# Spécifications du Test de Recrutement pour Développeur Frontend/Backend JavaScript

**Entreprise :** HIGH REFERENCE SARL

Nous sommes ravis de vous convier à une série de tests pratiques conçus pour évaluer vos compétences en développement JavaScript, tant sur la partie **Frontend** que **Backend**. L'objectif est de recruter des talents capables de s'intégrer rapidement à notre équipe et de contribuer activement au développement de nos applications en microservices.

Ce processus de sélection se déroulera en trois phases distinctes, chacune visant à évaluer des compétences spécifiques et votre capacité à collaborer.

---

## Processus de Sélection

### Étape 1 : Développement Individuel de l'Application "TrueNumber"

- **Objectif :** Évaluer votre autonomie, votre maîtrise des technologies JavaScript (Node.js pour le backend, Next.js ou autre framework JS pour le frontend), votre capacité à implémenter des fonctionnalités complètes et à respecter les bonnes pratiques de développement.
- **Durée :** 5 jours
- **Date limite de livraison : Lundi 21 juillet 2025 à 23h59 (WAT)**

### Étape 2 : Collaboration et Ajout de Fonctionnalités en Groupe

- **Objectif :** Évaluer votre capacité à travailler en équipe, à comprendre et adapter du code existant, et à contribuer à un projet commun en respectant une architecture définie (Frontend / Backend).
- **Sélection :** Les meilleurs candidats de l'Étape 1 seront regroupés pour cette phase.
- **Déroulement :** Vous serez assigné(e) à un rôle (Frontend ou Backend) et travaillerez en équipe pour ajouter une fonctionnalité cruciale à l'application "TrueNumber" développée à l'Étape 1. La fonctionnalité exacte sera communiquée ultérieurement.

### Étape 3 : Entretiens Approfondis

- **Objectif :** Comprendre votre raisonnement technique, votre approche de résolution de problèmes, vos motivations et votre adéquation avec notre culture d'entreprise.

- **Sélection :** Les membres des groupes s'étant le plus démarqués lors de l'Étape 2 seront conviés à cette étape finale.

---

# Détails de l'Étape 1 : Développement de l'Application "TrueNumber"

## Objectif Principal

Implémenter le **Frontend** et le **Backend** d'une application nommée "TrueNumber" en utilisant les technologies JavaScript (de préférence **Node.js** pour le backend et **Next.js** pour le frontend, mais d'autres frameworks JS seront acceptés). Le code source doit être hébergé sur **GitHub** et l'application doit être déployée.

## Fonctionnalités Attendues

### 1. Système d'Authentification (Frontend & Backend)

- **Création de compte :** Un utilisateur doit pouvoir s'inscrire via une interface client dédiée. Lors de la création du compte, le **numéro de téléphone** de l'utilisateur doit également être collecté.
- **Connexion :** Un utilisateur doit pouvoir se connecter avec ses identifiants.
- **Déconnexion :** Un utilisateur doit pouvoir se déconnecter de son compte.

### 2. Interface Client "Jeu TrueNumber" (Frontend & Backend)

Une fois connecté, l'utilisateur accède à une interface client où il peut :

- **Jouer :** Un bouton "Générer un nombre" doit être présent. Au clic, un nombre aléatoire entre **0 et 100** doit être généré.
    - Si le nombre est **inférieur ou égal à 70**, un message (ou "toast") indique à l'utilisateur qu'il a **perdu** le jeu.
    - Si le nombre est **supérieur à 70 et inférieur ou égal à 100**, un message (ou "toast") indique que l'utilisateur a **gagné** le jeu.
- **Afficher son Solde :** Un compteur de points, initialisé à **0**, doit être visible.
    - Si l'utilisateur gagne, son solde **augmente de 50 points**.
    - Si l'utilisateur perd, son solde **diminue de 35 points**.
- **Consulter son Historique :** Un menu ou une section doit permettre de visualiser l'historique des parties jouées et les résultats obtenus (gagné/perdu, nombre généré, variation du solde).

