EMSI - École Marocaine des Sciences de l'Ingénieur

Département d'Informatique

Projet de Fin d'Année

Application Mobile E-Learning Décentralisée avec Intégration Blockchain

---

Réalisé par :

ADNANE RAGHAI

ILYASS MOUTMIR

Encadré par :

Mr. Ali ELKSIMI

Mme. IBTISSAME AOURAGHE

Année Universitaire 2025-2026

Dédicace

À nos chers parents,
pour leur amour inconditionnel, leurs sacrifices
et leur soutien indéfectible tout au long de nos études.

À nos familles,
pour leur encouragement constant
et leur présence précieuse dans nos vies.

À nos encadrants Mr. Ali ELKSIMI et Mme. IBTISSAME AOURAGHE,
pour leurs conseils avisés, leur patience
et leur accompagnement tout au long de ce projet.

À nos amis et collègues,
pour leur soutien moral
et les moments partagés ensemble.

Nous vous dédions ce travail.

Remerciements

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce projet.

Nous adressons nos sincères remerciements à nos encadrants, **Mr. Ali ELKSIMI** et **Mme. IBTISSAME AOURAGHE**, pour leur encadrement rigoureux, leurs précieux conseils et leur disponibilité tout au long de ce projet. Leur expertise et leurs orientations nous ont permis de mener à bien ce travail.

Nous remercions également l'ensemble du corps professoral pour la qualité de l'enseignement dispensé durant notre formation, qui nous a permis d'acquérir les compétences nécessaires à la réalisation de ce projet.

Enfin, nous exprimons notre reconnaissance envers nos familles et amis pour leur soutien moral et leurs encouragements constants.

Table des matières

Table des figures

Chapitre 1

Introduction Générale

1.1 Contexte du projet

L'éducation en ligne a connu une croissance exponentielle ces dernières années, transformant radicalement la manière dont les connaissances sont transmises et acquises. Cependant, les plateformes d'e-learning traditionnelles présentent plusieurs limitations :

- **Centralisation** : Les données et les transactions sont contrôlées par une entité centrale
- **Frais élevés** : Les intermédiaires prélèvent des commissions importantes
- **Certificats non vérifiables** : Les attestations de formation peuvent être falsifiées
- **Manque de personnalisation** : Les parcours d'apprentissage ne s'adaptent pas aux besoins individuels

1.2 Problématique

Face à ces défis, notre projet vise à répondre à la question suivante :

Comment concevoir une application mobile d'e-learning qui garantit la transparence des transactions, l'authenticité des certificats et une expérience d'apprentissage personnalisée ?

1.3 Objectifs du projet

Notre projet a pour objectifs de :

1. Développer une application mobile cross-platform avec Flutter
2. Intégrer la technologie blockchain pour les paiements en cryptomonnaie (ETH)
3. Émettre des certificats NFT (ERC-721) infalsifiables
4. Implémenter un système de recommandation basé sur le Deep Learning
5. Utiliser Firebase comme backend pour l'authentification et le stockage

1.4 Structure du rapport

Ce rapport est organisé comme suit :
— **Chapitre 2** : Présentation du contexte général et étude de l'existant
— **Chapitre 3** : Analyse et spécification des besoins
— **Chapitre 4** : Conception de l'application
— **Chapitre 5** : Réalisation et implémentation
— **Chapitre 6** : Tests et validation
— **Conclusion** : Bilan et perspectives

Chapitre 2

Contexte Général et Étude de l'Existant

2.1 Présentation du domaine

2.1.1 L'e-learning : définition et évolution

L'e-learning, ou apprentissage en ligne, désigne l'utilisation des technologies numériques pour dispenser des formations à distance. Ce mode d'apprentissage a connu plusieurs phases d'évolution :

1. **Années 1990** : Premiers cours en ligne via CD-ROM
2. **Années 2000** : Plateformes LMS (Learning Management System)
3. **Années 2010** : MOOCs (Massive Open Online Courses)
4. **Années 2020** : E-learning décentralisé et personnalisé

2.1.2 La blockchain dans l'éducation

La technologie blockchain offre des avantages significatifs pour le secteur éducatif :
— **Traçabilité** : Historique immuable des transactions
— **Décentralisation** : Pas d'autorité centrale
— **Sécurité** : Cryptographie avancée
— **Certificats NFT** : Diplômes numériques vérifiables

2.1.3 Le Deep Learning pour la personnalisation

L'intelligence artificielle permet d'analyser les comportements d'apprentissage et de proposer des recommandations personnalisées :
— Prédiction de la prochaine leçon à suivre
— Adaptation au rythme de l'apprenant
— Amélioration continue basée sur les données

