المدرسة الوطنية للعلوم التطبيقية بتطوان
ⵜⵜⴰⵙⵙⴰⵔⵜ ⵜⴰⵏⴰⵎⵓⵔⵜ ⵏ ⵜⵣⵔⴼⵜ ⵜⵜⴰⵡⵓⵙⵙⴰⵏⵉⵜ ⵜⴰⵎⴰⵣⵉⵖⵜ
Ecole Nationale des Sciences Appliquées de Tétouan

Abdelmalek Essaâdi University

National School of Applied Sciences

Tetouan, Morocco

Data Science, Big Data & Artificial Intelligence

End-of-Project Report

Blockchain Based Assignment Management System

Prepared By:

Anouar Bouzhar

Hamza Kholti

Supervised By:

Prof. Imad Sassi

Module: Blockchain Fundamentals

Academic Year: 2025–2026

Contents

List of Figures

List of Tables

1 General Introduction

1.1 Project Context

The rapid expansion of online education has fundamentally transformed teaching and learning practices, offering unprecedented flexibility and global access to knowledge. In recent years, this transformation has been strongly reinforced by national digitalization strategies aimed at modernizing public services and higher education systems. In our country, the acceleration of digital transformation has led universities and engineering schools to increasingly adopt e-learning platforms, online assessments, and digital administrative processes.

Despite these advances, conventional online learning platforms continue to exhibit significant limitations in transparency, fairness, and security. Interactions between students and instructors often lack reciprocity, while administrative control remains concentrated among a small group of privileged users. This centralized structure frequently gives rise to misunderstandings, unnecessary disputes, and opportunities for academic misconduct, including cheating and unjustified grading.

In this context, the final project for the "Fundamentals of Blockchain" module at the National School of Applied Sciences of Tetouan (ENSA-Tétouan) aims to address these challenges by aligning with the ongoing digitalization efforts in the country. The project proposes the application of blockchain technology to design and implement a secure, transparent, and equitable system for managing assessments, ensuring the confidentiality of student submissions while effectively preventing plagiarism and reinforcing trust in digital academic processes.

1.2 Issues in Educational Platforms

Traditional online educational platforms suffer from inherent structural weaknesses that erode trust and academic integrity. The centralization of authority restricts full visibility and control to administrators, creating a clear power imbalance among participants. This lack of openness can encourage conflicts and facilitate misconduct, such as the sharing of assignment answers or biased evaluation practices. Furthermore, submitting work in plain text on inherently transparent systems exposes student submissions to unauthorized

access, significantly increasing the risk of plagiarism. The dependence on a single trusted entity also introduces vulnerabilities related to potential tampering and the absence of verifiable mechanisms to guarantee fairness throughout the assessment process. These shortcomings underscore the need for a more robust, decentralized approach to educational technology.

1.3 Project Objectives

The main objective of this project is to develop a blockchain based assessment management system that addresses the limitations of existing platforms. The system is designed to preserve the confidentiality of all communications, eliminate opportunities for cheating, and foster a trustworthy environment in which students, instructors, and administrators possess appropriate rights and engage in fully verifiable interactions. The core functional requirements include the secure distribution and submission of assignments, the protected delivery of grades to students, and the reliable dissemination of course-related announcements. Additional modules may be incorporated to improve overall usability and administrative efficiency. Ultimately, the platform seeks to establish a secure, transparent, and equitable framework that inspires confidence across all stakeholders.

1.4 Benefits of Using Blockchain Technology

Blockchain technology provides a powerful set of characteristics that directly resolve many of the deficiencies observed in conventional educational platforms. Its inherent immutability guarantees that once academic interactions are recorded, they cannot be modified, thereby creating permanent and auditable records of submissions, grades, and announcements. The decentralized nature of validation removes reliance on a single controlling authority, distributing trust among network participants while preserving accountability through cryptographic identities. Metadata and transaction histories remain transparent for auditing purposes, yet sensitive content is protected via encryption, achieving an optimal balance between openness and privacy. The capability to automate rule enforcement through smart contracts further minimizes human intervention and potential bias, for instance by strictly applying submission deadlines or grade release protocols. Together with strong cryptographic safeguards, these features enable the construction of an educational ecosystem that is simultaneously transparent, tamper-resistant, private, and fair qualities

that are exceedingly difficult to attain with traditional centralized architectures.

2 Theoretical Concepts and State of the Art

2.1 Blockchain: General Principles

Blockchain is a distributed ledger technology enabling transparent, secure, and decentralized information storage and transmission. Introduced in 2008 by Satoshi Nakamoto with Bitcoin [1], this technology organizes data in cryptographically chained blocks. Each block contains transactions, a timestamp, and a cryptographic hash of the previous block, creating an immutable chain. This architecture makes falsification practically impossible, as modifying an old block requires recalculating all subsequent blocks in a distributed network.

Blockchain functions on fundamental principles. Decentralization eliminates central authority by distributing the ledger across peer nodes. Each node maintains a complete or partial blockchain copy and validates new transactions. This distributed architecture strengthens resilience against failures and attacks. Transaction validation occurs through consensus mechanisms like Proof of Work for Bitcoin, Proof of Stake for Ethereum 2.0 [2], and Practical Byzantine Fault Tolerance for private blockchains [3]. These mechanisms ensure all participants agree on blockchain state without mutual trust.

Immutability constitutes blockchain's essential property. Once validated and recorded, transactions become virtually impossible to modify due to cryptographic chaining and consensus mechanisms [4]. Any modification attempt is immediately detected by network nodes. Blockchain transparency allows all participants to verify complete transaction history, though this varies by type. Public blockchains make all transactions visible, while private blockchains implement restrictive confidentiality mechanisms.

2.2 Smart Contracts

Smart contracts represent a major blockchain evolution. Initially proposed by Nick Szabo in 1994 [5], they found practical implementation with Ethereum in 2015 [6]. A smart contract is an autonomous program executing automatically on blockchain when predefined conditions are met. Unlike traditional contracts requiring intermediaries, smart contracts

self-execute deterministically and transparently. Contract code deployed on blockchain becomes immutable, guaranteeing rules cannot be unilaterally modified. This eliminates trust requirements between parties, as blockchain protocol guarantees execution.

Smart contracts operate on a simple yet powerful model. They contain code defining conditions and actions. When transactions trigger contracts with appropriate parameters, they automatically verify condition satisfaction and execute corresponding actions. This execution replicates across network nodes, ensuring consensus and traceability [7].

In educational contexts, smart contracts offer significant opportunities. They automate grade management, assignment submission verification, and academic credit allocation [8]. For example, smart contracts can automatically record student grades upon teacher validation, creating immutable and verifiable records. This automation reduces human errors and increases grading system confidence. Smart contracts also implement complex business rules including evaluation criteria, submission deadlines, and delay penalties. These transparent and verifiable rules reinforce educational process fairness.

2.3 Public vs Private Blockchain

The distinction between public and private blockchains constitutes a fundamental architectural choice profoundly influencing system characteristics and use cases. These paradigms address different needs with distinct trade-offs in decentralization, performance, and control [9].

Public blockchains like Bitcoin or Ethereum are open without restriction. Anyone can join networks, read transactions, submit transactions, and participate in validation. This total openness maximizes decentralization and resilience. No one can censor transactions or stop networks, as no central control exists. However, this openness costs performance and confidentiality. Public blockchain consensus mechanisms, particularly Proof of Work, are intentionally slow and energy-intensive to guarantee trustless environment security.

Private blockchains restrict network access to predefined authorized participants. Organizations deploying private blockchains control network membership, data access, and transaction validation. This restriction enables faster and more efficient consensus mechanisms, as participants are known and relatively trustworthy [10]. Private blockchains offer better confidentiality with non-public data access.

Hybrid consortium or permissioned blockchains exist where organization groups share net-

work control. This model combines decentralization advantages with private blockchain control and performance. Hyperledger Fabric and Quorum exemplify platforms designed for this deployment type [11].

For academic assignment management systems, private blockchains present decisive advantages. They enable access control ensuring only authorized students and teachers interact with platforms. Sensitive student data remains confidential while benefiting from blockchain immutability and traceability. Superior private blockchain performance processes numerous assignment submissions without public blockchain throughput limitations.

2.4 Ethereum and Geth

Ethereum represents second-generation blockchain technology, introducing decentralized virtual machine concepts capable of executing arbitrary programs. Launched in 2015 by Vitalik Buterin's team [2], Ethereum transformed blockchain from simple transaction ledgers into complete distributed computing platforms.

Ethereum's distinctive characteristic is the Ethereum Virtual Machine (EVM), a Turing-complete execution environment enabling smart contract deployment and execution. Unlike Bitcoin's financial transaction limitations, Ethereum enables complex decentralized applications covering wide use case ranges. Ethereum smart contracts are typically written in Solidity [12], a contract-oriented programming language similar to JavaScript. Ethereum's account model differs fundamentally from Bitcoin's UTXO model. Ethereum uses state-based accounts where each account maintains balances and stores code and data. This approach simplifies complex application development and enables intuitive state management.

Geth, short for Go Ethereum, is the reference Ethereum protocol implementation written in Go language. Developed and maintained by the Ethereum Foundation, Geth enables deploying complete Ethereum nodes capable of participating in main networks or creating private networks. This flexibility makes it the preferred tool for developers and organizations deploying private Ethereum blockchains.

Geth offers complete command-line interfaces for blockchain interaction, allowing account creation, block mining, smart contract deployment, and transaction sending. For production applications, Geth exposes JSON-RPC APIs enabling programmatic blockchain in-

teraction. This standardized API is compatible with numerous client libraries like Web3.js for JavaScript or Web3.py for Python.

Within educational systems, Geth enables creating private Ethereum networks controlled by academic institutions. This configuration offers all Ethereum technology advantages, including smart contracts and immutability, while maintaining control over network access and performance. Deploying multiple Geth nodes ensures data replication and fault tolerance.

2.5 Asymmetric Cryptography

Asymmetric cryptography, or public-key cryptography, constitutes blockchain system security foundations. This technique uses mathematically linked key pairs: public keys freely shared and private keys kept secret. Data encrypted with one key decrypts only with the paired key [13].

Asymmetric cryptography relies on mathematically difficult problems. The RSA system, developed by Rivest, Shamir, and Adleman in 1977 [13], uses large prime number factorization difficulty. System security depends on computational infeasibility of deriving private keys from public keys, despite mathematical linkage.

In RSA systems, public keys encrypt messages only corresponding private key holders can decrypt, ensuring communication confidentiality. Conversely, private keys create digital signatures verifiable with public keys, authenticating message authors and guaranteeing integrity [14].

Blockchains extensively use asymmetric cryptography for critical functions. Each user possesses key pairs defining network identity. Public keys serve as transaction receiving addresses, while private keys sign transactions proving ownership and authorizing transfers. This architecture eliminates centralized authentication system requirements.

In assignment management systems, asymmetric cryptography implements essential security mechanisms. Students encrypt submissions with teachers' public keys, ensuring only teachers read content. Digital signatures ensure submission authenticity and prevent repudiation; students cannot deny submissions once signed with private keys.

Private key management represents major challenges in asymmetric cryptography systems. Private key loss results in irreversible access loss to associated resources. In educational contexts, recovery and backup mechanisms must be implemented while maintaining

system security.

2.6 Blockchain Contribution to Education

Blockchain application in education represents promising innovation addressing fundamental traditional system issues [15]. Educational institutions manage considerable sensitive data volumes including grades, diplomas, certifications, and academic works. Securing and verifying this data constitutes major challenges blockchain elegantly resolves. Blockchain immutability offers unprecedented academic record integrity guarantees. Once grades or diplomas are recorded on blockchain, fraudulent modification becomes impossible. This eliminates diploma falsification risks, a widespread problem costing employers and institutions considerable verification resources [8]. Blockchain-stored diplomas can be instantly verified by stakeholders with appropriate access, without requiring issuing institution contact.

Transparency and traceability constitute significant advantages. Students access complete and verifiable academic journey histories. Each interaction, assignment submission, and evaluation leaves permanent and auditable traces. This transparency strengthens trust between students and institutions, as grading and evaluation processes become verifiable and objective [16].

Blockchain decentralization reduces dependence on centralized systems vulnerable to failures and attacks. In traditional models, university central database loss could lead to decades of academic record disappearance. With blockchain distributed across multiple nodes, this vulnerability is eliminated. Even if some nodes fail, networks continue functioning and data remains accessible.

Smart contracts automate time-consuming administrative processes. Automatic academic credit allocation, prerequisite verification for course enrollment, and diploma distribution can be codified in smart contracts. This automation reduces human errors, accelerates processes, and frees administrative staff for higher value-added tasks [17].

For assignment management specifically, blockchain brings unique benefits. Timestamped and immutable submissions prevent submission deadline disputes. Students cannot modify assignments after submission, and teachers cannot deny receiving submissions. This clarity eliminates many educational process conflict sources.

Protection against plagiarism and cheating also benefits from blockchain. By combining

immutable timestamping with cryptographic hashing techniques, proof of anteriority can be created for academic works. If multiple students submit similar works, blockchain determines who submitted first, providing objective evidence in plagiarism accusations. Finally, blockchain facilitates academic mobility and international qualification recognition. Students can present verifiable credits and diplomas to any institution worldwide without long and costly bureaucratic processes. This academic credential portability takes increasing importance as education becomes increasingly international and distributed [15].

3 Analysis of Requirements and Specifications

3.1 System Actors

The system involves four main actors: the teacher, the student, the administrator, and the blockchain.

The teacher is responsible for creating and distributing assignments, publishing course-related announcements, and grading submissions. To ensure confidentiality, the teacher generates a public-private key pair and shares the public key with students alongside each assignment.

The student submits encrypted responses using the teacher's public key, receives grades and feedback securely, and accesses announcements. Submissions are protected against plagiarism through encryption and immutable recording on the blockchain.

The administrator serves as a supporting actor, handling tasks such as managing course distribution to teachers in accordance with administrative decisions and facilitating student enrollment in courses. These functions, while not directly related to the examination process, are essential for overall system administration and operational efficiency.

