sustainability

MDPI

Article

Enhancing Transparency and Trust in Higher Education Institutions via Blockchain: A Conceptual Model Utilizing the Ethereum Consortium Approach

Yerlan Kistaubayev [1], Francisco Liébana-Cabanillas [2,*], Aijaz A. Shaikh [3], Galimkair Mutanov [3], Olga Ussatova [3] and Ainura Shinbayeva [4]

[1] Faculty of Information Technologies, Al-Farabi Kazakh National University, Almaty 050010, Kazakhstan; kistaubayev_yerlan@kaznu.edu.kz
[2] Marketing and Market Research Department, University of Granada, 18071 Granada, Spain
[3] Department of Information Security, The Institute of Information and Computational Technologies, Almaty 050010, Kazakhstan; aijaz.a.shaikh@ipic.kz (A.A.S.); galimkairmutanov@gmail.com (G.M.); olgaussatova@gmail.com (O.U.)
[4] Department of Standardization, Certification and Metrology, Satbayev University, Almaty 050013, Kazakhstan; shinbayeva_a@hotmail.com
* Correspondence: franlieb@ugr.es

Abstract

It has been recognized that Blockchain technology contributes to environmentally sustainable development goals (SDGs). It has emerged as a disruptive innovation capable of transforming various economic and social sectors significantly. This conceptual paper is driven by the need to explore how blockchain, specifically a consortium-based Ethereum architecture, can be integrated into higher education institutions to ensure data sovereignty, integrity, and verifiability while adhering to legal and ethical standards such as GDPR. We propose a multi-layered blockchain-based model for Kazakhstan's Unified Platform of Higher Education (UPHE). This model employs hybrid on-chain/off-chain data storage, smart contract automation, and a Proof-of-Authority consensus mechanism to address system limitations, including data centralization and inadequate verification of academic credentials. Empirical simulations using Blockscout and Ethereum-compatible tools demonstrate the model's feasibility and performance. This paper contributes to the growing discussion on educational blockchain applications by presenting a scalable, secure, and transparent architecture that aligns with institutional governance and Environmental, Social, and Governance (ESG) principles. It also supports the objectives of UN SDG 4 (i.e., Quality education) by fostering trust, transparency, and equitable access to verifiable educational credentials.

Keywords: blockchain adoption; interdisciplinary analysis; sectoral impact; organisational innovation; technology integration

check for updates

Academic Editor: Fernando Moreira

Received: 9 September 2025
Revised: 3 October 2025
Accepted: 11 October 2025
Published: 21 October 2025

Citation: Kistaubayev, Y.; Liébana-Cabanillas, F.; Shaikh, A.A.; Mutanov, G.; Ussatova, O.; Shinbayeva, A. Enhancing Transparency and Trust in Higher Education Institutions via Blockchain: A Conceptual Model Utilizing the Ethereum Consortium Approach. *Sustainability* **2025**, *17*, 9350. https://doi.org/10.3390/su17209350

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Fourth Industrial Revolution (Industry 4.0) has transformed our lives and society, revolutionized production methods, and created opportunities to improve our quality of life. However, there is a growing recognition that this transformation must also address the need to develop a more environmentally sustainable society and institutions [1]. Blockchain technology, a key component of Industry 4.0, is now utilized in various fields, including education, and many governments have adopted this technology to promote environmental

sustainability [2]. Among the 17 UN SDGs, SDG-4 focuses on quality education and building an inclusive and equitable education system that offers lifelong learning opportunities for all. Therefore, applying blockchain technology in education is an important step toward aligning technological progress with the human-centered goals of SDG-4 [3].

In the digital era, higher education institutions (hereinafter referred to as HEIs), including universities and degree-awarding institutions, face numerous challenges related to the security, sustainability, authenticity, transparency, and verifiability of academic records. These issues undermine institutional trust, hinder student mobility, and diminish employer confidence. Aligned with the United Nations Sustainable Development Goal (SDG-4), this paper conceptualizes how blockchain technology (a consortium-based Ethereum model) could address these challenges within the context of a developing country's higher education system.

Blockchain is a distributed database or ledger. The data in a blockchain like Ethereum is not "encrypted". They are secured through cryptographic hashes and digital signatures. Additionally, blocks are connected not just chronologically but through cryptographic references to previous blocks; timestamps are present but are unreliable and do not serve as the main method of ordering.

Blockchain technology is recognized for enabling secure transactions, ensuring data integrity, supporting user anonymity, and functioning within a decentralized and distributed framework [4]. Since its inception with Bitcoin, blockchain has evolved into a versatile and practical technology through the development of smart contracts on platforms like Ethereum. Nevertheless, a promising application of blockchain technology in education is its role in certificate authentication. With blockchain, educational institutions can securely issue and verify digital certificates that are stored on the blockchain. These digital certificates may encompass degrees, diplomas, certifications, and continuing education credits. By leveraging blockchain, educational institutions can ensure their certificates are secure and tamper-proof, providing confidence to students and employers [5,6].

One of the main benefits of blockchain is its high security level. Transactions are stored across a decentralized network, removing a single point of failure that hackers could target. They are digitally signed to verify authenticity, while the inability to modify them comes from their inclusion in the cryptographically linked block structure and consensus algorithm, not from the transaction itself.

These features make blockchain ideal for securely managing sensitive academic records, including certificates and transcripts. It is therefore increasingly believed that blockchain technology can transform the higher education system, both private and public, by offering decentralized, secure, and transparent data storage methods. Thus, integrating blockchain technology into the higher education system can significantly enhance the efficiency, security, and reliability of educational processes, as well as administrative tasks and records management, ensuring economic growth and social stability [7,8]. By creating secure and transparent platforms for tracking and verifying students' academic achievements, blockchain can help develop a more accessible and trustworthy educational ecosystem. This, in turn, facilitates students' ability to demonstrate their skills and knowledge to prospective employers.

Kazakhstan, since gaining independence from the Soviet Union in 1991, has made notable progress in various areas, including digitalizing government services, establishing stable financial and payment systems, and advancing HEIs that equip citizens with modern, skill-based knowledge. Against this backdrop, the Government of Kazakhstan developed the Unified Platform of Higher Education (UPHE). The purpose of the UPHE is to oversee and monitor the activities of HEIs in the country by integrating their Learning Management Systems (hereinafter referred to as LMS) into a centralized data repository. Currently,

113 HEIs in Kazakhstan are connected to UPHE, which consolidates the academic data of over 500,000 students, approximately 3.5 million graduates, nearly 50,000 academic staff members, and 13,000 educational programs across all levels of study [9]. This data illustrates the advanced level of digital maturity achieved by Kazakhstan's HEIs and the national higher education system. Higher education institutions' business processes are configured to transmit educational data streams, including information on students, their academic performance, and mobility, to UPHE on a monthly basis.

Despite these key advancements in Kazakhstan's higher education ecosystem through the UPHE platform, UPHE remains centralized and lacks strong mechanisms for certificate verification, auditability, and resistance to data tampering. These limitations weaken transparency, mobility, and employer trust. Furthermore, existing studies [10–12] primarily focus on isolated blockchain applications, such as credential verification or credit transfer, but fail to address the needs of national, large-scale, integrated systems like UPHE. This creates a significant gap in research and implementation.

Against this backdrop, this study aims to explore how blockchain, specifically a consortium-based Ethereum architecture, can be conceptually integrated into national education infrastructure to ensure data sovereignty, integrity, and verifiability while aligning with legal and ethical standards, such as GDPR. Furthermore, this conceptual study has four primary objectives. First, the limitations of the current UPHE architecture regarding data integrity and transparency need to be critically examined. Second, a blockchain-based consortium model for secure storage of academic records and automated diploma verification should be proposed. Third, compliance with national and international data protection standards, including GDPR, must be ensured. Fourth, the proposed model needs to be evaluated against scalability and ESG (Environmental, Social, Governance) benchmarks.

By addressing these objectives, this research contributes to SDG 4, aiming to build inclusive, verifiable, and trustworthy educational systems globally. Also, this research seeks to enhance the management and verification of student records in HEIs. The conceptual paper demonstrates how a specialized blockchain network (Ethereum consortium) can improve the security, verification, and tamper-resistance of academic records. The conceptualized model seeks to address current issues in Kazakhstan's education data system by making it more transparent, trustworthy, and compliant with international data protection laws. This can help students showcase their qualifications more easily and give employers greater confidence in their academic documents.

Next, we discussed blockchain technology, its process, principles, and applications in HEIs.

2. Materials and Methods

2.1. Blockchain Technology and SDG 4: The Process, Principles, and Applications in Higher Education

Blockchain technology plays a vital role in advancing governance and sustainability [13]. Sustainability is defined as meeting "the needs of the present without compromising the ability of future generations to meet their own needs" [14]. Current research [3] indicates that education is essential for achieving the SDGs, especially SDG 4: Quality Education. As a catalyst for social change, blockchain has demonstrated strong potential to support sustainable practices and assist society, organizations, and governments in reaching the United Nations' SDGs [14].

Blockchain is a distributed digital ledger technology that enables secure, transparent, and tamper-proof data recording across multiple computers (nodes) without relying on a central authority. Once data is recorded in a block and confirmed by the network, it becomes immutable—no one can change or delete it without detection. In the case of student records,

a transaction might look like this: A student completes a course; the university system generates a transaction that includes the student's ID, course code, grade, ECTS credits, academic period, and a digital signature from the university. Once a transaction is created, it is broadcast to the network.

Once a set of transactions is validated, they are grouped into a block according to the rules of the underlying consensus protocol. Each transaction is first cryptographically hashed, producing a fixed-length output known as a transaction hash. These hashes are then combined into a Merkle tree, where the hierarchical structure offers an efficient and secure way to represent all transactions in the block.

A Merkle tree is a binary structure where each leaf node represents the hash of a single transaction. Non-leaf nodes are created by hashing the combined data of their two child nodes. This process continues upward until a single hash, called the Merkle root, is reached. The Merkle root provides a compact cryptographic summary of all transactions in the block. Changing any one transaction will alter its hash and all parent hashes up to the root. This feature allows verification of a transaction's inclusion and integrity by checking only a small set of hashes, rather than rehashing the entire block, ensuring efficiency and tamper detection.

The block header contains the Merkle root, along with other essential elements such as the hash of the previous block and a nonce. The cryptographic hash of the block header acts as the unique identifier for the block. Blockchain systems use different hash functions: for example, Bitcoin employs SHA-256, while Ethereum relies on Keccak-256 (SHA-3 variant). These hash functions are computationally efficient yet practically irreversible. As a result, once a block is added to the blockchain, any attempt to alter its transactions will change the Merkle root and the block hash, thereby invalidating all subsequent blocks and making tampering immediately detectable.

In the case of a consortium blockchain (e.g., the one used in higher education), a Proof-of-Authority (PoA) consensus algorithm is typically utilized. In PoA, the validators are pre-approved, trusted entities (such as universities or regulatory bodies). Furthermore, the PoA algorithm ensures that only authorized validators can approve blocks, enhancing security and decreasing the likelihood of malicious actors undermining the network.

Once a block is mined or validated, it is proposed for inclusion in the blockchain. Other nodes in the network independently verify the block to ensure that all transactions are valid and that the block complies with the consensus protocol rules. This validation process prevents fraudulent actions or inconsistent state updates, such as the submission of a credential that was never legitimately issued. Integrity of transactions is further ensured through the verification of digital signatures and the consistency of transaction data with the current blockchain state. After successful validation, the block is appended to the blockchain, and its transactions are considered final and confirmed.

After a transaction is included in a block and added to the blockchain, the data becomes immutable: it cannot be changed, deleted, or tampered with. This immutability is ensured by the cryptographic hashes that link each block to its predecessor. For example, once a student's academic record (grades, credits, etc.) is added to the blockchain, it cannot be altered, providing a secure and verifiable record of their achievements.

2.2. Kazakhstan—The Context of the Study

According to the United Nations [15], Kazakhstan ranks 28th in the E-Government Development Index (EGDI) and 8th in the Online Service Index. In Kazakhstan, 1208 government services (or 92%) are provided electronically. This achievement is made possible through the development of a robust e-government infrastructure, including key state databases, sectoral information systems, data exchange and integration components

(such as the "e-Government Gateway" and Smart Bridge), and a wide range of digital services [16].

Additionally, the World Competitiveness Centre of the International Institute for Management Development (IMD) in Lausanne, Switzerland, has announced the results of the 2022 Digital Competitiveness Ranking. This study assesses how countries adopt and implement digital technologies that drive the transformation of government policies, business models, and society. Kazakhstan ranked 36th globally in this ranking [17].

In the context of governance, Kazakhstan has shown notable progress in the digital transformation of its higher education system. One of the most significant advancements is the development and implementation of the UPHE—a multi-level, integrated information system designed to encompass all stakeholders, implementers, and beneficiaries involved in the governance of higher education. This system reflects the high level of infrastructure and digital maturity of HEIs and the Ministry of Science and Higher Education (MSHE), positioning them well for the adoption of blockchain technology.

From an infrastructure perspective, implementing blockchain would primarily require deploying network nodes using existing computational resources, eliminating the need for significant hardware upgrades. On the software side, the current business process digitization architecture, which relies on REST API interactions, would remain largely unaffected. Only minor adjustments to the backend components of UPHE would be necessary to facilitate blockchain integration.

Furthermore, the widespread implementation of UPHE significantly enhances the feasibility of scaling blockchain-based solutions, particularly in automating the issuance and verification of educational credentials. All HEIs have already integrated their local LMS with the digital ecosystem of MSHE by adapting to UPHE protocols, which align with the Bologna Process. This experience provides a strong foundation for adopting decentralized ledger technologies to improve transparency, security, and interoperability in higher education governance.