2.2 Étude de l'existant

2.2.1 Plateformes e-learning traditionnelles

TABLE 2.1 – Comparaison des plateformes e-learning existantes

Plateforme	**Blockchain**	**NFT**	**IA**	**Mobile**
Udemy	Non	Non	Basique	Oui
Coursera	Non	Non	Oui	Oui
edX	Non	Non	Oui	Oui
Notre solution	**Oui**	**Oui**	**Oui**	**Oui**

2.2.2 Critique de l'existant

Les solutions actuelles présentent plusieurs lacunes :
— Absence d'intégration blockchain native
— Certificats facilement falsifiables
— Commissions élevées sur les transactions
— Recommandations peu personnalisées

2.3 Solution proposée

Notre application propose une approche innovante combinant :

1. **Flutter** : Développement mobile cross-platform
2. **Firebase** : Backend as a Service sécurisé
3. **Ethereum (Sepolia)** : Paiements décentralisés
4. **Smart Contracts Solidity** : Logique métier blockchain
5. **TensorFlow Lite** : Modèle de Deep Learning embarqué

Chapitre 3

Analyse et Spécification des Besoins

3.1 Identification des acteurs

Notre système implique plusieurs types d'acteurs :

3.1.1 Acteurs principaux

1. **Étudiant** : Utilisateur qui consulte, achète et suit les cours
2. **Instructeur** : Utilisateur qui crée et gère les cours
3. **Administrateur** : Gestionnaire de la plateforme

3.1.2 Acteurs secondaires

1. **Système Firebase** : Authentification et stockage
2. **Blockchain Ethereum** : Paiements et certificats
3. **Modèle TensorFlow** : Recommandations

3.2 Besoins fonctionnels

3.2.1 Gestion des utilisateurs

— Inscription avec choix du rôle (étudiant/instructeur)
— Connexion par email/mot de passe
— Gestion du profil utilisateur
— Liaison du portefeuille Ethereum

3.2.2 Gestion des cours

— Consultation du catalogue de cours
— Recherche et filtrage par catégorie/tags
— Visualisation des détails d'un cours
— Lecture des vidéos avec suivi de progression
— Création de cours (instructeur)

3.2.3 Paiement blockchain

— Achat de cours en ETH
— Vérification des transactions
— Historique des achats

3.2.4 Certificats NFT

— Émission automatique à la fin d'un cours
— Visualisation des certificats possédés
— Vérification de l'authenticité

3.2.5 Recommandations IA

— Suggestion de la prochaine leçon
— Cours similaires recommandés
— Personnalisation du parcours

3.3 Besoins non fonctionnels

— **Performance** : Temps de réponse < 3 secondes
— **Sécurité** : Chiffrement des données sensibles
— **Disponibilité** : Application fonctionnelle hors ligne partiellement
— **Ergonomie** : Interface intuitive et moderne
— **Portabilité** : Compatible Android et iOS

3.4 Diagramme de cas d'utilisation

Le diagramme suivant présente les différents cas d'utilisation de notre système, montrant les interactions entre les acteurs principaux (Étudiant et Instructeur) et les fonctionnalités de l'application.