### 3. Interface Administrateur (Frontend & Backend)

Un utilisateur avec un rôle "admin" doit pouvoir se connecter à une interface d'administration dédiée. Cette interface doit permettre à l'administrateur de :

- Effectuer toutes les actions d'un utilisateur normal (jouer, voir solde, historique).

- **Voir tous les utilisateurs** ayant interagi avec l'application.
- **Supprimer** un utilisateur.
- **Créer** un nouvel utilisateur.
- **Modifier les informations d'un utilisateur**, notamment son **rôle** (passer de "client" à "admin" et inversement).

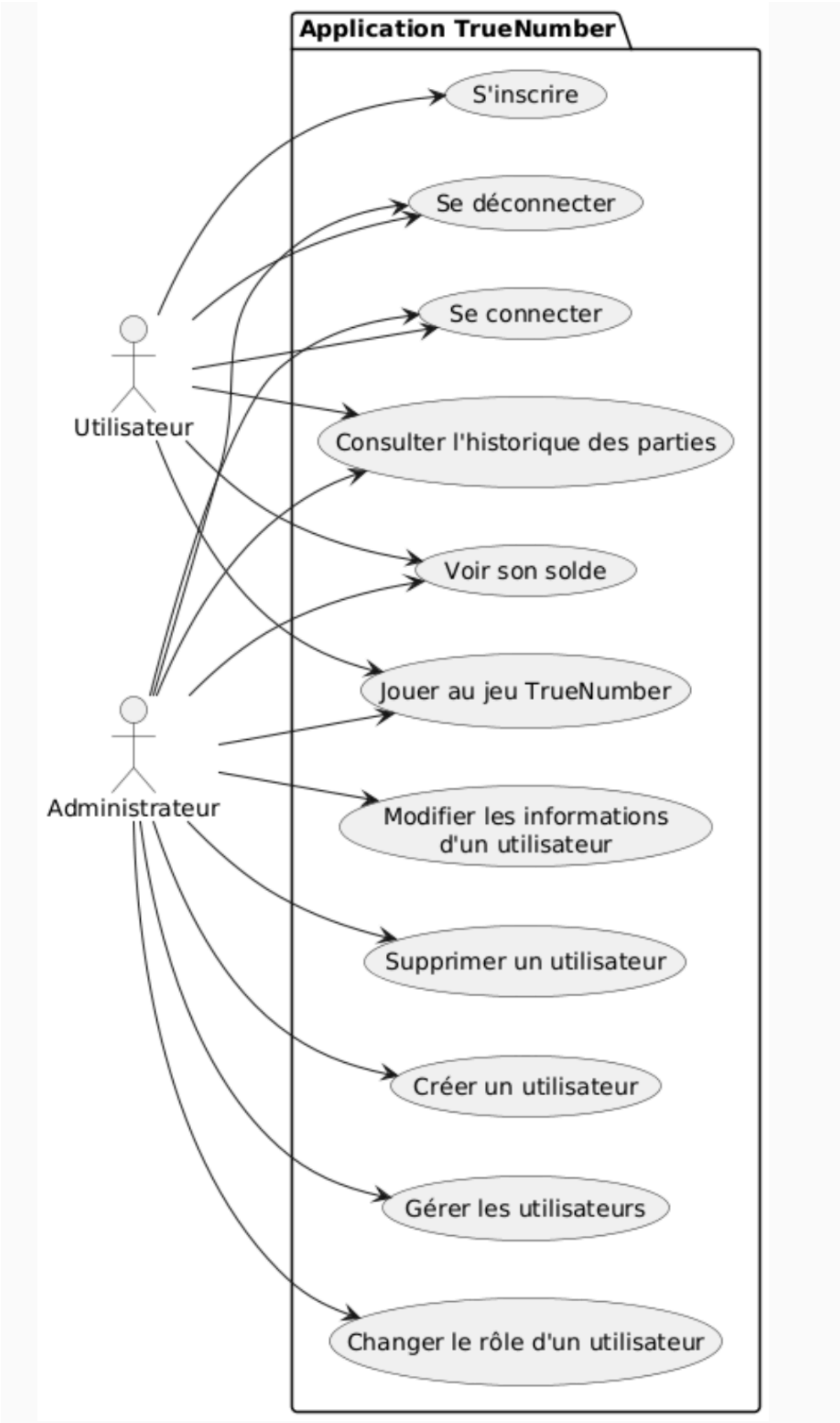
### Exigences Techniques et Bonnes Pratiques