The blockchain acts as a neutral, decentralized infrastructure that provides immutable traceability for all interactions. It records transaction hashes for assignments, submissions, grades, and announcements, ensuring transparency of metadata while preserving the privacy of encrypted content.

3.2 Functional Requirements

The system must implement the following core functionalities as specified in the project guidelines:
The assignment distribution and submission module enables teachers to publish assignments accompanied by their public key. Students encrypt their responses and submit them before the deadline, with the system enforcing temporal constraints.
The grade delivery module allows teachers to send encrypted feedback and scores to individual students, with all grading actions recorded immutably for auditability.
The announcement module supports the dissemination of course-related information, either globally or targeted to specific courses, ensuring reliable and tamper-proof communication.
Additional useful modules, such as user management, course administration, and enrollment handling, have been incorporated to enhance the platform's completeness and usability.

3.3 Non-Functional Requirements

The system must satisfy several critical non-functional requirements to ensure its reliability and effectiveness:

- Security: All sensitive communications must be protected using cryptographic encryption (RSA-2048 recommended) to prevent unauthorized access or tampering.
- Confidentiality: Private keys are never stored on the server, and submission content remains encrypted on-chain, visible only to intended recipients.
- Immutability and Traceability: Every significant interaction (assignment creation, submission, grading, announcement) generates a transaction hash recorded on the blockchain, providing permanent, verifiable audit trails.
- Performance and Scalability: The platform must handle concurrent users efficiently while maintaining reasonable response times for common operations.
- Usability: Interfaces for students, teachers, and administrators must be intuitive and responsive.

These requirements collectively ensure a robust, trustworthy, and user-friendly educational platform.

3.4 Technical and Legal Constraints

The project is subject to specific technical and legal constraints:
On the technical side, the solution employs Django for the backend with RESTful APIs, React for the frontend, and cryptographic libraries for RSA encryption. The blockchain component is prepared for integration (transaction hashes stored), allowing future connection to permissioned networks such as Hyperledger Fabric or Quorum. Data persistence uses relational databases (PostgreSQL-compatible), and the architecture supports microservices principles.
Legally, the system respects data protection regulations (including GDPR principles) by minimizing personal data storage and ensuring confidentiality through encryption. Participant anonymity is preserved on the blockchain while maintaining accountability through cryptographic identities. All recorded activities remain traceable without compromising privacy. The development is carried out by teams of up to four students, with the final deliverable including a functional prototype demonstrating the three mandatory modules.

4 Global System Architecture

4.1 Architecture Overview

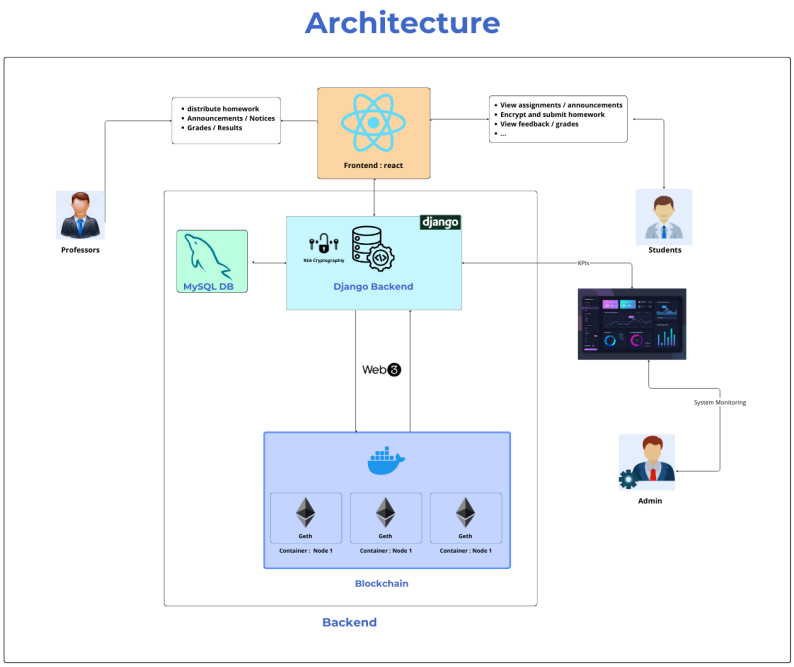


Figure 1: Global System Architecture

The blockchain-based assignment management system follows a three-tier architecture separating presentation, business logic, and blockchain layers. This design provides modularity and scalability while leveraging blockchain's unique properties of immutability and transparency. The system handles the complete assignment lifecycle, from creation and submission to evaluation and grade publication, with cryptographic guarantees of integrity.

The architecture builds on decentralized trust principles, where the blockchain serves as the single source of truth for academic transactions. A private Ethereum network replaces traditional centralized databases, eliminating single points of failure and ensuring that academic records remain accessible even during individual node failures. The design prioritizes security, transparency, and user experience across all layers.

4.2 System Layers Description

4.2.1 Frontend Layer: React

The presentation layer uses React for building dynamic user interfaces with component-based architecture. React's virtual DOM provides excellent performance for real-time up-

dates, essential for tracking assignment submissions and grade publications. The frontend implements role-based interfaces adapting to user identity. Teachers access dashboards for managing assignments and publishing grades, while students browse assignments, submit encrypted work, and view results.

State management utilizes React hooks and context API for clean data flow without external libraries. The frontend communicates with the backend exclusively through RESTful APIs, maintaining separation of concerns. Cryptographic operations for submission encryption occur client-side using Web Crypto API, ensuring data protection before transmission. Public keys retrieved from the blockchain establish direct cryptographic channels between students and teachers.

4.2.2 Backend Layer: Django

Django serves as the application server, providing robust security with built-in protection against SQL injection, XSS, and CSRF attacks. The backend manages authentication, authorization, and session state using industry-standard password hashing. It maintains a relational database for user profiles and caches blockchain data for improved query performance.

The backend bridges the web application and blockchain network through a service layer encapsulating all blockchain interactions. Django REST Framework exposes comprehensive RESTful endpoints for user registration, assignment operations, submissions, and grade retrieval. Each endpoint enforces role-based access control, ensuring users perform only authorized actions. The backend manages RSA key pairs for teachers, securely storing private keys for decrypting student submissions. Django's encryption utilities protect this sensitive material in the database. The Web3.py library facilitates blockchain communication, handling transaction construction, signing, and broadcasting to the network.

4.2.3 Blockchain Layer: Private Ethereum

A private Ethereum network with three nodes forms the trust foundation. This blockchain provides immutable storage for academic transactions and executes smart contracts enforcing business rules transparently. The network uses Proof of Authority consensus with the Clique algorithm, where authorized signers produce blocks at one-second intervals.

Node configuration includes one sealer creating blocks and two observers maintaining

synchronized copies. This setup achieves Byzantine fault tolerance with near-real-time confirmation. The network operates with a unique chain ID preventing replay attacks and zero gas prices eliminating transaction costs, ensuring equal access for all users.

Docker containerization provides consistent deployment environments. Each node runs Geth, the official Ethereum implementation. Docker Compose orchestrates the multi-container deployment, managing network configuration and persistent storage. Smart contracts written in Solidity define data structures for assignments, submissions, and results, enforcing deadline validation, access control, and submission immutability.

4.3 Component Interactions

Component interactions follow well-defined protocols ensuring data consistency and security. When teachers create assignments, the frontend generates RSA key pairs and sends requests to Django with assignment details. The backend constructs blockchain transactions calling smart contract functions, signs them, and broadcasts to the network. The sealer node validates and includes transactions in blocks, typically within seconds.

For submissions, students encrypt content with teachers' public keys retrieved from the blockchain. The backend stores encrypted submissions on-chain, ensuring only teachers with private keys can decrypt them. Grade publication involves smart contract verification that only assignment creators can evaluate submissions, creating immutable grade records.

All nodes maintain synchronized state through consensus. When one node receives transactions or blocks, it propagates information to peers. Observer nodes validate incoming blocks, ensuring proper signatures and protocol compliance. This distributed validation prevents academic record manipulation.

4.4 Justification for Private Blockchain

The private blockchain choice balances decentralization benefits with practical institutional requirements. Performance considerations favor private deployment, as public blockchains process transactions slowly with limited throughput. Private networks with Proof of Authority achieve sub-second block times and higher throughput, crucial during peak submission periods.

Cost management represents another advantage. Public blockchains require cryptocurrency gas fees, creating financial barriers. Private networks with zero gas prices allow

unlimited transactions without cost, providing equitable access regardless of economic circumstances. Privacy requirements also favor private deployment, as public blockchains expose all data. Private networks implement access controls protecting student privacy while maintaining data protection regulation compliance.

Regulatory compliance becomes manageable with institutional control over network operations and participant identification. Governance remains with the institution for network upgrades and policy decisions, enabling rapid responses to security issues without external consensus.

Despite being private, the blockchain maintains core value propositions. Immutability persists through cryptographic chaining and distributed replication. Transparency exists among authorized participants with visible transactions. Multi-node architecture provides fault tolerance and prevents unilateral manipulation. This approach captures essential blockchain benefits while avoiding public network drawbacks, delivering a functional, performant, and compliant system for educational needs.

5 Solution Modeling and System Design

5.1 Use Case Diagram

The use case diagram provides a comprehensive overview of the system's functionalities and the interactions between the different actors. It is organized into logical packages to reflect the main modules of the platform, ensuring clarity and readability.

The diagram identifies four primary actors:

- The **Teacher**, who creates and publishes assignments (including the associated public key), grades submissions, and publishes announcements and results.
- The **Student**, who views available assignments, downloads the teacher's public key, encrypts and submits responses, and consults grades, feedback, and announcements.
- The **Administrator**, who performs supporting administrative tasks such as user management, course creation, professor assignment, and student enrollment. These operations, while not directly part of the core examination process, are essential for the proper operation and setup of the platform.
- The **Blockchain Network**, modeled as an external system actor, which handles transaction validation, immutability, and traceability of all recorded interactions.

The use cases are grouped into functional packages: Assignment Management, Grading Management, Announcement Management, Blockchain Core Operations, User Management, and Security Services. Key relationships such as `<<include>>` (e.g., submitting an assignment includes encryption and transaction creation) and `<<extend>>` (e.g., plagiarism detection as an optional extension of submission decryption) highlight the dependencies and optional behaviors within the system.

This diagram effectively captures the secure workflow centered on RSA encryption for confidentiality, combined with blockchain recording for transparency and anti-cheating measures.

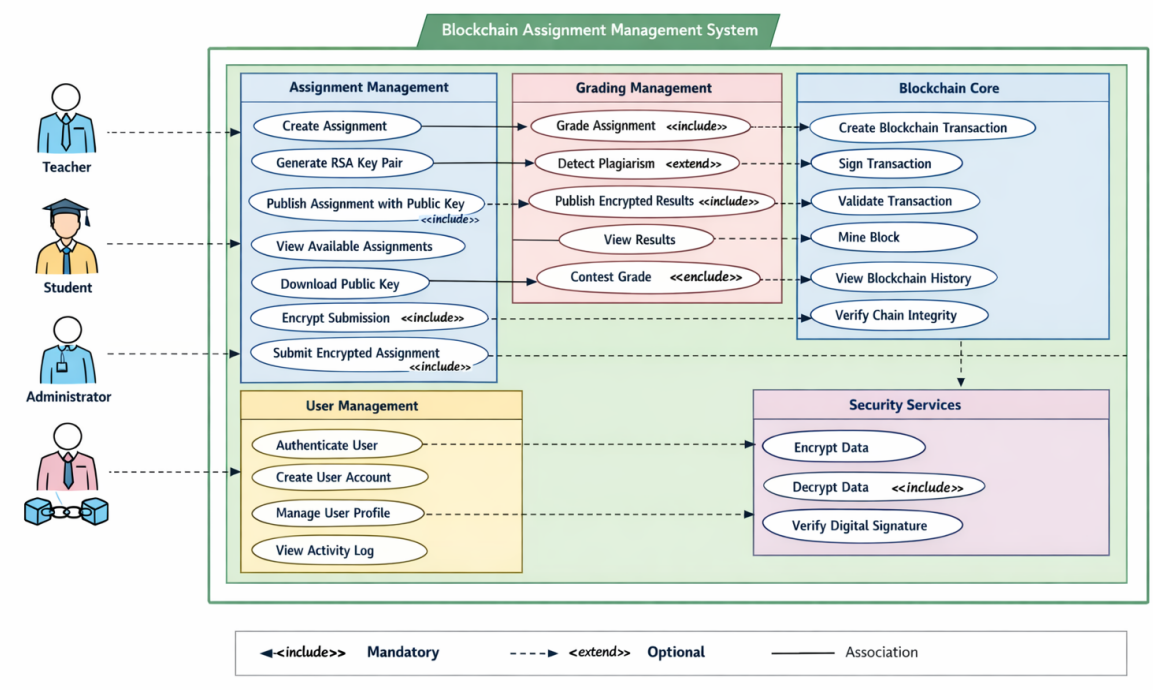


Figure 2: Use Case Diagram of the Blockchain System

5.2 Sequence Diagrams

Two sequence diagrams have been developed to illustrate the core business processes of the system at the application level: the submission of an assignment and the consultation of results. These diagrams focus on the logical flow of interactions between actors and system components, emphasizing the integration of cryptographic mechanisms and blockchain recording for security and traceability.

The first diagram, titled "Encrypted Assignment Submission via Blockchain", depicts the complete workflow from assignment retrieval to immutable recording on the blockchain.

It highlights the critical role of RSA encryption in protecting submission confidentiality and the binding of student identity to the response as an anti-plagiarism measure. The process includes authentication, encryption, digital signing, transaction broadcasting, consensus validation, and final confirmation with blockchain proof (transaction hash and timestamp). Alternative scenarios cover deadline expiration, authentication failure, and network unavailability.