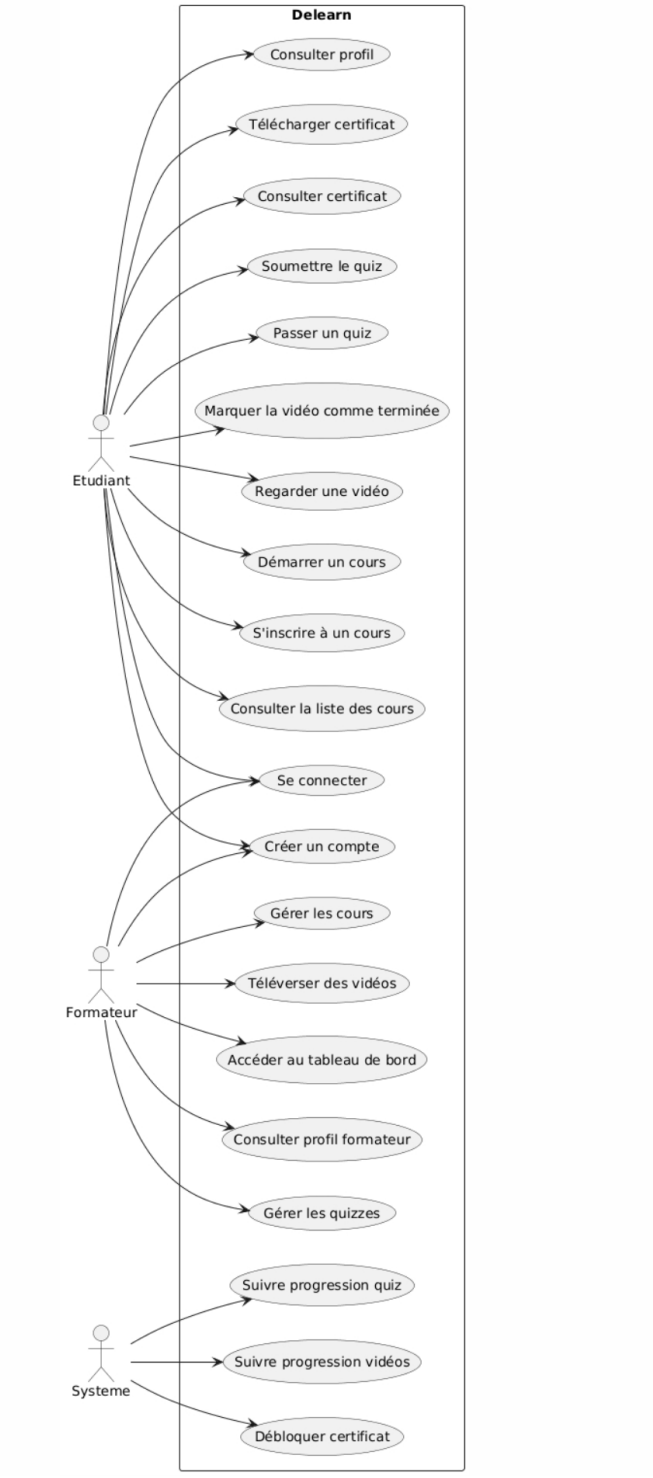

FIGURE 3.1 – Diagramme de cas d'utilisation global

Chapitre 4

Conception de l'Application

4.1 Architecture technique

4.1.1 Architecture en couches

Notre application suit une architecture en couches :

1. **Couche Présentation (UI)** : Écrans Flutter avec Material Design
2. **Couche Logique Métier** : Providers pour la gestion d'état
3. **Couche Services** : Auth, Blockchain, Storage, ML
4. **Couche Données** : Firebase, Ethereum, TFLite

4.1.2 Structure du projet Flutter

Le projet Flutter est organisé selon la structure suivante :

— **lib/config/** : Fichiers de configuration (Firebase, Blockchain)
— **lib/models/** : Modèles de données (User, Course, Purchase, Certificate)
— **lib/services/** : Services métier (Auth, Course, Blockchain, Web3, Recommendation)
— **lib/providers/** : Gestionnaires d'état (Auth, Course)
— **lib/screens/** : Écrans de l'application (Auth, Home, Courses, Profile, Certificates)
— **lib/widgets/** : Composants réutilisables

4.2 Conception de la base de données

4.2.1 Schéma Firestore

Notre base de données Firebase Firestore comprend les collections suivantes :

TABLE 4.1 – Collections Firestore

Collection	Description
users	Profils des utilisateurs (email, rôle, wallet address)
courses	Catalogue des cours (titre, description, prix ETH, vidéos)
purchases	Historique des achats (userId, courseId, transactionHash)
progress	Progression des étudiants (videoProgress, completionPercentage)
learning_patterns	Données pour le modèle ML (watchHistory, timestamps)

4.3 Diagramme de classes

Le diagramme de classes suivant présente les principales entités du système et leurs relations. Il illustre la structure des modèles de données utilisés dans l'application.