- **Backend (API) :**
    - Développé en JavaScript (Node.js est recommandé).
    - Doit exposer des **routes API RESTful** pour la gestion des utilisateurs, des soldes et de l'historique des jeux.
    - Toutes les routes API doivent être **documentées via Swagger** et accessibles publiquement.
- **Frontend (Interface Client & Admin) :**
    - Développé en JavaScript (Next.js est recommandé, mais tout autre framework JS est acceptable). L'utilisation de **TypeScript** est un atout.
    - Peut être développé pour le **web ou le mobile**, selon votre préférence.
    - L'interface client, même simple, doit être **ergonomique, intuitive et responsive**.
- **Base de Données :** L'application doit utiliser **MongoDB** pour la persistance des données.
- **Hébergement et Déploiement :**
    - Le code source doit être hébergé sur un **dépôt GitHub public**.
    - L'application frontend et/ou le backend (si nécessaire) doivent être déployés sur un service permettant un accès public (Vercel, Heroku, Netlify, Render, etc. – le choix est libre pour le candidat).
- **Qualité du Code :** Respect des bonnes pratiques de développement, code propre, maintenable, commenté si nécessaire, et structuré. L'organisation du projet sera un critère d'évaluation.

---

# Diagramme des Cas d'Utilisation

Le diagramme de cas d'utilisation ci-dessous illustre les interactions possibles des différents acteurs (Utilisateur, Administrateur) avec le système "TrueNumber".


Application TrueNumber
S'inscrire
Se déconnecter
Se connecter
Consulter l'historique des parties
Voir son solde
Jouer au jeu TrueNumber
Modifier les informations d'un utilisateur
Supprimer un utilisateur
Créer un utilisateur
Gérer les utilisateurs
Changer le rôle d'un utilisateur
Utilisateur
Administrateur

**Description des Cas d'Utilisation :**

- **S'inscrire :** Permet à un nouvel utilisateur de créer un compte en fournissant ses informations (y compris le numéro de téléphone).
- **Se connecter :** Permet à un utilisateur (client ou admin) d'accéder à l'application avec ses identifiants.
- **Se déconnecter :** Permet à un utilisateur de mettre fin à sa session.
- **Jouer au jeu TrueNumber :** Permet à un utilisateur connecté de générer un nombre aléatoire et de voir le résultat (gagné/perdu) ainsi que l'impact sur son solde.
- **Voir son solde :** Permet à un utilisateur de consulter son solde de points actuel.
- **Consulter l'historique des parties :** Permet à un utilisateur de voir la liste de ses parties jouées.
- **Gérer les utilisateurs (Administrateur) :** Permet à l'administrateur d'accéder à l'interface de gestion des utilisateurs.
- **Créer un utilisateur (Administrateur) :** Permet à l'administrateur d'ajouter manuellement un nouvel utilisateur.
- **Supprimer un utilisateur (Administrateur) :** Permet à l'administrateur de supprimer un compte utilisateur.
- **Modifier les informations d'un utilisateur (Administrateur) :** Permet à l'administrateur de mettre à jour les détails d'un utilisateur.
- **Changer le rôle d'un utilisateur (Administrateur) :** Permet à l'administrateur de basculer le rôle d'un utilisateur entre "client" et "admin".

---

# Propositions de Routes API (Backend)

Pour faciliter et uniformiser le travail, voici une proposition de structure de routes API RESTful. Vous êtes libre d'adapter ces routes si vous estimez qu'une autre organisation est plus pertinente, à condition de maintenir une logique claire et documentée dans Swagger.

## Authentification

- **POST** `/api/auth/register`
  - **Description :** Création d'un nouveau compte utilisateur.
  - **Corps de la requête :** `{ "username": "...", "email": "...", "password": "...", "phone": "..." }`
  - **Réponse :** `{ "message": "Compte créé avec succès", "user": { ... } }` ou erreur.
- **POST** `/api/auth/login`
  - **Description :** Connexion d'un utilisateur.
  - **Corps de la requête :** `{ "email": "...", "password": "..." }`
  - **Réponse :** `{ "token": "...", "user": { "id": "...", "username": "...", "role": "..." } }` ou erreur.