From a social responsibility perspective within the ESG (Environmental, Social, and Governance) framework, the implementation of blockchain technology in the UPHE adheres to data privacy and ethical standards, particularly the principles outlined in the General Data Protection Regulation (GDPR). In accordance with national legislation and GDPR-like provisions, learners consent to the processing of their data within Learning Management System (LMS) platforms. This consent allows HEIs to possess and securely transmit necessary data to supervisory authorities through protected communication channels. This commitment to ethical data handling directly supports the principles of SDG 4, ensuring that digital transformation in education is equitable and secure.

Before discussing the proposal, we categorized it according to the protocol-level classification categories for blockchain-based projects [18]. Figure 1 illustrates this classification and demonstrates the exemplary placement of the project solution within the classification of blockchain-based projects.

2.3. The UPHE Architecture

Figure 2 depicts the current architecture of UPHE. The platform features two user interaction channels: a REST API for automating synchronization processes with the Learning Management System (LMS) of HEIs and a web portal for manual operations. Through the REST API, HEIs manage tasks such as registering educational programs, students, and their academic performance. The UPHE portal facilitates the manual preparation of student graduation data (e.g., diploma issuance, student-to-graduate transitions), which requires digital signature (EDS) authorization from the head of the HEI to validate actions. Additionally, the portal provides an open-access diploma verification service for all user

categories, including employers, HEIs, graduates, and parents. Verification is conducted by searching diploma data using the graduate's Individual Identification Number (IIN), diploma number, and series. The special markers assigned to the MSHE and Employer roles indicate that the former is deprived of the authority to grant or revoke access rights for universities and other organizations, whereas the latter, in turn, is endowed with the capability to verify the authenticity of a job applicant's educational credentials. Figure 3 illustrates a sequence diagram that shows the interaction processes between users and systems.

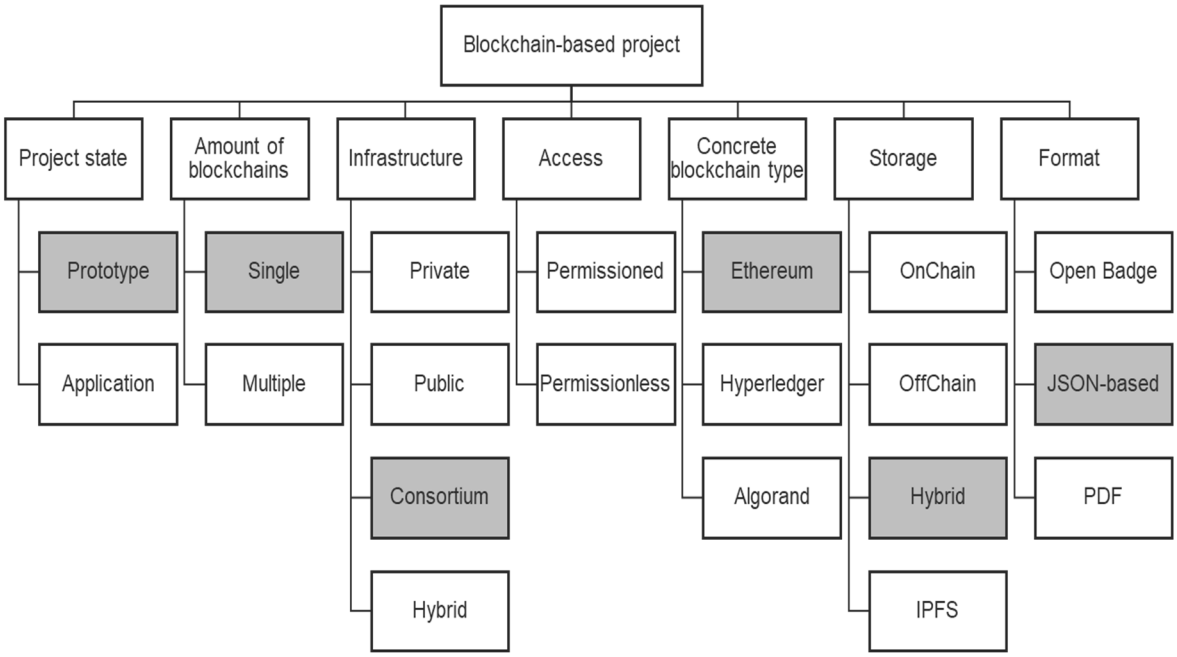


Figure 1. Classification of blockchain-based projects and exemplary placement of the proposal.

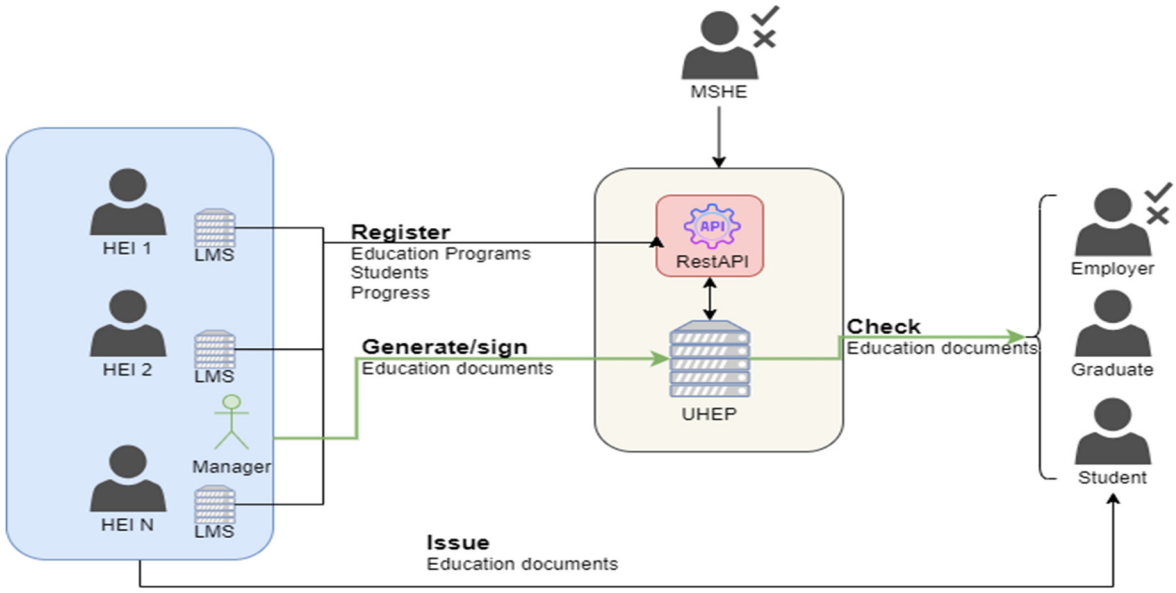


Figure 2. Current architecture of a UPHE.

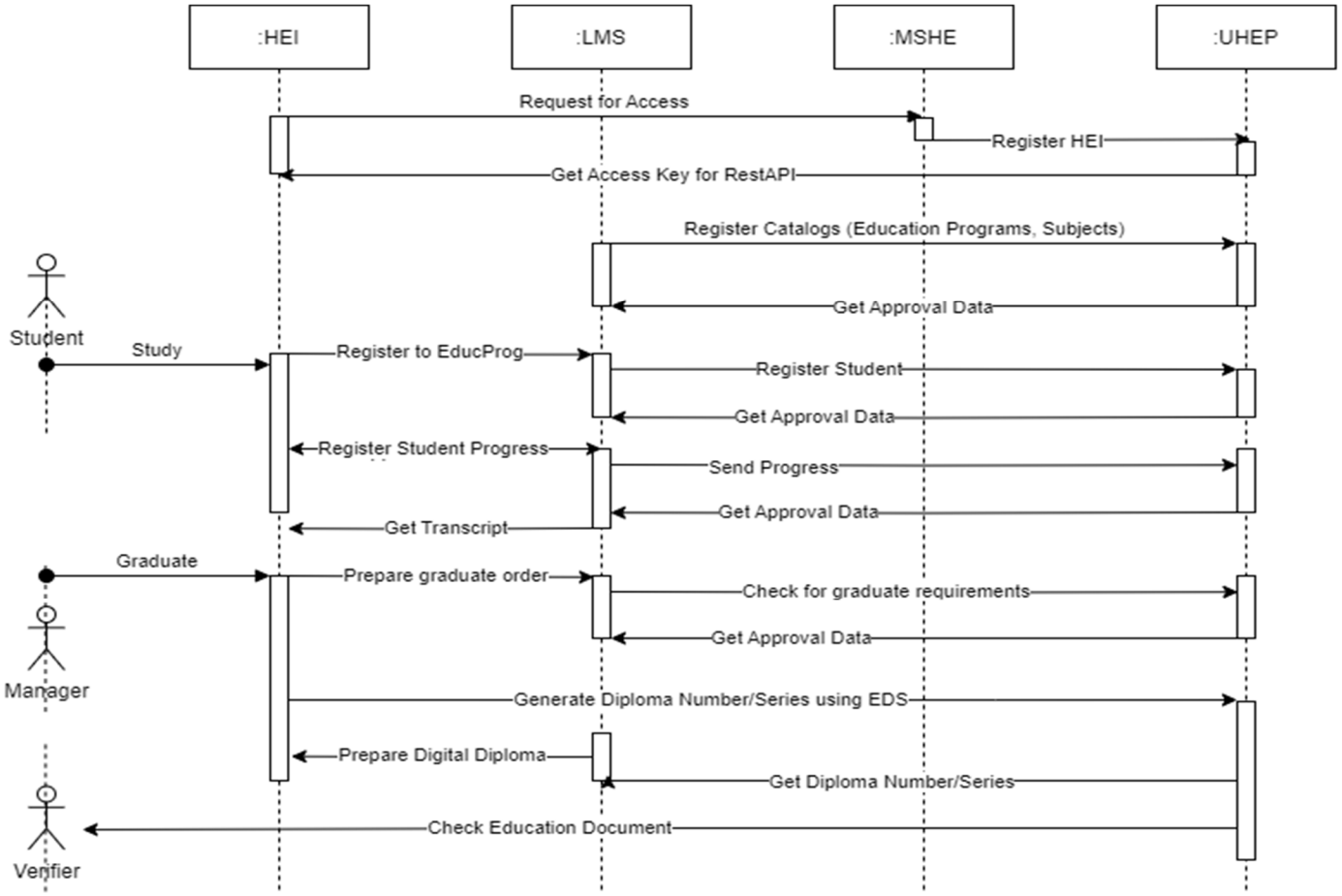


Figure 3. User Interaction Sequence Diagram within the System Workflow.

The figures and accompanying descriptions illustrate the current state of digitized services within the higher education system in Kazakhstan. However, the existing system has several limitations that require optimization, especially considering the opportunities presented by blockchain technology. Among these limitations are the following:

Limitations of the Current UPHE Infrastructure

First, the architecture illustrates that data replication occurs exclusively between two nodes, specifically the HEI's LMS and the UPHE database. However, data from one HEI is not replicated in any other database. In the event of risks associated with the loss of the centralized UPHE repository or the LMS, all critical data could be lost.

Second, although the MSHE mandates that HEIs submit timely and up-to-date data to UPHE, the REST API channel lacks the technical means to prevent HEIs from retroactively overwriting students' academic performance records and other critical information. Furthermore, data stored in UPHE is not encrypted, which further exacerbates its vulnerability.

Third, as shown in the sequence diagram (Figure 2), educational document numbers are generated manually due to the institutional head's requirement for a digital signature (EDS) to authorize the list of students eligible for graduation. However, the graduation requirements—based on level, mode of study, and academic performance—are strictly regulated by MSHE and can be processed algorithmically.

Fourth, the UPHE portal provides open access to diploma authenticity verification based on three primary parameters: diploma number, diploma series, and the graduate's Individual Identification Number (IIN). This service is primarily intended for employers, enabling them to validate a candidate's diploma using a physical copy. However, in

instances where students transfer to another HEI through academic mobility or when a graduate has lost their educational documents and cannot access the system to retrieve digital copies, the service is inadequate. One of the main challenges of traditional certificate authentication is that the process can be time-consuming and costly. Employers often must request transcripts and certificates from multiple institutions to verify their authenticity. This can be a tedious and expensive procedure, especially for large organizations that need to validate the certificates of numerous candidates [6,19].

2.4. Performance Comparison of Consensus Algorithms and ESG Criteria

Consensus algorithms should also be evaluated based on performance parameters such as throughput (measured in transactions per second, TPS) and scalability. In the blockchain context, throughput refers to the number of transactions that the system can process and confirm per second. This metric reflects the efficiency of the consensus mechanism in handling transaction load. Scalability, in turn, describes the system's ability to maintain or improve its performance as the number of nodes and transactions increases.

According to experimental studies [20], a throughput of 113,000 transactions per second (TPS) can be achieved with four nodes in the network. However, this figure drops significantly to 2400 TPS as the network size increases to 64 nodes. Furthermore, when the transaction size reaches 500 bytes, these throughput values decrease by approximately one third [21].

A method known as sharding is used to enhance throughput [22]. Sharding, adapted from database systems, involves partitioning large datasets into smaller pieces (shards) that are distributed across various machines. Consequently, each node no longer needs to handle the entire transaction load of the network, which improves performance and efficiency.