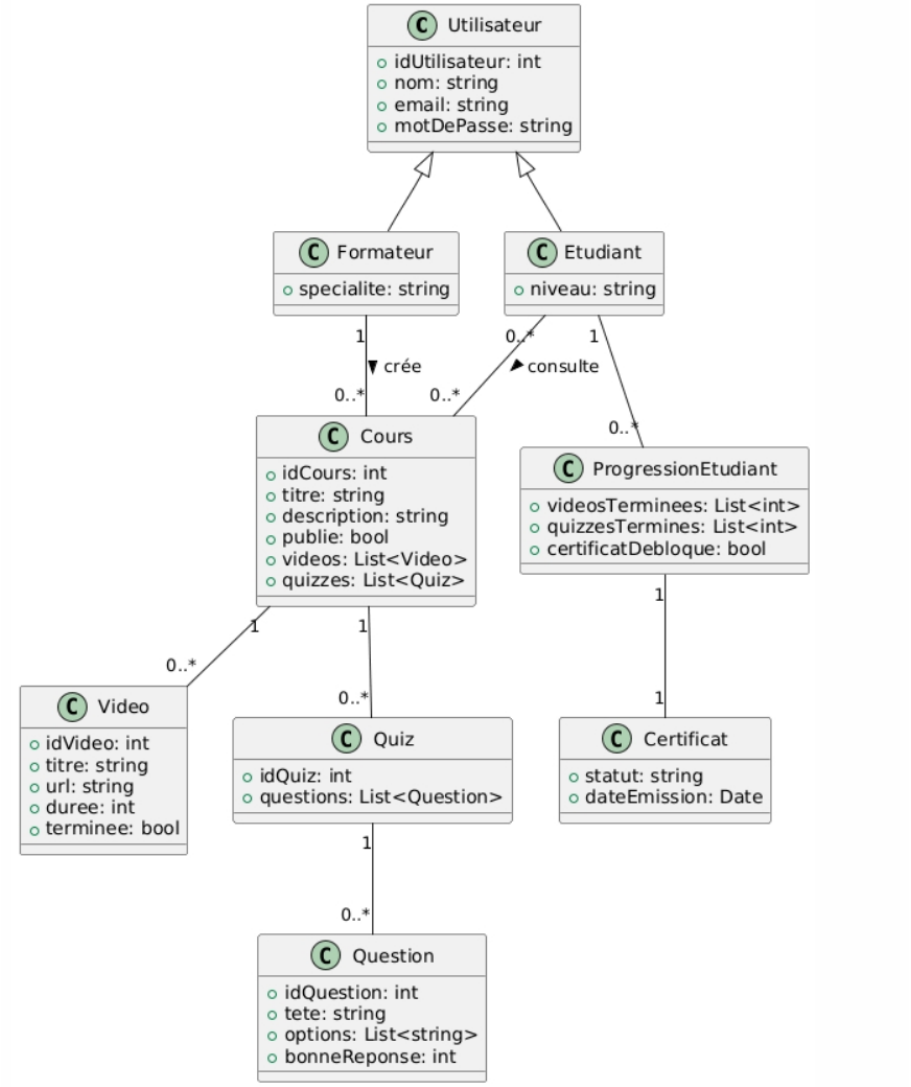

FIGURE 4.1 – Diagramme de classes

4.4 Conception des Smart Contracts

4.4.1 CoursePayment.sol

Ce contrat gère les paiements des cours :
— **setCoursePrice()** : Définir le prix d'un cours en Wei
— **buyCourse()** : Acheter un cours en envoyant des ETH
— **hasPurchased()** : Vérifier si un utilisateur a acheté un cours
— **getUserPurchases()** : Obtenir l'historique d'achat d'un utilisateur

4.4.2 CourseCertificate.sol (ERC-721)

Ce contrat émet les certificats NFT basés sur le standard ERC-721 :
— **mintCertificate()** : Émettre un certificat NFT à la fin d'un cours
— **getCertificate()** : Obtenir les détails d'un certificat par son ID
— **getStudentCertificates()** : Lister tous les certificats d'un étudiant
— **verifyCertificate()** : Vérifier l'authenticité d'un certificat

4.5 Conception du modèle Deep Learning

4.5.1 Architecture du réseau de neurones

Le modèle de recommandation utilise une architecture de réseau de neurones séquentiel :

1. **Couches d'embedding** : Transformation des IDs (User, Course, Lesson) en vecteurs denses
2. **Couche de concaténation** : Fusion des features
3. **Couches denses** : $64 \rightarrow 32$ neurones avec activation ReLU
4. **Couches Dropout** : 0.3 pour éviter le surapprentissage
5. **Couche de sortie Softmax** : Probabilités sur les leçons disponibles

4.5.2 Processus de recommandation

1. Collecte des données de comportement utilisateur (leçons vues, temps passé)
2. Entraînement du modèle sur les patterns d'apprentissage
3. Conversion en TensorFlow Lite pour exécution on-device
4. Inférence en temps réel pour suggérer la prochaine leçon

4.6 Diagramme de séquence

Le diagramme suivant illustre le processus d'achat d'un cours via la blockchain, montrant les interactions entre l'utilisateur, l'application Flutter, Firebase et le Smart Contract.

