WebappCi

📘 Cahier des Charges - Projet SCOLIA. Développé par WebappCi

1. Présentation du Projet

Scolia est une plateforme **SaaS (Software as a Service) Éducative Multi-Tenant**. Elle permet à plusieurs établissements scolaires de gérer leur administration, leur pédagogie et leurs finances de manière centralisée, sécurisée et cloisonnée.

Objectif principal : Moderniser la gestion scolaire (Afrique de l'Ouest/CI) en intégrant le suivi financier (Mobile Money), la communication temps réel et l'intelligence artificielle.

---

2. Architecture Technique

2.1. Stack Technologique

- **Frontend :** React.js, Vite, TypeScript.
  - *Gestion d'état :* Context API (AuthContext).
  - *Communication :* Axios (avec Intercepteurs JWT).
  - *UI :* CSS-in-JS / Styled, React Hot Toast (Notifications), React Icons.
- **Backend :** NestJS (Node.js), TypeScript.
  - *ORM :* TypeORM.
  - *Architecture :* Modulaire, Event-Driven (pour découplage), Services/Controllers.
- **Base de Données :** PostgreSQL (Hébergé sur Neon Tech).
- **Infrastructure :**
  - *Backend :* Render (Web Service).
  - *Frontend :* Vercel.
- **Services Tiers :**
  - *IA :* Google Gemini API (Génération emplois du temps).
  - *Notifications :* Firebase Cloud Messaging (FCM).
  - *Mails :* Nodemailer (SMTP).
  - *Visio :* Jitsi Meet Integration.

2.2. Principes d'Architecture

- **Multi-Tenancy (Cloisonnement) :** Chaque donnée (User, Grade, Fee, etc.) est liée à un `schoolId`. L'API force ce filtre pour empêcher les fuites de données entre écoles.
- **Sécurité :**
  - Authentification JWT (Access Tokens).
  - Hachage des mots de passe via BCrypt.
  - RBAC (Role-Based Access Control) via Guards et Décorateurs `@Roles`.
  - Protection CORS et Helmet.
  - Rate Limiting (Throttler) pour éviter le brute-force.

---

3. Acteurs et Rôles

1. **Super Admin (Plateforme) :**
    - Création de nouvelles écoles (Onboarding).
    - Activation/Désactivation des modules par école (Feature Flipping).
    - Suspension d'accès (Coupure abonnement).
2. **Admin (Directeur d'école) :**
    - Gestion des utilisateurs (Profs, Parents, Élèves).
    - Import en masse (CSV).
    - Validation des paiements.
    - Génération des emplois du temps.
    - Vue Radar de Risque.
3. **Enseignant :**
    - Saisie des notes (Bulk).
    - Saisie de l'appel (Présence/Retard).
    - Évaluation des compétences (Soft skills).
    - Déclaration d'absence/retard (Notification Admin/Parents).
    - Création de devoirs.
4. **Parent :**
    - Consultation des notes et bulletins.
    - Paiement de la scolarité (Soumission référence Mobile Money).
    - Réception des alertes (Absences, Retards prof).
5. **Élève :**
    - Consultation notes, devoirs et emploi du temps.

---

4. Fonctionnalités Détaillées (Modules)

4.1. Module Gestion Utilisateurs (`Users` / `Students`)

- **CRUD complet :** Création, modification, suppression.
- **Liaison User-Student :** Architecture One-to-One séparant le compte de connexion (User) du profil métier (Student).
- **Import CSV :** Création en masse avec détection automatique des classes et création des comptes financiers.

4.2. Module Pédagogie (`Grades` / `Skills` / `Bulletins`)

- **Notes :** Saisie rapide par lot (Bulk Insert).
- **Moyennes :** Calcul automatique de la moyenne pondérée (sur 20) tenant compte des coefficients.
- **Bulletins :** Génération par trimestre, appréciation du conseil, calcul de moyenne générale.
- **Compétences :** Évaluation par étoiles (1-5) des savoir-être.

- **Blocage Financier :** Le bulletin est masqué si la scolarité n'est pas à jour (Flag `isBlocked`).

4.3. Module Finance (`Payments`)

- **Configuration :** Définition du montant total et de la date limite par élève.
- **Paiement :** Le parent soumet une référence de transaction (ex: Orange Money).
- **Validation :** L'Admin valide ou rejette la transaction. Le solde se met à jour.
- **Automatisation :** Création automatique du compte `Fee` à l'inscription de l'élève (via Event Emitter).

4.4. Module Vie Scolaire (`Attendance` / `Timetable`)

- **Appel Numérique :** Saisie des absents/retards par classe.
- **Alertes Professeurs :** Le prof signale son retard -> Notification Push aux parents concernés.
- **Emploi du Temps IA :** Génération automatique du planning via **Google Gemini** en fonction des contraintes (matières/heures) fournies.

4.5. Module Communication (`News` / `Notifications`)

- **Fil d'actualité :** Annonces ciblées (Tout le monde, Profs uniquement, etc.).
- **Push Notifications :** Intégration Firebase pour alertes immédiates.
- **Visio :** Salle de classe virtuelle intégrée (Jitsi) avec lien sécurisé unique par jour.

4.6. Module Analytics (`Risk Radar`)

- **Algorithme prédictif :** Détecte les élèves à risque de décrochage.
- **Critères :**
  - Risque Financier (Paiement < 30%).
  - Risque Académique (Moyenne < 10/20).
- **Action :** Boutons d'action rapide (WhatsApp/Appel Parent).

---

5. Modèle de Données (Schéma Simplifié)

- **School :** `id, name, modules (jsonb), isActive`
- **User :** `id, email, password, role, schoolId`
- **Student :** `id, userId (OneToOne), classId, parentId`
- **Class :** `id, name, schoolId`
- **Grade :** `id, value, coef, studentId, period`
- **Fee :** `id, totalAmount, amountPaid, studentId`
- **Transaction :** `id, amount, status (Pending/Validated), mobileMoneyRef`
- **Attendance :** `id, status, date, studentId`
- **TimetableEvent :** `id, day, start, end, subject`

6. Livrables Attendus

1. **Code Source Backend :** Dépôt Git structuré NestJS (propre, typé, sans dépendances circulaires).
2. **Code Source Frontend :** Dépôt Git React/Vite.
3. **Base de Données :** Schéma relationnel PostgreSQL déployé.
4. **URLs de Production :**
    - API : `https://scolia-backend.onrender.com`
    - Client : `https://scolia.vercel.app`