Structural scalability refers to the network's ability to increase the number of nodes participating in consensus or functioning as clients without degrading system performance [23]. A primary objective of blockchain technology is to support business applications through a decentralized and secure network architecture. While PoS-based networks can accommodate thousands of participants [24], they do not ensure finality in consensus, which is crucial for certain applications. This limitation has led to the adoption of consortium-based networks, in which the number of nodes is governed by a central authority [25,26], allowing system performance to be more predictable and manageable.

An important aspect to consider when analyzing consensus algorithms is their alignment with ESG criteria—environmental, social, and governance. This approach has gained significance due to the growing global trend toward sustainable and responsible technologies.

The PoS and PoA have minimal environmental impact since they do not require energy-intensive mining operations. In contrast, PoW results in a significant carbon footprint due to its computational demands. While PoW is theoretically open to anyone, the high entry costs limit participation in practice. PoS requires a certain number of tokens to participate, which can exclude users with fewer resources. PoA is even more restrictive, as validators are pre-selected, reducing inclusivity. PoA offers strong controllability but risks centralization. PoS provides more flexibility in governance but could be dominated by large stakeholders. Proof-of-work (PoW) is the most decentralized yet the most difficult to manage effectively. Therefore, from an ESG perspective, PoS is the most balanced solution, providing energy efficiency with moderate social and governance sustainability (See Table 1).

Table 1. ESG Comparison of Consensus Algorithms.

S	Algorithm	Environmental	Social	Governance	Overall ESG Score
1	PoW	High energy consumption	High barrier to entry	Partial decentralization	Low
2	PoS	Energy efficient	Participation is limited by stakeholder	Stakeholder dominance risk	Medium
3	PoA	Very low energy use	Low inclusivity	Centralized control	Medium–Low

2.5. Consortium-Based Ethereum Network for Enterprise Applications

Deploying consortium blockchain networks is crucial for adopting blockchain technologies in enterprise environments. Unlike public blockchains, such as the leading Ethereum network [27], consortium blockchains feature a partially decentralized architecture where consensus participation and data access are limited to a predefined set of trusted entities. This model is well-suited for business use cases that demand governance, enhanced performance, and regulatory compliance.

As a flexible and expressive platform supporting Turing-complete smart contracts, Ethereum offers a solid foundation for building consortium networks. These networks enable multiple organizations to jointly maintain a distributed ledger while utilizing the full suite of Ethereum tools, including the Solidity programming language, developer frameworks like Truffle, and node orchestration utilities such as Geth and Besu. Typically, these networks adopt alternative consensus mechanisms, such as Proof of Authority (PoA) or Istanbul Byzantine Fault Tolerance (IBFT), which are more suitable for permissioned environments, ensuring high throughput and low latency.

From a governance perspective, nodes in a consortium network are pre-selected and verified, enabling the enforcement of access control policies and accountability mechanisms. This controlled participation enhances the trust model and significantly reduces the risks associated with anonymous validators or unpredictable network behavior.

Performance is another critical advantage. Due to the reduced network size and optimized consensus protocols, consortium-based Ethereum deployments demonstrate substantially higher transaction throughput than their public counterparts. Empirical evaluations have shown that permissioned Ethereum networks can process thousands of transactions per second, depending on configuration parameters such as block time and the number of validators. These properties make the model especially viable for mission-critical applications involving high-frequency transactions and strong consistency requirements, such as interbank settlements, supply chain traceability, digital identity, and document verification systems [28].

Additionally, consortium networks enable fine-grained privacy through permissioned access and data partitioning mechanisms, aligning blockchain infrastructures with enterprise data protection standards (e.g., GDPR, ISO/IEC 27001 [29]). These characteristics are generally unattainable in fully public blockchains, reinforcing the suitability of this model for organizational contexts.

According to blockchain taxonomy, MSHE is the central certified authority that registers and validates consortium nodes. Unauthorized or Byzantine node behavior is immediately detected and penalized [30]. Consortium blockchains provide greater control by restricting read/write permissions to pre-authorized members, ensuring efficient transaction validation and chain consistency.

In summary, the consortium-based Ethereum network strikes a balanced trade-off between decentralization and control, providing a scalable, secure, and enterprise-friendly architecture while maintaining the Ethereum platform's programmable flexibility and ecosystem maturity.

2.6. The Conceptual Architecture of a Solution

Based on our analysis of the current architecture of the UHEP and the digital services within Kazakhstan's higher education system, we propose an optimized and enhanced model of UHEP, which is called the decentralized platform for higher education (hereinafter referred to as DPHE), and presented as a multi-layered component architecture in Figure 4.

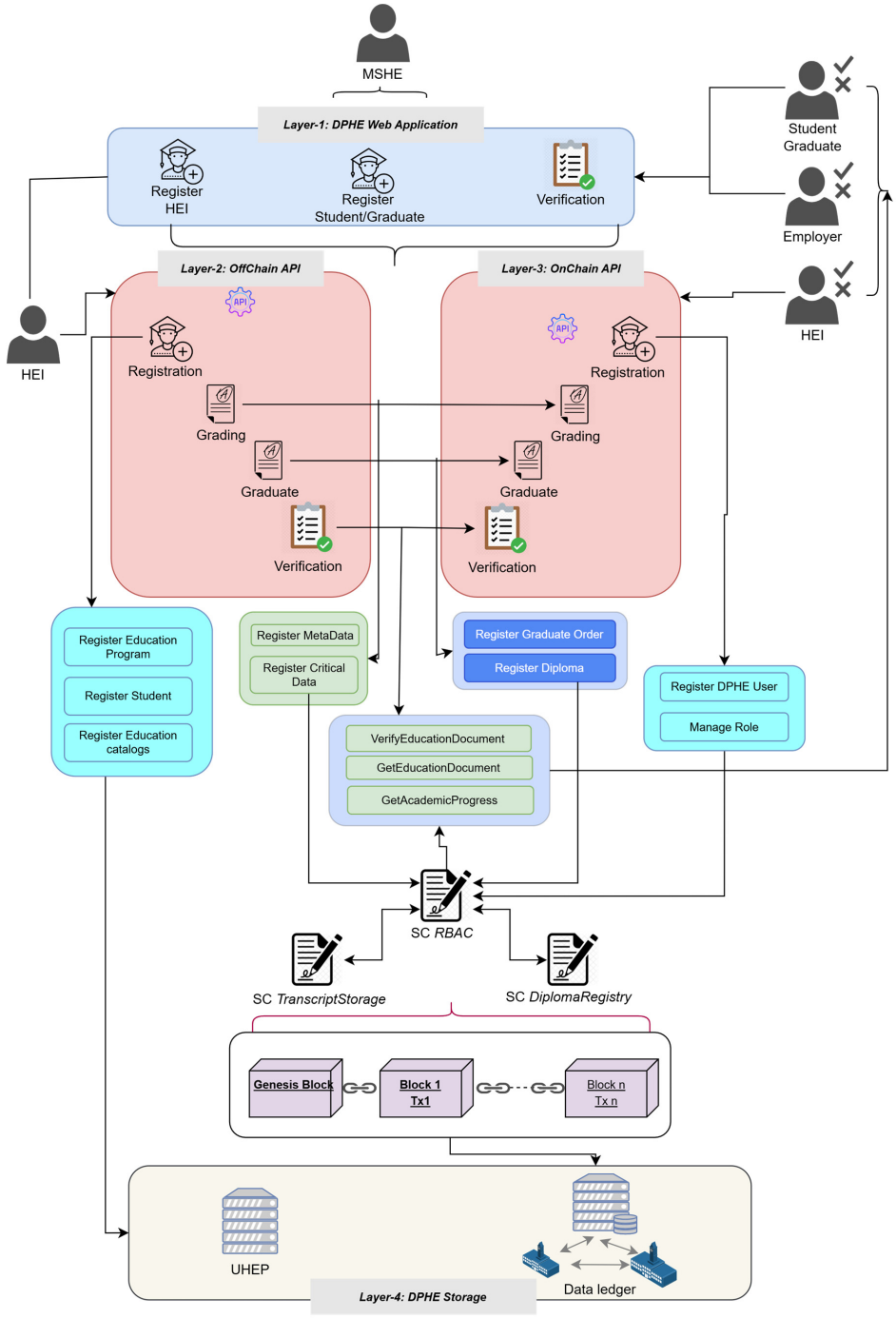


Figure 4. Conceptual Architecture of the DPHE with Blockchain Integration.

Layer-1. DPHE web application: This layer represents the web platform for HEIs, students and graduates, and employers. Unlike the UHEP, it features an enhanced architecture with authenticated access for graduates and students to obtain digital versions of educational documents (certificates, diploma supplements, academic transcripts, etc.). This will eliminate the need to contact HEIs or the relevant ministry or government agency for duplicate educational documents. HEIs can view their catalogs, student and graduate demographics, and educational documents within this system. Students and graduates can register, view their educational documents (transcripts and diplomas), and download them. Employers and other organizations can then verify applicants' educational documents without needing registration.

Layer-2: OffChain API—this layer represents the RestAPI technology used to work with data from the centralized storage owned by the MSHE validator. The API channel offers various methods, such as registering HEI catalogs (disciplines, educational programs, curricula, student/graduate demographics); registering academic performance metadata (metadata includes the title, type, and other details about the discipline, as well as the obtained qualification); registering the graduation process and diploma issuance (issuance of the graduation order); verifying academic performance (individual services for extracting metadata from the OffChain database and complex services for combined data extraction from OffChain and OnChain databases); verifying educational document data (searching for document data through the diploma number/series and student identifier); and generating a digital equivalent of an educational document (academic transcript, diploma, diploma supplement).

Layer-3: OnChain API—this layer represents a REST API channel for interacting with the decentralized storage of the blockchain network. By default, the OnChain component is deployed on the primary blockchain node owned by the government or MSHE. It is accessible to network participants (i.e., HEIs) who cannot host their nodes. A deployment guide and a ready-to-use Docker package are provided for participants managing their nodes to establish a local OnChain API channel. This enables participants to operate independently of the MSHE when utilizing OnChain API services (using the version deployed on their node).

The OnChain API services can be categorized as:

- Access Provisioning for HEIs involves account creation, permission assignment, and token issuance for authorized access to API methods.
- Student Performance Registration: methods for registering essential academic performance data (grades, credits, academic period); retrieval of the transaction hash/block number after successful block formation; verification of the completion of necessary academic credits.
- Graduation Process and Diploma Issuance: Approaches for Recording Graduation-Related Data.

Layer-4: Data storage—this project utilizes a hybrid data storage structure, combining on-chain and off-chain approaches. This method aligns with our commitment to preserving high-quality data, particularly the complete academic performance history. While many existing studies store only digital versions of diplomas or their hashes on the blockchain, we aim to retain all qualitative data necessary to generate any educational document (academic transcript, diploma, diploma supplement).

Due to the high storage demands of blockchain systems, we have separated critical data from metadata. Metadata encompasses textual, reference, and auxiliary information, including catalogs of disciplines, educational programs, curricula, and parts of transcript content. This data is stored in the centralized UHEP repository or the OffChain database, which is managed by the Ministry of Science and Higher Education (MSHE).

In the OffChain database, each entity (reference data, students, and graduates) is stored using the original identifiers from the university's LMS. For instance, a student with a unique number 12,345 in their home university's LMS will be saved in the OffChain DB with the same number. To prevent duplicate student IDs across different universities, additional fields were added to the tables that include the university ID from the official university registry. Therefore, a database record can be uniquely identified using a composite key that consists of the student ID and the university ID. This method eliminates any dependency of platform participants on parameters specific only to the OffChain DB.

In contrast, the data ledger stores critical information that is particularly vulnerable to falsification. This includes academic performance records (grades, number of credits, academic period) and graduation data (diploma number and series, obtained qualification, start and end dates of study, university information).

2.7. Smart Contracts

The limitations of the Ethereum blockchain platform restrict the transaction payload to 1 KB. Therefore, we ensured that the data for the TranscriptStorage and DiplomaRegistry smart contracts does not exceed this limit.

2.7.1. Smart Contract "TranscriptStorage"

This smart contract is designed to manage the processes of storing, retrieving, and verifying academic performance data within the data ledger. All operations related to a student's academic performance can be categorized into metadata registration and critical data registration. Metadata is transferred to the OffChain database, while critical data is stored in the ledger. Metadata is represented in JSON format (Figure 5) and contains textual information such as the course title and code, student ID, and course type (e.g., mandatory or elective, internship or research work).

A composite key that combines university and student identifiers enables efficient indexing and pagination. Integration with an external Role-Based Access Control (RBAC) module ensures strict access control: only authorized actors (e.g., those assigned the roles of issuer or admin) are allowed to modify the transcript repository. Authorization is enforced through the RBAC contract, which encapsulates the role verification logic.

Data immutability and auditability in blockchain are primarily guaranteed by its structural properties, such as cryptographic hashing, Merkle tree organization, and consensus validation, which ensure data integrity, accountability, and non-repudiation. Blockchain events, while not responsible for immutability themselves, serve as auxiliary logs that facilitate off-chain monitoring and efficient tracking of contract interactions. Within this design, the TranscriptStorage contract remains particularly well-suited for decentralized academic data management systems in higher education institutions (HEIs).