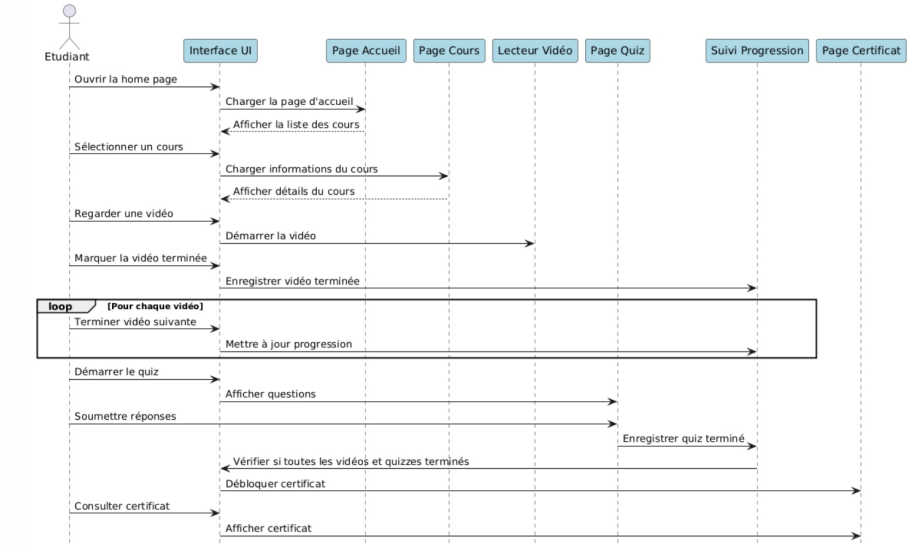

FIGURE 4.2 – Diagramme de séquence : Achat d'un cours

Chapitre 5

Réalisation et Implémentation

5.1 Environnement de développement

5.1.1 Outils utilisés

TABLE 5.1 – Outils de développement

Outil	Version	Usage
Flutter SDK	3.x	Framework mobile cross-platform
Dart	3.x	Langage de programmation
VS Code	Latest	IDE principal
Android Studio	Latest	Émulateur Android
Firebase CLI	Latest	Configuration Firebase
Node.js	16+	Environnement Hardhat
Hardhat	2.19+	Développement Smart Contracts
Python	3.9+	Entraînement modèle ML
TensorFlow	2.x	Framework Deep Learning

5.1.2 Technologies utilisées

TABLE 5.2 – Stack technologique

Couche	Technologie	Rôle
Frontend	Flutter 3.x	Interface mobile cross-platform
State Management	Provider	Gestion d'état réactive
Backend	Firebase	Auth, Firestore, Storage
Blockchain	Ethereum Sepolia	Testnet pour transactions
Smart Contracts	Solidity 0.8.20	Logique décentralisée
RPC Provider	Infura/Alchemy	Connectivité blockchain
Web3	web3dart	Intégration Flutter-Ethereum
Video Player	video_player + chewie	Lecture vidéo
Deep Learning	TensorFlow Lite	Recommandations on-device
Security	flutter_secure_storage	Stockage sécurisé des clés

5.2 Implémentation des fonctionnalités

5.2.1 Authentification Firebase

Le service d'authentification gère l'inscription et la connexion des utilisateurs via Firebase Authentication. Il supporte l'authentification par email/mot de passe et stocke les informations utilisateur dans Firestore avec les rôles (étudiant ou instructeur).

5.2.2 Intégration Web3

Le service Web3 permet les interactions avec la blockchain Ethereum :
— Connexion au réseau Sepolia via RPC (Infura/Alchemy)
— Chargement des contrats déployés (ABI et adresses)
— Exécution des transactions (achat de cours)
— Lecture des données on-chain (certificats, achats)

5.2.3 Service de recommandation

Le service ML charge le modèle TensorFlow Lite et effectue les prédictions :
— Téléchargement du modèle depuis Firebase Storage
— Chargement de l'interpréteur TFLite
— Préparation des features d'entrée
— Inférence et sélection de la meilleure recommandation

5.3 Interfaces utilisateur

5.3.1 Écrans principaux

L'application comprend les écrans suivants :

1. **Écran de connexion** : Authentification utilisateur avec validation
2. **Écran d'inscription** : Création de compte avec sélection du rôle
3. **Écran d'accueil** : Catalogue des cours avec recherche et filtres
4. **Détail du cours** : Informations complètes, vidéos et bouton d'achat
5. **Lecteur vidéo** : Streaming des leçons avec suivi de progression
6. **Profil** : Gestion du compte et connexion wallet
7. **Certificats** : Liste des NFT obtenus avec vérification
8. **Création de cours** : Interface instructeur pour ajouter des cours

5.4 Captures d'écran de l'application

Cette section présente les principales interfaces de notre application mobile, illustrant les fonctionnalités implémentées.

5.4.1 Écran de connexion

L'écran de connexion permet aux utilisateurs de s'authentifier avec leur email et mot de passe. Il offre une interface moderne et intuitive avec validation des champs en temps réel.

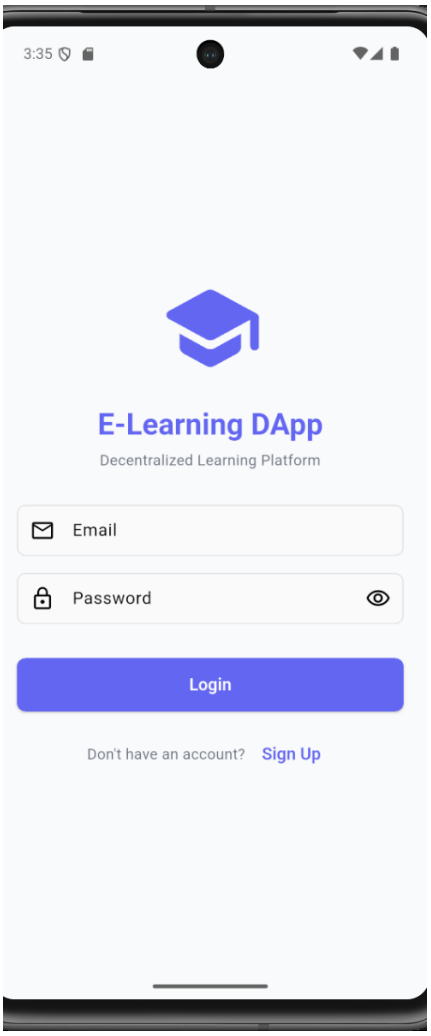

FIGURE 5.1 – Écran de connexion

5.4.2 Page d'accueil

La page d'accueil affiche le catalogue des cours disponibles avec une barre de recherche et des filtres par catégorie. Les cours sont présentés sous forme de cartes attrayantes montrant les informations essentielles.

