# **Spécification Fonctionnelle et Technique – Plateforme de Réservation de Cours & des ateliers**

## **🌟 Objectif du projet**

Créer une plateforme web permettant :

* Aux professeurs indépendants de proposer des cours (en ligne ou présentiel)
* Aux élèves de rechercher, réserver, payer, discuter, et évaluer des cours

Ajout d’une couche d’intelligence artificielle avec deux modules IA :

* Génération automatique de quiz à partir des descriptions de cours
* Détection automatique de comportements suspects (fraude/spam)

## **🔧 Technologies utilisées**

* **Frontend** : Next.js 13+, Tailwind CSS, React Hook Form, Zustand / Redux
* **Backend** : Django + Django REST Framework (DRF)
* **Base de données** : PostgreSQL
* **Authentification** : JWT ou OAuth2
* **Paiement** : Stripe ou simulateur
* **Messagerie temps réel** (optionnel) : Django Channels / WebSocket
* **IA (Quiz + Anomalie)** : Hugging Face (T5), scikit-learn / PyOD
* **Déploiement** : Vercel (frontend), Railway / Render (backend)

## **📅 Fonctionnalités principales**

### **1. Authentification & Gestion utilisateur**

* Connexion / Inscription (email + mot de passe)
* Rôle : professeur ou élève
* Gestion du profil, photo, bio

### **2. Gestion des professeurs**

* Création de profil professionnel
* Ajout de cours avec description, tarif, localisation, disponibilités

### **3. Recherche & Découverte**

* Filtres (matière, niveau, localité, avis)
* Page détaillée d’un cours ou professeur

### **4. Réservation & Paiement**

* Choix d’un créneau disponible
* Paiement via Stripe / simulateur
* Confirmation et e-mail automatique

### **5. Avis & Notation**

* Commentaires, étoiles
* Affichage moyen des notes

### **6. Messagerie (optionnel)**

* Discussions privées élève-professeur
* Notifications temps réel

### **7. Tableau de bord personnalisé**

* Interface différente pour professeur ou élève
* Historique des réservations ou cours donnés

### **8. Interface responsive**

* Optimisation mobile / tablette / desktop

## **🤖 Fonctionnalités IA intégrées**

### **✍️ 1. Génération automatique de quiz**

#### **Objectif**

* Générer automatiquement un quiz (QCM) à partir de la description du cours saisie par le professeur.

#### **Fonctionnement**

* Analyse du texte avec un modèle NLP (flan-t5-base, t5-small, etc.)
* Génération de 3 à 5 questions avec choix multiples + réponse correcte
* Sauvegarde des quiz dans la base de données
* Affichage pour les élèves avant ou après un cours

#### **Technologies**

* Hugging Face Transformers en local/API gratuite
* Django (modèles Quiz, Question, Option)

### **🕵️ 2. Détection de comportement suspect**

#### **Objectif**

* Identifier automatiquement les comportements à risque (spam, fraudes, abus)

#### **Critères de détection**

* Nombre anormal de réservations sans paiement
* Multiples commentaires négatifs ou toxiques
* Messages identiques / suspects

#### **Fonctionnement**

* Traitement périodique (batch) ou temps réel
* Analyse comportementale via scikit-learn (Isolation Forest) ou PyOD
* Génération d'une alerte admin et blocage temporaire automatique si seuil atteint

#### **Technologies**

* Backend Django
* Modèle d’apprentissage non supervisé
* Dashboard admin dédié pour examiner les alertes

## **🎓 Utilisateurs cibles**

* **Professeurs** : gestion de cours, disponibilités, paiements, messagerie, quiz
* **Elèves** : recherche, réservations, quiz, historique, messagerie
* **Admins** : surveillance, modération, détection IA

## **🔍 Priorités de développement (MVP)**

1. Authentification / Profil utilisateur
2. Création / réservation de cours
3. Paiement intégré
4. Recommandation de professeurs (optionnel IA)
5. IA Quiz automatique (idée 15)
6. IA Détection d’abus (idée 16)
7. Messagerie (optionnel)

## **📊 KPIs et suivi**

* Nombre de cours réservés / donnés
* Taux de conversion (visite -> réservation)
* Nombre de quiz réalisés
* Alertes de comportement suspect traitées

## **📥 Déploiement**

* **Frontend** : Vercel
* **Backend** : Railway / Render
* **Base de données** : PostgreSQL (Railway, Supabase)
* **Modèles IA** : exécutés en local ou API HuggingFace (plan gratuit)

## **🚀 Scalabilité future**

* Suggestions IA plus avancées (recommandation personnalisée)
* IA de correction de rédaction (LanguageTool / GPT open-source)
* Traduction multilingue automatique
* Gamification autour des quiz (scores, badges)