In turn, the critical part of the data comprises the student ID, final course grade, number of ECTS credits, the academic period in which the course was completed, and the course type. The final academic performance for a given study period is serialized in JSON (Figure 5) format and transmitted to the TranscriptStorage smart contract. The contract records this data as a transaction on the blockchain ledger, returning a temporary GUID generated by the Message broker (RabbitMQ) (Figure 6). After a certain period, once the transaction is accepted and recorded in a block, it is possible to retrieve the remaining data using the temporary GUID, which includes the transaction hash, processing status, execution time, block number, and the sender and recipient addresses. Additionally, the smart contract contains an algorithm to verify whether the student has accumulated sufficient academic credits according to their mode of study, which is required for graduation. For example, suppose a student is enrolled in a bachelor's degree program and must complete

at least 240 ECTS academic credits according to their curriculum. In that case, the algorithm will check this condition after each transaction and automatically execute the graduation procedure once the requirement is fulfilled.


```
{
  "id": 1213546,
  "studentId": 178709,
  "subjectCode": "Mat (1) 1201",
  "subjectNameRu": "Математика I",
  "subjectNameEn": "-",
  "subjectNameKz": "Математика I",
  "accepted": 0,
  “wasretaken": 0,
  "codeEn": "-",
  "codeRu": "Mat (1) 1201",
  "deleted": false,
  "isGeneralExam": false,
  "isCoursework": false,
  "notIncludedScholarship": false,
  "hash": null,
  "markid": 0,
  "isPassed": true,
  "typeCode": "TRANSCRIPT",
  "universityId": 32
}
```


Figure 5. Meta-data of transcript for OffChain DB.

Along with storing data in the data ledger, the smart contract provides methods for retrieving this data using the two major parameters, i.e., (1) by student ID and university ID, it is possible to retrieve all academic transcript records; (2) by transaction hash, a specific academic transcript record can be retrieved.



```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.10;

import "./IRBAC.sol";

contract TranscriptStorage {
    IRBAC private rbac;

    struct Transcript {
        uint universityId;
        uint studentId;
        bytes32 subjectCode;
        uint credits;
        bytes32 alphaMark;
        uint courseNumber;
        uint term;
        string subjectNameRu;
        string subjectNameKz;
        string subjectNameEn;
        bool isPassed;
        bool isGeneralExam;
        uint ects;
        address addedBy;
    }

    mapping(uint => Transcript[]) private transcripts;

    event TranscriptSaved(uint studentId, bytes32 subjectCode, uint
credits, bytes32 alphaMark, address addedBy);
    event RBACAddressUpdated(address newAddress);

    constructor(address rbacAddress) {
        rbac = IRBAC(rbacAddress);
    }

    function getRBACAddress () public view returns (address) {
        return address(rbac);
    }

    function updateRBACAddress(address newRBACAddress) public
onlyRole("admin") {
        require(newRBACAddress != address(0), "Invalid RBAC address");
        rbac=IRBAC(newRBACAddress);
        emit RBACAddressUpdated(newRBACAddress);

    }

    modifier onlyRole(string memory role) {
        require(rbac.hasRole(msg.sender, role), "Access denied: You do
not have the required role");
        _;
    }
```

```
    function saveTranscript(
        uint universityId,
        uint studentId,
        bytes32 subjectCode,
        uint credits,
        bytes32 alphaMark,
        uint courseNumber,
        uint term,
        string memory subjectNameRu,
        string memory subjectNameKz,
        string memory subjectNameEn,
        bool isPassed,
        bool isGeneralExam,
        uint ects
    ) public onlyRole("issuer"){

        require(subjectCode != bytes32(0), "Subject code cannot be
empty");
        require(alphaMark != bytes32(0), "Alpha mark cannot be empty");
        require(bytes(subjectNameRu).length > 0, "Subject name RU cannot
be empty");
        require(bytes(subjectNameKz).length > 0, "Subject name KZ cannot
be empty");
        require(bytes(subjectNameEn).length > 0, "Subject name EN cannot
be empty");
        require(universityId>0, "University ID cannot be empty");
        require(studentId>0, "Student ID cannot be empty");

        Transcript memory newTranscript = Transcript(
            universityId,
            studentId,
            subjectCode,
            credits,
            alphaMark,
            courseNumber,
            term,
            subjectNameRu,
            subjectNameKz,
            subjectNameEn,
            isPassed,
            isGeneralExam,
            ects,
            msg.sender
        );

        transcripts[studentId].push(newTranscript);

        emit TranscriptSaved(studentId, subjectCode, credits, alphaMark,
msg.sender);
    }

    function getTotalTranscripts(uint studentId) public view returns
(uint) {
        return transcripts[studentId].length;
    }
```

```
    function getTranscripts(uint studentId, uint page, uint pageSize)
public view returns (Transcript[] memory) {
        uint totalTranscripts = transcripts[studentId].length;
        require(totalTranscripts > 0, "No transcripts found for this
student");

        uint startIndex = page * pageSize;
        require(startIndex < totalTranscripts, "Page index out of
bounds");

        uint endIndex = startIndex + pageSize;
        if (endIndex > totalTranscripts) {
            endIndex = totalTranscripts;
        }

        Transcript[] memory result = new Transcript[](endIndex -
startIndex);

        for (uint i = startIndex; i < endIndex; i++) {
            Transcript storage transcript = transcripts[studentId][i];
            result[i - startIndex] = Transcript(
                transcript.universityId,
                transcript.studentId,
                transcript.subjectCode,
                transcript.credits,
                transcript.alphaMark,
                transcript.courseNumber,
                transcript.term,
                transcript.subjectNameRu,
                transcript.subjectNameKz,
                transcript.subjectNameEn,
                transcript.isPassed,
                transcript.isGeneralExam,
                transcript.ects,
                transcript.addedBy
            );
        }
        return result;
    }
}
```

```
{
  "transcriptId": 1213546,
  "studentId": 178709,
  "totalMark": 72.0,
  "courseNumber": 1,
  "term": 1,
  "type": 0,
  "subjectId": 27,
  "ects": 5.00
}
```

Figure 6. The TranscriptStorage smart contract source code and a format of academic transcript data for storing in data ledger.

2.7.2. Smart Contract "DiplomaRegistry"

This smart contract differs from the TranscriptStorage contract as it stores and processes a different category of data. Specifically, this includes the diploma number and series, the number and date of the official record (protocol) confirming the awarding of an academic degree based on the graduate's study format, the academic degree and qualification, the educational program (code and title), the dates of university admission and graduation, and the name of the university. The contract ensures the uniqueness of each diploma issued by a university to a student, preventing the duplicate issuance through pre-checks for existing entries.

Security and access control mechanisms are enforced through an external modular RBAC (Role-Based Access Control) contract, which restricts diploma issuance operations to users with designated roles (e.g., issuer). Blockchain technology ensures the immutability and transparency of records: all issuance events are logged in the smart contract's event log.

This approach greatly lowers the risk of diploma forgery, streamlines verification processes, and builds trust among educational institutions, especially in international academic interactions.

The smart contract can be invoked by the previous contract or directly via the OnChain API. The latter case is used for graduates whose academic performance data is not stored at the university, but for whom official graduation documents (e.g., university decree) are available. In such instances, the graduation process can be recorded in the data ledger without traces of academic performance.

In all the scenarios described above, the resulting data structure for the graduate is identical, as illustrated in Figure 7. The list indicates that the dataset is sufficient to generate a digital version of the diploma.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.10;

import "./IRBAC.sol";

contract DiplomaRegistry {
    IRBAC private rbac;

    struct Diploma {
        uint256 universityId;
        uint256 studentId;
        bytes32 minutesNumber;
        bytes32 dateOfIssue;
        bytes32 registrationNumber;
        string fio;
        uint256 year;
        bytes32 dateOfMinutes;
        bytes32 iacDiplomaSeries;
        bytes32 iacDiplomaNumber;
        address addedBy;
    }

    mapping(uint256 => Diploma) private diplomas;
    mapping(uint=>mapping(uint=> bool) ) private issuedDiplomas;

    event DiplomaCreated(uint256 indexed studentId, bytes32
diplomaNumber, bytes32 series, address addedBy);
    event RBACAddressUpdated(address newAddress);

    constructor(address rbacAddress) {
        rbac = IRBAC(rbacAddress);
    }

    modifier onlyRole(string memory role) {
        require(rbac.hasRole(msg.sender, role), "Access denied: You do
not have the required role");
        _;
    }

    function updateRBACAddress(address newRBACAddress) public
onlyRole("admin") {
        require(newRBACAddress != address(0), "Invalid RBAC address");
        rbac=IRBAC(newRBACAddress);
        emit RBACAddressUpdated(newRBACAddress);
    }
```

```
{
  "studentId": 12904,
  "specializationCode": "6B0605055",
  "degreeType": "Bachelor degree",
  "enrollYear": 2021,
  "graduateYear": 2025,
  "diplomNum": "BD12356",
  "diplomSeries": "BD",
  "protocolDate": "05.05.2025",
  "protocolNum": "01/05/25-65",
  "universityId": 32
}
```

```
    function createDiploma(
        uint256 universityId,
        uint256 studentId,
        bytes32 minutesNumber,
        string memory fio,
        bytes32 dateOfMinutes,
        bytes32 registrationNumber,
        bytes32 iacDiplomaNumber,
        bytes32 iacDiplomaSeries,
        bytes32 dateOfIssue
    ) public onlyRole ("issuer"){

        require(universityId != 0, "University ID cannot be empty");
        require(studentId != 0, "Student ID cannot be empty");
        require(minutesNumber != bytes32(0), "Minutes number cannot be
empty");
        require(bytes(fio).length > 0, "FIO cannot be empty");
        require(dateOfMinutes != bytes32(0), "Date of minutes cannot be
empty");
        require(registrationNumber != bytes32(0), "Registration number
cannot be empty");
        require(iacDiplomaNumber != bytes32(0), "IAC Diploma number
cannot be empty");
        require(iacDiplomaSeries != bytes32(0), "IAC Diploma series
cannot be empty");
        require(dateOfIssue != bytes32(0), "Date of issue cannot be
empty");
        require(!issuedDiplomas[universityId][studentId], "This student
has already issued Diploma");
        issuedDiplomas[universityId][studentId] = true;

        diplomas[studentId] = Diploma({
            universityId: universityId,
            studentId: studentId,
            minutesNumber: minutesNumber,
            dateOfIssue: dateOfIssue,
            registrationNumber: registrationNumber,
            fio: fio,
            year: getCurrentYear(),
            dateOfMinutes: dateOfMinutes,
            iacDiplomaSeries: iacDiplomaSeries,
            iacDiplomaNumber: iacDiplomaNumber,
            addedBy: msg.sender
        });

        emit DiplomaCreated(studentId, iacDiplomaNumber,
iacDiplomaSeries, msg.sender);
    }

    function getCurrentYear() internal view returns (uint256) {
        return (block.timestamp / 365 days) + 1970;
    }

    function getRBACAddress () public view returns (address) {
        return address(rbac);
    }

    function getDiploma(uint256 studentId) public view returns (Diploma
memory) {
        return diplomas[studentId];
    }
}
```

Figure 7. The DiplomaRegistry smart contract source code and a format of graduation data for storing in the data ledger.

From the perspective of verification services, the smart contract includes methods to retrieve the necessary information based on the student's and the university's unique identifiers. If needed, the data can also be queried using the transaction hash. The obtained data is then utilized to generate a digital diploma version on the DPHE web platform.

2.7.3. Smart Contract "RBAC"

RBAC—Role-Based Access Control for managing permissions to modify records in the data ledger (Figure 8).

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.10;

import "./IRBAC.sol";

contract RBAC is IRBAC{

    mapping(address => mapping(string => bool)) private roles;
    address public admin;

    event RoleGranted(address indexed account, string role);
    event RoleRevoked(address indexed account, string role);
    event AdminTransferred(address indexed from, address indexed to);


    constructor() {
        admin = msg.sender;
    }

    modifier onlyAdmin() {
        require(hasRole(msg.sender, "admin"), "Only admin can perform this action");
        _;
    }

    function grantRole(address account, string memory role) public override  onlyAdmin {
        require(account != address(0), "Invalid account address");
        require(!hasRole(account, role), "Account already has this role");
        require(
            keccak256(bytes(role)) != keccak256(bytes("admin")),
            "Admin role can only be changed via transferAdmin"
        );
        roles[account][role] = true;
        emit RoleGranted(account, role);
    }

    function revokeRole(address account, string memory role) public override  onlyAdmin {
        require(account != address(0), "Invalid account address");
        require(hasRole(account, role), "Account does not have this role");
        require(
            keccak256(bytes(role)) != keccak256(bytes("admin")),
            "Admin role can only be changed via transferAdmin"
        );
        roles[account][role] = false;
        emit RoleRevoked(account, role);
    }

    function hasRole(address account, string memory role) public view override returns (bool) {
        if (keccak256(bytes(role)) == keccak256(bytes("admin"))) {
            return account == admin;
        }
        return roles[account][role];
    }

    function transferAdmin(address from, address to) public override  onlyAdmin {
        require(from != address(0), "Invalid donor admin address");
        require(to != address(0), "Invalid recipient admin address");
        require(to != from, "Donor and recipient must be different accounts");
        require(msg.sender == from, "Caller must be the donor admin");
        require(to != admin, "Same address");
        require(hasRole(to, "admin"), "Recipient already has admin role");
        address oldAdmin = admin;
        admin = to;
        roles[oldAdmin]["admin"] = false;
        roles[to]["admin"] = true;
        emit AdminTransferred(from, to);
    }
}
```

Figure 8. The RBAC smart contract source code.

The RBAC interface establishes a modular contract that verifies whether a given blockchain address has the necessary role to perform specific actions. This component separates authorization logic from the core business logic of the TranscriptStorage and DiplomaRegistry smart contracts, thereby improving their reusability and maintainability.