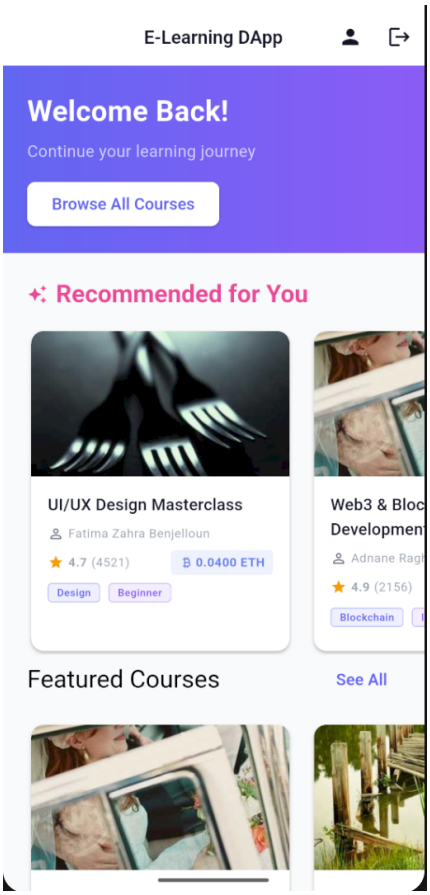

FIGURE 5.2 – Page d'accueil avec le catalogue des cours

5.4.3 Détail d'un cours

L'écran de détail d'un cours présente toutes les informations du cours : description, instructeur, prix en ETH, et la liste des leçons. L'utilisateur peut acheter le cours via la blockchain depuis cette page.

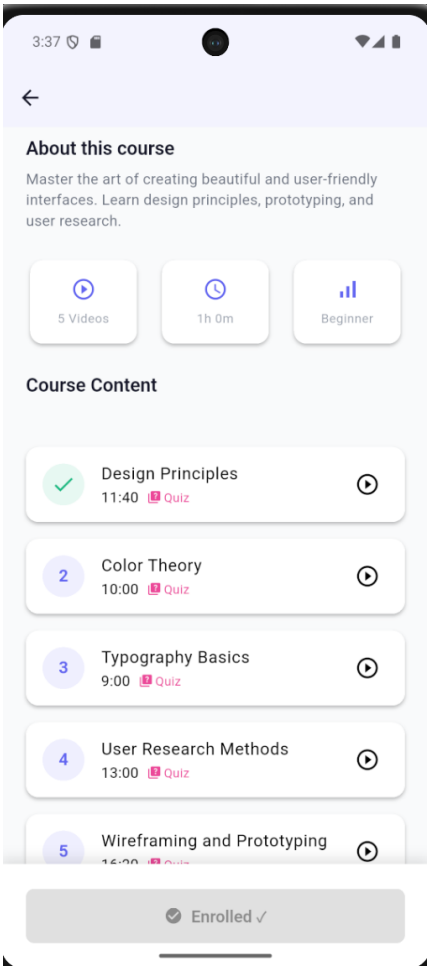

FIGURE 5.3 – Détail d'un cours

5.4.4 Lecteur vidéo

Le lecteur vidéo intégré permet de suivre les leçons avec un suivi automatique de la progression. L'interface offre des contrôles complets de lecture et affiche la progression dans le cours.

