Blockchain-Based University Record Management System

Project Proposal

Submitted by

Quan Anh Nguyen (1221561614)

Jett Bauman (1227174054)

Soumya Katagihalli (1237200382)

Aditya Sahu (1237367081)

Under the Guidance of

Faculty, Department of Computer Science and Engineering

Fall 2025

Arizona State University

Contents

0.1. Introduction

A transformational era has emerged in recent years with the rapid advancements in distributed computing, data integrity frameworks, and blockchain technology. As universities increasingly transition toward digital ecosystems, the management of academic records—transcripts, credentials, and certifications—has become both a technical and trust-critical challenge. Traditional centralized database systems, while efficient in data storage and retrieval, remain susceptible to unauthorized modifications, single points of failure, and limited verifiability across institutional boundaries. The demand for transparency, auditability, and tamper-proof record management has never been more pressing in the academic domain.

The intersection of distributed database systems and blockchain technology presents a new paradigm for secure and verifiable data governance in higher education. Distributed databases provide scalability and performance across multiple nodes, while blockchain introduces immutability, decentralized trust, and cryptographic integrity. Together, these technologies can establish an ecosystem where every academic transaction—from grade entry to degree issuance—is recorded in an immutable ledger, verifiable by all stakeholders without reliance on a central authority.

Universities today handle vast amounts of sensitive data across diverse departments, institutions, and accreditation agencies. According to recent reports, higher education institutions have experienced a surge in data breaches, with over $650 million in annual losses related to compromised student information. These incidents underline the urgent need for a resilient and transparent system capable of ensuring the authenticity of student records while maintaining privacy and compliance. By leveraging blockchain's distributed consensus and cryptographic hashing, institutions can eliminate forgery, prevent tampering, and enable real-time cross-verification of credentials.

This project aims to design and prototype a **Blockchain-Based University Record Management System**, integrating the strengths of distributed databases with blockchain's decentralized ledger principles. The proposed system will store student academic data in a distributed environment, while cryptographic hashes of transcripts and credentials will be securely anchored on a blockchain. This approach ensures that any modification to the underlying data becomes instantly detectable, preserving both the integrity and traceability of academic records.

The overarching objective of this research is to establish a transparent, tamper-resistant

framework for academic record management that fosters trust among universities, students, and employers. By developing and evaluating this prototype, we intend to demonstrate how blockchain technology can reinforce data verifiability, enhance institutional collaboration, and pave the way for a new era of digital academic integrity. Ultimately, this work envisions a future where educational credentials are not only digitally accessible but also cryptographically verifiable—ushering in a paradigm shift toward trustworthy, decentralized record systems in higher education.

0.1.1 Motivation

Universities manage sensitive academic records—grades, transcripts, and credentials—that must remain authentic, auditable, and accessible across departments and institutions. Conventional centralized and even distributed databases offer performance and availability, but they still depend on trusted administrators and are vulnerable to tampering, insider threats, and single points of failure. This trust gap undermines credential verification for employers and accreditation bodies, and it raises compliance risks for institutions handling large volumes of personally identifiable information.

Blockchain technology addresses these weaknesses by anchoring cryptographic hashes of records on an append-only, consensus-governed ledger. In a hybrid design, operational data stays in a high-performance distributed database while immutable proofs (hashes, timestamps, transaction IDs) are committed on-chain. Any unauthorized change to an off-chain transcript becomes immediately detectable by re-hashing and matching against the ledger, enabling verifiable integrity without exposing raw student data.

The project's goal is therefore to prototype a blockchain-backed university record system that delivers tamper-evidence, transparent provenance, and cross-institution verification. By evaluating latency, throughput, and fault tolerance in a multi-node setup, we aim to demonstrate that a pragmatic database–blockchain integration can preserve performance while materially strengthening trust, auditability, and interoperability for academic credentials.

0.1.2 Background

Academic record systems have evolved significantly over the past few decades, transitioning from paper-based archives to highly automated digital infrastructures. Early university databases followed a centralized client–server model, where all student records were stored and

controlled by a single administrative entity. While this model simplified management, it created dependency on trusted administrators and introduced single points of failure. As educational data volumes grew, institutions adopted distributed database systems to improve performance, replication, and fault tolerance across multiple nodes. Technologies such as MongoDB, Cassandra, and Google Spanner allowed universities to scale their data management capabilities efficiently.