Typically, the RBAC interface includes functions for role assignment, revocation, and verification, although these functions may be implemented in a dedicated contract. In practice, such an architecture enables decentralized governance bodies to manage access permissions without altering each application-level contract. The RBAC model is foundational in aligning blockchain-based platforms with institutional policies and native security models, serving as an infrastructure layer for managing sensitive academic operations in distributed systems.

To ensure a consistent and secure governance model, the contract enforces a single-administrator design. The admin role cannot be granted or revoked through generic role-management functions. Still, it can only be transferred explicitly via the transferAdmin operation, which requires authorization from the current administrator. This prevents the creation of multiple administrators, eliminates the risk of arbitrary privilege reassignment, and guarantees that the contract cannot be left without an active administrator. Such an approach provides predictable role management on-chain while delegating advanced security and recovery mechanisms (e.g., multi-signature wallets, custody solutions, or off-chain IAM policies) to the surrounding infrastructure.

The smart contract acts only as an executor of externally validated decisions, which reduces on-chain complexity and costs while providing flexibility to introduce new roles without contract modifications. This separation of responsibilities ensures consistency at the infrastructure level and aligns with the principle of optimizing on-chain computations.

To ensure a certain degree of fault tolerance, smart contracts are replicated across multiple nodes. If one node is compromised, other nodes can execute the transaction and advance the ledger to the next state. The term "replication" originates from this concept, referring to smart contracts that are duplicated across several nodes [31].

2.8. The Sequence of DPHE Participants' Actions

As illustrated in Figure 9, the smart contracts are owned by the Ministry of Science and Higher Education (MSHE), a certified trustworthy authority. To ensure fault tolerance, the contracts are replicated across all nodes. In the event of a node compromise, execution continues on the remaining nodes, preserving ledger consistency.

The diagram (Figure 9) illustrates the sequence of interactions among key platform participants, including students, university representatives, and verifiers (e.g., employers). It outlines essential steps such as university account provisioning, academic data registration and retrieval, and document verification procedures.

It is important to note that this scenario does not address the process for registering a university's blockchain node. Instead, it illustrates a typical connection flow for institutions using the central node provided by the Ministry of Science and Higher Education. The diagram emphasizes the platform's functionality for participants without a dedicated node.

Given the consortium structure of the blockchain network, the Proof of Authority (PoA) consensus algorithm was intentionally selected to prevent malicious nodes from gaining control over the network. The algorithm defines authoritative validators who can partake in the mining process and ensures that unauthorized validators cannot perform transactions for their advantage.

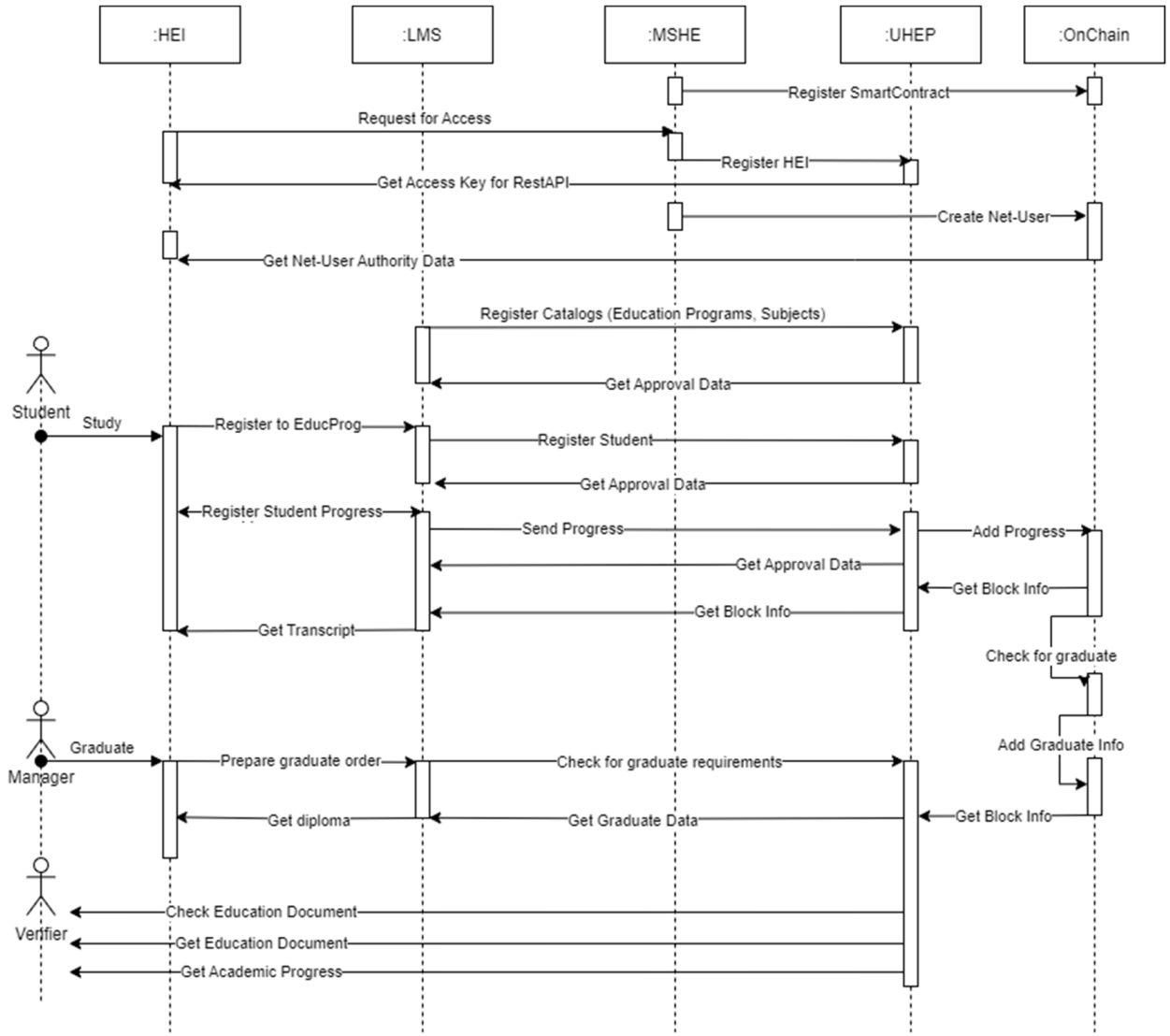


Figure 9. Sequence diagram of user operations within the DPHE platform.

A separate block diagram illustrates the university's operational algorithm when interacting with the platform. As shown in Figure 10, the registration of a university as a platform participant is designed to be inclusive and flexible. For universities that cannot deploy their blockchain node, the platform provides access to the OnChain API hosted by the central validator, maintained by the Ministry of Science and Higher Education (MSHE). Such institutions can upload their academic catalogs and student/graduate records to the centralized OffChain database and use the OnChain API to register critical data in the blockchain-based data ledger.

At the same time, universities equipped with adequate computational resources may request a server-side deployment utility from the platform registrar (MSHE). By following the documentation, these institutions can setup a local node and join the consortium network. This setup enables them to interact with the blockchain independently through their node and a locally deployed OnChain API service.

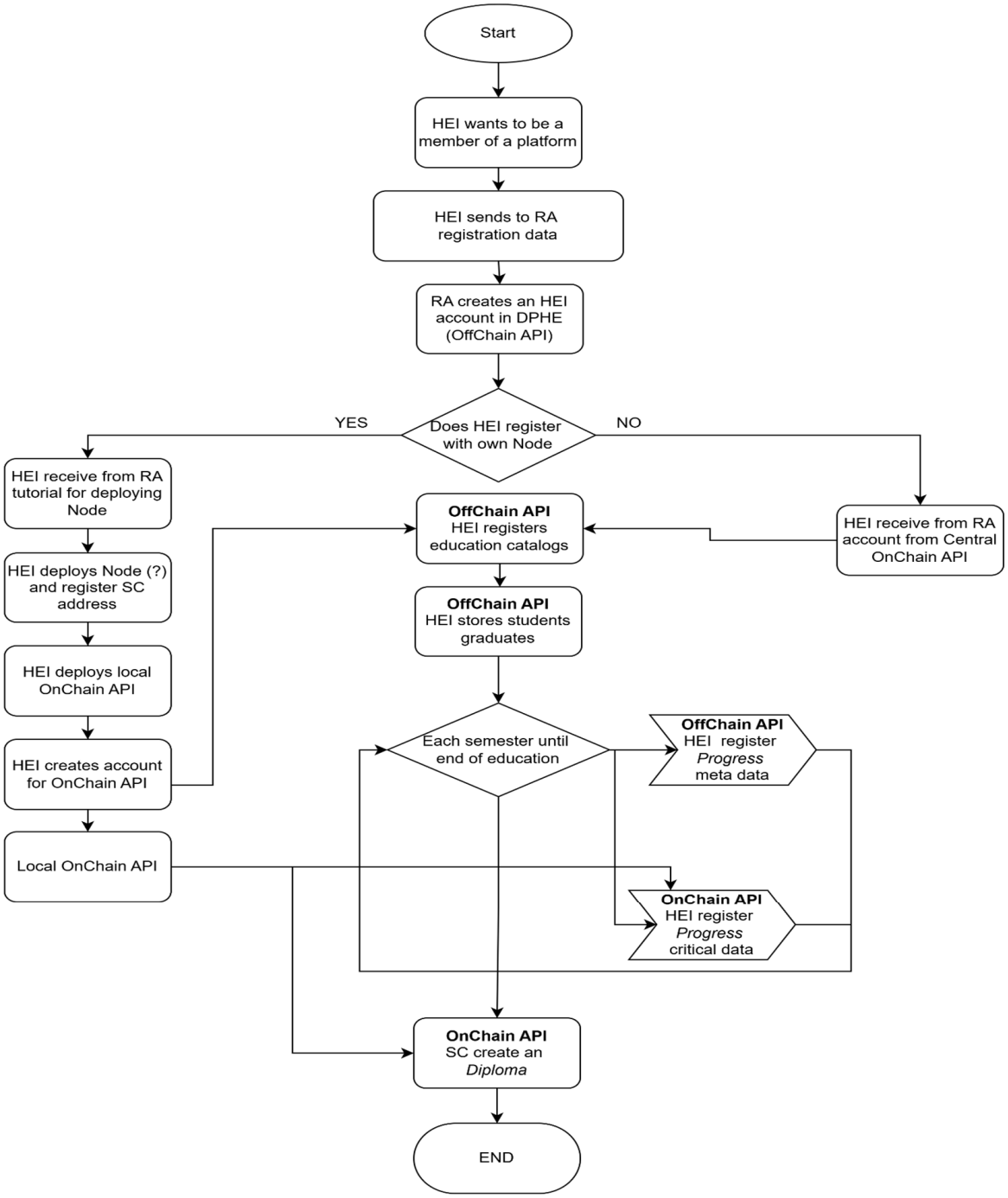


Figure 10. HEI interaction algorithm within the DPHE platform.

2.9. Data Verification Use Cases

Verification is a crucial platform process that enables users to conveniently validate data through digital services. The verification process can be categorized into three main groups:

Verification of education documents: This service is publicly accessible through the DPHE web platform. It enables users to retrieve graduation data by entering the document's registration details (diploma number, series, and the student's Individual Identification Number). Core data is extracted from the data ledger, while supplementary information is retrieved from the off-chain database. The service is designed for quick verification of paper diplomas and is particularly beneficial for employers to confirm the authenticity of submitted documents.

Retrieval of education documents: Available through the DPHE web platform and the OffChain API (with prior authorization), this service offers access to a digital version of a student or graduate diploma. The diploma features an embedded QR code that links to a blockchain explorer (e.g., Blockscout), allowing for the verification of the diploma's authenticity. This service benefits students, graduates, and universities. After registering on the platform, students and graduates can obtain verified digital certificates, usable for employment, university admissions, or in situations where paper originals are lost. Universities can utilize this service to verify the authenticity of incoming students' qualifications.

Retrieval of academic transcripts or diploma supplements: Also available through the DPHE web platform and the OffChain API (with prior authorization), this service provides access to a student's or graduate's digital transcript. Each record within the transcript can be verified via a link to the blockchain explorer (Blockscout), where one can confirm the authenticity of grades, ECTS credits, and the course completion date. This service is especially relevant for students, graduates, and universities, particularly when a diploma supplement is required or when applying to another institution under academic mobility programs. Platform-member universities can find the student and download their academic transcript to facilitate the transfer of credits.

3. Results

3.1. System Architecture and Performance Considerations

The illustrated architecture (Figure 11) outlines the core components and operational workflow of a blockchain-based transaction processing system, designed for high performance, security, and deterministic execution within a permissioned blockchain network. This system utilizes a microservices architecture and employs asynchronous processing mechanisms to ensure efficiency, scalability, and fault isolation.

At the application layer, the OnChain API, developed using the NestJS framework, serves as the main interface for interacting with the blockchain. It handles incoming requests and connects to a PostgreSQL database for persistent data storage. Meanwhile, a RabbitMQ-based transaction task queue is utilized to separate the API from the transaction execution logic, thus enabling non-blocking and asynchronous processing.

Upon submitting a transaction request, Worker #1 retrieves the task from RabbitMQ and initiates the transaction preparation process. This phase involves structuring transaction metadata, validating input data, and ensuring nonce consistency. A Redis-based Nonce Store is utilized to mitigate nonce-related transaction errors, such as "nonce too low" or "replacement transaction underpriced." This store maintains a mapping between blockchain addresses and their respective nonces, ensuring the correct sequencing of transactions in a concurrent environment.

