IEEE CS Bangalore Chapter Internship and Mentorship Program - 2025

Duration: 1st April 2025 to 30th September 2025

Monthly Progress Report

Project ID: P100

Name of the Students:

1. Snehal Suryavanshi
2. Vinayak Agasimani
3. Surajkumar Kottal
4. Srushti Utturkar

University/ College Name: Jain College of Engineering

Name of the Mentor: Vishal Rathod

Mentor's Organization: C-DAC, Bengaluru

Title of the Project Decentralized Student Result Storage and verification System using Blockchain and Cloud Computing

1.Problem Statement:

Educational institutions frequently encounter challenges in maintaining secure, transparent, and easily accessible repositories for student academic records. Traditional systems often present vulnerabilities regarding data integrity, lack efficient mechanisms for student access and management of their documents, and can be ineffective for administrators responsible for uploading and organizing these files. This can lead to concerns about the authenticity and immutability of academic achievements, as well as inefficiencies in administrative workflows related to record management.

2.Literature Review

The application of blockchain technology in educational systems has garnered significant interest for its potential to secure, verify, and decentralize the management of student academic records. Pawar et al. (2019) laid the groundwork by addressing issues such as result tampering, fake mark sheets, and the lack of transparency in traditional university result systems. Their blockchain-based framework introduces role-based access for administrators and students, ensuring improved data integrity, decentralized validation, and real-time access to academic records while preserving student privacy. Expanding on this, Enwerem and Okolie (2023) implemented a hybrid result and transcript management system at the Federal University of Technology, Owerri. Their architecture blends a Web2 backend using PHP and MySQL with a Web3 Ethereum-based smart contract layer. SHA-256 hashes of student results are stored on the blockchain, enabling secure verification and transcript generation. Node.js is used to bridge the frontend and blockchain, ensuring smooth interaction.

Several other researchers have explored more specialized applications of blockchain in academic verification. Kong (2024) examined the design of a secure and transparent system for managing school records, emphasizing confidentiality and integrity. Rustemi et al. (2024) proposed "DIAR," a system for generating and verifying academic diplomas using blockchain's immutability to prevent tampering. Kaneriya and Patel (2023) focused on privacy-preserving mechanisms in student credential verification systems, demonstrating how blockchain can maintain both security and data confidentiality. Rahman et al. (2023) introduced "Verifi-Chain," which integrates blockchain with the InterPlanetary File System (IPFS), storing hashes on-chain and documents off-chain, thus combining security with decentralized file access. Ghani et al. (2022) used

Hyperledger Fabric to build a permissioned blockchain for managing student certificates, prioritizing controlled access within academic networks.

Other notable implementations include the work of Maranto et al. (2024), who developed a public Ethereum-based system for diploma and transcript verification, showcasing the use of a public blockchain for open credential authentication. Mishra et al. (2025) examined a decentralized document verification system applicable to academic use cases, reinforcing blockchain's role in broader document security. While not directly blockchain-based, Kaviya et al. (2025) developed a web-based academic platform aimed at optimizing student database management and communication, presenting potential opportunities for future blockchain integration.

The paper by Pawar et al. (2025) offers a broader perspective by evaluating blockchain as a distributed database system for managing university results. Their study not only addresses common issues such as hacking and false data, but also explores applications in crypto-voting, privacy in cryptocurrencies like Bitcoin and Zerocash, and secure ledger-based data management. These studies collectively demonstrate how blockchain can revolutionize academic data management by introducing immutability, transparency, and verifiability while enabling decentralized control.

3.Proposed System

To address these limitations, this project proposes the development of a decentralized Student Result Storage System. This innovative solution leverages the security and immutability of blockchain technology, specifically the Ethereum platform, integrated with the scalability and accessibility of cloud computing services. The system will establish distinct user roles for administrators, who will securely upload and manage student records, and for students, who will have secure view-only access with the added convenience of automated downloads to their linked Google Drive accounts. Furthermore, the implementation of an intelligent file suggestion feature during the administrator upload

the process will streamline data entry and improve overall efficiency. This approach aims to enhance the security, transparency, and accessibility of student academic records while optimizing administrative processes.

4. Architectural Framework

This flowchart illustrates the process of uploading, storing, accessing, and verifying student results using a system that integrates cloud storage and blockchain technology. Let's break down each step:

The system architecture is depicted in **Figure 1**, which illustrates the interaction between users, cloud storage, and blockchain verification.