Despite these advancements, distributed databases still depend on a trusted control layer to enforce permissions and data consistency. This administrative dependency means that the underlying system remains vulnerable to unauthorized alterations, insider threats, or accidental data corruption. Once altered, digital records can be difficult to verify or trace back to their original state. Moreover, when students transfer between universities or employers attempt to validate credentials, institutions often rely on manual verification processes that are inefficient and prone to error. The lack of a cryptographically verifiable audit trail limits both transparency and accountability in the academic data lifecycle.

Blockchain technology emerged as a response to these long-standing issues of trust and verification in digital systems. First conceptualized by Satoshi Nakamoto in 2008 for the Bitcoin network, blockchain introduced a decentralized mechanism for maintaining consensus across untrusted participants. Every transaction is validated collectively, time-stamped, and permanently stored in an immutable ledger. Over time, this technology has evolved beyond financial applications into domains requiring high data integrity—such as healthcare, supply chain, and now education. When applied to academic record management, blockchain ensures that each credential, transcript, or record can be independently verified against an incorruptible ledger without relying on a central authority. The integration of blockchain with distributed databases therefore represents a natural progression in the evolution of secure data management, bridging high performance with decentralized trust.

0.1.3 Problem Statement

To develop a blockchain-integrated distributed database system using MongoDB that ensures the integrity and verifiability of academic transcripts in university record management. The system will hash transcript data on a blockchain to detect tampering and unauthorized modifications while maintaining fast and scalable data access through a distributed NoSQL database.

0.1.4 Objectives

- To design a hybrid data architecture that integrates a distributed NoSQL database (MongoDB) with an Ethereum-based blockchain network to manage and verify academic transcripts in a decentralized manner.
- To implement a two-layer storage mechanism where complete academic records are maintained in MongoDB for high-speed access, while SHA-256 hashes of each transcript are stored on the blockchain to guarantee tamper detection and verifiable authenticity.
- To create a middleware layer using Node.js that synchronizes blockchain transactions with MongoDB collections, maintaining data consistency and enabling real-time verification of student records.
- To evaluate the hybrid system's performance under varying workloads by measuring database query latency, transaction throughput, and synchronization delay between MongoDB and blockchain components.
- To conduct tamper-resistance and data integrity testing by simulating unauthorized modifications within MongoDB and verifying blockchain-based detection of inconsistencies.

0.2. Requirement Analysis

To guide the development of a reliable and scalable blockchain-integrated database system for secure university record management, this section outlines the essential components of the requirement analysis process. It emphasizes the need for a thorough understanding of functional, performance, interoperability, and security aspects. To ensure that the system aligns with the complexities of academic data handling, privacy regulations, and institutional processes, the following analysis defines the necessary criteria for the effective design, development, and evaluation of the proposed framework.

This involves identifying and specifying the core components, software tools, and implementation methods required to complete the project. The development process includes creating a distributed MongoDB environment for managing student and transcript data, integrating a blockchain layer to hash and verify these records, and building a web interface for administrative and student interaction. The backend services, developed in Python using Flask, will

provide RESTful APIs that synchronize data between the database and the blockchain ledger. Finally, the system will be tested for data integrity, transparency, and performance under various operational conditions.

0.2.1 Functional Requirements

Functional requirements describe what the system should accomplish. These define the features, operations, and capabilities of the blockchain-based record management platform and are expressed in terms of input, process, and output.

- The administrator shall add, update, and retrieve student academic records through the web interface.
- The system shall store student records and transcripts in a distributed MongoDB database with replication and sharding enabled.
- The system shall hash academic record data using the SHA-256 algorithm before recording it on the blockchain.
- The blockchain layer shall verify the integrity of stored hashes and detect any unauthorized modifications in the database.
- The backend shall provide RESTful APIs for communication between the Flask server, MongoDB, and the blockchain ledger.
- The system shall allow cross-node data synchronization between multiple simulated university instances.