Then, the prepared transaction is pushed into a BullMQ transaction dispatch queue, which uses Redis as its underlying in-memory storage system. This architecture enables high-speed queuing and delivery to the following worker processes. Worker #2, in charge of the dispatch phase, pulls transactions from BullMQ and sends them to the blockchain network for inclusion in the ledger.

Transactions are then held in the mempool, a temporary staging area, while awaiting block confirmation. The blockchain utilizes a Proof of Authority (PoA) consensus mechanism, wherein a pre-authorized set of validator nodes (Node 1, Node 2, and Node 3) is responsible for validating and finalizing blocks. The network is configured to mine new blocks at deterministic intervals (e.g., every 15 s), ensuring timeliness and consistency in transaction processing.

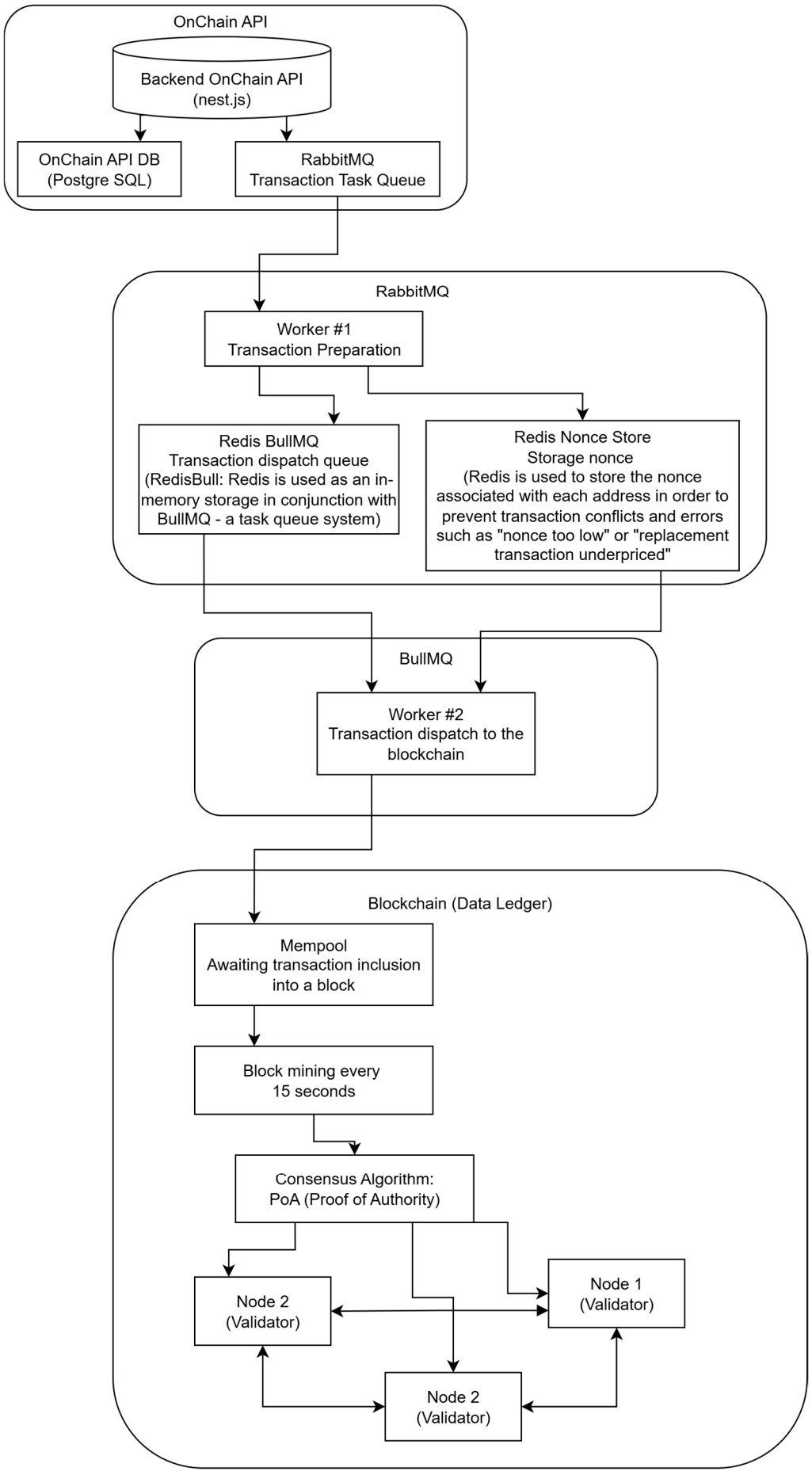


Figure 11. Detailed architectural flow of a blockchain-based transaction processing system.

This modular and layered architecture addresses several key requirements for enterprise and institutional blockchain applications. By combining asynchronous processing, secure nonce management, and deterministic consensus, the system demonstrates robustness against transaction collisions, enhances throughput, and ensures data integrity. The adoption of a Proof of Authority (PoA) further ensures trust and governance within the network, making this architecture particularly suitable for applications such as academic credential verification, governmental record-keeping, and regulated financial transactions.

The presented architecture illustrates a robust and scalable blockchain transaction processing system based on a microservices approach. It integrates asynchronous messaging via RabbitMQ and BullMQ, nonce conflict prevention through Redis, and a Proof of Authority (PoA) consensus mechanism for secure block validation. By separating transaction preparation and dispatch into distinct workers and leveraging both in-memory and persistent storage, the system ensures high throughput, fault tolerance, and deterministic execution—ideal for institutional applications such as credential verification and secure record management.

Predicting the deployment patterns or complexity of smart contracts on a public Ethereum blockchain is nearly impossible, as any participant can freely deploy contracts. This openness leads to challenges when estimating future performance or storage requirements. In contrast, on a consortium-based Ethereum blockchain, performance and storage requirements can be estimated with greater accuracy because of the following characteristics: participants are authorized nodes; smart contracts are relatively fixed, and the underlying business model determines their complexity; the transaction volume is limited by the specific enterprise use case's scale.

In Ethereum, the transaction execution time can be divided into two main components: the time spent by the Ethereum Virtual Machine (EVM) and the cost associated with updating the "World State" [32]. Similarly, storage overhead is primarily influenced by the size of the transaction and the changes made to the "World State" data. Consortium blockchains typically operate with predetermined and stable contracts, so the EVM execution time and transaction size remain relatively constant. Therefore, variations in performance and storage utilization are mainly driven by modifications to the "World State."

Each transaction executed through a smart contract ultimately alters the nodes of the World State tree. The Ethereum state maps a 160-bit address to the corresponding account state, which is stored in a data structure known as the State Trie. The state of an account corresponds to a leaf node, and its address forms a path from the root to that node. Smart contracts are treated as special types of accounts. Each contract account is associated with a separate data structure for persistent storage called the Storage Trie. This storage is also implemented using a Merkle Patricia Trie (MPT) [33].

Consequently, Ethereum's World State consists of two primary hierarchical layers: the Account Trie (upper layer) and the Storage Trie (lower layer). In the current Ethereum implementation, when a node is searched for or inserted along a path, the system executes the sha3() hashing function and performs read/write operations to LevelDB. These actions result in measurable time consumption, and the size of the new node corresponds to the incremental data stemming from the modification. Therefore, if the current height of a new leaf node can be predicted, the performance and storage implications can be inferred accordingly. When state changes occur within the MPT structure, transaction performance and storage increments can be projected by analyzing the modification time of the World State or by empirically testing business-specific smart contracts. This analysis focuses on the current height of the World State nodes within the MPT.

According to Devroye [34], the expected height of a Merkle Patricia Trie (MPT) can be calculated using the following formula:

$$H_n = \frac{2\sqrt{2log2n} + log2n}{4} \tag{1}$$

When the number of transactions reaches $\boldsymbol{n}$ and a new transaction is executed, the average time cost associated with modifying the Merkle Patricia Trie (MPT) is determined as follows:

$$T_{mpt}(n) = \frac{t}{2}\log(an) \tag{2}$$

where $\boldsymbol{a}$ is the number of contract deployments in the network, $\boldsymbol{n}$ is the transaction volume, $\boldsymbol{t}$ is the execution time for creating a node in the MTP tree (sha3() calculation and database access).

Ethereum uses LevelDB as its key–value storage database. Due to the discrete nature of hashing, the keys used for accessing the database are irregular. While LevelDB performs better for sequential read and write operations, its efficiency decreases with random key access patterns [26]. As a result, the access time $\boldsymbol{t}$ to LevelDB increases with the growth of the stored data volume. Experimental results also show that when $\boldsymbol{n}$ becomes sufficiently large, the value of $\boldsymbol{t}$ rises, and performance significantly degrades for specific data entries that are no longer cached in LevelDB.

Subsequently, for brilliant contract execution within the Ethereum Virtual Machine (EVM), after processing $\boldsymbol{n}$ transactions, the average execution time and the maximum transaction time are defined as follows:

$$T_{avg}(n) = T_{exec} + T_{mpt}(n) \tag{3}$$

where $\boldsymbol{T_{exec}}$ is the time to execute the contract on the virtual machine.

After the execution of n transactions, the storage increment resulting from the modification of the Merkle Patricia Tree (MPT) can be computed as follows:

$$S_{mpt}(n) = s_b\left(\frac{\log(an)}{2} - 1\right) + s_1 + s_a + s' \tag{4}$$

where $\boldsymbol{s_b}$ is the size of the filled branch, $\boldsymbol{s_1}$ is the size of the end node, $\boldsymbol{s_a}$ is the storage space occupied by the account state and $\boldsymbol{s'}$ is the sum of storage trees for each account. If we include here the size of the $\boldsymbol{S_t}$ transaction itself, then the final formula becomes

$$S_{sumavg}(n) = nS_t + \sum_{i=0}^{n} S_{mpt}(n) \tag{5}$$

The authors conduct experiments to validate the proposed performance prediction method. Three key metrics are collected during the experiment: the height of the State Trie in the Ethereum "World State," the transaction execution time, and the storage utilization after each transaction. The experimental setup involves the deployment of a single test node, with each transaction encapsulated in an individual block to eliminate the impact of Ethereum's caching mechanisms on performance measurements. A smart contract with minimal logic is utilized, and one million transactions are executed to ensure sufficient data for analysis.

Based on the empirical data obtained from the conducted experiment, the following conclusions were drawn:

The actual heights of the State Trie did not exceed the values predicted by Equation (1). Consequently, the predictive model's accuracy improves when the empirical relationship

between the LevelDB read time t and the volume of stored data is included, whether through experimental observations or theoretical estimation.

Under "one transaction per block," the observed storage growth in the testing environment closely matches the predicted values.

This study is significant for validating the choice of blockchain network architecture in our proposed solution—the Ethereum-based consortium blockchain. We acknowledge that HEIs are not profit-oriented organizations; their primary objective is delivering quality education rather than generating financial returns. Therefore, promoting digital transformation in HEIs by adopting public blockchain platforms (e.g., Ethereum, Bitcoin), which incur high operational and computational costs due to cryptocurrency incentives and infrastructure maintenance, is economically unfeasible.

In contrast, a consortium Ethereum network allows for accurate predictions of blockchain performance and storage utilization under clearly defined and controlled conditions. The ability to estimate transaction scale supports the design of smart contracts that minimize performance and storage overhead. As a result, optimal data distribution between the State Trie and Storage Trie can be achieved, enabling a performance–storage trade-off that significantly lowers operational costs.

3.2. The Tool for Monitoring and Auditing of Transactional Activity Within the Blockchain Network

As part of this study, the Blockscout system [35]—an open-source blockchain explorer designed for Ethereum Virtual Machine (EVM)-compatible networks—was integrated to enable the monitoring, analysis, and auditing of transactional activity within the private blockchain network. The selection of this tool was driven by its comprehensive feature set, adaptability to private networks, and support for modern interoperability standards through REST API and GraphQL interfaces.

Blockscout was deployed in a dedicated infrastructure environment consisting of an application server and a PostgreSQL database, which stores indexable data on transactions, blocks, and smart contracts. During configuration, indexing parameters were modified to enhance the processing speed of internal transactions and optimize performance under high write-throughput conditions. The contract source code verification feature was also enabled, allowing for more thorough auditing of executed operations.

The main features of Blockscout used in this research are (1) indexing all transactions on the network, including internal contract-to-contract calls; (2) tracking of user account states and balances; (3) automated verification and publication of smart contract source code; (4) run custom data queries through GraphQL to produce analytical insights on network activity across different intervals (Table 2 below summarizes the advantages and limitations of using Blockscout in private blockchain network).

Table 2. Advantages and Limitations of Using Blockscout in Private Blockchain Networks.

No.	Aspect	Advantages	Limitations
1	Openness	Fully open-source codebase with customization capabilities	Requires technical resources for setup and maintenance
2	Integration	Support for REST API and GraphQL enables seamless integration	Limited support for non-standard protocols
3	Analytics	Indexes internal transactions and contract events	High database load under large-scale data volumes
4	User Interface	Intuitive and user-friendly web interface	Limited UI customization options without additional development

Table 2. *Cont.*

No.	Aspect	Advantages	Limitations
5	Private Network Support	Adaptable for private and consortium blockchain environments	Potential scalability challenges in large-scale networks
6	Performance	Efficient in networks with moderate transactional activity	Performance degradation under sudden spikes in transaction volume

3.3. The Key Experiment Parameters

As demonstrated by the data presented, using Blockscout in private blockchain networks offers high functionality, transparency, and integration flexibility. The tool's open-source architecture and modern interface support effective adaptation to various research and enterprise scenarios. At the same time, the identified limitations—such as its resource-intensive nature and the need for technical optimization under high transactional loads—highlight the importance of thoroughly assessing infrastructural capabilities before deploying Blockscout in large-scale systems. Analyzing its strengths and weaknesses confirms the feasibility of applying Blockscout in environments where operational transparency and real-time data accessibility are prioritized.

