUD sur un tournoi en entrant les informations liées au tournoi

Implémentation :

FRONTEND

- Ajouter un formulaire de création de tournoi.
- Inclure les champs pour le nom, type, dates, règles, description, et image.
- Ajouter un bouton pour soumettre le formulaire.
- Envoyer les données à l'API pour créer le tournoi.
- Afficher un toast avec un message de confirmation ou d'erreur après la soumission.
- Créer une vue pour afficher la liste des tournois existants.
- Inclure des informations comme le nom, type, dates et nombre de participants.
- Ajouter des boutons pour voir les détails, éditer ou supprimer un tournoi.
- Faire une requête à l'API pour récupérer les données des tournois.
- Afficher les tournois dans un tableau ou une liste.
- Ajouter un formulaire de mise à jour pour un tournoi sélectionné.
- Pré-remplir le formulaire avec les données existantes du tournoi.
- Inclure des champs pour le nom, type, dates, règles, description et image.
- Ajouter un bouton pour soumettre les modifications.
- Envoyer les données mises à jour à l'API.
- Afficher un toast avec un message de confirmation ou d'erreur après la soumission.
- Ajoutez un bouton de suppression pour chaque tournoi dans la liste.
- Afficher une confirmation avant de supprimer le tournoi.
- Faire une requête à l'API pour supprimer le tournoi.
- Afficher un toast avec un message de confirmation ou d'erreur après la soumission.

Durée : 2 j/homme (16h de travail)

Difficulté : Intermédiaire

Prix : 50 €

BACKEND

- Créer une route /api/tournament/ en CRUD (donc plusieurs routes différentes)
- Permettre la création de tournoi, la modification des détails et la suppression d'un tournoi
- Ecrire des tests unitaires

Durée : 1 j/homme (08h de travail)

Difficulté : Facile

Prix : 16 €

Action :

Je veux créer de manière aléatoire des versus entre les participants d'un tournoi (si besoin)

Implémentation :

FRONTEND

- Ajouter un formulaire de création de tournoi.
- Ajouter une option ou un bouton pour générer des *versus* aléatoires entre les participants du tournoi.
- Faire une requête à l'API pour récupérer la liste des participants du tournoi.
- Définir une variable d'état pour stocker les données des participants récupérées de l'API.
- Mettre à jour cette variable d'état avec les données obtenues lors de la réponse de l'API.
- Récupéré depuis le backends les résultats du versus
- Afficher une prévisualisation des *versus* générés pour permettre à l'administrateur de les vérifier.
- Afficher un message de confirmation ou d'erreur après l'enregistrement des *versus*. Mettre à jour l'interface utilisateur pour refléter les nouveaux *versus* dans le tournoi.

Durée : 1 j/homme (04h de travail)

Difficulté : Facile

Prix : 8 €

BACKEND

- Définir une route POST `/api/tournaments/:id/generate-versus` pour générer des versus aléatoires.
- Écrire un algorithme pour générer des *versus* aléatoires en pairant les participants entre eux.
- Retourner cette liste au front
- Ecrire des tests unitaires
- Créer un contrôleur pour gérer la récupération de la liste des participants depuis MongoDB.
- Définir une route POST `/api/tournaments/:id/generate-versus` pour générer des versus aléatoires.
- Écrire une fonction pour sélectionner et paire aléatoirement les participants du tournoi.
- Mettre à jour le modèle du tournoi dans MongoDB avec les versus générés.
- Gérer les erreurs et retourner des messages d'erreur clairs en cas de problème.
- Mettre en place des tests unitaires pour vérifier le bon fonctionnement des routes et des contrôleurs.

Durée : 1 j/homme (08h de travail)

Difficulté : Intermédiaire

Prix : 32 €

Action :

Je veux pouvoir entrer les résultats de chaque manche des tournois (si tournoi avec versus)

Implémentation :

FRONTEND

- Ajouter une option pour entrer les résultats.
- Afficher les *versus* en cours.
- Créer un formulaire pour entrer les résultats (score ou gagnant).
- Ajouter un bouton pour soumettre les résultats.
- Faire une requête API pour enregistrer les résultats.
- Afficher un message de confirmation ou d'erreur.
- Mettre à jour l'interface utilisateur avec les résultats.
- Calculer les scores et mettre à jour l'état du tournoi.