The second diagram, titled "Authenticated Grade Retrieval with Cryptographic Verification", models the secure consultation of results. It emphasizes the mandatory verification of the teacher's digital signature before any decryption, ensuring grade authenticity. The workflow encompasses student authentication, retrieval of encrypted grade transactions from the blockchain, signature validation, authorized decryption, and assembly of a comprehensive report with audit trail elements (timestamps, transaction references). An optional branch illustrates the grade contestation process, which generates an immutable dispute record on the blockchain.

These diagrams collectively demonstrate how the system combines asymmetric cryptography with blockchain immutability to provide confidentiality, non-repudiation, transparency, and fairness throughout the assessment lifecycle.

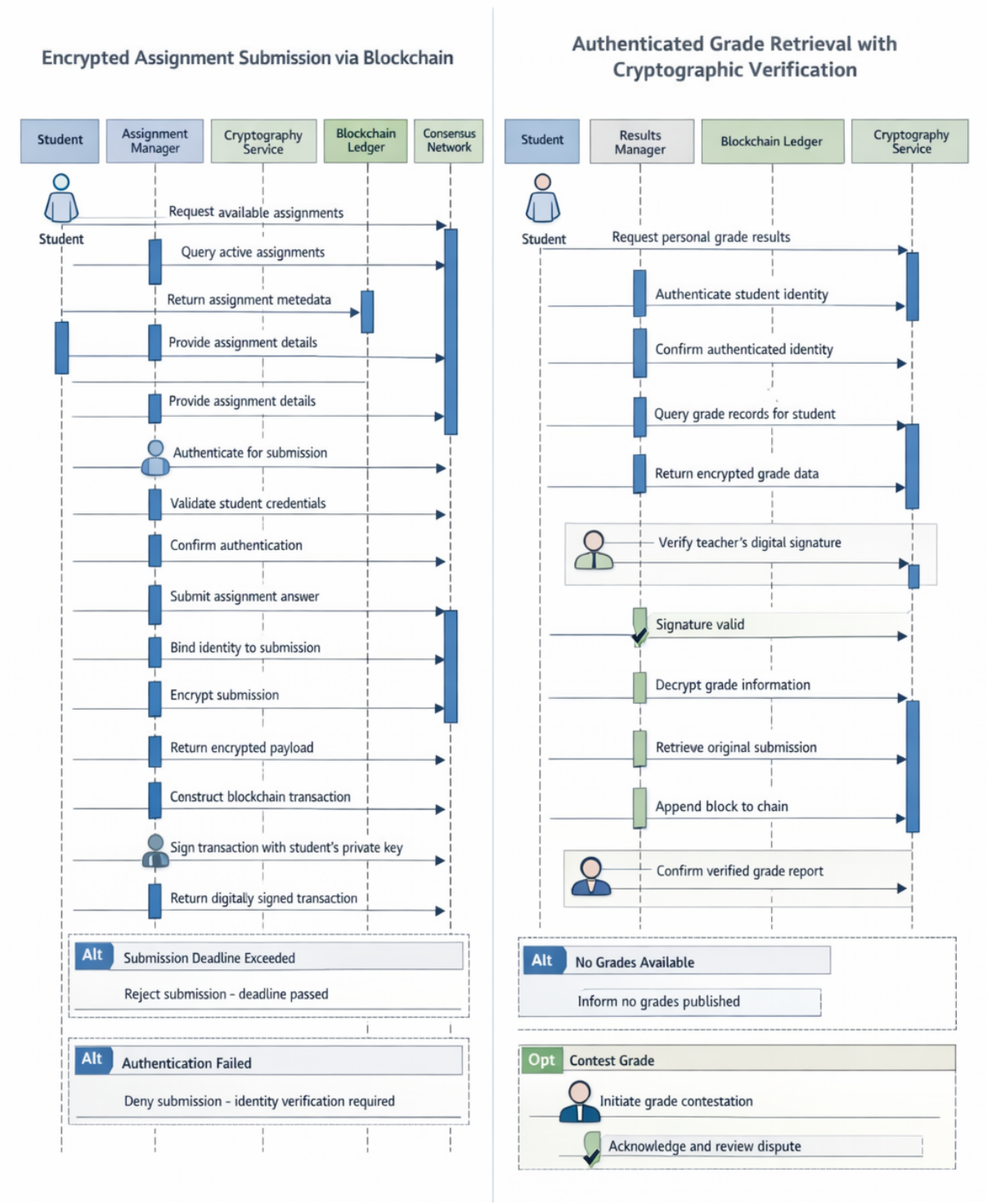


Figure 3: Sequence Diagrams for Assignment Submission and Grade Consultation

5.3 Data Model

The data model of the system is implemented using Django ORM, providing a structured and relational representation of all entities involved in the assessment management process. It is designed to support the core requirements of confidentiality, traceability, and

administrative flexibility while preparing for blockchain integration.

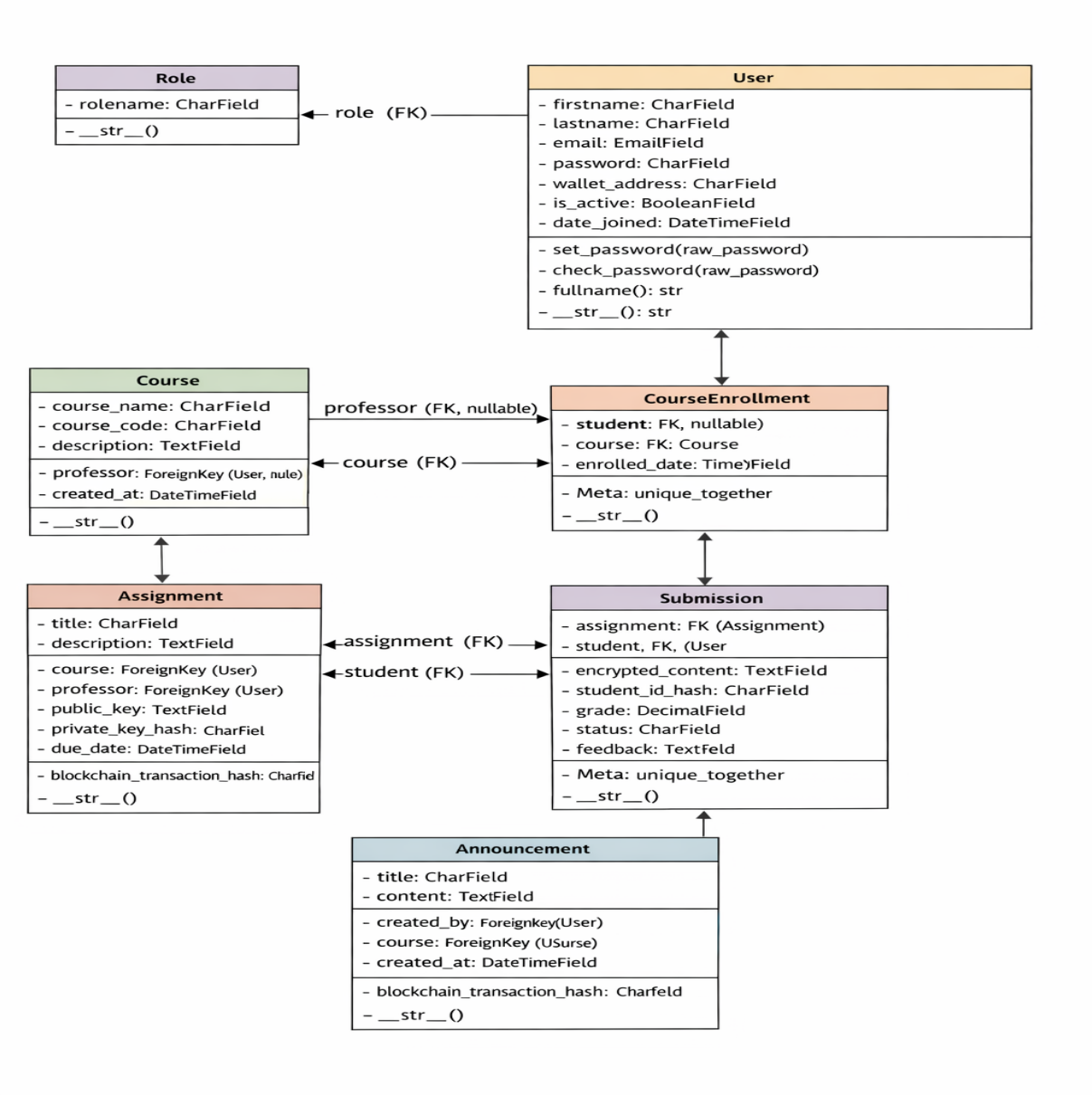


Figure 4: Entity-Relationship Diagram of the Data Model

The model is organized around five principal entities:

- **User Management**: The `User` entity, linked to a `Role` (student, professor, or admin), stores personal information, credentials, and account status. The optional `wallet_address` field anticipates future blockchain wallet association. Additional attributes such as `is_active` and `date_joined` facilitate administrative control and auditing.

- **Course Management**: The `Course` entity includes basic descriptive fields and an optional foreign key to a professor, allowing courses to be created before a teacher is assigned. The `CourseEnrollment` model manages many-to-many relationships between

students and courses, ensuring unique enrollments.

- **Assignment Management**: The `Assignment` entity is tied to a specific course and professor, storing the public key for encryption and a hash of the private key. It includes due dates and a field for the future blockchain transaction hash, enabling immutable recording of assignment publication.

- **Submission Management**: The `Submission` entity captures encrypted content and a hash combining student identity with the response to prevent undetectable plagiarism. It tracks submission status, grades, feedback, and associated blockchain transaction hashes, with validation logic preventing submissions after the deadline.

- **Announcement Management**: The `Announcement` entity supports both global and course-specific communications, authored by professors or administrators, with an optional link to a course and a field reserved for blockchain traceability.

This relational structure ensures data integrity, supports efficient querying, and aligns with the project's security objectives by separating sensitive encrypted content from metadata intended for blockchain recording.

5.4 Blockchain Transaction Flow

The blockchain transaction flow represents the mechanism by which key operations in the system achieve immutability, transparency, and traceability. Although the full blockchain network integration is planned for future development, the current implementation reserves dedicated fields in the data model to store transaction hashes, ensuring seamless compatibility with a permissioned blockchain (such as Hyperledger Fabric or Quorum) when deployed.

Each critical action assignment publication, encrypted submission, grade release, and announcement dissemination generates a transaction that is signed cryptographically and broadcast to the blockchain network. The resulting transaction hash is stored in the corresponding database record (e.g., `blockchain_transaction_hash` in `Assignment`, `Submission`, and `Announcement` models). This hash serves as an immutable reference, allowing any participant to verify the existence, timestamp, and integrity of the operation without accessing the encrypted content.

The typical flow proceeds as follows:

1. The initiating actor (teacher or student) performs the action through the application

interface.

2. The system constructs a transaction payload containing relevant metadata (e.g., assignment identifier, actor public address, timestamp, and hash of encrypted data).
3. The payload is digitally signed using the actor's private key to ensure non-repudiation.
4. The signed transaction is submitted to the blockchain network.
5. Validator nodes reach consensus and append the transaction to a new block.
6. Upon confirmation, the transaction hash and block reference are returned and persisted in the relational database.

This approach combines the confidentiality provided by RSA encryption (content remains private) with the transparency and auditability of blockchain (metadata and proofs are publicly verifiable). It effectively prevents retroactive modifications, guarantees submission timestamps, and provides cryptographic evidence for potential disputes, fully aligning with the project's objectives of fairness and trust in academic assessment processes.

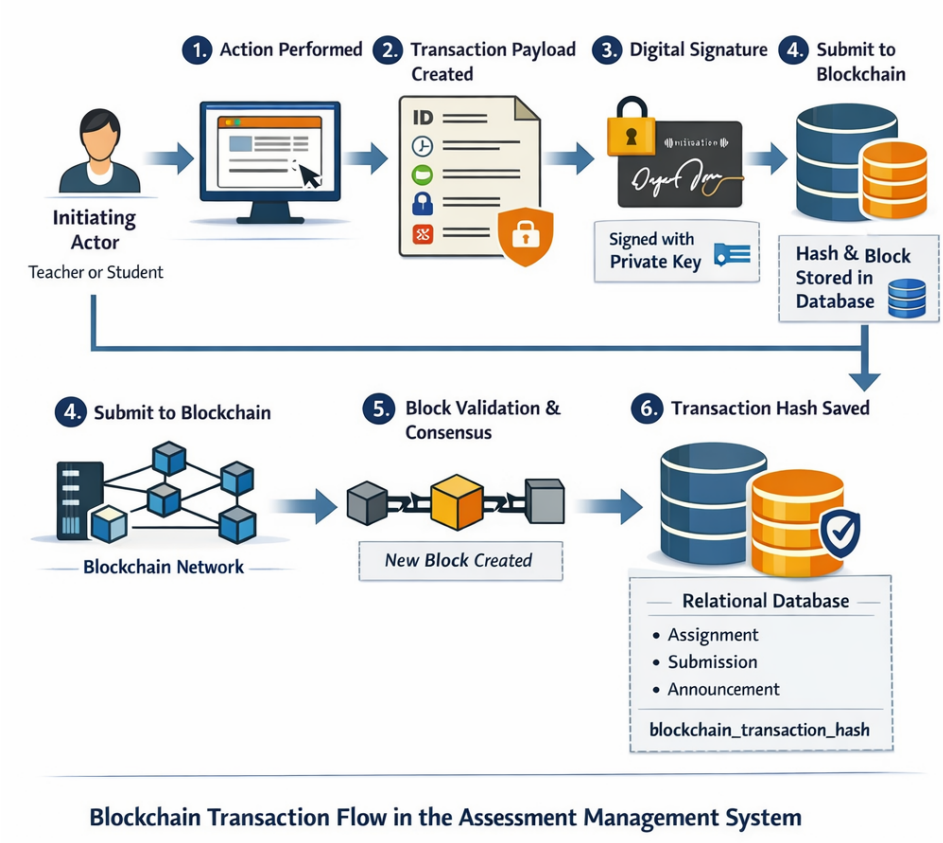


Figure 5: Blockchain Transaction Flow in the Assessment Management System

6 Blockchain Technical Implementation

6.1 Development Environment

The blockchain system is built using a well prepared development environment that combines several technologies. To make deployment easy and consistent on different machines, the whole blockchain infrastructure is containerized. Docker Desktop is used to manage the environment, allowing the Ethereum private network to run with three separate but connected nodes inside isolated containers.

