Cahier des charges

Projet d’un site scolaire

1. Contexte et objectifs du projet

1.1 Contexte

L'objectif est de concevoir un site web scolaire accessible à tous, permettant de fournir des services différenciés selon le type d'utilisateur : étudiants ou professeurs.

- Les étudiants pourront accéder à des cours en ligne moyennant un un compte et devoir payer un droit d’auteur pour chaque cours.
- Les professeurs pourront gérer les cours et consulter la liste des élèves inscrits.

1.2 Objectifs

- Créer une plateforme fonctionnelle pour la gestion et la consultation de cours.
- Mettre en place un système d'authentification et de gestion des rôles (professeur et étudiant).
- Intégrer un module de paiement en ligne pour les étudiants.
- Fournir une interface simple et ergonomique.

---

2. Périmètre fonctionnel du projet

2.1 Fonctionnalités pour les visiteurs (non connectés)

- Accès à une page d'accueil publique présentant le site.
- Possibilité de s'inscrire ou de se connecter.

2.2 Fonctionnalités pour les étudiants

- Accès à une liste de cours disponibles.
- Possibilité d'effectuer un paiement pour accéder à un cours spécifique.
- Consultation des cours payés.
- Suivi de leur historique de paiements.
- Gestion de leur profil utilisateur (nom, email, mot de passe).

2.3 Fonctionnalités pour les professeurs

- Consultation de la liste des étudiants inscrits.
- Visualisation des cours disponibles.

- Gestion des cours :
    - Ajout de nouveaux cours (titre, description, fichier PDF, prix).
    - Modification ou suppression des cours existants.

2.4 Fonctionnalités administratives (optionnel)

- Supervision des paiements et de l'activité sur la plateforme.

---

3. Technologies utilisées

Frontend :

- **HTML, CSS, JavaScript** : pour la structure et le design du site.
- Framework suggéré : **React.js** ou **Vue.js** pour un site dynamique.
- **Bootstrap** ou **Tailwind CSS** pour un design responsive.

Backend :

- **Node.js avec Express.js** ou **PHP (Laravel)** pour gérer les requêtes serveur.
- **Base de données relationnelle** : MySQL ou PostgreSQL (gestion des utilisateurs, cours, et paiements).

Paiements :

- Intégration d'un système de paiement en ligne pour le droit d’auteur (exemple : **PayPal**, **Stripe**, ou **Flutterwave**).

3.2 Authentification

- Utilisation de **JWT (JSON Web Tokens)** pour la gestion des sessions utilisateurs.
- Hashage des mots de passe (exemple : **bcrypt**).

3.3 Sécurité

- Mise en place d'un certificat SSL pour chiffrer les échanges de données.
- Protection contre les attaques (CSRF, XSS, injections SQL).

---

4. Architecture de la base de données

Tables principales

1. **Utilisateurs** :
    - `id` (int, PK)
    - `nom` (varchar)

- `email` (varchar, unique)
- `mot_de_passe` (varchar)
- `role` (enum : "étudiant" ou "professeur")

2. **Cours** :
    - `id` (int, PK)
    - `titre` (varchar)
    - `description` (text)
    - `auteur` (varchar)
    - `prix` (decimal)
    - `url_fichier` (varchar)
3. **Paiements** :
    - `id` (int, PK)
    - `id_utilisateur` (int, FK vers Utilisateurs)
    - `id_cours` (int, FK vers Cours)
    - `date_paiement` (datetime)
    - `montant` (decimal)

---

5. Maquettes et design

- Création de maquettes pour chaque page (avec **Figma**).
- Design épuré et responsive, adapté aux écrans mobiles.

Pages à maquetter :

1. Page d'accueil.
2. Interface étudiant (tableau de bord, consultation des cours, paiement).
3. Interface professeur (gestion des cours, liste des étudiants).
4. Pages de connexion et d'inscription.

---

6. Planning prévisionnel

- Création des maquettes.
- Mise en place de la base de données.
- Frontend : Conception des pages web.

- Backend : Mise en place des APIs et gestion des fonctionnalités.
- Intégration des paiements.

Tests et déploiement

- Tests unitaires et fonctionnels.
- Correction des bugs.
- Déploiement sur une plateforme en ligne (**Github, Netlify ou autre**).

7.Livrables attendus

- Un site web fonctionnel avec les interfaces étudiant et professeur.
- Documentation technique pour le code.
- Manuel utilisateur pour les étudiants et les professeurs.
- Rapport final sur les étapes du projet.

8. Contraintes et limites

- Le projet doit respecter les délais impartis.
- Priorisation des fonctionnalités essentielles pour garantir une livraison minimale viable.
- Les fonctionnalités additionnelles pourront être intégrées dans des versions futures.