Using Blockscout's analytical capabilities, we identified key performance indicators (KPIs) relevant to evaluating the efficiency of the data ledger and blockchain configuration used in our experiment. These indicators can be broadly categorized into configurable input parameters, empirical metrics collected during experimentation, and calculated parameters for forecasting future data ledger requirements. (Table 3 below presents the experimental values of indicators with our notes).

Table 3. Blockchain Performance Metrics for the Data-Ledger System.

No		Indicator	Unit	Value	Note
1	B_{block}	Total number of blocks (at the time of the review)	block	322,994	
	B_{block}^{Full}	Total number of non-empty blocks (at the time of the review)	block	30,453	
2	TX	Total number of transactions (at the time of the review)	transactions	294,698	
3	TX_{block}^{Max}	Maximum number of transactions per block	transactions	135	
4	AVG_{tx}^{Block}	Average number of transactions per block	transactions	0.912	considering only non-empty blocks
5	AVG_{gas}^{Tx}	Average gas consumed per transaction	gas	241,352.94	
6	AVG_{eth}^{Tx}	Average ether consumed per transaction	eth	0.00024141315162027664	
7	AVG_{gas}^{Block}	Average gas consumed per block	gas	2,334,916.551441139781	
8	AVG_{eth}^{Block}	Average ether consumed per block	eth	0.0023354990798568325	
9	AVG_{weight}^{Tx}	Average size of a single transaction	byte	292.081890613442	considering only transactions without fund transfers

Table 3. *Cont.*

No		Indicator	Unit	Value	Note
10	W_{store}	Total storage consumed (at the time of the review)	Gb	1.2	
11	T_{block}	Time taken to create one block	second	15 s	The genesis block specifies the time for the formation of the next block
12	N	Number of validator nodes	node	2	
13	N_{miner}	Number of miner nodes	node	1	
14	N_{sc}	Number of deployed smart contracts	smart contract	3	described in the "Smart Contracts" section
15	$AVG_{weight}{}^{Block}$[nonempty]	Average size (in KB) of a non-empty block	Kb	4.68	
16	$AVG_{weight}{}^{Block}$[empty]	Average size (in KB) of an empty block	Kb	0.60	
17	G_{block}	Gas limit for block creation.	gas	30 mln. gas	The Data-ledger is used only at the end of academic periods in HEI. To prevent the generation of blocks with few transactions, the gas limit was increased.
18	$AVG_{tx}{}^{Block}$	Average number of transactions that can fit into a block.	transactions	$G_{block}/AVG_{gas}{}^{Tx}$~124	This parameter is calculated based on the gas limit per block and average gas consumption per transaction. The importance of this parameter is further discussed in the section "Impact of the Number of Transactions in a Block"
	AVG_{TPS}	Average transactions per second (TPS)	transactions	$AVG_{tx}{}^{Block}/T_{block}$~8.26	For Ethereum, the average TPS should be $\geq$100 TPS, with a block generated every second. In our case, the block generation frequency is 15 s, resulting in a TPS~8.26, which is acceptable.
	$AVG_t{}^{Tx}$ [1 mln]	Average time to process 1 million transactions	hour	1,000,000/AVG_{TPS} = 121,065 s~33 h	Average time to process 1 million transactions, which is approximately 33 h
	$AVG_{weight}{}^{Tx}$ [1 mln]	Average storage consumption for 1 million transactions	Mb	$AVG_{weight}{}^{Tx}$ × 1 mln~300 Mb	Average storage consumption for 1 million transactions, which is approximately 300 MB.

3.3.1. Configurable Input Parameters

T_{block}—Block creation time. This parameter was defined during the creation of the genesis block.
N—Number of nodes in the network.
N_{miner}—Number of mining nodes.
N_{sc}—Number of deployed smart contracts.
G_{block}—Maximum gas limit per block.

3.3.2. Experimentally Measured Parameters

B_{block}—Total number of blocks produced.
B_{block}^{Full}—Total number of non-empty blocks.
TX—Total number of transactions (excluding value transfers, i.e., contract function calls).
TX_{block}^{Max}—Maximum number of transactions recorded in a single block.
AVG_{weight}^{Block}[empty]—Average size of an empty block.
AVG_{weight}^{Block}[nonempty]—Average size of a non-empty block.
W_{store}—Total storage volume used.
AVG_{gas}^{Tx}—Average gas consumed per transaction.
AVG_{eth}^{Block}—Average ether cost per block.
AVG_{weight}^{Tx}—Average size of a single transaction.

3.3.3. Derived and Forecasting Parameters (Calculated Based on Collected Metrics)

AVG_{tx}^{Block}—Average number of transactions per block.
AVG_{TPS}—Average number of transactions processed per second by the network.

The calculated parameters enable us to anticipate resource demands, such as the registrar—administrator's early replenishment of miner wallets and allocation of additional storage for the data ledger.

3.4. Key Performance Indicators for Network, Dapp and Infrastructure

3.4.1. Comparing Indicator Results with Benchmark Values

In addition to the parameters outlined above, we compared our experimental results with benchmark indicators commonly used in blockchain technology to assess the network's performance, underlying infrastructure, and decentralized applications. The comparative analysis revealed that not all optimal or recommended values were achieved during the experiments. This discrepancy can be attributed to the platform's specific operational characteristics and the predefined input parameters that govern its operation.

The analysis of the observed performance indicators, as summarized in Table 4, demonstrates a generally stable and operationally viable blockchain environment, tailored for academic credential management within a private network. Several network-level metrics are utilized to approach or meet the target thresholds. For instance, block propagation time was measured at approximately 856 milliseconds, aligning with optimal expectations ($\leq$1 s). The fork/error rate remained at 0%, thanks to the single-validator configuration, which ensures deterministic finality without chain splits.

However, block finalization and transaction response times notably exceeded typical enterprise-grade application benchmarks. The 15 s interval, predetermined in the genesis block for block creation, directly impacted transaction throughput (TPS), limiting it to 8.26 transactions per second, which is well below the $\geq$100 TPS target considered optimal for high-load systems. This is acceptable within the current use case, where transaction volumes are low and periodic, but it may pose scalability challenges with increased usage.

Table 4. Key Performance Indicators for Blockchain Network, DApp, and Infrastructure.

Network	**Node Availability**	**% of Time Nodes Are Online**	**≥99.9%**	**99%**
	Block Finalization Time	Time to confirm a block on the network	≤5 s (Clique consensus)	15 s (by algorithm)
	Block Propagation Time	Average time for a block to reach all nodes	≤1 s	856.302 ms (logged by the blockchain system)
	Blockchain Size Growth	Growth rate in GB per month	≤5% per month	See details in the section "OnChain Storage Growth Dynamics"
	Fork/Error Rate	% of chain mismatches	≤0.1%	0% (validator = 1)
	TPS (Throughput)	Transactions per second	≥100 TPS (for enterprise-grade systems)	8.26 tx/s
	Validator Participation Rate	Share of validators actively involved in consensus	≥80%	100%
Application (DApp)	Transaction Response Time	Time from submission to confirmation	≤3 s	15 s (from entering mempool)
	Gas Utilization	Average Gas Used/Gas Limit	50–70% (optimal)	80–90%—see details in "Transaction Gas Limit" section
	Smart Contract Code Coverage	% of functions actually invoked	≥80%	90%
	DApp Interface Uptime	Availability of frontend and APIs	≥99.9%	99.9%
Infrastructure	Monitoring and Alerting	Full coverage of critical network and API components	100% coverage	100% (Zabbix, Blockscout)
	Recovery Time After Failure	Average time to recover network/nodes	≤10 min	5–10 min (after sync initiation)
Strategic Metrics	Stakeholder Engagement	Share of active participants in consensus	≥80% of validators/nodes	100%
	System Upgrade Complexity	Time required for system/network/contract updates including testing	≤2 h	1–2 h
Category	Metric	Description	Target/Acceptable Value	Actual Value

From an application performance standpoint, the smart contract code coverage (90%) and gas utilization (80–90%) indicate a well-optimized deployment. However, the latter slightly exceeds the recommended 50–70% range, suggesting potential for congestion in more intensive usage scenarios. The interface uptime and monitoring coverage remained at 99.9% and 100%, respectively, demonstrating a mature infrastructure setup with strong observability through tools like Zabbix and Blockscout.

Strategically, the system demonstrated full validator engagement (100%) and maintained a low-complexity upgrade cycle, with updates completed in 1–2 h. This highlights the environment's maintainability and operational readiness, supporting its use in semi-critical academic and cross-institutional scenarios.

In conclusion, the experimental results affirm the system's viability for its intended purpose, highlighting performance bottlenecks related to design-time parameters. These factors should be considered in future iterations that require greater scalability or responsiveness.

3.4.2. On-Chain Storage Growth Dynamic

Table 5 shows the monthly growth dynamics of blockchain storage, differentiating between empty and non-empty block sizes. Data from January and February are excluded from percentage growth calculations due to the initial network bootstrapping phase. A significant increase in storage consumption is noted in March (+43.2%) and April (+72.63%), signaling an upward trend in data usage as transaction activity intensifies.

Table 5. Blockchain Storage Growth Dynamic.

Month	Size of Empty Blocks (Bytes)	Size of Non-Empty Blocks (Bytes)	Total Size (Bytes)	Growth (%)
January	627	0		—
February	21,295,537	54,099,233	75,394,770	—
March	72,386,481	35,575,403	107,961,884	(107,961,884 − 75,394,770)/75,394,770 ≈ +43.2%
April	93,350,123	93,019,267	186,369,390	(186,369,390 − 107,961,884)/107,961,884 ≈ +72.63%
May	105,205,886	97,761,186	202,967,072	(202,967,072 − 186,369,390)/186,369,390 ≈ +8.91%
June	64,098,297	0	64,098,297	

The reported figures are split into the size contributions from empty blocks—typically produced to maintain the block generation schedule—and non-empty blocks, which contain actual transaction data. The combined monthly storage increments reflect both network protocol characteristics (e.g., block time, block size limit) and user behavior (e.g., transaction frequency, contract interaction density).

This progression underscores the importance of proactive storage resource planning in permissioned blockchain networks. The predictable yet accumulative nature of block data necessitates efficient indexing, compression, and archival strategies to ensure long-term operability, particularly in domains such as higher education, where transaction frequency may correlate with academic cycles.

3.4.3. Impact of Transaction Volume per Block

The volume of transactions stored within a single blockchain block is crucial for determining the system's overall performance, scalability, and responsiveness. A high transaction density per block can influence not only execution efficiency but also the interaction between system components and the retrieval of specific transaction records.

One of the immediate consequences of an increased transaction volume is the rise in transaction processing latency. Since each transaction in a block must be validated and executed independently, the overall computational burden grows proportionally with the number of transactions. This added complexity can slow down block verification, leading to delays in block propagation across the network and reduced system responsiveness, particularly in high-throughput environments.

Besides processing latency, a high transaction load affects transaction search efficiency. Specifically, when queries target certain attributes like transaction hashes, performance can decline if suitable indexing mechanisms are not in place. Without indexing, search

operations within a block may involve linear time complexity, causing delays as each transaction must be scanned sequentially. This problem becomes more noticeable as the number of transactions per block increases.

Scalability is another area negatively affected by large block sizes. Higher transaction volumes require greater computational resources to ensure timely validation and execution. At the same time, storage needs increase, as blocks with dense transaction payloads take up more disk space, putting further pressure on nodes throughout the network. The necessity to store the entire blockchain state worsens memory and storage limitations, especially for nodes operating with limited resources.

Furthermore, the propagation of large blocks across the network may be delayed due to increased data volume, especially in consensus mechanisms like Proof of Work (PoW) or Proof of Stake (PoS), where every node must verify each block. This latency can impede synchronization among full nodes, ultimately impacting consensus finality and transaction finalization timeframes.

State rollback and system recovery also become increasingly complex when managing large transaction sets per block. In the event of an error or failure within a transaction, the rollback mechanism must consider numerous state transitions, making error handling and system restoration more computationally intensive and error-prone.

To address these challenges, several optimization strategies can be employed. Transaction indexing can significantly enhance query efficiency, particularly by transaction hash or relevant metadata. Moreover, distributing transaction loads across multiple smaller blocks may improve scalability and reduce computational strain, depending on the consensus protocol used. Storage optimization techniques, such as data compression, can reduce disk usage, while parallel processing of transactions, if supported by the blockchain architecture, can speed up block validation and boost system throughput.

In summary, storing numerous transactions in a block can negatively affect search and processing efficiency, as well as overall system performance if not properly optimized. However, these effects can be alleviated through effective indexing, storage optimization, and architectural enhancements.

3.4.4. Transaction Gas Limit

Efficient management of transaction gas consumption is essential for optimizing the performance and reliability of a blockchain-based system. One key strategy involves limiting the gas usage of individual transactions—to no more than 80% of the estimated limit, for example. This not only prevents resource exhaustion but also helps maintain predictable system behavior.