Figure 6: Containerized blockchain environment using Docker

Python is the main programming language used to interact with the blockchain and connect it to the backend. The project requires Python version 3.8 or higher, along with specific libraries such as Web3.py to communicate with Ethereum nodes, eth-account to manage cryptographic keys and transactions, and solcx to compile Solidity smart contracts. These tools make it possible to deploy contracts, sign transactions, and interact with the blockchain directly from Python scripts, without using external tools manually.

Figure 7: Python environment for blockchain interaction using Web3.py

Smart contracts are written in Solidity using version 0.8.17. This version was chosen because it provides better security features and is fully compatible with the selected Geth version. The development process follows a clear workflow, contracts are written, compiled locally, and deployed automatically using Python scripts. This automation reduces human errors and ensures that deployments are reproducible during development.

Figure 8: Smart contract development and deployment workflow with solidity

Git / GitHub are used for version control to organize the project and track changes. Separate branches are maintained for the blockchain setup, smart contracts, and backend integration. Before merging changes into the main branch, each part is tested independently to make sure it works correctly. This approach helps keep the project stable and makes debugging and collaboration easier.

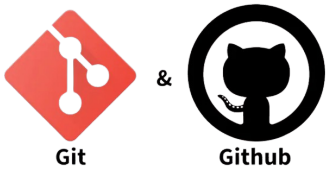


Figure 9: Version control and collaboration using Git and GitHub

The application platform is developed using a clear separation between frontend and backend. The frontend is built with React, which is used to create a dynamic and interactive user interface for students and teachers. The backend is developed with Django, which handles business logic, authentication, and communication with the blockchain. A REST API is implemented using Django REST Framework to allow secure and structured data exchange between the frontend and the backend. MySQL is used as the relational database to store users, assignments, submissions, keys, and application related data, ensuring data persistence and efficient querying.

(a) React (b) Django (c) REST API (d) MySQL

Figure 10: Application platform architecture and technologies

6.2 Private Blockchain Setup with Geth

The private Ethereum network is created using Geth, which is the official Ethereum client written in Go. This network is completely independent from the public Ethereum mainnet. Running a private blockchain gives full control over how the network behaves, including the consensus method, block creation speed, and gas rules. The network starts from a genesis block, which defines the initial state of the blockchain and the parameters that all nodes must follow to join the network.

The genesis file defines the core configuration of the private blockchain. A Chain ID of 1337 is used to uniquely identify the network. This value is important because it prevents transaction replay attacks, meaning transactions created for this private blockchain cannot be reused on public Ethereum or other networks. The genesis configuration also includes pre funded accounts, giving specific addresses enough Ether to perform transactions without worrying about gas costs.

Block generation is configured to happen every one second, which is much faster than the public Ethereum network where blocks are created approximately every fifteen seconds. This fast block time allows transactions to be confirmed almost instantly, which is especially useful in an educational platform where users expect quick feedback. This setup is possible because the network uses Proof of Authority, which does not require heavy computation like Proof of Work.

The gas limit is set to 134 million gas per block, which is significantly higher than standard public Ethereum limits. This high limit allows complex smart contract operations to execute without running out of gas. In addition, the gas price is set to zero, removing any economic cost for using the blockchain. This design choice makes the system more accessible and more suitable for educational use.

6.3 Node Configuration

6.3.1 Three-Node Architecture

The private blockchain network is composed of three nodes, each assigned a specific role. Two of these nodes act as sealer nodes, meaning they are authorized to create and sign new blocks. The sealer nodes run continuously and generate new blocks every one second, even when there are no transactions to process. This ensures that the blockchain progresses at

a constant rate and that block timestamps remain consistent.

The third node functions as an observer node. Its role is to stay synchronized with the sealer nodes and validate the blocks they produce. The observer maintains a full copy of the blockchain, which adds redundancy to the system and strengthens data immutability. When a new block is received, the observer node verifies that the block was signed by an authorized sealer and rejects any block coming from an unauthorized source. This verification process helps preserve the integrity of the network.

Each node is associated with its own Ethereum account generated in a deterministic manner. The sealer nodes use predefined private keys that generate known addresses, which are registered as authorized block signers in the genesis configuration. The observer node uses a different private key, giving it a unique identity while remaining fully synchronized with the network. This deterministic configuration simplifies node recreation and supports efficient recovery in case of failure.

6.3.2 Consensus Mechanism

The blockchain network uses a Proof of Authority consensus mechanism based on the Clique algorithm. Unlike Proof of Work, which requires heavy computational power to solve cryptographic puzzles, Proof of Authority relies on a predefined set of trusted addresses. Only these authorized addresses, called sealer nodes, are allowed to create and sign new blocks. The sealer nodes take turns producing blocks, which ensures a fair and balanced block creation process.

The Clique consensus mechanism also supports epoch checkpoints that occur every 30,000 blocks. During these checkpoints, the list of authorized sealers can be updated through a voting process. Although this feature is not actively used in the current configuration, it allows the network to be easily extended in the future by adding new sealer nodes. This provides flexibility while maintaining high performance and low overhead.

Block validation is based on cryptographic signature verification. When a new block is produced by a sealer node, the observer node receives it and checks that the block signature matches one of the authorized sealer addresses defined in the genesis configuration. If the signature is invalid or comes from an unauthorized node, the block is rejected. This mechanism ensures that only legitimate sealers can influence the blockchain state.

All nodes in the network stay synchronized using a peer to peer communication model.

When a sealer node creates a new block, it broadcasts the block to the other nodes in the network. The observer node validates the block and then propagates it if valid. This exchange of information ensures that all nodes maintain a consistent view of the blockchain, even in the presence of temporary disconnections or network delays.

6.4 Containerization with Docker

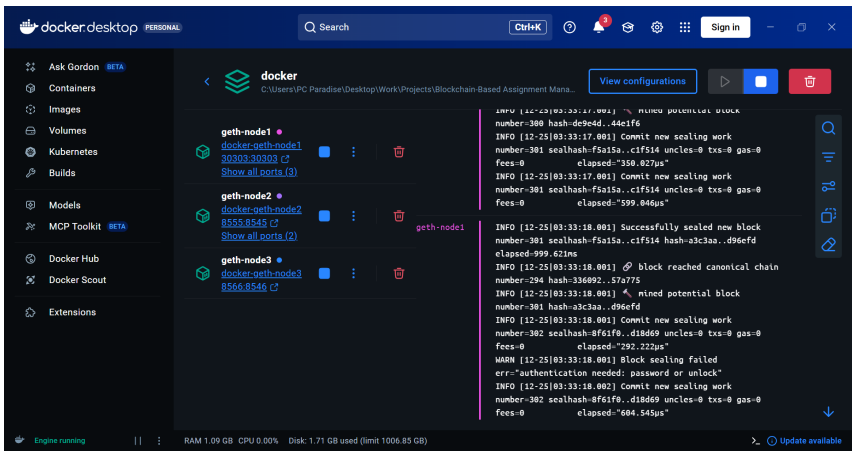


Figure 11: Containerized three-node Geth blockchain running in Docker

Docker is used to simplify the deployment of the multi node blockchain network and make it reproducible. Each Geth node runs inside its own Docker container with isolated resources, network configuration, and persistent storage. This isolation avoids conflicts between nodes while still allowing them to communicate securely through Docker's internal network.

Each container is built from a Docker image that defines the full environment required for a node to run. The image is based on the official Ethereum client and includes the genesis file, initialization scripts, and network settings. This approach ensures that all nodes run in identical environments, while still allowing specific configurations to be adjusted using environment variables.

Docker Compose is used to manage and coordinate all containers. A single Compose file defines the three nodes and their configurations, including network settings, exposed ports, and data volumes. Docker Compose also controls the startup order, making sure that observer nodes start only after the sealer nodes are available, which guarantees proper synchronization.

Blockchain data is stored using Docker volumes to ensure persistence. Each node has its own dedicated volume that stores the full blockchain data, account information, and transaction history. This allows the blockchain to continue from the same state even after containers are stopped or restarted. Using separate volumes also prevents data mixing between nodes and simplifies backup and recovery.

Port mapping is configured to allow access from the host machine. The sealer nodes expose port 8545 to enable HTTP RPC communication, allowing external applications to send transactions and query blockchain data. The observer node uses a different host port while keeping the same internal container port, enabling access to multiple nodes when needed.

All containers are connected through a private Docker bridge network. A dedicated subnet is defined with static IP addresses assigned to each node, ensuring predictable and stable communication between them. This setup simplifies peer discovery, debugging, and network management.

Startup procedures are fully automated using initialization scripts. When a container starts, the script checks if the blockchain is already initialized, imports the required accounts, and launches Geth with the correct parameters. Sealer nodes start in sealing mode, while the observer node starts in synchronization mode. This automation reduces manual setup, minimizes errors, and makes deployment faster and more reliable.

6.5 Smart Contract Deployment

Smart contract deployment is the process of turning Solidity source code into a program that runs on the blockchain. The process starts by compiling the smart contracts using the Solidity compiler. This step converts the contract code into EVM bytecode and generates the Application Binary Interface, which defines how the contract can be interacted with. Compilation is handled automatically using Python scripts, removing the need for manual compilation.

Once the contract is compiled, the deployment script connects to the blockchain using Web3 through the RPC endpoint of a sealer nodes. Since the network uses Proof of Authority, a specific middleware is added to ensure compatibility with Geth and the Clique consensus mechanism. This middleware allows transactions to be formatted correctly and accepted by the network.

To deploy a contract, the deployer account must be accessed using its private key. The private key is securely decrypted and loaded, which then signs the deployment transaction. This transaction includes the compiled contract bytecode, any constructor parameters, and the required gas settings. The signed transaction is sent to the sealer nodes, which validates it and includes it in the next block.

Before sending the transaction, the system estimates the amount of gas required for deployment. This estimation is based on the size of the contract and its initialization logic. A safety margin is added to avoid deployment failures due to insufficient gas. Since the gas price is set to zero, this step is only used to ensure successful execution rather than cost calculation.

After the transaction is broadcast, the sealer node executes the contract creation. If the execution is successful, the blockchain assigns a unique and permanent address to the smart contract. This address is generated deterministically based on the deployer's address and the transaction nonce.

Finally, the deployment process verifies that the contract has been successfully deployed by waiting for the transaction receipt. The contract address is retrieved and stored in configuration files so that the backend can interact with the smart contract in future operations. This completes the deployment process and makes the contract available for use within the platform.

6.6 Cryptographic Key Management

The security of the system depends mainly on how cryptographic keys are generated, stored, and used. Each user has cryptographic keys that identify them on the blockchain and protect sensitive data. Two types of keys are used, Ethereum keys for blockchain identity and RSA keys for securing assignment submissions.

For blockchain operations, the system creates Ethereum accounts based on elliptic curve cryptography. Each account has a private key and a public key. The private key is randomly generated and must stay secret, while the public key is derived from it. From the public key, an Ethereum address is generated, which represents the user's identity on the blockchain.

To prevent unauthorized access, private keys are never stored in plain text. Before being saved in the database, they are encrypted using Django's built-in cryptographic tools.

The encryption is based on a master key stored in the environment configuration. This means that even if the database is compromised, the private keys remain protected and can only be decrypted by the system when needed.
In addition to Ethereum keys, RSA key pairs are used to protect assignment submissions. When a teacher creates an assignment, the system generates a 2048-bit RSA key pair. The public key is published on the blockchain so students can use it, while the private key is stored in the database in encrypted form and is accessible only to the teacher.

6.7 Student Submission Encryption

Student submissions are encrypted to ensure confidentiality during submission, storage, and grading. When a teacher creates an assignment, the system generates an RSA key pair. The public key is stored on the blockchain and the database so students can access it, while the private key is encrypted and saved securely in the database.
The encryption process happens directly in the student's browser. Students write their submission in the interface, and when they submit, the application retrieves the assignment's public key from the blockchain. This public key is used to encrypt the submission content before it is sent to the server, meaning the data is already protected during transmission.
Once encrypted, the submission is sent to the Django backend as a base64-encoded string. The backend stores the encrypted data in the database and also records it on the blockchain using a smart contract. This approach combines fast access from the database with the immutability and traceability provided by the blockchain. At no point can administrators or other students read the original content.
Decryption is only performed when the teacher reviews the submissions. The system retrieves the encrypted submission and the teacher's encrypted private key. After decrypting the private key using the system's master key, the submission is decrypted and displayed to the teacher for evaluation. This ensures that only the assignment creator can access the original content.
In addition to encryption, the system verifies submission integrity using cryptographic hashing. A SHA-256 hash of the submission is computed before encryption and stored on the blockchain. When the teacher decrypts the submission, the hash is recomputed and compared with the stored value. This allows detection of any modification attempts and

provides proof that the submission has not been altered.