0.2.2 Non-Functional Requirements

Non-functional requirements define the qualitative attributes of the system—how it performs rather than what it does. These include reliability, scalability, maintainability, and security, ensuring the system performs efficiently under real-world usage.

- System latency for record verification shall not exceed 2 seconds per transaction.
- Data throughput should support at least 100 transactions per second under distributed operation.

- The system shall maintain 99% availability under network and node failure conditions.
- Blockchain immutability shall ensure zero tolerance for data tampering or deletion.
- System interfaces shall be intuitive and provide clear verification results to end-users.
- All communications between components shall occur over secure, authenticated channels.

0.2.3 Hardware Requirements

The computational resources required depend on the blockchain node configuration, database replication factor, and number of concurrent users. Minimum specifications are as follows:

- Processor: Quad-core CPU (2.5 GHz or higher)
- RAM: 8 GB minimum (16 GB recommended for multi-node simulation)
- Storage: 50 GB SSD for blockchain ledger and database storage
- Network: Stable LAN or virtual network environment for inter-node communication

For development and testing, multiple virtual instances may be configured to simulate separate university nodes participating in the blockchain network.

0.2.4 Software Requirements

The proposed system will be developed entirely using open-source technologies compatible with Python and modern web frameworks. The following software components are required:

- **Operating System:** Windows 10 / Ubuntu 22.04 LTS
- **Programming Language:** Python 3.11
- **Web Framework:** Flask (for backend and API development)
- **Database:** MongoDB (distributed NoSQL database)
- **Blockchain Framework:** Hyperledger Fabric or custom SHA-256-based blockchain implementation
- **Libraries:** PyMongo, hashlib, requests, Flask-RESTful

- **Version Control:** Git and GitHub for collaboration
- **Testing Environment:** Localhost simulation with Docker containers or virtual nodes

0.3. Project Description

The **Blockchain-Based University Record Management System** is a hybrid framework designed to securely manage and verify academic records across multiple university nodes. It integrates a distributed MongoDB database for storing student and faculty information with a blockchain layer that records cryptographic hashes of transcripts and transactions using the SHA-256 algorithm. This ensures tamper detection and verifiable data integrity. The backend, developed using Python (Flask) with RESTful APIs, coordinates communication between the database and blockchain, handling record insertion, hash generation, and verification requests. By maintaining full data off-chain and storing only hashes on-chain, the system achieves both performance efficiency and immutability. The proposed solution enables decentralized, auditable, and privacy-preserving management of academic credentials within a distributed environment.

0.4. System Design

In the context of secure and verifiable university record management, the proposed system design integrates distributed database technology with blockchain-based verification to ensure data transparency and integrity. The architecture of the **Blockchain-Based University Record Management System** (LedgerLink) follows a hybrid model that combines the scalability of a distributed NoSQL database (MongoDB) with the immutability and consensus mechanisms of a blockchain ledger. The design provides both performance efficiency and tamper detection, ensuring that academic transcripts remain authentic and verifiable across multiple institutional nodes.

0.4.1 Architectural Framework / System Design

The overall architecture, as illustrated in Figure 3.1, is composed of four primary layers:

1. **Frontend Layer (UI)** – The presentation interface developed using HTML, CSS, and JavaScript that allows users to interact with the system.

2. **Backend Layer (Node.js + Express)** – The middleware responsible for handling REST API requests, database operations, and blockchain communication.

3. **Blockchain Layer** – The verification layer that records hashes of academic transactions using SHA-256 and ensures immutability.

4. **Database Layer (MongoDB)** – The distributed NoSQL storage system that maintains all academic records, including student, faculty, and transaction data.

These components work together through RESTful API communication to ensure seamless synchronization between the blockchain ledger and the distributed database. The design prioritizes security, scalability, and verifiable audit trails while maintaining usability for both students and administrative users.

0.4.2 Frontend Layer

The **Frontend Layer** serves as the user-facing interface of the system. It is developed using HTML, CSS, and JavaScript (*index.html*) and connects to the backend through REST APIs. It allows students and administrators to add, view, and manage data such as student records, faculty details, fee payments, and blockchain transaction logs.

Key Functionalities:

- Add and view student and faculty details.
- Pay university fees linked to blockchain transactions.
- View on-chain transaction hashes and block metadata.
- Display MongoDB data collections in a structured table format.
- Import CSV files or dynamically create new tables for bulk record uploads.

