Enoncé

Nous avons un système de queue FIFO.
La queue est une liste d'action en attente d'execution.

Chaque type d'action possède un nombre de crédits d'exécution.
Les differents types d'actions sont définis par le développeur.
Les credits d'execution ont une valeur maximale définis aussi par le développeur.
Ils seront calculés de manière aléatoire entre 80% et 100% cette valeur maximale.
Ils sont recalculés si au moins 10min sont passées depuis le dernier calcul.

Ajouter une action à la queue ne coûte pas de crédit, l'execution en consomme 1.
Via une interface, l'utilisateur peut ajouter autant d'actions qu'il souhaite dans sa queue. (Même s'il n'a plus de crédits).
Toutes les 15 secondes la prochaine action disponible est executée.

Exemple :
Je suis dans un système avec 3 type d'action, A, B, C ayant respectivement 10, 10, 15 en valeur maximale de credits.
L'utilisateur, après le premier calcul on lancement de l'application, possède en crédits pour les types d'actions A, B, C, dans l'ordre, 9, 10, 13.
Il ajoute 2 actions à la queue de type A, puis 3 de B puis 1 de A puis 2 de C.
Il aura donc une queue : First-In→[A, A, B, B, B, A, C, C]→Last-In
15 secondes plus tard, la première action de type A s'execute et consomme 1 crédit.
La queue sera donc devenue: First-Out→[A, B, B, B, A, C, C]→Last-Out, et les crédits seront passés à 8, 10, 13 (A, B, C)

Objectif :
Tu devras rendre un code mettant en place le système décrit ci-dessus.

- Il devra comporter un frontend (React) et un backend (Express, Node) et qu'il soit écrit en TypeScript. ✅

- Le refresh de la page ne doit avoir aucune conséquence sur l'état de la queue et des timers.
- Il devra etre rendu sous forme de repo Git.
- Il devra être implémenté à partir de ce boilerplate Nx : https://github.com/Waapi-Pro/test-boilerplate.
- Les commandes `nx start frontend` et `nx start backend` devront respectivement lancer le frontend et le backend.
- Des tests backends devront être écrits, la commande `nx test backend` devra lancer les lancer.
- Le choix de la persistance est libre (JS, fichier, SQL, NO-SQL, ...).
- Il n'y a pas besoin de mettre en place de système d'utilisateur.
- L'interface devra donc afficher au moins la queue d'action à exécuter, les crédits disponibles et donner la possibilité de rajouter des actions à exécuter à la queue.
- L'architecture de la codebase devra être scalable, une attention particulière sera portée dessus. Mettez en avant vos connaissances.
- Le serveur devra être robuste aux erreurs.

Il n'y a aucune limite de temps.
Une review du code avec des questions pourra avoir lieu.
L'énoncé peut cacher des subtilités, n'hésite pas si tu as des questions !