7 Platform Design and Implementation

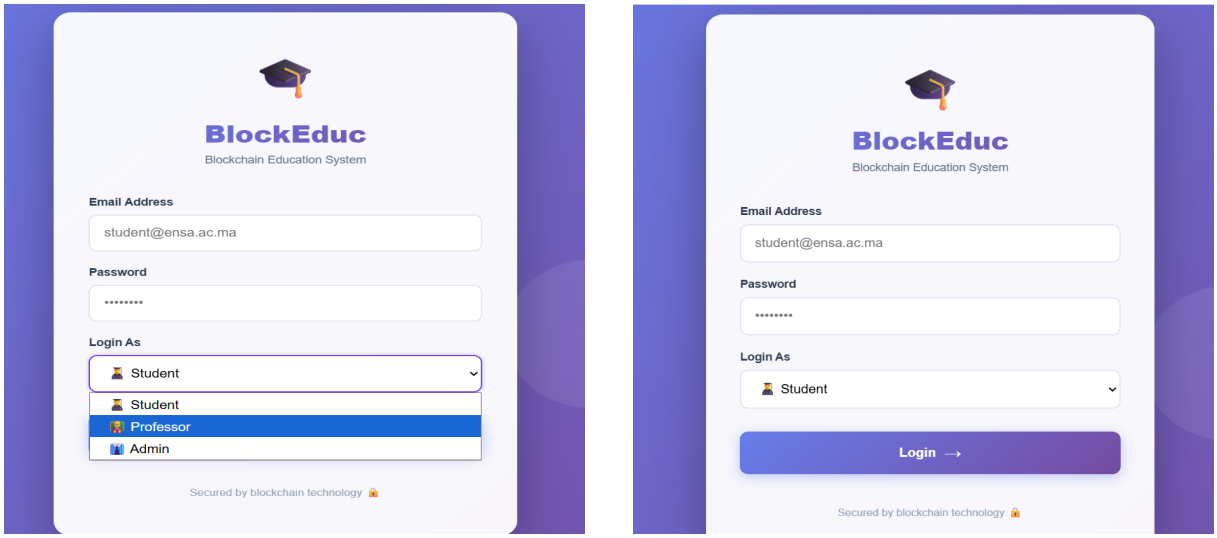
7.1 Frontend Development with React

The frontend of the platform was developed using React, a modern and widely adopted JavaScript library specifically designed for building dynamic, component-based, and highly responsive user interfaces. This technology choice facilitated the creation of a sophisticated single page application characterized by a clean modular architecture, efficient state management through React hooks, and seamless asynchronous communication with the Django REST framework backend. The implementation adheres to industry best practices, ensuring code reusability, maintainability, and optimal performance. The overall design places strong emphasis on clarity, accessibility, and professional aesthetics, carefully presenting complex cryptographic operations and blockchain integration features in an intuitive and user-friendly manner, making them comprehensible even to non technical users such as students and professors.

7.1.1 Main Interfaces

The main interfaces consist of role specific dashboards that serve as the primary workspaces for each type of user, complemented by a unified login screen that acts as the secure entry point to the system.

The login interface represents the common entry point for all users of the platform. It includes clearly labeled input fields for email address, password, and an explicit role selection dropdown (student, professor, or administrator). This deliberate design ensures immediate enforcement of role based access control, proper routing to the corresponding dashboard upon successful authentication, and informative feedback in case of invalid credentials or missing information.



(a) Role Selection Step (b) Authentication Form

Figure 12: Login Interface with Role Selection

The Professor Dashboard provides a comprehensive and well-organized environment dedicated to managing courses and the entire assessment lifecycle. It features intuitive tab navigation for accessing assigned courses, creating and publishing assignments, reviewing pending student submissions, and posting announcements. The dashboard presents detailed overviews of active courses, submission statistics, deadline reminders, and participation rates, enabling professors to monitor student engagement effectively and take timely actions when necessary.

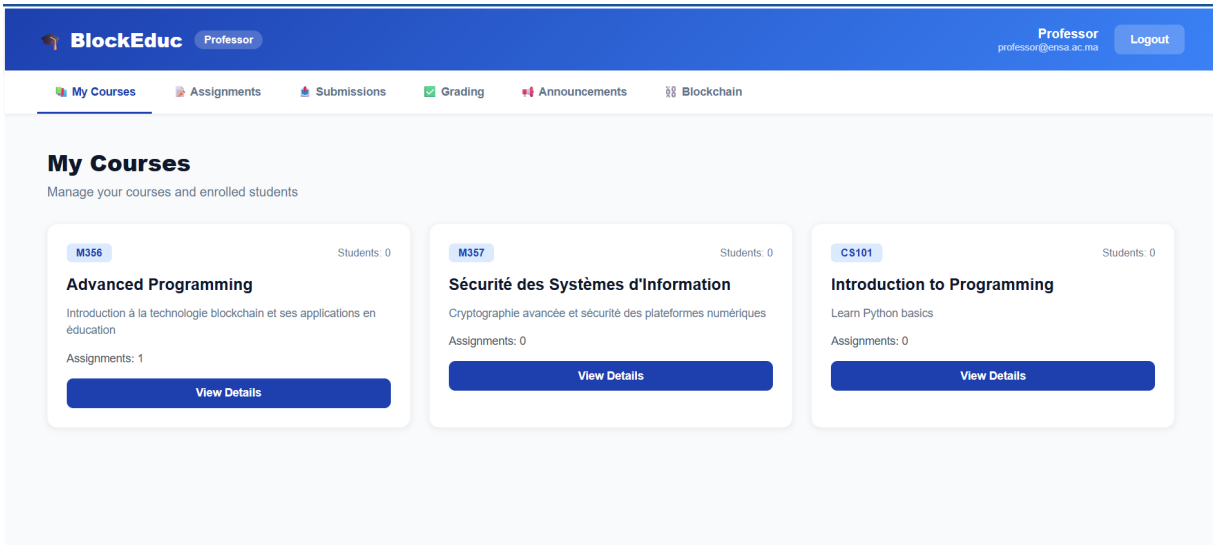


Figure 13: Professor Dashboard – Main Interface with Tab Navigation and Detailed Course Overview

The Student Dashboard serves as a personalized learning hub designed to support stu-

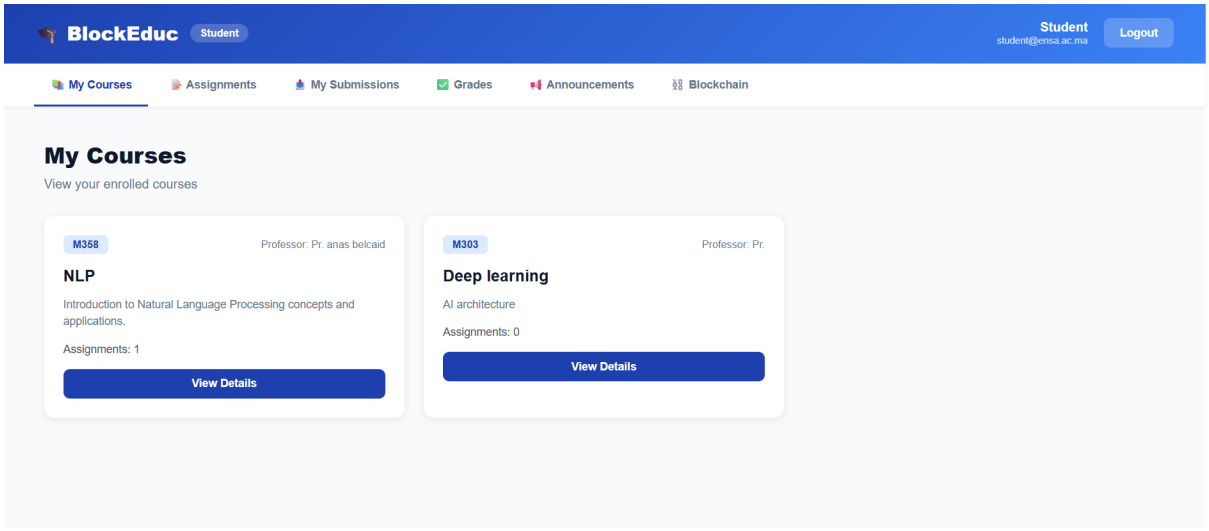


Figure 14: Student Dashboard – Main Interface Overview with Assignment Tracking and Progress Indicators

dents in staying organized and informed throughout their academic journey. It includes dedicated tabs for viewing enrolled courses, browsing active assignments with prominent deadline indicators, consulting grades and feedback, and accessing both global and course-specific announcements. The layout prioritizes clear presentation of upcoming tasks, submission status, and progress tracking to promote proactive participation.

The Administrator Dashboard functions as the central governance console, offering powerful administrative tools for platform management and maintenance. It provides tabbed access to user administration, course creation and configuration, student enrollment operations (individual and bulk), and comprehensive system monitoring, complete with summary statistics, search functionality, and operational insights for efficient oversight.

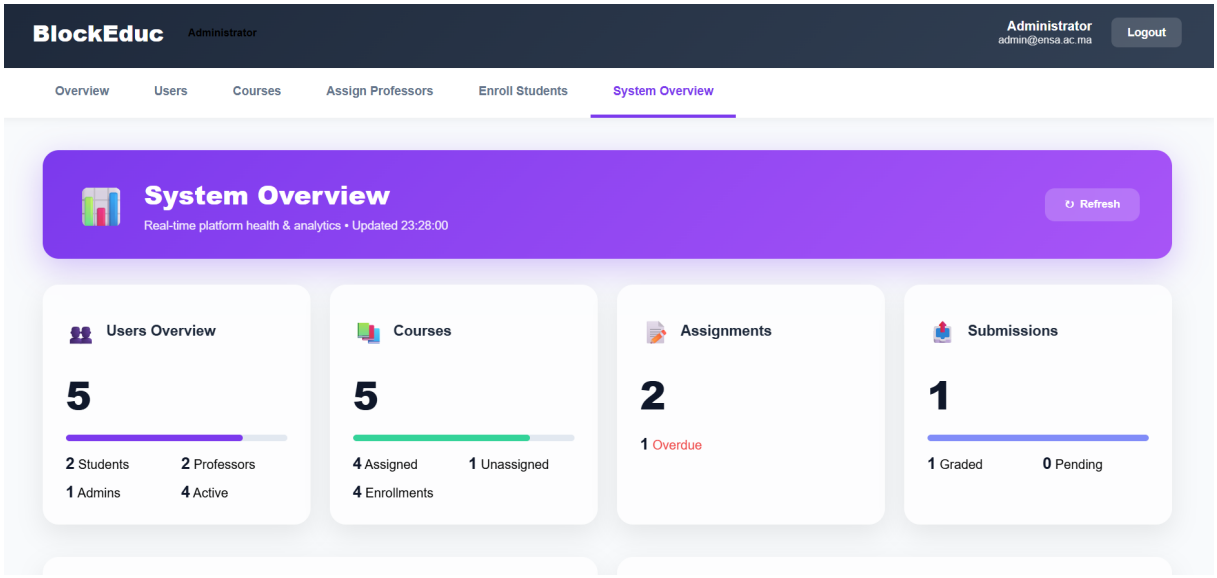
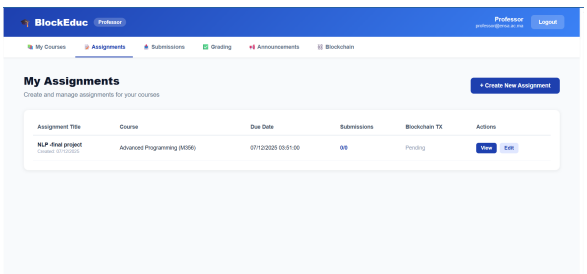


Figure 15: Administrator Dashboard – Main Governance Interface with Comprehensive Management Tools

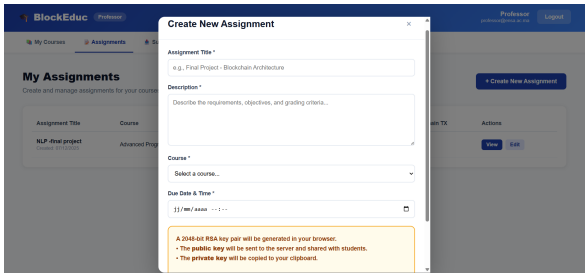
7.1.2 User Interaction

User interaction focuses on the detailed functionalities and dynamic elements that enable specific actions within the main interfaces, delivered through carefully crafted modals and interactive components that guide users step by step.

For professors, assignment creation is handled through a dedicated modal that captures essential information including the assignment title, detailed description, due date, and automatically generates an RSA key pair for encryption. The process concludes with the secure publication of the assignment along with the corresponding public key, ensuring students receive all necessary elements for confidential submission.



(a) Assignment Creation Form

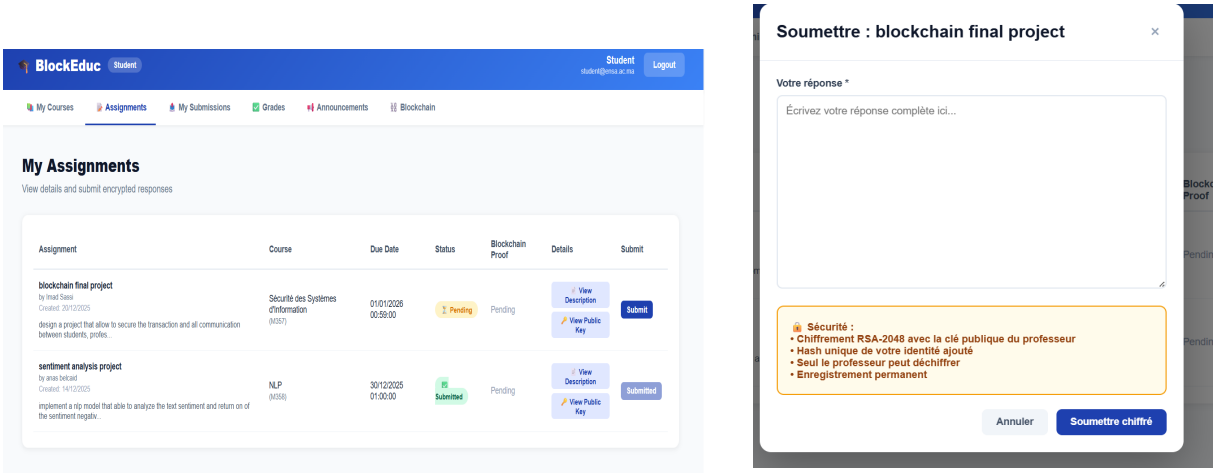


(b) Key Generation and Publication Confirmation

Figure 16: Professor Interaction – Assignment Creation Modal with Automatic Key Generation

Students submit assignments using a secure modal that supports detailed response entry,

performs client-side hybrid encryption utilizing the professor's public key, and provides real-time progress feedback. Upon successful submission, the interface prominently displays the blockchain transaction hash as immutable proof of record.