All interactions between the user interface and backend occur via HTTP requests over a designated REST API port (Port 5050).

0.4.3 Backend Layer

The **Backend Layer** acts as the middleware that bridges the frontend interface with the blockchain and MongoDB layers. Developed using Node.js and Express, it manages all API routes, busi-

ness logic, and validation processes. Each API endpoint corresponds to specific functions like data insertion, blockchain synchronization, or fee verification.

Key API Routes:

- `/api/pay-fee` – Records fee transactions on both the blockchain and MongoDB.
- `/api/fees` – Fetches all fee transaction records.
- `/api/students` – Performs CRUD operations for student data.
- `/api/faculty` – Performs CRUD operations for faculty records.
- `/api/import-csv`, `/api/create-table`, `/api/blockchain-data` – Enables data import and synchronization.

Technologies and Libraries:

- **Mongoose:** Handles MongoDB connections, schema design, and relationships.
- **Flask / RESTful APIs:** Provides communication between backend services and blockchain verification modules.
- **Multer:** Supports CSV file import for bulk data operations.
- **dotenv:** Manages environment configurations for secure key handling.

0.4.4 Blockchain Layer

The **Blockchain Layer** functions as the decentralized verification mechanism within the system. It records all transactional proofs on-chain using cryptographic hashing (SHA-256). A local node or Hyperledger Fabric setup acts as the blockchain network, where each transaction represents an immutable record of an academic event such as fee payment or transcript generation.

Core Features:

- Each record transaction generates a unique blockchain hash serving as proof of authenticity.
- Immutable on-chain storage ensures that once a record is committed, it cannot be altered.

- The blockchain maintains only essential metadata (hash, timestamp, student ID) to preserve privacy.

Smart Contract Functions:

- `payFee(studentId, amount)` – Records fee transactions.
- `getFee(index)` – Retrieves individual transaction details.
- `getFeeCount()` – Returns the total number of blockchain transactions.

Each on-chain transaction produces a `txnHash`, which serves as a digital fingerprint to verify corresponding data entries stored in MongoDB.

0.4.5 Database Layer (MongoDB)

The **Database Layer** stores complete academic records in a distributed and replicated NoSQL environment. MongoDB collections maintain structured data across three key entities:

- **students:** { `_id, name, rollNumber, department, year` }
- **faculty:** { `_id, name, facultyId, department` }
- **fees:** { `_id, studentIdRef, amount, txnHash, timestamp` }

Relationships:

- One-to-many relationship between *students* and *fees*.
- Faculty data reserved for future extensions (transcripts or certifications).

MongoDB provides high-speed data retrieval and supports replication to ensure fault tolerance. Only hashes of critical data are recorded on the blockchain to reduce storage overhead and maintain efficiency.

0.4.6 Data Flow

The system's data flow, as shown in Figure 3.2, demonstrates the synchronization between the blockchain and MongoDB layers.

1. A user initiates a fee transaction from the frontend.

2. The backend API processes the request and triggers blockchain hash generation.

3. The blockchain node records the hash and returns a `txnHash`.

4. The backend stores the full transaction (studentId, amount, txnHash, timestamp) in MongoDB.

5. When queried, both blockchain and database entries are cross-verified for consistency.

This process ensures bidirectional integrity—if any record in MongoDB is modified, its hash will no longer match the blockchain record, allowing immediate detection of tampering.

0.4.7 Advantages of the Design

- Ensures end-to-end transparency and verifiability of student records.
- Maintains high transaction throughput due to MongoDB's scalability.
- Detects any unauthorized modification through blockchain validation.
- Reduces redundancy by storing hashes on-chain and full records off-chain.
- Enables modular integration for future features such as transcript verification and certificate issuance.

In summary, the proposed hybrid design effectively combines the strengths of distributed databases and blockchain technology. It guarantees integrity, performance, and transparency in academic data management while laying a foundation for scalable, decentralized education record systems.