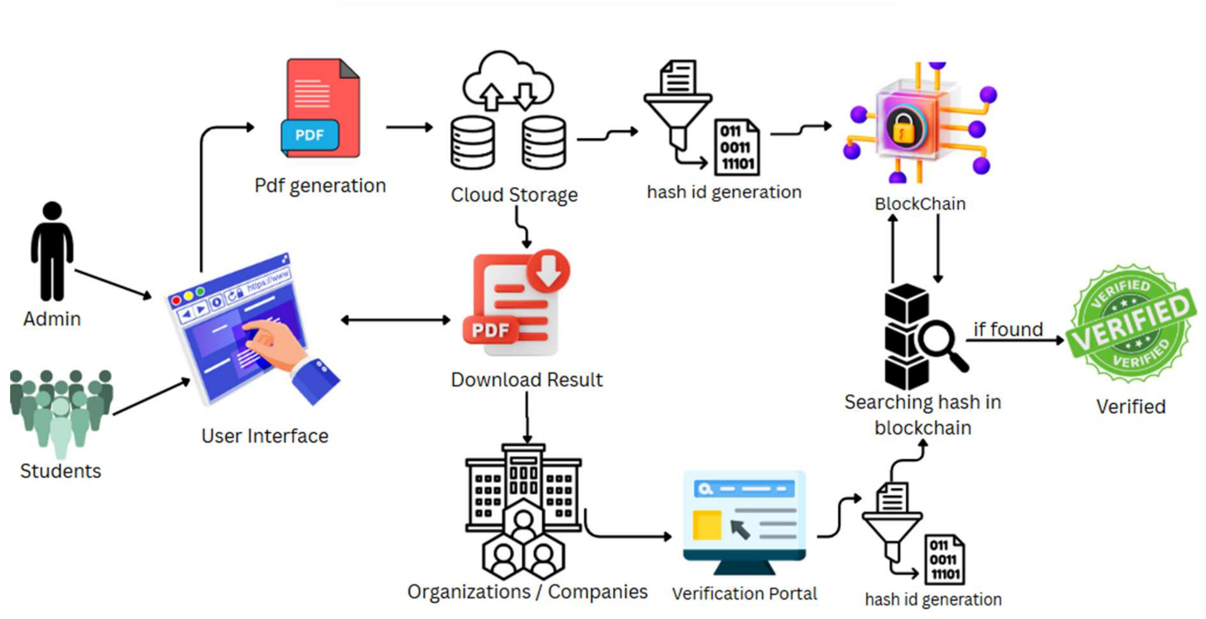


Figure 1

Block diagram of the student result verification process using blockchain and cloud storage (created by the author).

1. **Admin & Students:** The flowchart starts with two types of users:
    - **Admin:** Represents the administrator of the system, likely responsible for uploading and managing student results.
    - **Students:** Represent the students who will access and download their results.

2. **User Interface:** Both the admin and students interact with the system through a user interface (likely a web application). The hand clicking on the screen suggests interaction.
3. **Admin Action -> PDF Generation:** The arrow from the "Admin" to "User Interface" and then to "Pdf generation" indicates that the administrator initiates the process by uploading student results, which are then converted or exist in PDF format.
4. **PDF -> Cloud Storage:** The generated PDF files of student results are then stored in "Cloud Storage." This signifies the use of a cloud-based storage solution (like AWS S3, Google Cloud Storage, etc.) to hold the actual result documents.
5. **Cloud Storage -> Download Result (Student Action):** Students, through the "User Interface," can access and "Download Result" (the PDF file) from the "Cloud Storage." The arrow going directly from "Cloud Storage" to "Download Result" and then to "Organizations / Companies" via the "Verification Portal" suggests a direct download path for authorized users.
6. **Cloud Storage -> Hash ID Generation:** Simultaneously, after the PDF is stored in the cloud, the system generates a unique "hash id" for that specific PDF file. A hash function takes the content of the file and produces a fixed-size string (the hash) that acts as a digital fingerprint. Any change to the original PDF will result in a different hash.
7. **Hash ID Generation -> Blockchain:** The generated "hash id" is then recorded on the "Blockchain." This is the crucial step where the integrity of the result is secured. By storing the hash on a blockchain, which is immutable and transparent, any future attempts to alter the result can be detected by recalculating the hash and comparing it to the one on the blockchain.
8. **Organizations / Companies -> Verification Portal:** External entities like "Organizations / Companies" that need to verify the authenticity of a student's result can do so through a "Verification Portal."
9. **Verification Portal -> Hash ID Generation:** The organization/company, through the verification portal, would likely upload the student's result document (presumably a PDF they received). The system then generates a "hash id" for this uploaded document.
10. **Hash ID Generation -> Searching hash in blockchain:** The newly generated hash id from the uploaded document is then used to search for a matching hash within the "Blockchain."