(a) Assignment Details and Public Key Display

(b) Response Entry and Submission Confirmation

Figure 17: Student Interaction – Encrypted Assignment Submission Modal with Blockchain Confirmation

Professors grade submissions via a secure decryption modal that requires temporary private key input (never stored), followed by a comprehensive grading form allowing score assignment and detailed feedback composition, with final blockchain recording upon submission.

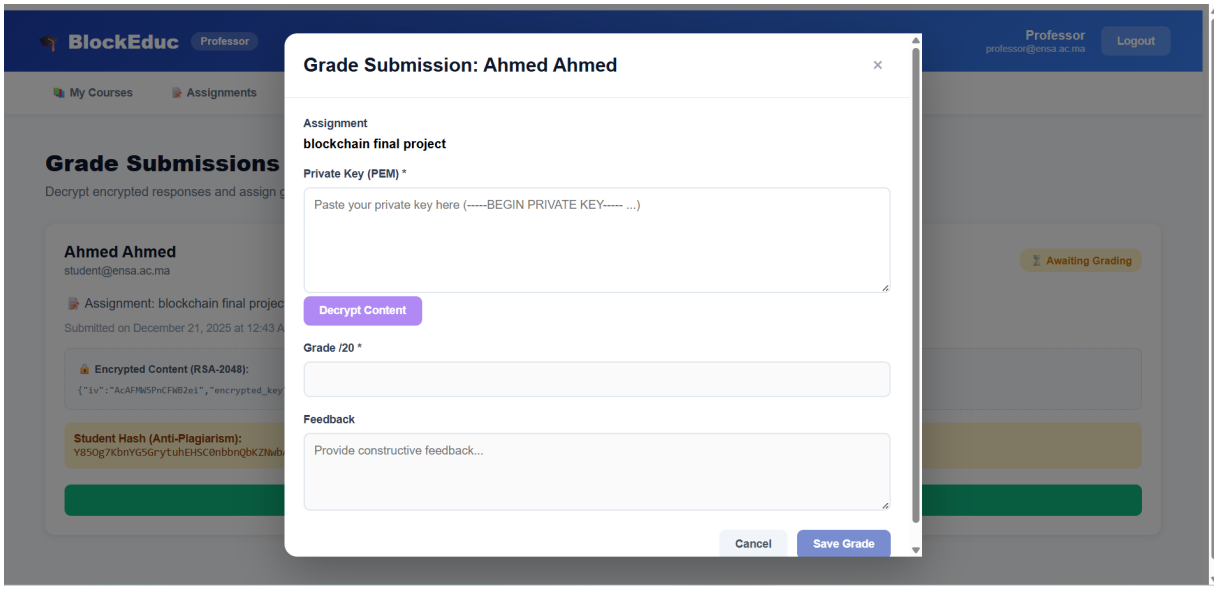


Figure 18: Professor Interaction – Submission Decryption and Grading Interface

Students view grades through expandable panels that reveal comprehensive feedback, final

scores, and associated blockchain verification details, providing complete transparency and cryptographic proof of authenticity.

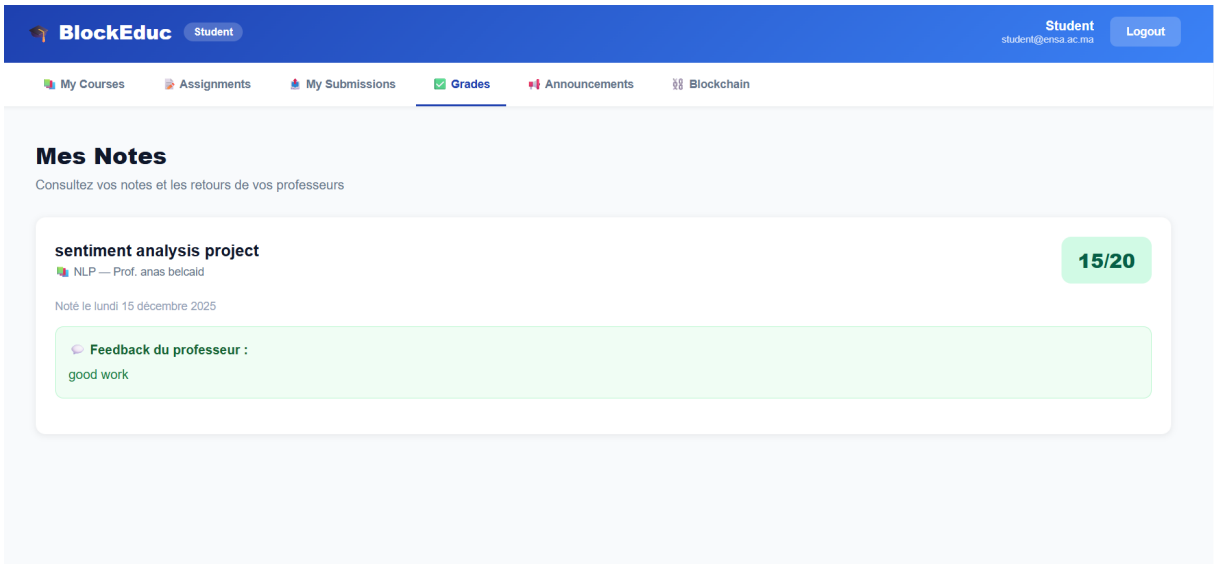


Figure 19: Student Interaction – Grade Consultation with Audit Trail and Blockchain Verification

At the same, This tab provides a real-time overview of blockchain activity within the platform, including network status, block information, smart contract statistics, and a transparent history of submissions and grades recorded on the blockchain. It ensures traceability, integrity, and trust in academic processes.

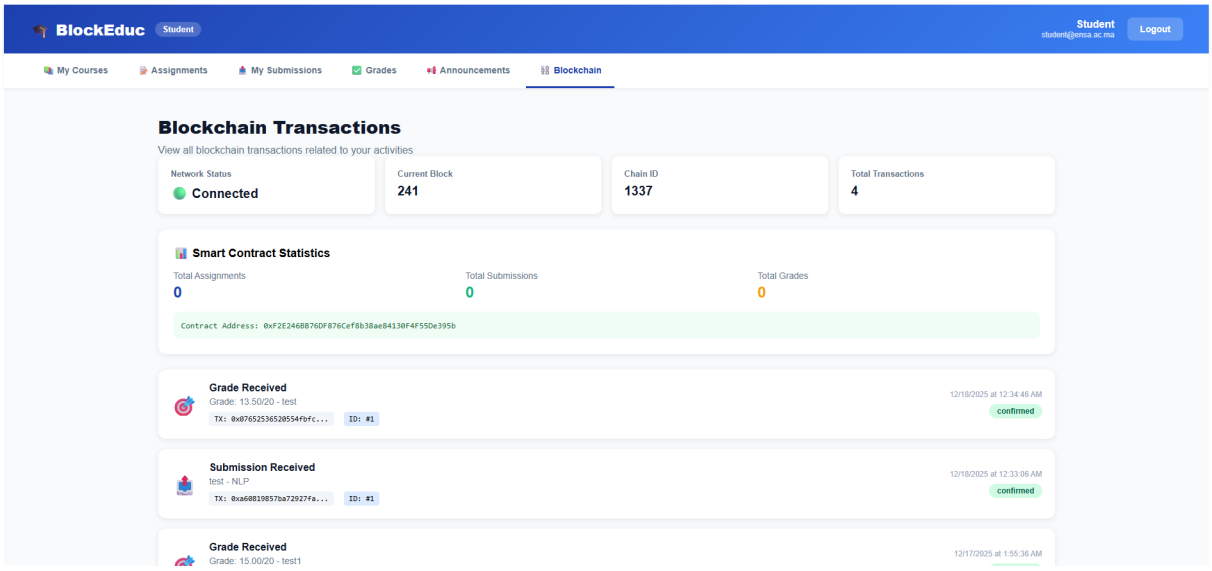


Figure 20: Blockchain Statistics and Transaction History.

Administrators manage users through intuitive modals supporting full account lifecycle

operations creation with role assignment, editing of personal information, activation/deactivation controls, and secure deletion—ensuring precise user administration.

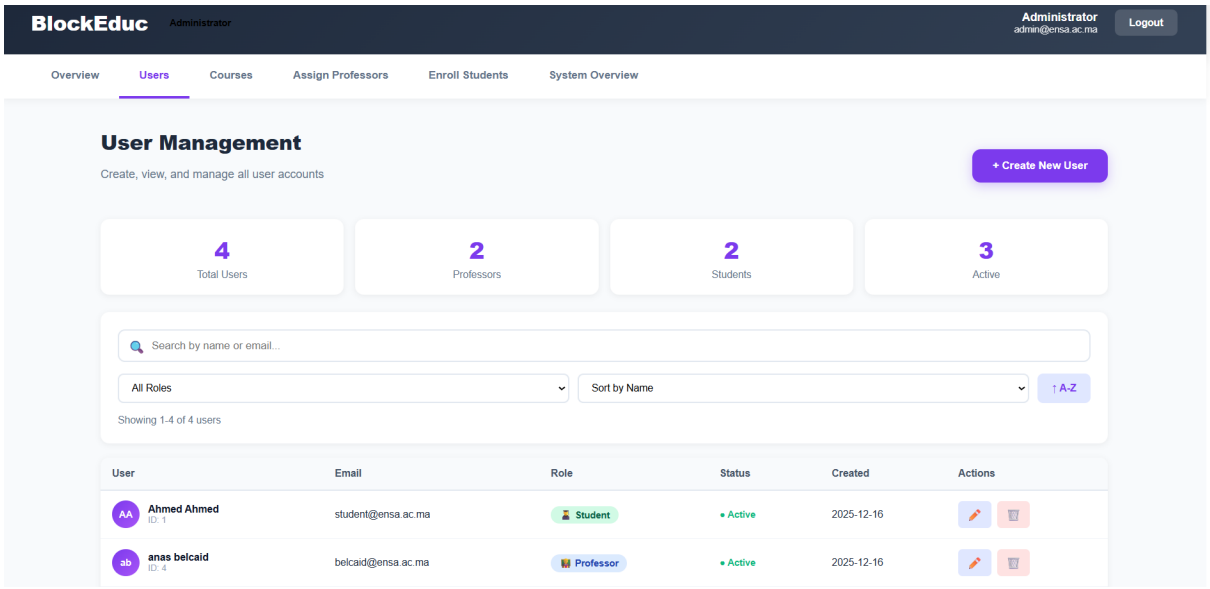


Figure 21: Administrator Interaction – User Account Management Modal

Administrators perform bulk enrollment operations using searchable selection interfaces and multi-step confirmation dialogs, streamlining the process of assigning multiple students to courses efficiently and accurately.

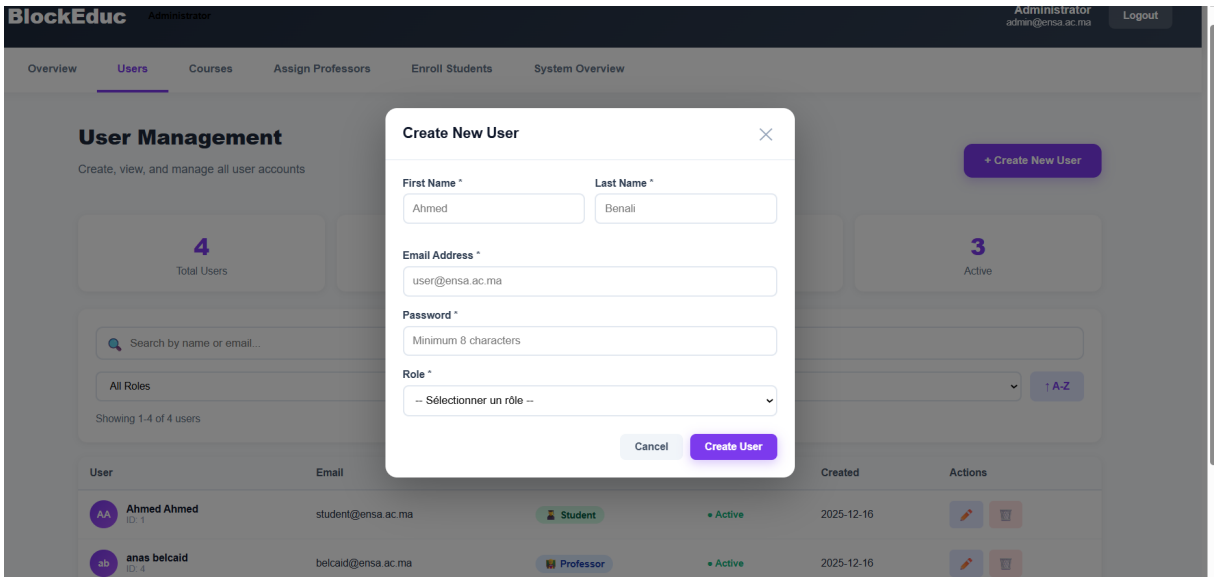


Figure 22: Administrator Interaction – User Account Creation

Administrators also perform course management operations through dedicated interfaces that allow the creation and configuration of new courses. This includes defining the unique course code, full name, detailed description, and optionally assigning a professor

immediately or leaving the position open for later allocation. The interface supports efficient course setup, ensuring that all necessary metadata is captured while maintaining administrative flexibility in staffing decisions.