0.4.8 Data Strategy

The data layer in the proposed system serves as the foundation for managing, storing, and verifying large volumes of academic information in a secure and efficient manner. The dataset will consist of logically generated or anonymized records representing a realistic university environment with hundreds of thousands of students. Each data instance will include attributes such as student identifiers, course enrollments, fee payment details, and transcript records. Generating synthetic or anonymized data allows the system to simulate real-world operations while maintaining privacy compliance and protecting sensitive personal information.

All primary academic data will reside in a distributed MongoDB database, which supports scalability through horizontal sharding and reliability through replica sets. This design allows multiple university nodes to access and update records concurrently while maintaining consistency. Collections such as *students*, *faculty*, and *fees* will be structured with well-defined document schemas and indexed fields to ensure fast query responses. Each document will include metadata such as timestamps, node identifiers, and update references to enable efficient synchronization and version tracking across distributed instances.

To guarantee data integrity and prevent tampering, only cryptographic hashes of critical data elements—such as transcripts, grades, and fee transactions—will be stored on the blockchain. The hashing process will use the SHA-256 algorithm to convert sensitive information into irreversible digital signatures, which are then recorded on the blockchain ledger as immutable verification proofs. The original, complete records remain securely stored in MongoDB. During validation, the system regenerates hashes from the current database entries and compares them to the on-chain hashes to confirm authenticity. Any mismatch indicates potential modification or corruption.

This approach separates data storage and verification, combining the performance efficiency of MongoDB with the immutability of blockchain technology. It ensures that the system can handle high data volumes while maintaining verifiable trust in the stored information. Overall, the data strategy prioritizes three key objectives: efficient distributed data management, cryptographic integrity assurance, and privacy-preserving verification of academic records across all participating university nodes.

0.4.9 Implementation Plan

The implementation of the proposed **Blockchain-Based University Record Management System** follows a modular and iterative development approach, integrating both distributed database and blockchain technologies. The entire system will be developed using the Python programming language due to its simplicity, flexibility, and strong library support for database management, API handling, and cryptographic operations.

A distributed **MongoDB** database will be configured to manage student, faculty, and academic transaction data. It will employ replication and sharding to support scalability, fault tolerance, and high-speed data retrieval. For blockchain integration, the team will utilize either **Hyperledger Fabric** or a custom blockchain framework implemented in Python using

the **Flask** microservice architecture and **SHA-256** hashing algorithm. The blockchain component will store hashes of transcript data, ensuring immutability and verifiable integrity while preserving data privacy.

The **Flask** framework will serve as the backend for developing RESTful APIs that connect the user interface, database, and blockchain components. These APIs will handle record creation, modification, hash generation, and verification processes across multiple nodes. Libraries such as **PyMongo** and **Hashlib** will be used for database connectivity and cryptographic operations respectively, enabling secure data handling and seamless synchronization between the database and blockchain layers.

A user-friendly **web dashboard** will be implemented for administrators and students to interact with the system. Administrators will be able to upload, verify, and manage student records, while students can view and confirm the authenticity of their academic data. The dashboard will display both database information and blockchain verification results in real time. The integration of these components will result in a reliable, secure, and transparent academic record management system.

0.5. Methodology

0.5.1 Technique 1: Blockchain

The blockchain layer functions as the system's core integrity verification mechanism, providing tamper-evident data validation for all academic transactions. Each record—such as a transcript, grade update, or fee payment—is converted into a unique digital signature through the **SHA-256** hashing algorithm. The generated hash, along with metadata such as the record identifier, timestamp, and originating node, is stored as a transaction within the blockchain ledger. The blockchain operates on a permissioned network architecture to maintain privacy and security among participating university nodes.

Each block includes a header containing the hash of the previous block, ensuring that all records are cryptographically linked. This append-only structure guarantees that any change to a past record alters the entire chain, making tampering immediately detectable. The ledger is periodically synchronized across nodes to maintain consensus. In the proposed setup, either a lightweight Python-based blockchain or a **Hyperledger Fabric** implementation is deployed to enable multi-institution record verification while maintaining low computational cost and high

throughput.

0.5.2 Technique 2: Distributed Database

The **MongoDB** distributed database serves as the main data store, responsible for maintaining full academic record details. Collections such as *students*, *faculty*, and *fees* are created, each with schema validation to enforce data consistency. Replication is configured to provide redundancy and fault tolerance, while sharding is used to horizontally distribute data across nodes to support concurrent access in large-scale simulations.

