Secure Access Control and Certificate Validation Using Blockchain and Python

Asst. Prof. Nishchitha T S*[1] Dept. of ISE, RRIT, Likhitha T R*[2], Nagumuttu*[3], Nikhil T S*[4], Nithin R*[5]

Dept. of ISE, R R Institute of Technology, Bangalore, Karnataka, India

ABSTRACT

This paper presents a blockchain-based solution for secure access control and certificate validation. Traditional certificate systems are prone to forgery, manipulation, and inefficiencies due to centralized storage and manual verification. The proposed system uses Python and blockchain technology to build a tamper-proof, transparent, decentralized platform for certificate issuance, storage, and verification. The system leverages smart contracts, cryptographic hashing, QR code integration, and OTP-based user authentication to ensure real-time, secure, and scalable certificate management.

Keywords: Blockchain, Certificate Validation, Cryptographic Hashing, Python, OTP Verification, Access Control.

I. INTRODUCTION

In today's digital-first world, the way we store, share, and validate important documents like academic certificates is rapidly evolving. Yet, many institutions continue to depend on outdated systems that are easy to manipulate. With increasing incidents of certificate forgery, unauthorized duplication, and difficulty in verifying credentials across institutions, it has become clear that a more secure and efficient solution is needed.

This project aims to solve these real-world issues by introducing a secure, blockchain-based certificate generation and validation system. Blockchain technology offers a decentralized and tamper-proof method of recording data. Once a certificate is generated and its details are stored as a block, it becomes practically impossible to alter or delete it without affecting the entire chain making it ideal for applications where trust, transparency, and data integrity are crucial. Python is used as the core programming language to build this system due to its simplicity, flexibility, and strong support for data encryption and backend development. The project includes modules for admin-controlled certificate generation, user registration, QR code integration for quick verification, and OTP-based email authentication to add an extra layer of security. The overall goal is to design a solution that not only prevents fraud but also simplifies the verification process for institutions, students, and employers. With just a scan or a click, certificates can be verified in real time — no paperwork, no delays, and no doubts.

II. METHODOLOGY

In this project, the first step involved requirement analysis to understand the need for a secure access system and certificate validation using blockchain technology. The system was then designed with a clear structure, comprising a user-friendly frontend using HTML, CSS, and JavaScript, and a robust backend using Python and Flask. A relational database was planned to manage user data and certificate details efficiently. For blockchain integration, a private blockchain network was set up, and smart contracts were developed to securely store and verify certificate information. The project involved generating digital certificates and storing their cryptographic hashes on the blockchain for tamper-proof validation. A secure access control mechanism was implemented, featuring user authentication and role-based access management. After development, integration testing was performed to ensure seamless communication between the frontend, backend, and blockchain components. Rigorous security and performance testing were carried out to validate the system's reliability. Finally, the complete application was deployed on a cloud platform, ensuring scalability, accessibility, and continuous monitoring for optimal performance.

Data Collection: Gather the required user credentials and certificate details needed for authentication and validation. Collect information such as user IDs, roles, and associated certificates. Ensure certificates contain essential metadata like issuance dates and authorized signatures.

Data Pre-processing: Pre-process the collected data by securing it with cryptographic techniques like hashing. Format the certificates properly in a digital structure (JSON/XML) suitable for blockchain storage and verification.

Blockchain Setup: Set up a private or test blockchain network. Develop and deploy smart contracts that will store the hashed certificates and validate authenticity during access control checks.

Certificate Generation: Generate digital certificates for users with unique identifiers. Hash the certificates using cryptographic algorithms like SHA-256 and store the hash values securely on the blockchain.

Secure Access Control: Implement authentication mechanisms (such as login systems) using Python and Flask. Use role-based access control to define different privileges for users like Admins and General Users.

Certificate Validation: When a user requests access, fetch their certificate, hash it, and verify the hash against the blockchain records. If a match is found, the system grants access and validates the certificate's authenticity.

Integration and Testing: Integrate the front end (HTML, CSS, JavaScript) with the backend and blockchain. Perform testing to ensure data flows correctly, certificates are validated accurately, and access is securely controlled.

Deployment: Deploy the application to a secure cloud environment (like AWS, Azure, or GCP). Ensure continuous monitoring, regular updates, and backup strategies to maintain system security, reliability, and scalability.

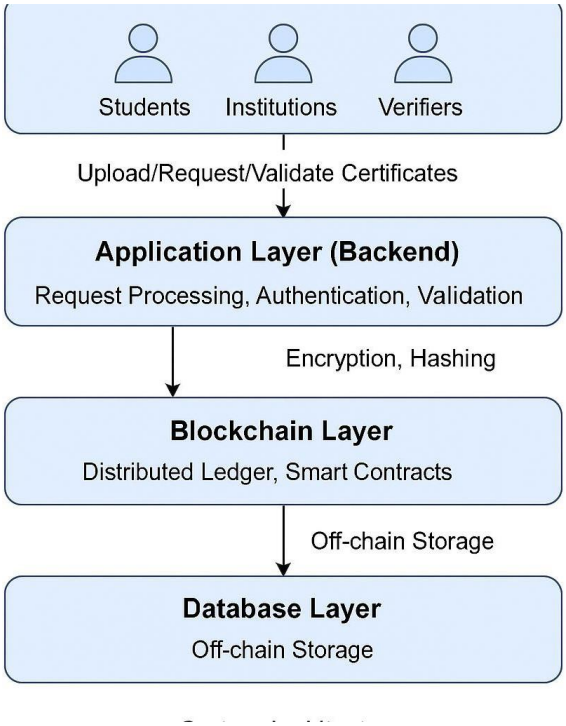

Figure 1. An overview of the proposed Secure Access Control and Certificate Validation