- **POST** `/api/auth/logout`
    - **Description :** Déconnexion de l'utilisateur (peut invalider le token côté serveur si implémenté).

### Utilisateurs (Nécessite authentification, certaines routes nécessitent un rôle 'admin')

- **GET** `/api/users/me`
    - **Description :** Récupérer les informations de l'utilisateur connecté.
    - **Réponse :** `{ "id": "...", "username": "...", "email": "...", "phone": "...", "role": "...", "balance": "..." }`
- **GET** `/api/users` (Admin seulement)
    - **Description :** Récupérer la liste de tous les utilisateurs.
    - **Réponse :** `[ { "id": "...", "username": "...", "email": "...", "phone": "...", "role": "..." }, ... ]`
- **GET** `/api/users/:id` (Admin seulement)
    - **Description :** Récupérer les informations d'un utilisateur spécifique par ID.
- **POST** `/api/users` (Admin seulement)
    - **Description :** Créer un nouvel utilisateur (par l'administrateur).
    - **Corps de la requête :** `{ "username": "...", "email": "...", "password": "...", "phone": "...", "role": "client" }`
- **PUT** `/api/users/:id` (Admin seulement)
    - **Description :** Modifier les informations d'un utilisateur (y compris le rôle).
    - **Corps de la requête :** `{ "username": "...", "email": "...", "phone": "...", "role": "admin" }` (les champs à modifier)
- **DELETE** `/api/users/:id` (Admin seulement)
    - **Description :** Supprimer un utilisateur.

### Jeu "TrueNumber"

- **POST** `/api/game/play` (Nécessite authentification)
    - **Description :** Lancer une partie du jeu TrueNumber, générer un nombre, mettre à jour le solde et enregistrer l'historique.
    - **Réponse :** `{ "result": "gagné" | "perdu", "generatedNumber": 75, "newBalance": 125 }`

### Solde

- **GET** `/api/balance` (Nécessite authentification)
    - **Description :** Récupérer le solde actuel de l'utilisateur connecté.
    - **Réponse :** `{ "balance": 150 }`

### Historique des Parties

- **GET** `/api/history` (Nécessite authentification)
    - **Description :** Récupérer l'historique des parties jouées par l'utilisateur connecté.
    - **Réponse :** `[ { "gameId": "...", "date": "...", "generatedNumber": 50, "result": "perdu", "balanceChange": -35, "newBalance": 65 }, ... ]`
- **GET** `/api/history/all` (Admin seulement)
    - **Description :** Récupérer l'historique de toutes les parties jouées par tous les utilisateurs.

---

# Livrables pour l'Étape 1

Une fois votre projet finalisé et déployé, veuillez envoyer un message à l'administrateur en respectant le format suivant :

Nom: [Votre Nom Complet]
Technologie: [Technologies utilisées - Ex: Next.js, Node.js, Express, MongoDB, TypeScript]
Lien Swagger: [Lien vers la documentation Swagger de votre API]
Lien Appli: [Lien vers l'application déployée (Frontend)]
Observations: [Toute observation pertinente, explication technique ou point que vous souhaitez souligner]

---

# Conseils pour Réussir

- **Lisez attentivement toutes les spécifications.** Prenez le temps de bien comprendre chaque fonctionnalité.
- **Commencez par la base :** Authentification et la logique principale du jeu.
- **Privilégiez la clarté et la propreté du code.** Un code bien structuré et lisible est essentiel.
- **Documentez votre API avec Swagger.** C'est un point clé pour l'interopérabilité et la compréhension de votre backend.
- **Pensez à l'expérience utilisateur** pour l'interface client (ergonomie, intuitivité, responsivité).
- **N'hésitez pas à poser des questions** si un point n'est pas clair.

Nous avons hâte de découvrir vos réalisations et votre approche du développement. Bonne chance !