Three primary control layers are commonly used:

- Sender-Level Control (Off-chain): The sender manually restricts the gas limit during transaction formation, usually setting it to 80% of the estimated gas. Although this method provides strict control, it may result in transaction failures if the actual gas consumption exceeds the set threshold.
- Infrastructure-Level Control (Middleware/API): Intermediate layers such as RPC endpoints or middleware can dynamically analyze and estimate gas values and block transactions considered too resource-intensive. This enables real-time filtering and enforces system-wide gas usage policies.
- Monitoring and Post-Factum Analysis: Tools like Blockscout can perform SQL-based analytics on recent transactions to evaluate gas consumption patterns. Although this method does not prevent overuse in real time, it provides valuable insights that can inform infrastructure tuning and smart contract optimization.

Combining these approaches creates a comprehensive gas management strategy, ensuring that transaction execution stays within acceptable resource limits. This enhances system stability, improves user experience, and supports the scalability of the underlying decentralized application.

4. Discussion and Conclusions

The proposed consortium-based Ethereum architecture tackles the critical challenges faced by centralized systems, such as UPHE, specifically vulnerability to tampering, opaque verification mechanisms, and the risks of system downtime. The implementation of smart contracts (TranscriptStorage and DiplomaRegistry) ensures immutable, auditable, and programmable credentialing workflows, while the integration of PoA consensus and node replication boosts resilience. Performance benchmarks, including consistent block propagation time and stable TPS under light load conditions, validate the model's operational viability for academic use cases. These findings reinforce the argument that blockchain can be effectively tailored for public-sector systems, where decentralization must coexist with institutional oversight.

The present conceptual research makes a significant contribution to the study of blockchain technology from a holistic, cross-sector, and interdisciplinary perspective. In a context marked by rapid technological evolution and a surge in academic publications on blockchain, this work addresses the need to synthesize current advances and relate them to the practical challenges faced by productive sectors and institutional settings [24,36]. The proposed model presents a useful approach for utilizing technology to achieve key targets of SDG 4, specifically by ensuring the recognition of measurable learning outcomes through secure and accessible verification mechanisms.

While the proposed model shows technical feasibility and benefits for institutions, its success also depends on how key stakeholders view it. Higher education institutions might face resistance to change from staff and administrators who are unfamiliar with blockchain systems. The digital divide also creates a challenge, as not all institutions or users have equal access to advanced digital tools. Additionally, there may be initial distrust of the system, considering its transparency and reliability. Privacy and ethics concerns must be addressed, since academic data is highly sensitive and higher education institutions will act as the primary validators of the model. Recognizing these issues is crucial for building acceptance and ensuring that the model is both technically effective and socially responsible.

Author Contributions: Conceptualization, Y.K. and F.L.-C.; methodology, Y.K. and A.A.S.; software, Y.K. and G.M.; validation, F.L.-C., G.M. and A.A.S.; formal analysis, Y.K.; investigation, O.U.; resources, G.M.; data curation, A.S.; writing—original draft preparation, all authors contributed; writing—review and editing, all authors contributed; visualization, Y.K., O.U. and F.L.-C.; supervision, G.M. and A.A.S.; project administration, F.L.-C. and G.M.; funding acquisition, G.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was carried out within the framework of the project BR24993014-OT-24 "The development of an intelligent anti-corruption system for information protection and validation of the results of academic achievements and official documents of students and graduates of the universities of the Republic of Kazakhstan," which is being implemented at the Institute of Information and Computational Technologies, Almaty, Kazakhstan.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: During the preparation of this manuscript/study, the authors used a licensed version of a Grammarly application for the purposes of improving the language. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HEI	Higher education institute
LMS	Learning management system
GDPR	General Data Protection Regulation
UPHE	Unified Platform of Higher Education
ESG	Environmental, Social, Governance
ID	Identity
ECTS	European Credit Transfer and Accumulation System
PoA	Proof-of-Authority
PoS	Proof-of-Stack
PoW	Proof-of-Work
EGDI	E-Government Development Index
IMD	Institute for Management Development
MSHE	Ministry of Science and Higher Education
EDS	Electronic digital signature
IIN	Individual identification number
TPS	Transactions per second
IBFT	Instanbul Byzantine Fault Tolerance
DPHE	Decentralized platform for higher education
JSON	JavaScript Object Notation
RBAC	Role-based Access control
EVM	Ethereum Virtual Machine
MPT	Merkle Patricia Trie
API	Application Programming Interface

Glossary of Terms

Bologna Process	A European higher education reform initiative aimed at standardizing degree structures, quality assurance, and mobility across universities.
Sharding	A scalability technique in blockchain where the network is partitioned into smaller subsets (shards), each processing a portion of transactions in parallel.
PoS (Proof of Stake)	A consensus mechanism where validators are selected to propose and confirm blocks based on the amount of cryptocurrency they lock as stake.
PoW (Proof of Work)	A consensus mechanism in which miners solve computational puzzles to validate transactions and add new blocks to the blockchain.
Smart contracts	Self-executing programs stored on a blockchain that automatically enforce predefined rules and agreements.
Solidity	A high-level programming language for writing smart contracts on Ethereum and Ethereum-compatible blockchains.
Truffle	A development framework for Ethereum that provides tools for smart contract compilation, deployment, and testing.

Geth (Go-Ethereum)	The official Ethereum client written in Go, used to run nodes, deploy contracts, and interact with the Ethereum blockchain.
Besu	An open-source Ethereum client written in Java, designed for both public and permissioned blockchain networks.
Istanbul Byzantine Fault Tolerance (IBFT)	A consensus protocol used in permissioned blockchains, providing fast finality by requiring validator agreement despite faulty or malicious nodes.
RabbitMQ	A widely used open-source message broker that enables applications to communicate via messaging queues.
Redis	An in-memory key–value database often used for caching, real-time analytics, and message brokering.
BullMQ	A Node.js library for managing distributed job queues, built on top of Redis.
GraphQL	A query language and runtime for APIs that allows clients to request only the data they need in a flexible and efficient way.
PostgreSQL	An advanced open-source relational database system known for reliability, extensibility, and SQL compliance.

References

1. Parmentola, A.; Petrillo, A.; Tutore, I.; De Felice, F. Is blockchain able to enhance environmental sustainability? A systematic review and research agenda from the perspective of Sustainable Development Goals (SDGs). *Bus. Strategy Environ.* **2022**, *31*, 194–217. [CrossRef]
2. Glavanits, J. Sustainable public spending through blockchain. *Eur. J. Sustain. Dev.* **2020**, *9*, 317–327. [CrossRef]
3. Choi, E.; Choi, Y.; Park, N. Blockchain-centered educational program embodies and advances 2030 sustainable development goals. *Sustainability* **2022**, *14*, 3761. [CrossRef]
4. Iansiti, M.; Lakhani, K.R. The truth about blockchain. *Harv. Bus. Rev.* **2017**, *95*, 118–127.
5. San, A.M.; Chotikakamthorn, N.; Sathitwiriyawong, C. Blockchain-based learning credential verification system with recipient privacy control. In Proceedings of the 2019 IEEE International Conference on Engineering, Technology and Education (TALE), Yogyakarta, Indonesia, 10–13 December 2019; IEEE: New York, NY, USA, 2019; pp. 1–5. [CrossRef]
6. Rhemananda, H.; Simbolon, D.R.; Fachrunnisa, O. Blockchain technology to support employee recruitment and selection in industrial revolution 4.0. In *Proceedings of International Conference on Smart Computing and Cyber Security*; Pattnaik, P.K., Sain, M., Al-Absi, A.A., Kumar, P., Eds.; Springer: Singapore, 2021; pp. 305–311. [CrossRef]
7. Alotaibi, E.M.; Issa, H.; Codesso, M. Blockchain-based conceptual model for enhanced transparency in government records: A design science research approach. *Int. J. Inf. Manag. Data Insights* **2025**, *5*, 100304. [CrossRef]
8. Dias, P.; Gonçalves, H.; Silva, F.; Duque, J.; Martins, J.; Godinho, A. Blockchain Technologies: A scrutiny into Hyperledger Fabric for Higher Educational Institutions. *Procedia Comput. Sci.* **2024**, *237*, 213–220. [CrossRef]
9. Available online: https://epvo.kz/#/statistics/opvo (accessed on 10 May 2025).
10. Baniata, H.; Kertesz, A. Prifob: A privacy-aware fog-enhanced blockchain-based system for global accreditation and credential verification. *J. Netw. Comput. Appl.* **2022**, *205*, 103440. [CrossRef]
11. Qu, X.; Yang, Z.; Chen, Z.; Sun, G. A consent-aware electronic medical records sharing method based on blockchain. *Comput. Stand. Interfaces* **2025**, *92*, 103902. [CrossRef]
12. Zhao, L.; Guo, D.; Luo, L.; Shen, Y.; Ren, B.; Zhu, S.; Yang, F. Concordit: A credit-based incentive mechanism for permissioned redactable blockchain. *Comput. Netw.* **2024**, *255*, 110848. [CrossRef]
13. Ali, M.S.E.; Agarwal, R.; Afzal, M. Block Chain for a Sustainable Future: Paving the Way towards Achieving Sustainable Goals. *J. Inform. Educ. Res.* **2024**, *4*, 22–32.
14. Jiang, S.; Jakobsen, K.; Bueie, J.; Li, J.; Haro, P.H. A tertiary review on blockchain and sustainability with focus on Sustainable Development Goals. *IEEE Access* **2022**, *10*, 114975–115006. [CrossRef]
15. United Nations. Kazakhstan. 2024. Available online: https://publicadministration.un.org/egovkb/en-us/Data/Country-Information/id/87-Kazakhstan (accessed on 10 July 2025).
16. Gamal, A.E.; Mammen, J.; Prabhakar, B.; Shah, D. Throughput-delay trade-off in wireless networks. In Proceedings of the IEEE INFOCOM 2004, Hong Kong, China, 7–11 March 2004; Volume 1.
17. Gupta, P.; Kumar, P.R. The capacity of wireless networks. *IEEE Trans. Inf. Theory* **2000**, *46*, 388–404. [CrossRef]
18. Gottlieb, M.; Deutsch, C.; Hoops, F.; Pongratz, H.; Krcmar, H. Expedition to the Blockchain Application Potential for Higher Education Institutions. *Blockchain Res. Appl.* **2024**, *5*, 100203. [CrossRef]

19. Cao, S.F.; Johnson, H.; Tulloch, A. Exploring blockchain-based traceability for food supply chain sustainability: Towards a better way of sustainability communication with consumers. *Procedia Comput. Sci.* **2023**, *217*, 1437–1445. [CrossRef]
20. Lepore, C.; Ceria, M.; Visconti, A.; Rao, U.P.; Shah, K.A.; Zanolini, L. A Survey on Blockchain Consensus with a Performance Comparison of PoW, PoS and Pure PoS. *Mathematics* **2020**, *8*, 1782. [CrossRef]
21. Croman, K.; Decker, C.; Eyal, I.; Gencer, A.E.; Juels, A.; Kosba, A.; Miller, A.; Saxena, P.; Shi, E.; Sirer, E.G.; et al. On scaling decentralized blockchains. In Proceedings of the International Conference on Financial Cryptography and Data Security, Christ Church, Barbados, 22–26 February 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 106–125.
22. Li, S.; Yu, M.; Yang, C.S.; Avestimehr, A.S.; Kannan, S.; Viswanath, P. Polyshard: Coded sharding achieves linearly scaling efficiency and security simultaneously. *IEEE Trans. Inf. Forensics Secur.* **2020**, *16*, 249–261. [CrossRef]
23. Bondi, A.B. Characteristics of scalability and their impact on performance. In Proceedings of the 2nd International Workshop on Software and Performance, Ottawa, ON, Canada, 17–20 September 2000; ACM: New York, NY, USA, 2000; pp. 195–203.
24. Yli-Huumo, J.; Ko, D.; Choi, S.; Park, S.; Smolander, K. Where is current research on blockchain technology-A systematic review. *PLoS ONE* **2016**, *11*, e0163477. [CrossRef]
25. Brewer, E.A. Towards Robust Distributed Systems (Invited Talk in ACM Symposium on the Principles of Distributed Computing (PODC)). 2000. Available online: https://dl.acm.org/doi/10.1145/343477.343502 (accessed on 1 October 2020).
26. Vukolić, M. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In Proceedings of the International Workshop on Open Problems in Network Security, Zurich, Switzerland, 29 October 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 112–125.
27. Buterin, V. A Next-Generation Smart Contract and Decentralized Application Platform, Ethereum White Paper. Available online: https://ethereum.org/en/whitepaper (accessed on 15 May 2025).
28. Koshiry, A.E.; Eliwa, E.; El-Hafeez, T.A.; Shams, M.Y. Unlocking the power of blockchain in education: An overview of innovations and outcomes. *Blockchain Res. Appl.* **2023**, *4*, 100165. [CrossRef]
29. *ISO/IEC 27001:2022*; Information Security, Cybersecurity and Privacy Protection—Information Security Management Systems—Requirements. ISO: Geneva, Switzerland, 2022.
30. Schneider, F.B. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Comput. Surv. (CSUR)* **1990**, *22*, 299–319. [CrossRef]
31. Schneider, F.B. Replication management using the state-machine approach. *Distrib. Syst.* **1993**, *2*, 169–198.
32. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
33. Patricia Tree. 2022. Available online: https://github.com/ethereum/eth-trie.rs (accessed on 10 October 2025).
34. Devroye, L. Universal asymptotics for random tries and PATRICIA trees. *Algorithmica* **2005**, *42*, 11. [CrossRef]
35. Blockscout. Blockscout Blockchain Explorer. Available online: https://blockscout.com (accessed on 28 April 2025).
36. Casino, F.; Dasaklis, T.K.; Patsakis, C. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telemat. Inform.* **2019**, *36*, 55–81. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.