……………………………………………………………………………………………………………

# **Frontend – Plateforme de réservation de cours**

## **🔧 Technologies**

* Next.js 13+ (App Router)
* Tailwind CSS (UI responsive)
* Zustand ou Redux (gestion d'état global)
* React Hook Form (formulaires)
* API REST (connexion au backend DRF)

## **📅 Pages / Routes principales**

* / : Accueil avec suggestions de professeurs
* /login, /register : Authentification
* /dashboard : Tableau de bord (élève ou professeur)
* /profile : Edition du profil utilisateur
* /teachers : Liste de professeurs avec filtres
* /teachers/[id] : Page détaillée avec réservation et avis
* /messages : Discussion privée (optionnelle)
* /admin/alerts : Vue admin pour alertes IA (comportement suspect)

## **🔊 Composants clés**

* Header / Footer adaptatif selon le rôle
* Composant de carte professeur
* Calendrier de disponibilités
* Formulaires : inscription, ajout de cours, filtre de recherche
* Paiement : intégration Stripe
* Notifications : confirmation de réservation, messages, alertes IA
* QuizViewer : afficher un quiz généré automatiquement

## **🚀 Intégration IA côté client**

### **Quiz automatique**

* Zone de prévisualisation du quiz dans la fiche cours
* Composant QuizPreview : rendu interactif du QCM
* Composant QuizResult : affichage des réponses de l'élève (en auto-évaluation)

### **Détection de comportement suspect**

* Interface admin : liste des alertes IA (utilisateur, type, date)
* Marquage visuel des comptes suspects dans le dashboard admin

## **🌐 Responsive Design**

* Utilisation extensive de Tailwind pour l'adaptation mobile / desktop
* Navigation adaptée : hamburger menu sur mobile, sidebar pour desktop

## **🔄 Données échangées avec le backend**

* Auth (JWT)
* Professeurs / cours (GET/POST)
* Réservations / paiements (GET/POST)
* Avis (GET/POST)
* Quiz (GET pour l’élève, POST par le prof)
* Alertes de comportement suspect (GET pour admin)

## **🔧 Outils recommandés**

* React Query ou SWR pour requêtes API
* Framer Motion pour animations
* Toast notifications (ex : sonneries, confirmations)

……………………………………………………………………………………………………………

# **Backend – Plateforme de réservation de cours**

## **🔧 Technologies**

* Django 4.x
* Django REST Framework (DRF)
* PostgreSQL
* Stripe SDK
* JWT Authentication (ou OAuth2)
* scikit-learn / PyOD (IA comportement)
* Transformers Hugging Face (IA quiz)

## **📆 Modèles principaux**

* User (rôle, bio, photo)
* TeacherProfile (spécialité, expérience, tarifs)
* Course (titre, description, prof, créneaux)
* Booking (cours, élève, date, statut)
* Review (note, texte, auteur)
* Message (optionnel)
* Quiz, Question, Option (liés à un Course)
* AnomalyAlert (utilisateur, type d’alerte, horodatage)

## **🚀 API Endpoints**

* /auth/login, /auth/register
* /users/profile, /users/update
* /courses/ (CRUD pour profs)
* /teachers/ (liste + filtres)
* /bookings/ (création + historique)
* /reviews/ (ajout + récup)
* /quiz/generate (POST)
* /quiz/:id (GET pour élève)
* /alerts/ (GET admin)

## **🤖 IA Quiz Automatique**

* Endpoint : POST /quiz/generate
* Entrée : description du cours
* Traitement : appel au modèle Hugging Face (T5 local/API)
* Sortie : JSON de questions + options + réponses
* Stockage lié au cours concerné

## **🕵️ IA Détection de comportement suspect**

* Traitement périodique (Celery / cron) ou appel live
* Analyse avec scikit-learn ou PyOD : IsolationForest
* Critères : réservations anormales, messages répétitifs, avis toxiques
* Création de AnomalyAlert
* Interface admin Django : supervision des alertes

## **🚪 Sécurité / Permissions**

* Permissions DRF (IsAuthenticated, IsAdminUser)
* JWT pour les routes privées
* Filtrage des données par rôle (prof/eleve/admin)

## **🚒 Services externes**

* Stripe : paiements et webhooks
* Hugging Face : modèles de langage (API ou local)
* (Optionnel) Redis + Django Channels pour messagerie

## **📅 Automatisations**

* Envoi de mails : confirmation de réservation, alertes admin
* Création de quiz à l’ajout d’un cours (trigger ou tâche asynchrone)
* Scans réguliers d’anomalies (cron / Celery Beat)