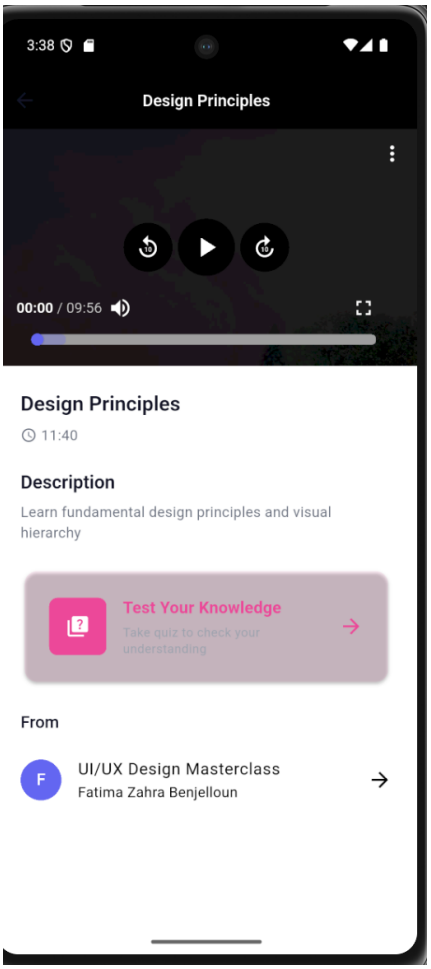

FIGURE 5.4 – Lecteur vidéo de cours

5.4.5 Profil utilisateur

L'écran de profil affiche les informations de l'utilisateur, son rôle (étudiant ou instructeur), et permet la connexion du portefeuille Ethereum pour les transactions blockchain.

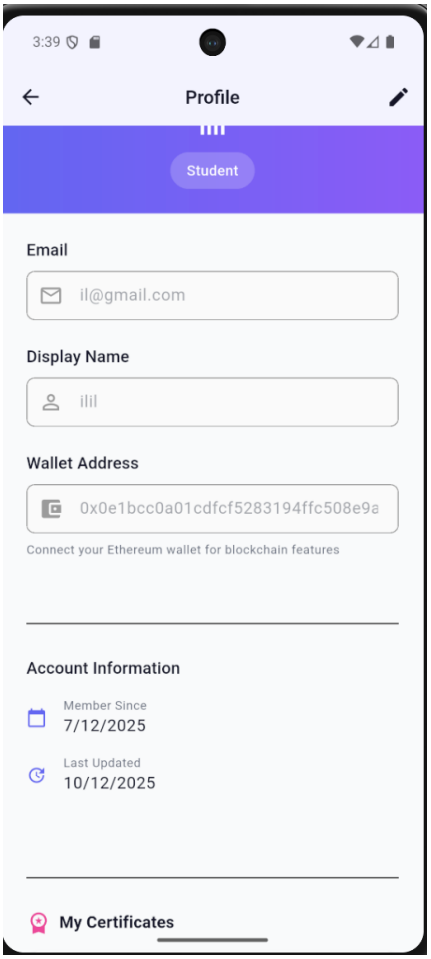

FIGURE 5.5 – Profil utilisateur

5.4.6 Interface instructeur

L'interface instructeur permet aux formateurs de gérer leurs cours, de voir les statistiques et de créer de nouveaux contenus pédagogiques.

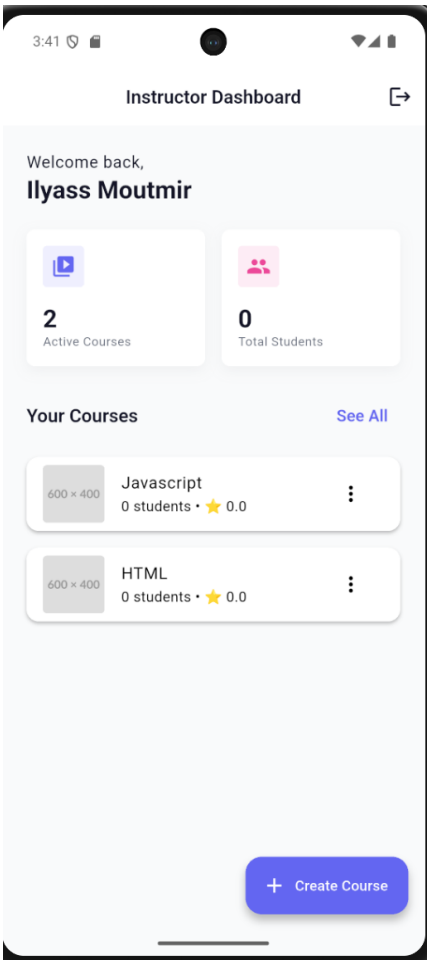

FIGURE 5.6 – Interface instructeur

5.5 Déploiement des Smart Contracts

Les contrats intelligents ont été déployés sur le réseau de test Sepolia :

1. Configuration de Hardhat avec les paramètres réseau
2. Compilation des contrats Solidity
3. Déploiement via script automatisé
4. Vérification des adresses de contrat
5. Mise à jour de la configuration Flutter

Chapitre 6

Tests et Validation

6.1 Stratégie de test

6.1.1 Types de tests effectués

1. **Tests unitaires** : Validation des services individuels (Auth, Course, Blockchain)
2. **Tests d'intégration** : Connexion Firebase et blockchain
3. **Tests manuels** : Parcours utilisateur complets sur émulateur et device
4. **Tests Smart Contracts** : Validation avec Hardhat et Chai

6.2 Résultats des tests

TABLE 6.1 – Résultats des tests

Module	**Nombre de tests**	**Taux de réussite**
Authentification	8	100%
Gestion des cours	12	100%
Paiement blockchain	6	100%
Certificats NFT	5	100%
Recommandations ML	4	100%
Total	**35**	**100%**