Durée : 1 j/homme (08h de travail)

Difficulté : Intermédiaire

Prix : 32 €

BACKEND

- Créer une route /api/tournament/game pour enregistrer les résultats d'une manche de jeux d'un tournoi
- Créer un controller
- Vérifier les données envoyé à la route
- Stocker les données dans la bd
- Procéder par websocket (temps réel)
    - emit un event pour trigger et refresh en temps réel l'interface qui affiche les résultats des manches d'un tournoi
- Designer automatiquement le vainqueur de la manche en utilisant les scores
- Ecrire des tests unitaires

Durée : ½ j/homme (04h de travail)

Difficulté : Intermédiaire

Prix : 25 €

Action :

Sinon je veux entrer le score de chaque joueurs (et le système se chargera de classer)

Implémentation :

FRONTEND

- Ajouter une option pour entrer les scores.
- Afficher les matchs en cours.
- Créer un formulaire pour entrer les scores des joueurs.
- Ajouter un bouton pour soumettre les scores.
- Faire une requête API pour enregistrer les scores.
- Afficher un message de confirmation ou d'erreur.
- Calculer automatiquement le classement des joueurs.
- Mettre à jour l'interface utilisateur avec les scores et le classement en temps réel.
- Afficher le classement des joueurs basé sur les scores.

- Calculer les scores et mettre à jour l'état du tournoi.

Durée : 1 j/homme (8h de travail)

Difficulté : Intermédiaire

Prix : 32 €

BACKEND

- Créer une route pour récupérer et stocker les scores de chaque utilisateur (si tournoi sans versus)
- Créer un controller
- Vérifier les données envoyées à la requête
- Mettre en place des tests unitaires
- Appliquer du real time

Durée : ½ j/homme (04h de travail)

Difficulté : Intermédiaire

Prix : 25 €

Action :

Je veux avoir les statistiques sur chaque joueur pour le moment le nombre de tournoi auquelle il a participé, le nombre de tournoi gagné ou perdu

Implémentation :

FRONTEND

- Ajoutez une option pour voir les statistiques des joueurs.
- Afficher la liste des joueurs du tournoi.
- Créer une interface pour afficher les statistiques du joueur sélectionné.
- Faire une requête API pour récupérer les statistiques du joueur.
- Définir une variable d'état pour stocker les données des statistiques.
- Mettre à jour cette variable avec les données obtenues de l'API.
- Afficher le nombre de tournois participés, gagnés et perdus.
- Appliquer des styles pour une présentation claire.
- Afficher un message d'erreur si les données ne sont pas disponibles.
- Mettre à jour l'interface utilisateur avec les statistiques.

Durée :1 j/homme (08h de travail)

Difficulté : Intermédiaire

Prix : 32€

BACKEND

- Définir un modèle de joueur dans MongoDB avec les champs appropriés pour les statistiques (tournois participés, gagnés, perdus).
- Définir une route GET `/api/players/:id/statistics` pour récupérer les statistiques d'un joueur par son identifiant.
- Créer un contrôleur pour gérer la récupération des statistiques du joueur depuis MongoDB.
- Créer un schéma de validation pour s'assurer que les données envoyées dans la requête sont correctes lors de la récupération des statistiques.
- Gérer les erreurs et retourner des messages d'erreur clairs en cas de problème.
- Mettre en place des tests unitaires pour vérifier le bon fonctionnement des routes et des contrôleurs.

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix : 16€

Action :

Je veux afficher les informations de l'administrateur

Implémentation :

FRONTEND

- Ajouter une option pour voir les informations de l'administrateur.
- Créer une page ou un modal pour afficher les informations.
- Faire une requête API pour récupérer les informations de l'administrateur.
- Définir une variable d'état pour stocker les données de l'administrateur.
- Mettre à jour la variable d'état avec les données récupérées.
- Afficher le nom, l'email, le rôle, et d'autres détails de l'administrateur.
- Appliquer des styles pour une présentation claire.
- Afficher un message d'erreur en cas de problème avec les données.
- Mettre à jour l'interface utilisateur avec les informations de l'administrateur.

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix : 16€

BACKEND

- Créer un modèle administrateur dans MongoDB avec les informations personnelles nécessaires.

- Définir une route GET `/api/user/admin` pour récupérer les informations personnelles de l'administrateur.
- Créer un contrôleur pour gérer la requête et retourner les informations personnelles.
- Implémenter une fonction de middleware pour vérifier que l'utilisateur est authentifié avant d'accéder aux informations personnelles.
- Gérer les erreurs et retourner des messages d'erreur clairs en cas de problème.
- Tester la route en écrivant des tests unitaires

Durée : ½ j/homme (04h de travail)

Difficulté : Facile

Prix : 16 €

Total : **760 €**

Technologies utilisées :

- React
- Nodejs
- Socket.io (pour websocket)
- Mongodb

À prendre en charge par le client :

- Hébergement du front et du back
- Fournir les noms de domaine
- S'assurer d'augmenter la capacité du stockage de la base de donnée
- Choisir le cloud provider pour la mise en place de l'infrastructure de la solution
- L'achat des serveurs sur le cloud provider choisit
- Choix des moyens de paiement locaux à utilisés

Détails:

- Plusieurs type de tournois
  - Single elimination
  - Double elimination
  - Phase de groupe

Deadline 02 jours
Deadline: Mi Août