Examen Angular 19 – Gestion d'état avec Signals

Noté sur 20 points

Consignes Générales

- Utilisez Angular 16 ou plus (idéalement 19).
- Implémentez la gestion d'état en utilisant `signal`, `computed`, et un store (`@Injectable`).
- Ne pas utiliser `localStorage`, ni `NgRx`.
- Utilisez `HttpClient` uniquement si spécifié.
- Structurez bien vos composants, modèles et stores.

-Créer un repository publique github pour chaque projet et envoyez le travail à

khalillakhdharatc@gmail.com

Barème Général

- Respect des consignes et architecture : 3 pts
- Fonctionnalité de base (état, actions) : 5 pts
- Utilisation correcte des signals : 4 pts
- Utilisation de computed / derived state : 3 pts
- Séparation logique / affichage (store vs component) : 2 pts
- UI claire, logique, dynamique : 3 pts

Exercice 1 : Gestion de favoris – 6 points

Créez une application affichant une liste d'articles. Chaque article doit pouvoir être ajouté ou retiré des favoris.
Utilisez un `FavoritesStore` pour stocker les IDs ou objets favoris.
Affichez dynamiquement la liste des favoris et leur nombre total (avec `computed`).

- Attendus :
- Signal pour stocker les favoris
- Méthodes : addFavorite(), removeFavorite()
- Computed pour totalFavoris()
- Affichage dynamique dans le composant

Exercice 2 : Panier e-commerce – 7 points

Créez une application simulant un panier e-commerce. L'utilisateur peut ajouter des produits, ajuster les quantités et voir le total.
Créez un `CartStore` qui gère l'état du panier et les opérations.

- Attendus :
- Signal avec une liste d'objets (produits avec quantité)
- Méthodes : addProduct(), removeProduct(), updateQuantity()
- Computed : totalAmount()
- Interface interactive + affichage du panier

Exercice 3 : Authentification avec profil – 7 points

Créez une interface de connexion avec deux champs (username/password).
Une fois connecté, affichez un message de bienvenue avec le nom d'utilisateur.
Utilisez un `AuthStore` global pour stocker les données reçues de l'API (simulée ou mockapi.io).

- Attendus :
- Méthode login() avec appel API (mockée)
- Signal pour user, token, loading, error
- Computed : isAuthenticated()
- Effect() pour log ou action secondaire
- Méthode logout() pour réinitialiser l'état