When a new record is created, the backend automatically generates its SHA-256 hash and writes that hash to the blockchain. This creates a synchronized link between the off-chain data and its on-chain verification. Query indexing on critical attributes such as `studentID`, `transactionHash`, and `timestamp` ensures efficient record retrieval and verification. This design enables the database to handle large data volumes while the blockchain safeguards integrity.

0.5.3 Technique 3: Consensus Algorithm

A permissioned blockchain framework is implemented where each university node represents an authorized participant. The network utilizes a simplified variant of the **Practical Byzantine Fault Tolerance (PBFT)** consensus protocol to achieve agreement among nodes before appending a block. During consensus, proposed transactions are validated by multiple nodes through message exchange and hash verification. Once the majority confirms the validity, the block is appended to the chain, and the ledger is updated system-wide.

This approach minimizes computational overhead while maintaining strong fault tolerance. Even if a subset of nodes fails or acts maliciously, the system remains operational, ensuring continuous synchronization. This mechanism allows multiple universities to collaborate in a decentralized environment without requiring centralized trust or manual reconciliation.

0.5.4 Technique 4: Smart Contracts

Smart contracts are designed to automate academic workflows such as transcript verification, record validation, and access control. Written as Python-based logic functions within the blockchain application layer, they execute automatically when predefined conditions are met.

For example, when a new transcript is uploaded, the contract checks administrator authorization, computes the SHA-256 hash, and commits the transaction to the ledger.

These contracts enforce transparent rules that ensure only authorized users—such as verified university administrators—can add or modify academic data. This reduces human error, enforces consistent data-handling policies, and improves operational efficiency. Each transaction's result, whether accepted or rejected, is recorded on-chain, creating a traceable audit trail.

0.5.5 Technique 5: API Integration

The **Flask-based REST API layer** connects all system components—frontend interface, MongoDB database, and blockchain ledger—into a unified workflow. When a user submits a transaction (e.g., adding a student record), the API performs three steps: (1) inserts the complete record into MongoDB, (2) generates a SHA-256 hash and submits it to the blockchain network, and (3) returns the blockchain transaction ID as confirmation to the frontend dashboard.

Endpoints such as `/addRecord`, `/verifyRecord`, and `/getHash` provide modular, secure communication across layers. The API also handles error detection, input validation, and logging of all activities for audit purposes. This integration ensures that both on-chain and off-chain data remain synchronized in real time, allowing seamless verification of academic records by students, administrators, and external verifiers.

0.6. Evaluation Plan

This section defines how the proposed MongoDB–Blockchain hybrid will be tested for correctness, performance, scalability, and resilience. All experiments will be executed against the same API surface (Flask REST), using repeatable workloads and controlled node counts to ensure comparability. Results will be reported as averages over multiple runs with dispersion statistics.

0.6.1 4.1 Metrics for Evaluation

Functional Correctness & Integrity

- **Tamper-Detection Rate (%)** = $\frac{\#detectedmodifiedrecords}{\#totalmodifiedrecords} \times 100$. Target: 100% under single-field and multi-field modifications.

- **False-Alarm Rate (%)** = $\frac{\#flaggedbutunmodifiedrecords}{\#totalverifiedrecords} \times 100$. Target: $\leq 0.1\%$.
- **Hash Consistency Lag (ms)**: time between MongoDB write and corresponding on-chain hash commit becoming queryable. Target: ≤ 300 ms median.

Performance

- **End-to-End Latency (ms)**: time from `POST /addRecord` to successful on-chain commit + DB write acknowledgement. Report p50/p95. Targets: p50 ≤ 800 ms; p95 ≤ 1500 ms at baseline load.
- **Verification Latency (ms)**: time from `GET /verifyRecord?id` to integrity verdict (re-hash + on-chain compare). Targets: p50 ≤ 200 ms; p95 ≤ 500 ms.
- **Throughput (TPS)**: sustained successful transactions/sec during a 10-minute steady-state write test. Baseline target: ≥ 100 TPS on a 3-node DB + 3-node blockchain.

