TP ANAM AYA 3ALIAL GA

### **🔹 Rapport Général sur le Code**

## **📌 1. Contrôleur (controllers/login\_controllers.js)**

Le contrôleur gère l'inscription et la connexion des utilisateurs :  
✔ **signUp(req, resp)** :

* Récupère les données de l'utilisateur depuis req.body.
* Hache le mot de passe avec bcrypt avant de stocker l'utilisateur.
* Renvoie un message confirmant la création de l'utilisateur.  
  ✔ **signIn(req, resp)** :
* Recherche un utilisateur avec l’email fourni.
* Vérifie le mot de passe avec bcrypt.compare().
* Génère un **token JWT** si l'authentification réussit.
* Renvoie une erreur **401** si l'authentification échoue.

## **📌 2. Base de Données (models/user.js)**

Le schéma MongoDB (mongoose.Schema) définit la structure des utilisateurs :  
✔ fname, lName, email, et pwd sont obligatoires.  
✔ mongoose.model("User", userSchema) permet d'utiliser le modèle.  
✔ **Bonne structuration** des données, mais manque **d'indexation** pour email (pour éviter les doublons).

## **📌 3. Routes (routes/login\_routes.js)**

✔ Définit deux routes :

* POST /signUp pour l'inscription.
* POST /signIn pour la connexion.  
  ✔ Utilisation de express.Router() pour une meilleure organisation du code.  
  ✔ Routes bien définies, mais **aucune validation des entrées utilisateur**.

## **📌 4. Dépendances Importantes**

✔ express : Framework Node.js pour gérer les routes et requêtes.  
✔ mongoose : ORM pour interagir avec MongoDB.  
✔ dotenv : Chargement des variables d’environnement.  
✔ bcrypt : Hachage sécurisé des mots de passe.  
✔ jsonwebtoken (jwt) : Gestion de l’authentification avec tokens.  
✔ cors : Sécurise les requêtes entre le frontend et backend.

## **📌 5. Bonnes Pratiques Utilisées ✅**

✔ **Architecture MVC** : Séparation des **contrôleurs**, **modèles** et **routes**.  
✔ **Utilisation de bcrypt pour hacher les mots de passe**.  
✔ **JWT pour gérer l’authentification**.  
✔ **Utilisation de dotenv** pour stocker les informations sensibles (SECRET\_KEY, DB\_URI).  
✔ **Gestion des erreurs de connexion MongoDB** avec catch(err).

**🟠 Points à améliorer :**❌ Ajouter **des validations** (email unique, longueur du mot de passe).  
❌ Gérer les **erreurs spécifiques** (ex: email non trouvé, mauvais mot de passe).  
❌ **Limiter les tentatives de connexion** pour éviter les attaques par force brute.

## **📌 6. Sécurité JWT 🔐**

✔ **Le token JWT est signé avec process.env.SECRET\_KEY**.  
✔ **Expiration du token définie (expiresIn: "2h")**.  
✔ **JWT stocké en réponse (à utiliser côté client pour les requêtes sécurisées)**.

**🟠 Points à améliorer :**❌ **Stocker JWT dans un cookie HttpOnly pour éviter le vol de token via XSS**.  
❌ **Implémenter une route protégée (middleware) pour vérifier JWT**.

