Test Technique – Développeur Fullstack MEAN Stack (Confirmé)

Contexte

Vous êtes chargé de développer une plateforme de blog collaboratif multi-auteurs avec :

- **Backend** : Node.js + Express + MongoDB
- **Frontend** : Angular 16+
- **Temps réel** : Socket.io
- **Sécurité & performance** : JWT, Rate Limiting, Hashing, Permissions dynamiques

La plateforme doit permettre aux utilisateurs de créer, modifier et commenter des articles avec une gestion avancée des permissions et notifications en temps réel.

Objectifs

- Démontrer une maîtrise avancée du MEAN stack.
- Concevoir une architecture **scalable, sécurisée et maintenable**.
- Implémenter un système de **permissions dynamiques** et rôles complexes.
- Optimiser les requêtes et performances MongoDB/Mongoose.
- Intégrer le temps réel avec WebSockets.
- Fournir une interface Angular fluide et réactive.

Fonctionnalités à implémenter

1. Gestion des utilisateurs & authentification

- Inscription et connexion via API sécurisée.
- Système de **rôles dynamiques** : Admin, Éditeur, Rédacteur, Lecteur.
- JWT + Refresh Token pour l'authentification.
- Les admins peuvent gérer les rôles via une interface Angular.

2. Gestion des articles (CRUD avancé)

- Les articles ont : titre, contenu, image, tags.
- Permissions :
  - Éditeur/Admin → modifier tout article
  - Rédacteur → modifier **seulement ses articles**
  - Admin → supprimer des articles uniquement
- Optimiser les requêtes MongoDB/Mongoose

3. Commentaires en temps réel (Socket.io)

- Les utilisateurs peuvent commenter et répondre à des commentaires existants (imbriqués).
- Notifications en temps réel pour l'auteur de l'article.

4. Sécurité et bonnes pratiques

- Rate Limiting (pour prévenir DDoS).
- Hashing des mots de passe avec bcrypt.
- Implémenter CORS correctement.
- Validation des entrées côté serveur (ex. injection MongoDB).

Bonus (Pour se démarquer)

- Notifications push (Web Push API).
- Dashboard Analytics avec Angular + Chart.js/Recharts pour statistiques articles.
- Architecture microservices : découper en UserService, ArticleService, NotificationService.
- ElasticSearch ou Redis pour optimisation des requêtes et caching.

Livrables

- Repo Git : backend + frontend.
- README clair :
  - Installation et exécution du projet
  - Structure du projet et choix techniques
- Tests unitaires ou e2e pour valider les rôles et permissions.

Durée estimée

16 à 20h, sur plusieurs jours.