Conception de l'Application d'Apprentissage pour Élèves

Septembre 2025

Table des matières

1 Introduction

Cette application aide les élèves à apprendre leurs leçons via des images de cours (notes, scans). Elle inclut l'extraction automatique de notions, la génération de quizzes/mini-évaluations, des concours de classe, un emploi du temps de révisions avec notifications, et des corrections détaillées par IA, entraînées sur les notions des cours et les corrections des enseignants. Ce document présente la conception complète.

2 Exigences Fonctionnelles

- **Enregistrement et Gestion des Cours** :
  - Capture/upload d'images (notes, scans) via caméra/galerie.
  - Extraction de texte via OCR, génération de résumés et notions (NLP).
  - Stockage de multiples versions d'un cours pour enrichissement.
  - Annotations manuelles sur images.
- **Génération de Contenu Pédagogique** :
  - Création de quizzes/mini-évaluations basés sur notions.
  - Génération de flashcards et mind maps.
  - IA entraînée sur cours pour varier questions.
  - Gamification : points, badges.
- **Planification des Révisions** :
  - Emploi du temps personnalisé (spaced repetition).
  - Notifications push, intégration calendrier.
  - Messages personnalisés (ex. : "Bienvenue Nom !").
- **Suivi des Progrès** :
  - Tableau de bord : scores, graphiques, recommandations IA.
  - Rapports exportables (PDF), analyses comparatives.
- **Concours et Compétitions** :
  - Concours périodiques entre élèves d'une classe (leaderboard).
  - Corrections IA détaillées à la fin (quiz, éval, concours).
  - IA entraînée sur corrections enseignants + notions.
- **Fonctionnalités Utilisateurs** :
  - Inscription/connexion (rôles élève/enseignant).
  - Messages personnalisés globaux.
  - Partage collaboratif, mode offline, multilingue.
- **Administration et Modération** :
  - Gestion des classes, vérification contenu, organisation concours.

3 Exigences Non-Fonctionnelles

— **Performance** : OCR $<$ 5s, quiz généré $<$ 3s, leaderboards real-time $<$ 1s.
— **Sécurité** : Conformité RGPD, chiffrement AES, JWT, détection abus.
— **Scalabilité** : 10k+ utilisateurs, queues asynchrones (Celery).
— **Fiabilité** : 99.9% disponibilité, backups, retry OCR.
— **Accessibilité** : Interface responsive, TTS, dyslexie-friendly.
— **Technologies** : OCR (Tesseract/Google Vision), NLP (Hugging Face), Django backend.

4 Modèle de Données

4.1 Entités

— **User** : id, nom, rôle (élève/enseignant), email, password, classe$_id$.

— **Class** : id, nom, enseignant$_id$, élèves($listeuser_ids$).

— **Course** : id, titre, image$_url$, $texte_ocr$, résumé, $notions$, $versions$.

— **Quiz** : id, course$_id$, $questions(JSON)$, $type(quiz/mini-éval/concours)$, $difficulté$.

— **Evaluation** : id, quiz$_id$, $user_id$, $score$, $date$, $correction_id$.

— **Schedule** : id, user$_id$, $revisions(JSON)$.

— **Contest** : id, nom, date$_début$, $date_fin$, $classe_id$, $quiz_id$, $leaderboard$.

— **Correction** : id, texte, source (IA/enseignant), validated.
— **Annotation** : id, course$_id$, $annotations$.

4.2 Relations

— User –N :1–> Class
— Class –1 :N–> Contest
— Contest –1 :1–> Quiz
— User –1 :N–> Course
— Course –1 :N–> Quiz
— User –1 :N–> Evaluation
— Quiz –1 :N–> Evaluation
— Evaluation –1 :1–> Correction

— User –1 :1–> Schedule

— Course –N :N–> Course (versions)

— Course –1 :N–> Annotation

5 Architecture Globale

— **Frontend** : React Native (mobile, responsive).

— **Backend** : Python/Django, API REST + Channels (real-time concours).

— **Base de Données** : PostgreSQL.

— **Services Externes** : AWS S3 (images), Firebase (notifications).