11. **Searching hash in blockchain -> Verified (if found):**
    - **If found:** If a matching hash is found on the blockchain, it strongly indicates that the document being verified is the original, unaltered result that was initially uploaded by the administrator. The "VERIFIED" stamp signifies a successful verification.
    - **If not found (implied):** If the hash is not found on the blockchain, it would suggest that the document being verified is either not an official result or has been tampered with since it was originally recorded. This outcome isn't explicitly shown with a negative indicator but is the logical consequence of a failed search.

5. Knowledge gained - Tools, Technology, Courses etc.

- **Cloud service :AWS ,**For cloud storage, we chose AWS EC2 due to its simplicity, flexibility, and strong security features—making it ideal for beginners. It offers a Free Tier with up to 750 hours/month on t2.micro or t3.micro instances, enabling cost-free initial deployment. EC2 allows full customization of virtual machines and integrates seamlessly with EBS for block storage and S3 for object storage. Its built-in features like auto-scaling, IAM roles, and security groups make it suitable for hosting file upload services, managing academic records, and running backend systems reliably.
- **BLOCKCHAIN: Ethereum Platform**: Ethereum is an open-source, public blockchain platform that enables the creation and execution of smart contracts. Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They automate processes and enforce rules without the need for intermediaries. Ethereum uses its own cryptocurrency called Ether (ETH) to pay for transaction fees, known as "gas." The Ethereum Virtual Machine (EVM) is the runtime environment for executing smart contracts on the Ethereum network. Ethereum's robust smart contract capabilities make it well-suited for defining the logic and rules for our student result storage system.

- **HASH GENERATION: Algorithm : SHA-256 (Secure Hash Algorithm 256-bit)**

In our project, we use the SHA-256 (Secure Hash Algorithm 256-bit) cryptographic function to generate a unique digital fingerprint for each uploaded PDF file. SHA-256 produces a fixed 256-bit hash value, ensuring that even a minor change in the file will result in a completely different hash. This makes it ideal for verifying file integrity and preventing tampering. We chose SHA-256 because it is widely adopted in secure systems such as blockchain and digital signatures, offers strong resistance to attacks, and maintains an excellent balance between speed and security. For implementation, we use the js-sha256 npm package, a lightweight and dependency-free library that supports both Node.js and browser environments. It efficiently handles binary data like PDFs, making it well-suited for our application.

- **Handling Uploaded PDFs (Backend)**

  On the backend, file uploads are managed using the built-in file handling capabilities of modern web frameworks like Express, Flask, or Django, which support multipart/form-data formats. Once received, uploaded PDF files are securely stored in cloud storage using the appropriate cloud SDK, such as boto3 for AWS S3, Google Cloud Client Libraries, or Azure Storage SDK. These SDKs enable seamless integration with cloud environments, allowing for efficient storage, retrieval, and access control of student academic documents.

- **PDF Generation from Raw Data**

  When administrators provide raw academic data instead of pre-generated PDFs, we generate the required files programmatically on the backend. In the Node.js environment, libraries such as pdfmake, html-pdf, and puppeteer are used to create structured PDF documents, often from HTML templates. This allows us to standardize the formatting of academic records and ensures consistency, readability, and security before storing them in the cloud or generating hashes for verification.

- User Interface

Frontend

The frontend of our system is developed using React.js, a component-based JavaScript library that enables the creation of dynamic, responsive, and interactive user interfaces. It supports real-time updates, making it ideal for applications where students and administrators need immediate feedback and access to results. React Router is used for managing in-app navigation without full page reloads, enhancing the user experience. For handling user input, Formik simplifies form management, while Yup is used for schema-based validation to ensure data accuracy. Axios facilitates secure and efficient HTTP communication between the frontend and backend services. Additionally, Ethers.js (or Web3.js) allows the frontend to interact directly with Ethereum smart contracts, enabling blockchain-based features such as result verification and hash retrieval.

