Exercice Technique

Développement d’une application MERN avec Redux et gestion multi-utilisateurs

Projet : TeamTask

1. Contexte

Vous êtes intégré(e) à une startup en pleine croissance souhaitant développer une application interne de gestion des tâches pour structurer et fluidifier la collaboration entre les membres de l’équipe.

Chaque collaborateur dispose d’un compte personnel sécurisé. Un chef d’équipe (manager) est responsable de la création et de l’assignation des tâches aux membres de l’équipe.

Dans le cadre de cet exercice, vous êtes chargé(e) de concevoir une première version opérationnelle de cette application en utilisant la stack MERN (MongoDB, Express.js, React, Node.js). Le frontend devra impérativement utiliser Redux Toolkit pour la gestion d’état.

2. Objectifs et Fonctionnalités attendues

2.1 Authentification (fonctionnalité bonus requise pour atteindre la note maximale)

- Mise en place d’un système d’inscription et de connexion des utilisateurs
- Authentification sécurisée via JSON Web Tokens (JWT), avec stockage dans le localStorage
- Middleware backend protégeant l’accès aux routes sensibles
- Accès restreint aux ressources selon le rôle de l’utilisateur

Fonctionnalités selon le rôle :

- Un utilisateur authentifié peut consulter uniquement ses propres tâches.
- Un manager peut créer de nouvelles tâches, les assigner à d'autres membres, et visualiser toutes les tâches.

2.2 Gestion des tâches (fonctionnalité obligatoire)

Implémentation d’un CRUD complet pour les tâches, avec les propriétés suivantes :

- `title` (obligatoire)
- `description` (facultatif)
- `status` (valeurs possibles : à faire, en cours, terminée)
- `assignedTo` (utilisateur concerné)

Routes backend attendues :

- `GET /tasks` : récupération des tâches de l’utilisateur connecté
- `POST /tasks` : création d’une tâche (seul le manager peut assigner une tâche à un autre utilisateur)
- `PUT /tasks/:id` : modification d’une tâche
- `DELETE /tasks/:id` : suppression d’une tâche

3. Gestion des rôles et permissions

Deux rôles distincts sont attendus dans l'application :

- **Utilisateur (User)**
  - Peut consulter uniquement ses propres tâches
  - Peut modifier le statut de ses tâches
- **Manager**
  - Dispose d’un accès global à toutes les tâches
  - Peut créer et assigner des tâches à n’importe quel utilisateur

4. Architecture technique attendue

Backend

- Technologies : Node.js, Express.js
- Authentification via JWT
- Base de données MongoDB, avec utilisation de Mongoose
- Schémas requis :
  - `User`

- `Task` (incluant une référence à l'utilisateur assigné)

Frontend

- Technologies : React, Redux Toolkit
- Gestion de l'état via Redux Toolkit
- Stockage local du token et des données utilisateur
- Pages à développer :
  - Page de connexion
  - Page d'inscription
  - Tableau de bord (dashboard)
  - Liste des tâches assignées avec possibilité de filtrer par statut
  - (Optionnel) Page Admin affichant la liste des utilisateurs

5. Livrables attendus

- Lien vers le dépôt GitHub contenant l'ensemble du code (frontend et backend)
- Fichier `README.md` complet incluant les instructions d'installation, de configuration et de lancement du projet
- Démonstration de l'application via :
  - Une vidéo de présentation, ou
  - Des captures d'écran annotées
- (Bonus) Lien de déploiement de l'application sur un service cloud (Render, Vercel, etc.)