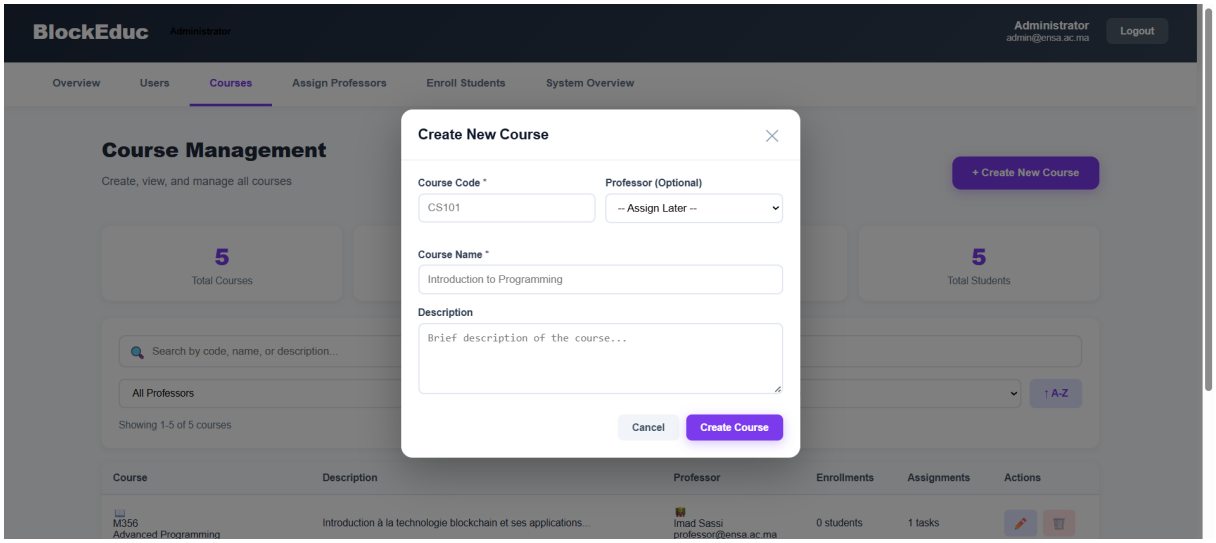


Figure 23: Administrator Interaction – Course Creation and Management Interface

These carefully crafted interaction patterns ensure secure, guided, and transparent workflows throughout the platform. By making advanced security features visible and understandable such as encryption status indicators, key management processes, and blockchain transaction confirmations the system builds strong user confidence while maintaining operational simplicity and efficiency across all roles.

7.2 Backend Development with Django

The backend of the platform was implemented using Django, a high-level Python web framework known for its robustness, security features, and rapid development capabilities. This choice allowed for the creation of a scalable server-side application with built-in support for database management, user authentication, and RESTful API development. The backend handles all business logic, data persistence, and integration with the blockchain layer, ensuring secure and efficient operations across the system. Key components include model definitions for data structure, serializers for API data handling, views for request processing, and signals for automated workflows such as blockchain profile creation upon user registration.

7.2.1 REST API

The REST API serves as the core communication layer between the frontend and backend, providing endpoints for all major functionalities. It is structured using Django REST Framework, with class-based views for clarity and maintainability. Authentication is managed through token-based mechanisms, ensuring that only authorized users can access role-specific endpoints.

The API includes endpoints for user authentication (login), course management (listing and creation), assignment handling (creation, listing, submission), grading operations, and announcement management. For example, the student submission endpoint validates deadlines and enrollment status before creating records, while professor endpoints allow secure grade assignment. Serializers validate and transform data, preventing invalid inputs and ensuring data integrity.

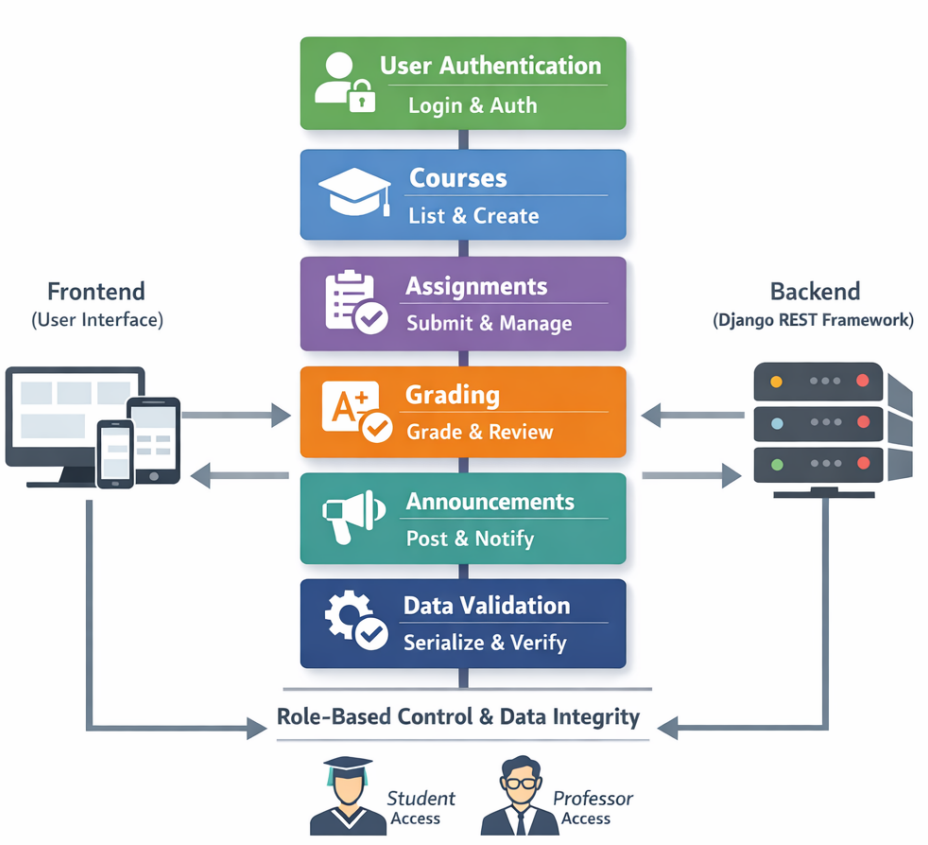


Figure 24: Overview of REST API Endpoints and Data Flow

7.2.2 Communication with the Blockchain via Web3

Communication with the blockchain is facilitated through Web3.py, a Python library for interacting with Ethereum-compatible networks. The backend includes a dedicated interaction layer (interact.py) that handles account creation, transaction signing, and contract calls using a local Geth node for a permissioned blockchain setup.

Upon creating professor or student accounts, Django signals automatically generate blockchain wallets, encrypt private keys (using Fernet symmetric encryption in utils.py), and fund accounts from predefined sources. Smart contract interactions such as recording assignment creation or submission hashes are performed asynchronously to maintain responsiveness. This integration ensures immutability and traceability without compromising the platform's performance.

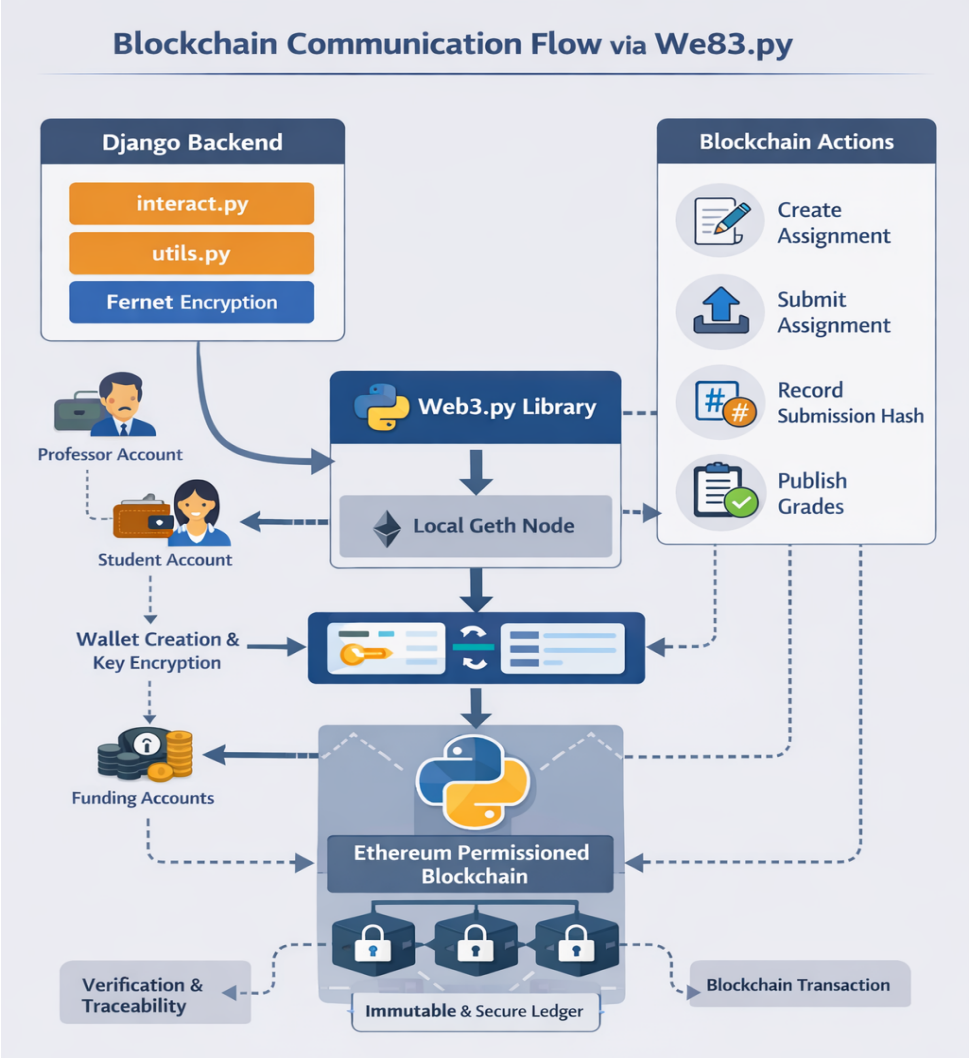


Figure 25: Blockchain Communication Flow via Web3.py

7.3 Role Management (Teacher / Student)

Role management is implemented through a dedicated Role model and foreign key relationships in the User model, enabling fine-grained access control. Teachers (professors) have privileges to create assignments, grade submissions, and post announcements, while students can submit encrypted responses and view their grades. The system enforces these roles at the API level using custom permissions and view restrictions, preventing unauthorized actions. This structure ensures that each user interacts only with features relevant to their role, maintaining security and workflow efficiency.

7.4 Security and Access Management

Security is paramount in the platform, with multiple layers of protection implemented throughout the backend. Passwords are hashed using Django's built-in mechanisms, and sensitive data like private keys are encrypted before storage. Access management uses role-based authorization in views, wwith IsAuthenticated restrictions enforced on all API endpoints to ensure that only logged-in users can access protected resources, further refined by custom permission checks based on user roles. API endpoints include input validation via serializers to prevent injection attacks, and rate limiting can be added for additional protection. The blockchain integration adds an extra layer of immutability, ensuring that critical actions (e.g., submissions) cannot be altered once recorded.

8 Security, Privacy and Compliance

The system protects sensitive data using multiple layers of encryption. Assignment submissions are encrypted using RSA-OAEP 2048-bit in the student's browser before transmission. The encrypted content is stored in the MySQL database, while cryptographic hashes are recorded on the blockchain for immutability verification. This ensures that only the authorized teacher can access the submitted work. Assignment private keys are never stored in the system. When creating an assignment, teachers generate an RSA-2048 key pair in their browser. The public key is saved to the database and shared with students for encryption, while the private key is copied to the teacher's clipboard with a critical warning to save it securely in their local environment. Only a SHA-256 hash

of the private key is stored for verification purposes. This zero-knowledge architecture provides maximum privacy but places full responsibility on teachers for key management. If a teacher loses their private key, all submissions for that assignment become permanently unreadable with no recovery mechanism. Blockchain wallet private keys require different handling. Both student and teacher Ethereum private keys are encrypted using the Fernet library (AES-128 in CBC mode with HMAC authentication) before storage in the database. The encryption uses a master key stored in environment variables (BLOCKCHAIN_ENCRYPTION_KEY). Keys are decrypted only temporarily in memory when signing transactions and are never stored in plaintext. While this protects against database theft, compromise of the master key would expose all wallet private keys, creating a single point of failure. User passwords follow industry-standard security practices. The system uses Django's PBKDF2-SHA256 hashing with 600,000 iterations, producing irreversible hashes. Even with database access, attackers cannot recover original passwords, ensuring credential security.

Table 1: Summary of Sensitive Data Protection Mechanisms

Sensitive Data	**Encryption Method**	**Storage**	**Security Level**
User Passwords	PBKDF2 with SHA-256 (hashed)	Not applicable (one-way)	Excellent
Assignment Private Keys	Not stored	Professor local environment only	Excellent
Submission Content	RSA-OAEP 2048-bit	Public key in DB, private key held by professor	Excellent
Wallet Private Keys	Fernet (AES-128 CBC)	`settings.BLOCKCHAIN_ENCRYPTION_KEY`	Medium
Student Submission Hash	SHA-256	Stored as hash value	Good

Table 2: Security Single Points of Failure

Failure Point	**Impact**	**Probability**	**Mitigation**
Professor loses assignment private key	All submissions for that assignment become permanently unreadable	Medium	User education, key backup reminders, no technical recovery
`BLOCKCHAIN_ENCRYPTION_KEY` compromised	All wallet private keys exposed, funds at risk	Low	Secure environment variables, restricted server access
Database breach without env file	Encrypted wallet keys safe, but public keys and encrypted submissions exposed (Privacy violation, no financial loss)	Medium	Database encryption at rest, access control
Blockchain network unavailable	Cannot record new transactions, offline mode required	Low	Queue transactions, retry mechanism, fallback to DB-only
Smart contract bug or exploit	Invalid data on blockchain, potential fund loss	Low	Thorough testing, audit, contract upgrades

8.1 Actor Anonymity

The system uses blockchain addresses to provide pseudonymous identities rather than full anonymity. Each user is represented on the blockchain by an Ethereum address that does not directly reveal personal information. This address is generated from the user's public key and serves as a persistent blockchain identity.

At the same time, the system preserves educational accountability. A link between blockchain addresses and user profiles is maintained in the relational database. This allows proper management of courses, assignments, and grades. Only authorized administrators can access this mapping, combining privacy on the blockchain with institutional control.

Student identity information is not stored directly on the blockchain. Instead, it is included inside the encrypted submission content. As a result, external observers cannot identify the author of a submission by analyzing blockchain data. The student's identity

becomes visible only after the teacher decrypts the submission.

From an external point of view, blockchain transactions show only interactions between addresses. Observers can verify that transactions exist and are valid, but they cannot access personal identities or submission content. This approach provides transparency for auditing while preserving user privacy.