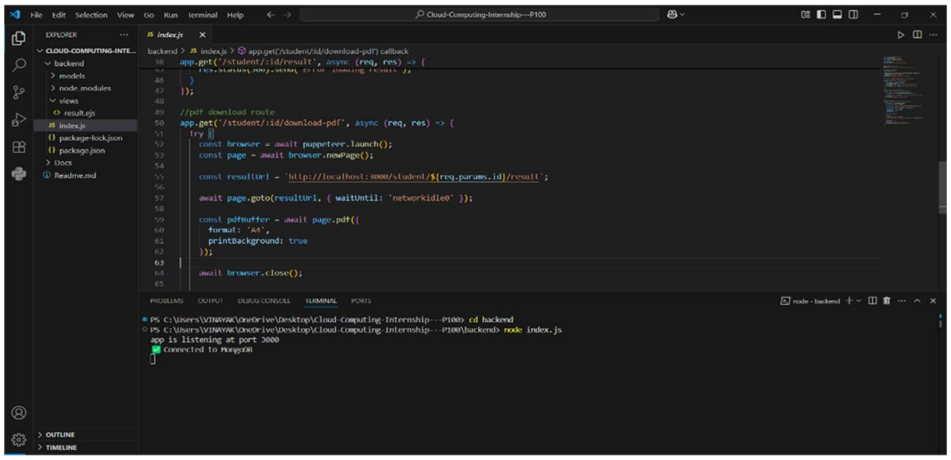
Backend

The backend is powered by Node.js, a non-blocking, event-driven runtime that efficiently manages asynchronous operations and API requests. Express.js, a minimalist web framework for Node.js, is used to define and handle RESTful APIs, manage routes, and process form data submitted from the frontend. The backend securely interacts with the blockchain layer, performing tasks like generating and verifying SHA-256 hashes, interacting with smart contracts, and communicating with cloud storage services such as AWS S3. This architecture ensures a clear separation of concerns, high scalability, and smooth integration between the traditional server operations and blockchain-based components.

6. Project Implementation :

The following implementation tasks have been completed:

- **Backend Development**
  - A Node.js and Express-based backend server has been set up and configured to run on port 3000.

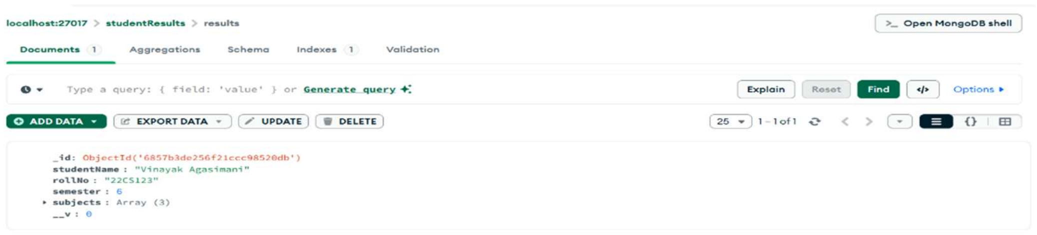


```
app.get('/student/:id/result', async (req, res) => {
    res.status(500).send('Error loading result');
  }
});

//pdf download route
app.get('/student/:id/download-pdf', async (req, res) => {
  try {
    const browser = await puppeteer.launch();
    const page = await browser.newPage();

    const resultUrl = `http://localhost:3000/student/${req.params.id}/result`;

    await page.goto(resultUrl, { waitUntil: 'networkidle0' });

    const pdfBuffer = await page.pdf({
      format: 'A4',
      printBackground: true
    });

    await browser.close();
```

```
PS C:\Users\VINAYAK\OneDrive\Desktop\Cloud-Computing-Internship---P100> cd backend
PS C:\Users\VINAYAK\OneDrive\Desktop\Cloud-Computing-Internship---P100\backend> node index.js
app is listening at port 3000
✅ Connected to MongoDB
```

