Cahier des Charges

Application de Gestion de Cours et d'Évaluations pour une École

➢ 1. Présentation du Projet

➢ 1.1 Contexte

➢ Les établissements scolaires font face à de nombreux défis dans la gestion quotidienne des cours et des évaluations. Les méthodes traditionnelles (papier, Excel) sont souvent sources d'erreurs, chronophages et ne permettent pas un suivi efficace. Notre application vise à résoudre ces problèmes en proposant une solution digitale centralisée.

➢ 1.2 Objectifs

➢ Simplifier la gestion des cours pour les professeurs

➢ Faciliter la création et la correction des évaluations

➢ Permettre aux étudiants de suivre leurs cours et leurs résultats

➢ Fournir des outils d'analyse pour l'administration

➢ Améliorer la communication entre les différents acteurs

➢ 2. Description Fonctionnelle

➢ 2.1 Public Cible

➢ **Professeurs**: Gestion des cours, création d'évaluations, notation

➢ **Étudiants**: Consultation des cours, participation aux évaluations, suivi des résultats

➢ 2.2 Fonctionnalités Principales

➢ Pour les Professeurs:

➢ Créer, modifier et supprimer des cours

➢ Ajouter du contenu pédagogique aux cours (documents, liens)

➢ Créer des évaluations (QCM, questions ouvertes, etc.)

➢ Noter les évaluations des étudiants

➢ Suivre la progression des étudiants

➢ Communiquer avec les étudiants inscrits à leurs cours

➢ Pour les Étudiants:

➢ S'inscrire aux cours disponibles

➢ Consulter le contenu des cours

➢ Participer aux évaluations

➢ Consulter leurs notes et leur progression

➢ Recevoir des notifications (nouvelles évaluations, notes)

➢ 3. Spécifications Techniques

➢ 3.1 Architecture

➢ Backend: Laravel (API RESTful)

➢ Frontend: HTML, CSS, JavaScript (DOM, Ajax)

➢ Base de données: MySQL

➢ 3.2 Modèles de données principaux

➢ Utilisateurs (User, Student, Professor)

- Cours (Course)
- Évaluations (Evaluation)
- Notes (Grade)
- Documents/Ressources (Resource)

3.3 Sécurité

- Authentification sécurisée (Laravel Sanctum)
- Autorisations basées sur les rôles
- Protection des données sensibles
- Validation des entrées utilisateur

4. Contraintes et Exigences

4.1 Techniques

- Architecture MVC
- Programmation Orientée Objet avancée
- Respect des principes SOLID
- API REST conforme aux standards
- Code documenté et maintenable

4.2 Performances

- Temps de réponse rapide (< 2 secondes)
- Capacité à gérer plusieurs utilisateurs simultanés
- Optimisation des requêtes de base de données

4.3 Design

- Interface intuitive et responsive
- Expérience utilisateur adaptée à chaque type d'utilisateur
- Design épuré et professionnel

5. Livrables Attendus

- Code source complet (backend et frontend)
- Documentation technique (structure de l'application, API)
- Démonstration fonctionnelle

6. Critères de Réussite

- Application fonctionnelle répondant aux besoins décrits
- Code respectant les bonnes pratiques de développement
- Interface utilisateur intuitive et réactive
- Démonstration convaincante du produit