— **IA Pipeline** : Image → OCR → Extraction notions → Génération quiz/corrections. Entraînement IA via Celery.

```
[Utilisateur] --> [App Mobile (Frontend)]
                  |
                  v
[API Backend (Django + Channels)] <--> [Base de Données (PostgreSQL)]
                  |
                  v
[IA Services] (OCR, NLP, Génération, Entraînement via Celery)
                  |
                  v
[Stockage Cloud] (Images, Datasets Entraînement)
```

6 Diagrammes UML

Pour inclure les diagrammes, suivez ces étapes :

1. Copiez chaque code Mermaid ci-dessous.
2. Collez dans `https://mermaid.live`.
3. Exportez l'image (PNG/SVG).
4. Placez les images dans le dossier du projet.
5. Décommentez les commandes `\includegraphics` ci-dessous.

6.1 Use Case Diagram

```
graph TD
    Eleve[  lve  ] --> UploadImage[Uploader une image cours]
    Eleve --> FaireQuiz[Faire un quiz/mini- val ]
    Eleve --> ParticiperConcours[Participer    un concours]
    Eleve --> VoirCorrections[Voir propositions de corrections]
    Eleve --> GererSchedule[G rer emploi du temps]
    Eleve --> RecevoirNotif[Recevoir notifications/messages
       personnalis s]
    Eleve --> VoirProgres[Voir progr s]
```

```
    Enseignant[Enseignant] --> UploadImage
    Enseignant --> GenererContenu[G n rer r sum s/quizzes]
    Enseignant --> FournirCorrections[Fournir corrections pour
       entra nement IA]
    Enseignant --> OrganiserConcours[Organiser concours de classe
       ]
    System[App] --> OCRExtraire[Extraire texte via OCR]
    System --> GenererResume[G n rer r sum /corrections]
    System --> Enrichir[Enrichir avec versions/corrections
       enseignants]
    System --> EntrainerIA[Entra ner IA sur corrections/notions]
```

6.2 Class Diagram

```
classDiagram
    User <|-- Eleve
    User <|-- Enseignant
    User --> Schedule
    User --> Class
    Course --> Quiz
    Course --> Summary
    Eleve --> Evaluation
    Evaluation --> Correction
    Class --> Contest
    Contest --> Quiz
    class User {
        +id: String
        +nom: String
        +role: Enum
        +classeId: String
        +login()
        +receivePersonalMessage()
    }
    class Class {
        +id: String
        +nom: String
        +enseignantId: String
        +eleves: Array<User>
    }
    class Course {
        +id: String
        +imageUrl: String
        +texteOcr: String
        +notions: Array
        +versions: Array<Course>
        +processOcr()
        +extractNotions()
    }
    class Quiz {
        +questions: Array
```

```
        +difficulte: Enum
        +type: Enum
        +generateCorrection()
    }
    class Contest {
        +id: String
        +dateDebut: Date
        +leaderboard: Array
        +startContest()
    }
    class Correction {
        +texte: String
        +source: Enum
        +trainIA()
    }
    class Schedule {
        +revisions: Array
        +sendNotification()
    }
```

6.3 Sequence Diagram (Concours et Correction)

```
sequenceDiagram
    participant Eleve
    participant App
    participant IA
    participant BD
    Eleve->>App: S'inscrire    un concours de classe
    App->>BD: V rifier classe et d marrer quiz
    Eleve->>App: Soumettre r ponses
    App->>IA:  valuer  et g n rer correction
    IA->>App: Correction d taill e
    App->>BD: Stocker  valuation  et leaderboard
    App->>Eleve: Afficher r sultats, correction, message ("Bravo
        [Nom] !")
```

7 Flux Utilisateur Principal

1. Connexion : Accueil avec "Bienvenue Nom !".
2. Upload image cours → OCR → Résumé → Quiz.
3. Participer à quiz/mini-éval/concours (notifs pour concours).
4. Résultats avec corrections IA + leaderboard (concours).
5. Schedule révisions avec messages personnalisés.
6. Enseignant upload corrections → Entraîne IA.

8 Technologies

— **Frontend** : React Native.
— **Backend** : Python/Django, Django Channels.
— **IA** : Tesseract/Google Vision (OCR), Hugging Face (NLP), Celery (jobs).
— **Autres** : PostgreSQL, AWS S3, Firebase.