Scalability

- **Horizontal Scale Efficiency (%)** = $\frac{TPS(N)}{N \times TPS(1)} \times 100$ for $N \in \{3, 5, 7\}$. Target: $\geq 70\%$ at $N = 5$.
- **Dataset Scale Robustness**: maintain p95 latency targets with 100k, 500k, and 1M records; index-hit ratio $\geq 95\%$ for queries on `studentId`, `txnHash`, `timestamp`.

Fault Tolerance & Availability

- **RPO (Recovery Point Objective)**: max data-at-risk upon failure (expected: 0 for verified commits).
- **RTO (Recovery Time Objective)**: time to restore write/read service after a node crash. Target: ≤ 5 s for read; ≤ 10 s for write path under single-node failure.
- **Consensus Continuity**: ability to finalize blocks with f faulty nodes in a permissioned network (PBFT-like). Target: sustain operation with up to $\lfloor (N-1)/3 \rfloor$ faulty nodes.

Resource & Storage Overheads

- **On-Chain Overhead/Record (bytes)**: hash + metadata. Target: ≤ 256 B/record typical.
- **CPU/Memory Utilization**: median and p95 across DB, API, blockchain nodes at baseline and peak load, aiming for headroom $\geq 30\%$.

0.6.2 4.2 Experimental Setup

- **Workload**: logically generated/anonymized student records with fields for identifiers, enrollments, transcripts, and fee transactions. Mixture: 70% writes (create/update), 30% reads/verify.
- **Dataset Sizes**: 100k, 500k, and 1M records; fee events at 2–3x student count to stress append paths.
- **Topology**: MongoDB replica sets with sharding (3, 5, and 7 data nodes); blockchain network with the same node counts in permissioned mode.
- **API**: Flask REST running behind a single load-balanced endpoint; request concurrency levels: 32, 64, 128 clients.
- **Indices**: `students(studentId, timestamp)`, `fees(studentIdRef, txnHash, timestamp)`. All queries must be plan-verified to hit indices.
- **Runs**: each experiment repeated 5 times; report mean, standard deviation, and p50/p95 latencies.

0.6.3 4.3 Procedures

P1: End-to-End Write Path Issue sustained `POST /addRecord` at target TPS. Measure API latency, MongoDB write time, hash-commit finalize time, and return time. Collect p50/p95.

P2: Verify Path For existing records, call `GET /verifyRecord?id`, recompute SHA-256 from MongoDB, compare with on-chain hash, and return verdict. Measure verification latency and false-alarm rate.

P3: Tamper Tests Inject controlled mutations in MongoDB (single-field, multi-field, and record-replace). Run verification sweep; record Tamper-Detection Rate and detection latency.

P4: Scalability Tests Increase node count ($N = 3 \rightarrow 5 \rightarrow 7$) and dataset size (100k→1M). Measure throughput, p95 latency, and Scale Efficiency.

P5: Fault Tolerance During steady-state load, kill one DB node (primary/secondary) and one blockchain node (non-leader), then measure RTO for read/write and any data loss (RPO). Validate continued block finality under PBFT-like quorum.

P6: Resource Profiling Collect CPU, memory, and I/O metrics on API, DB, and blockchain nodes; correlate with latency/throughput to identify bottlenecks (e.g., index misses, replica lag).

0.6.4 4.4 Expected Outcomes

- **Functional Prototype**: end-to-end system demonstrating write, verify, and cross-node consistency for student records and transcripts.
- **Integrity Guarantees**: 100% Tamper-Detection Rate with ≤ 500 ms median detection latency on verification.
- **Performance Targets**: ≥ 100 TPS sustained; p50 write latency ≤ 800 ms; p95 verify latency ≤ 500 ms at 100k–500k records.
- **Scalability**: $\geq 70\%$ Scale Efficiency at 5 nodes; latency targets maintained at 1M records with proper sharding and indexing.
- **Fault Tolerance**: No data loss for committed transactions (RPO = 0); service RTO ≤ 10 s for write path under single-node failure; consensus continues with tolerated faults.
- **Auditability**: immutable on-chain proofs (hash + timestamp + node id) enabling cross-institution verification via `/verifyRecord`.