8.2 Immutability and Traceability

Blockchain immutability guarantees that academic records cannot be altered after being recorded. Once an assignment is created, a submission is stored, or a grade is published, this information becomes permanent. Any attempt to modify past data would invalidate the blockchain, making tampering practically impossible.

This property offers important benefits in an educational context. Students cannot change their submissions after the deadline, and teachers cannot secretly modify grades once published. All changes are be recorded as new transactions, creating a visible and trustworthy history.

Traceability is ensured through detailed transaction records. Each transaction contains timestamps, participant addresses, and cryptographic hashes of related data. This allows students to verify submission times, teachers to prove grading actions, and administrators to review complete assignment histories.

Smart contracts also emit events when key actions occur, such as assignment creation, submission reception, and grade publication. These events make it easier to monitor system activity, generate reports, and provide real-time notifications without scanning the entire blockchain.

8.3 Anti-Cheating Protection

The encryption mechanism helps reduce cheating and plagiarism. Even if two students submit identical content, the encryption process produces different encrypted outputs due to random padding. This prevents students from comparing encrypted submissions or coordinating during submission.

Blockchain timestamps prevent manipulation of submission times. Each submission is recorded with a reliable timestamp, providing proof that it was submitted before or after the deadline. This removes disputes related to late submissions or claims of system errors.

To further detect plagiarism, the system uses cryptographic hashing. A SHA-256 hash of each submission is stored on the blockchain. Teachers can later compare hashes to identify identical or highly similar submissions, even after decryption.
Grade integrity is protected through smart contract access control. Only the teacher who created an assignment is allowed to publish grades for it. This rule is enforced at the blockchain level, preventing unauthorized or accidental grade modifications.

8.4 Privacy and Personal Data Protection

The system is designed to comply with data protection regulations such as GDPR and Moroccan Law 09-08. Personal data is stored only in the traditional database, where it can be modified or deleted if required. The blockchain stores only encrypted content, addresses, and cryptographic hashes, without exposing personal information.
This separation allows users to exercise their privacy rights. Personal data can be removed from the database while blockchain records remain intact. Since blockchain data does not directly identify users, immutability is preserved without violating privacy rules.
The system follows data minimization principles. Only essential information is stored on the blockchain, while descriptive details and personal data remain in the database under strict access control. This reduces unnecessary exposure of sensitive information.
Finally, consent is handled at the application level. Students and teachers agree to the system's terms of use, including the permanent nature of blockchain storage and the use of encryption. This informed consent provides a clear legal basis for data processing within the platform.

9 Results and Testing

9.1 Objectives Validation

The developed system successfully meets all the defined technical objectives. The three main functional components work as expected. Teachers can create and publish assignments with embedded public keys, students can submit their work in encrypted form, and grades are published as immutable records on the blockchain. In addition, the announcement module allows teachers to share course-related messages stored on the blockchain.

The assignment workflow has been fully validated. When a teacher creates an assignment, an RSA key pair is generated, and the public key is stored in the database for student access. Students retrieve this key, encrypt their submissions on the client side, and send the encrypted data to the server, where it is stored in the database. At the same time, a cryptographic hash of the submission is recorded on the blockchain to ensure immutability. Teachers use their privately held private keys, which are never stored in the system, to decrypt submissions, evaluate them, and publish grades through smart contract transactions. This process ensures confidentiality while preserving transparency for timestamps and transaction records.

The encryption implementation satisfies the required security specifications. RSA encryption with 2048-bit keys guarantees the confidentiality of submissions, even though blockchain transactions are publicly visible. Due to encryption padding, each student produces a unique encrypted output, which prevents plagiarism through submission comparison. Teacher identity verification is also enforced, as only the holder of the correct private key can decrypt student submissions.

Automation significantly reduces administrative effort. Smart contracts automatically register assignments, validate submission timestamps, and enforce access control without manual intervention. The entire assignment lifecycle, from creation to grading, is handled by the system without requiring continuous administrative supervision.

All critical actions are automatically signed through blockchain transactions. Assignment creation, submission, and grade publication each generate cryptographic signatures that prove transaction authenticity. These signatures provide strong non-repudiation guarantees, as their validity is ensured by blockchain consensus rather than a central authority.

The system provides advanced verification capabilities. Any participant can verify the existence of assignments, submission timestamps, and published grades directly on the blockchain. This verification does not rely on trust in a central system, as cryptographic proofs validate all recorded actions. Students can independently verify the authenticity of their grades by checking blockchain records.

Transparency is carefully balanced with privacy. While transaction existence and timing are publicly verifiable on the blockchain, submission content and student identities remain encrypted. This design ensures accountability without exposing sensitive information.

Immutability guarantees prevent data manipulation. Assignments cannot be modified

after creation, submissions remain unchanged once recorded, and published grades become permanent records. This permanence removes disputes related to data alteration and provides reliable proof of academic actions.

Finally, the system includes strong security protections against common threats. Encryption prevents unauthorized access, access control mechanisms prevent impersonation, and blockchain consensus prevents transaction forgery. The system is resistant to attacks such as man-in-the-middle interception, unauthorized grade modification, and submission time manipulation.

9.2 Performance Analysis

The system shows acceptable performance for an educational environment. Assignment creation usually takes between two and three seconds from submission to confirmation on the blockchain. This delay is reasonable since assignment creation is not time-critical, and teachers typically prepare assignments well before deadlines. The one-second block time helps ensure fast confirmation.

The system also handles multiple student submissions efficiently. During deadline periods, several students can submit their work at the same time without noticeable performance issues. Tests with simulated simultaneous submissions from thirty students resulted in average confirmation times of less than five seconds. This level of performance is suitable for typical classroom sizes and does not require students to submit at different times.

Query performance is optimized through the use of both a relational database and the blockchain. The database manages frequent and complex queries such as listing assignments, checking submission status, and retrieving grades, all of which complete in milliseconds. The blockchain is used only when verification is needed, adding minimal overhead for audits or dispute resolution.

Storage usage remains practical for institutional use. Encrypted submissions require more storage than plain text due to encryption overhead, but the total size remains manageable. For one academic term, the database uses around five hundred megabytes per hundred students for encrypted submissions, while the blockchain stores less than ten megabytes of data for transaction hashes and metadata. This storage growth is linear and suitable for long-term deployment.

Network bandwidth usage stays within normal limits. Internal communication within

the private blockchain network does not overload institutional networks. Communication between the Django backend and blockchain nodes generates low traffic, with temporary increases during peak submission times that remain well within acceptable network capacity.

9.3 System Limitations

Despite its strengths, the system has several limitations. The private blockchain relies on manual node management and does not support automatic scaling. Adding new nodes requires manual configuration and peer connection setup. This limits the ability to quickly scale the network when demand increases, although a three-node setup is sufficient for typical educational use cases.

Gas estimation can sometimes fail for complex smart contract operations. Even though gas prices are set to zero, the estimation process may underestimate the required resources, leading to transaction failures. In such cases, transactions must be retried with higher gas limits, which can cause minor delays in specific situations.

File storage is handled off the blockchain, with only cryptographic hashes stored on-chain. Large submissions, such as video files or extensive programming projects, require external storage solutions. The blockchain is used only to verify file integrity. This design improves storage efficiency but reduces decentralization. Future integration with systems like IPFS could address this limitation.

The current system does not support multi-signature mechanisms for collaborative grading. Only the assignment creator is allowed to publish grades, which prevents teaching assistants or multiple evaluators from participating in the grading process. Supporting multi-signature grading would require additional smart contract development.

There is no recovery mechanism for lost assignment private keys. If a teacher loses their assignment private key, student submissions cannot be decrypted, and recovery is not possible. In contrast, blockchain wallet private keys are encrypted and stored in the database, allowing potential administrative recovery. This difference reflects the zero-knowledge design of assignment encryption, where only the key holder can access content. While this approach aligns with blockchain's trustless model, educational environments may require future recovery solutions that balance security and usability.

Table 3: System Limitations

Limitation	**Impact**	**Potential Solution**
Manual node management	Limited scalability, requires manual configuration for new nodes	Implement auto-discovery and dynamic peer connection
Gas estimation failures	Occasional transaction failures requiring retry with higher limits	Implement adaptive gas limit calculation with safety margins
Off-chain file storage	Large files require external storage, only hashes on-chain	IPFS integration for decentralized file storage
No multi-signature grading	Only assignment creator can grade, no collaborative evaluation	Implement multi-sig smart contracts for committee grading
No recovery for assignment keys	Lost teacher keys = permanent loss of submission access	Key escrow system or social recovery mechanism

10 Conclusion and Future Perspectives

10.1 Challenges Encountered

During the implementation of the system, several technical challenges were encountered. One major difficulty was integrating blockchain technology with a traditional Django-based web architecture. Blockchain operations are asynchronous, while web requests are synchronous. To solve this, blockchain interactions were handled in background processes, which prevents request timeouts and keeps the user interface responsive.

Managing encryption keys was another important challenge. The system uses two different types of keys. RSA keys are used for encrypting assignment submissions and are generated on the client side without being stored. Ethereum keys are used for blockchain identity and are encrypted before being stored in the database. Keeping these two cryptographic systems separate while ensuring security required careful system design.

Docker networking also introduced initial difficulties. Creating reliable connections between blockchain nodes running in containers required a good understanding of Docker's networking and port mapping. This was solved by assigning static IP addresses within Docker's virtual network, which simplified node discovery and peer connections.

Testing blockchain functionality was more complex than traditional application testing. Standard unit tests were not sufficient because blockchain features require a running network. To address this, the testing strategy combined unit tests for individual components with integration tests that deploy a complete blockchain network, ensuring reliable validation of blockchain interactions.

Performance optimization required balancing blockchain immutability with system responsiveness. Direct blockchain queries were too slow for real-time user interfaces. The adopted hybrid architecture stores frequently accessed data in a relational database while keeping the blockchain as the authoritative source. This approach improves performance while preserving data integrity.

10.2 Future Enhancements

Several improvements can be considered for future versions of the system. Integrating IPFS would allow decentralized storage of large files such as videos or large code submissions. Files would be stored on IPFS, while their content hashes would be recorded on the blockchain, ensuring integrity without relying on centralized storage.

Adding multi-signature grading would support collaborative evaluation. Teaching assistants or grading committees could participate in grading through smart contracts that require multiple approvals before finalizing grades. This feature would make the system suitable for more complex academic structures.

Token-based incentive mechanisms could also be introduced. Students could earn tokens for timely submissions, high-quality work, or peer review participation. Smart contracts would automate token distribution based on predefined rules, creating transparent and fair incentive systems.

Cross-chain integration represents another potential enhancement. Important academic records, such as diplomas, could be recorded on public blockchains for global verification, while daily academic operations remain on a private network. This hybrid model would combine privacy with public trust.

Advanced analytics could be developed using blockchain data. Analysis of submission times, grading patterns, and student engagement could help institutions improve teaching strategies. These analytics would rely on immutable records while preserving privacy through data aggregation.

Finally, developing mobile applications would improve accessibility and user experience. Native mobile apps could offer smoother interaction with the system and leverage device-level security features, potentially improving cryptographic key protection.

10.3 Final Conclusion

This project demonstrates that blockchain technology can be effectively applied to educational assignment management. The developed system combines cryptographic security, transparency, and immutable record-keeping in a platform suitable for real academic environments. The use of Proof of Authority consensus enables a private blockchain that balances decentralization with institutional control.

The system architecture shows that blockchain integration does not require sacrificing usability or performance. By combining blockchain with traditional databases, the platform delivers fast user interactions while maintaining strong cryptographic guarantees. This approach addresses practical challenges that often limit blockchain adoption in institutions.

The technical implementation validates the chosen design decisions. RSA encryption protects submission confidentiality, smart contracts enforce academic rules automatically, and Docker ensures consistent deployment across environments. Together, these components demonstrate a system that is close to production-ready for educational use.

Educational environments benefit significantly from blockchain properties. Immutable records prevent grade manipulation, trusted timestamps resolve deadline disputes, and cryptographic verification removes the need for centralized trust. These features solve long-standing academic challenges while introducing new capabilities.

This work contributes to the growing understanding that blockchain technology has practical value beyond cryptocurrencies. Education, healthcare, and supply chain management are examples of domains that can benefit from blockchain-based solutions. This project provides a concrete implementation that can serve as a reference for future developments.

Future adoption of blockchain in education will depend on addressing current limitations and improving scalability, usability, and interoperability. This project establishes a strong foundation for continued research and development in blockchain-based educational systems, highlighting both current achievements and future potential.

Bibliography

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," tech. rep., Ethereum Foundation, 2014.

[3] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, pp. 173–186, 1999.

[4] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[5] N. Szabo, "Smart contracts," 1994.

[6] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[7] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *Ieee Access*, vol. 4, pp. 2292–2303, 2016.

[8] A. Grech and A. F. Camilleri, "Blockchain in education," in *JRC Science for Policy Report, European Commission*, 2017.

[9] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," *2017 IEEE international congress on big data (BigData congress)*, pp. 557–564, 2017.

[10] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," in *IEEE transactions on knowledge and data engineering*, vol. 30, pp. 1366–1385, IEEE, 2017.

[11] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, pp. 1–15, 2018.

[12] C. Dannen, *Introducing Ethereum and solidity*, vol. 1. Springer, 2017.

[13] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[14] W. Stallings, *Cryptography and network security: principles and practice*. Pearson, 2017.

[15] M. Sharples and J. Domingue, "The blockchain and kudos: A distributed system for educational record, reputation and reward," *European conference on technology enhanced learning*, pp. 490–496, 2016.

[16] M. Turkanović, M. Hölbl, K. Kosošić, M. Heričko, and A. Kamišalić, "Eductx: A blockchain-based higher education credit platform," in *IEEE access*, vol. 6, pp. 5112–5127, IEEE, 2018.

[17] Y. Chen, S. Ding, Z. Xu, H. Zheng, and S. Yang, "Blockchain based searchable encryption for electronic health record sharing," *Future generation computer systems*, vol. 95, pp. 420–429, 2018.