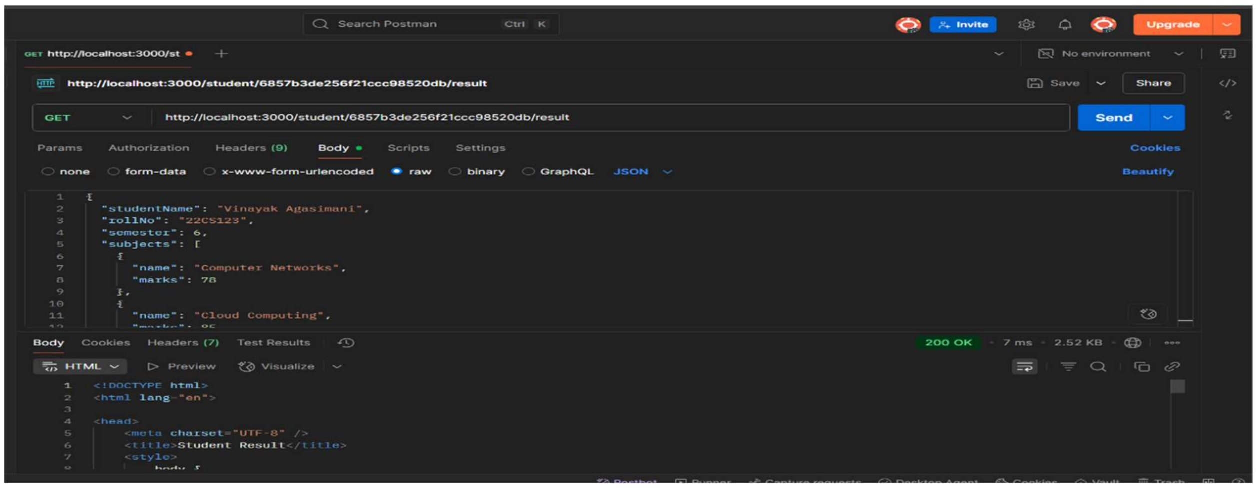
  - MongoDB has been integrated to serve as the primary database for storing student result records.

- **Data Acquisition and Storage**
  - A RESTful API endpoint has been developed to accept and parse JSON-formatted student result data.
  - Input data includes student name, roll number, semester, and subject-wise marks.
  - Submitted data is successfully stored in MongoDB using the Mongoose ODM.



```
_id: ObjectId('6857b3de256f21ccc98520db')
studentName : "Vinayak Agasimani"
rollNo : "22CS123"
semester : 6
▸ subjects : Array (3)
__v : 0
```

- **Dynamic Result Template Design**
    - An EJS (Embedded JavaScript) template has been designed to generate standardized result sheets for students.
    - The layout includes:
        - University name (configurable)
        - Student details (name, roll number, semester)
        - Tabular display of subjects, marks obtained, and pass/fail status
    - The same layout is reused dynamically for all students by injecting data from the backend.

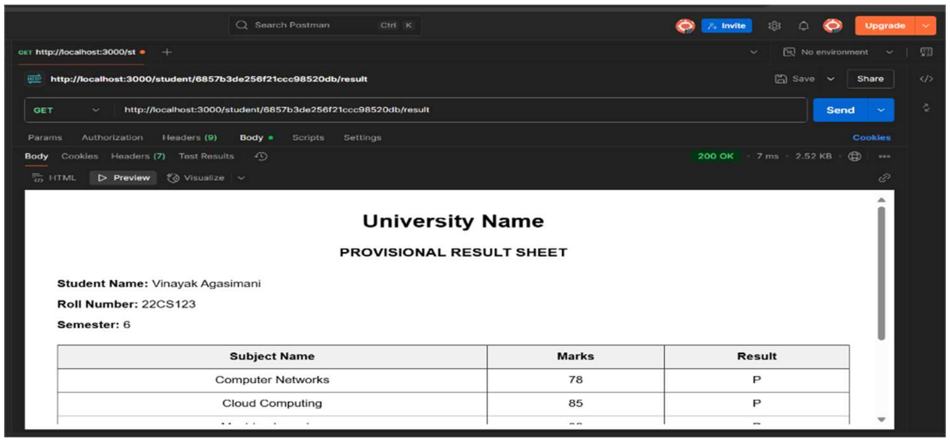


GET http://localhost:3000/student/6857b3de256f21ccc98520db/result

```
{
  "studentName": "Vinayak Agasimani",
  "rollNo": "22CS123",
  "semester": 6,
  "subjects": [
    {
      "name": "Computer Networks",
      "marks": 78
    },
    {
      "name": "Cloud Computing",
```