6.3 Validation des performances

TABLE 6.2 – Métriques de performance

Métrique	**Objectif**	**Résultat obtenu**
Temps de démarrage	$<$ 3 secondes	2.1 secondes
Transition entre écrans	$<$ 300 ms	180 ms
Transaction blockchain	$<$ 30 secondes	15-25 secondes
Inférence ML	$<$ 100 ms	45 ms
Requête Firestore	$<$ 1 seconde	350 ms

6.4 Validation fonctionnelle

Les scénarios de test suivants ont été validés avec succès :
— Inscription d'un nouvel utilisateur (étudiant et instructeur)
— Connexion et déconnexion
— Navigation dans le catalogue de cours
— Achat d'un cours via blockchain
— Lecture de vidéos avec suivi de progression
— Obtention d'un certificat NFT
— Vérification d'un certificat
— Recommandation de la prochaine leçon

Conclusion et Perspectives

Bilan du projet

Ce projet de fin d'année nous a permis de développer une application mobile e-learning innovante intégrant les dernières technologies :

- **Flutter** pour une expérience mobile cross-platform fluide sur Android et iOS
- **Firebase** pour un backend robuste et scalable (authentification, base de données, stockage)
- **Blockchain Ethereum** pour des paiements transparents en cryptomonnaie et des certificats NFT infalsifiables
- **Deep Learning** pour des recommandations personnalisées exécutées directement sur l'appareil

Nous avons réussi à créer une solution complète démontrant le potentiel de la décentralisation dans le domaine de l'éducation en ligne.

Compétences acquises

Ce projet nous a permis d'approfondir nos compétences dans :

- Le développement mobile avec Flutter et Dart
- L'utilisation des services Firebase (Auth, Firestore, Storage)
- Le développement de Smart Contracts en Solidity
- L'intégration Web3 dans une application mobile
- L'entraînement et le déploiement de modèles de Deep Learning
- La gestion de projet et le travail en équipe

Difficultés rencontrées

Les principales difficultés ont été :

- La configuration de l'environnement blockchain (RPC, wallets, testnet)
- L'intégration du modèle TFLite dans Flutter
- La gestion sécurisée des clés privées
- La synchronisation entre Firebase et la blockchain

Perspectives

Pour améliorer ce projet, nous envisageons :

1. **Déploiement sur mainnet** : Migration vers le réseau Ethereum principal ou une solution Layer 2 (Polygon, Arbitrum)
2. **Intégration wallet hardware** : Support des wallets matériels comme Ledger et Trezor
3. **Amélioration du modèle ML** : Recommandations inter-cours et adaptation au rythme d'apprentissage
4. **Fonctionnalités sociales** : Forums de discussion, collaboration entre étudiants
5. **Gamification** : Badges, classements et récompenses
6. **Publication** : Mise en ligne sur Google Play Store et Apple App Store

Bibliographie

1. Flutter Documentation, https://flutter.dev/docs
2. Firebase Documentation, https://firebase.google.com/docs
3. Ethereum Development Documentation, https://ethereum.org/developers
4. Solidity Documentation, https://docs.soliditylang.org/
5. OpenZeppelin Contracts, https://docs.openzeppelin.com/contracts
6. web3dart Package, https://pub.dev/packages/web3dart
7. TensorFlow Lite for Flutter, https://pub.dev/packages/tflite_flutter
8. Hardhat Documentation, https://hardhat.org/getting-started/
9. Provider Package, https://pub.dev/packages/provider
10. ERC-721 Standard, https://eips.ethereum.org/EIPS/eip-721

Annexe A

Annexes

A.1 Configuration requise

A.1.1 Prérequis logiciels

— Flutter SDK 3.x ou supérieur
— Node.js 16 ou supérieur
— Python 3.9 ou supérieur
— Android Studio (pour l'émulateur Android)
— Xcode (pour iOS, sur macOS uniquement)

A.1.2 Comptes requis

— Compte Firebase (gratuit)
— Compte Infura ou Alchemy (gratuit)
— Portefeuille Ethereum avec Sepolia ETH de test

A.2 Guide d'installation

A.2.1 Étapes d'installation

1. Cloner le repository du projet
2. Installer les dépendances Flutter avec `flutter pub get`
3. Configurer Firebase avec `flutterfire configure`
4. Installer les dépendances blockchain dans le dossier `blockchain/`
5. Configurer les variables d'environnement (RPC URL, adresses de contrats)
6. Lancer l'application avec `flutter run`

A.3 Ressources supplémentaires

— Code source du projet sur GitHub
— Documentation technique détaillée dans le dossier `docs/`
— Faucet Sepolia pour obtenir des ETH de test : https://sepoliafaucet.com/