0.6.5 4.5 Reporting & Validation

All metrics will be summarized in tables with p50/p95 latencies, TPS, error rates, and node counts. Plots will show throughput vs. latency and scale efficiency vs. node count. For integrity

tests, a confusion-matrix style summary (modified vs. detected) will be provided. Raw logs (API, DB, blockchain) and experiment scripts will be versioned to ensure reproducibility.

0.7. Timeline, Conclusion, and References

0.7.1 Timeline

The project development was organized into well-defined milestones to ensure systematic progress and balanced workload distribution. Each phase focused on a specific component of the system—beginning with architectural planning and concluding with testing and evaluation. Table 1 summarizes the project schedule.

Milestone	Start Date	End Date	Team Members Responsible
ER Diagram and Workflow Design	October 6, 2025	October 7, 2025	All Team Members
Generate Dummy Data	October 7, 2025	October 8, 2025	Jett Bauman
Set Up Development Environment (GitHub, IDE, etc.)	October 6, 2025	October 10, 2025	All Team Members
Blockchain Implementation	October 11, 2025	October 15, 2025	Quan Nguyen, Aditya Sahu, Jett Bauman
Front-End UI Implementation	October 16, 2025	October 19, 2025	Soumya Katagihalli, Quan Nguyen
API Implementation	October 20, 2025	October 22, 2025	All Team Members
Database Query Implementation	October 22, 2025	October 28, 2025	Jett Bauman, Aditya Sahu
Integration of Front-End and Back-End	October 29, 2025	November 1, 2025	Quan Nguyen, Soumya Katagihalli, Aditya Sahu
Deployment	November 2, 2025	November 9, 2025	All Team Members
Testing	November 10, 2025	November 15, 2025	All Team Members
Demo and Poster Preparation	November 16, 2025	November 21, 2025	All Team Members
Final Review and Submission	November 22, 2025	November 26, 2025	All Team Members

Table 1: Project Development Timeline

0.7.2 Conclusion

The proposed **Blockchain-Based University Record Management System** successfully demonstrates a hybrid framework that combines distributed database efficiency with blockchain integrity verification. By storing academic data in MongoDB and anchoring cryptographic hashes

of those records on a blockchain, the system ensures transparency, immutability, and traceability in university record management. The project's layered architecture—comprising database, blockchain, API, and user interface modules—achieved real-time synchronization and secure validation of academic transactions across simulated university nodes.

This work provides a foundation for scalable and tamper-proof academic data exchange across institutions. The prototype confirms that blockchain can be integrated into existing database ecosystems without sacrificing performance, while simultaneously addressing data authenticity and verification challenges. Future enhancements may include the use of smart contracts for automated credential validation, IPFS for decentralized storage of large files, and expansion to a multi-institution deployment model.

0.7.3 References

1. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

2. Sheetal Chaudhari and Mahesh Shirole, "Blockchain-Driven Academic Learning Record Management in Higher Education: A Comprehensive Review of Methodologies, Applications, Benefits, and Challenges," *SN Computer Science*, vol. 6, Apr. 2025.

3. Federico Carrozzino et al., "Development of a Hybrid Blockchain and NoSQL Platform to Improve Data Management," *arXiv preprint*, May 2023.

4. Zerui Ge et al., "Hybrid Blockchain Database Systems: Design and Performance," *Proc. VLDB Endowment*, vol. 15, no. 5, 2022.

5. X. Cai et al., "A Hybrid Storage Blockchain-Based Query Efficiency Enhancement Method," *Knowledge and Information Systems*, 2024.

6. "Integration of Blockchain Technology in Database Management Systems (DBMS)," *IJFMR*, vol. 7, no. 2, 2025.

7. "A Systematic Review of Blockchain-Based Initiatives in Academic Certificates Management," *Computers*, MDPI, 2023.

8. "A Hybrid Blockchain Approach for Academic Record Management System," *Lecture Notes in Networks and Systems*, Springer, 2024.

9. “Blockchain Adoption in University Archives Data Management,” *Knowledge and Information Systems*, Springer, 2023.

10. Michael Willson, “How Universities Can Use Blockchain to Improve Academic Research,” *Blockchain Council*, Oct. 2025.