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <title>Student Result</title>
    <style>
```

- **PDF Generation Module**
    - Puppeteer has been integrated to convert the result HTML page into a downloadable PDF format.
    - A dedicated route has been implemented to allow PDF download on-demand.
    - A “Download Result” button is embedded on the result page, which triggers the download without exposing internal backend routes.
    - CSS media queries have been used to exclude non-essential elements (e.g., download button) from the PDF.
    - The result PDF is clean, standardized, and ready for cloud upload and verification processes.


Search Postman
Invite
Upgrade
No environment
Save
Share
Send
Params Authorization Headers (9) Body Scripts Settings
Cookies
Body Cookies Headers (7) Test Results
HTML Preview Visualize

GET http://localhost:3000/student/6857b3de256f21ccc98520db/result

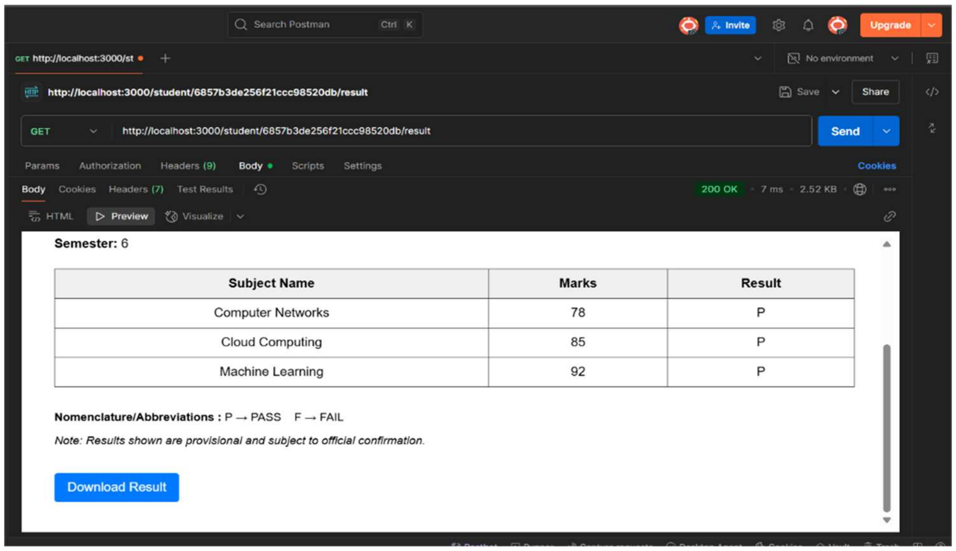
200 OK · 7 ms · 2.52 KB

University Name

PROVISIONAL RESULT SHEET

Student Name: Vinayak Agasimani
Roll Number: 22CS123
Semester: 6

Subject Name	Marks	Result
Computer Networks	78	P
Cloud Computing	85	P


Search Postman
Invite
Upgrade
No environment
Save
Share
Send
Params Authorization Headers (9) Body Scripts Settings
Cookies
Body Cookies Headers (7) Test Results
HTML Preview Visualize

GET http://localhost:3000/student/6857b3de256f21ccc98520db/result

200 OK · 7 ms · 2.52 KB

Semester: 6

Subject Name	Marks	Result
Computer Networks	78	P
Cloud Computing	85	P
Machine Learning	92	P

Nomenclature/Abbreviations : P → PASS F → FAIL

Note: Results shown are provisional and subject to official confirmation.

Download Result

- **Next Steps (Planned):**
  - Automate PDF generation and upload to AWS S3 upon result submission.
  - Generate a SHA-256 hash of the final PDF and store it on a blockchain for tamper-proof record verification.
  - Implement a secure verification portal to validate result authenticity using uploaded PDFs or embedded QR codes.

7. Results — yet to be done

8.Conclusion and Future Work

In this report, we have conducted a comprehensive study on the concept, relevance, and potential of implementing a blockchain-based student result storage system. Through detailed literature review and architectural analysis, we have identified key technologies, system requirements, and the advantages of integrating blockchain with traditional result management processes. Although the technical implementation of the system has not yet begun, our research has laid a strong foundation for the development phase.

The next phase of the project will focus on practical implementation using selected tools such as React for the frontend, Node.js for backend communication, and a blockchain platform (such as Ethereum or Hyperledger) for secure data storage and verification. We aim to apply the architectural insights gained during this phase to design and build a functional prototype that meets our security, scalability, and usability objectives.