III. SYSTEM DESIGN

1. Blockchain Integration with Flask:

In our project, blockchain functionality was integrated conceptually within the Flask application to secure certificate validation. We did not use any external blockchain network. Instead, we simulated blockchain behavior by generating SHA-256 hashes of certificates and storing them securely. These hashes ensured the integrity and authenticity of certificates without needing a full blockchain deployment. The entire logic was handled through Python functions inside the Flask backend.

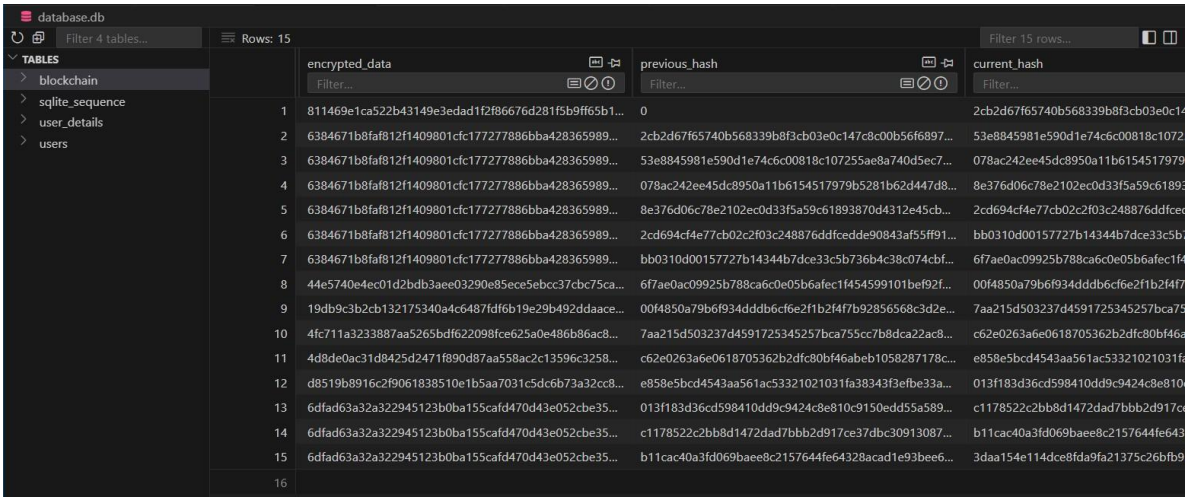

Figure 1. Blockchain Simulation Flow

2. Backend Development:

The backend was developed using **Python** and **Flask**, providing the core functionalities of user authentication, certificate generation, hashing, and validation.

- **Flask**: For API creation, routing, and server-side logic.
- **Hashlib**: To generate SHA-256 hashes simulating blockchain immutability.
- **SQLite3**: For managing user and certificate data locally.

APIs were developed for user registration, login, certificate issuance, and certificate validation. Flask ensured smooth communication between front end and back end.

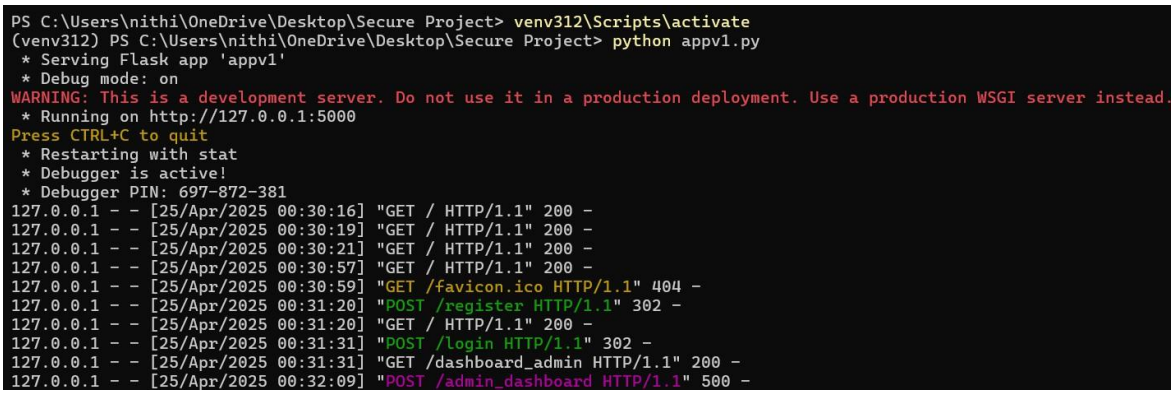

Figure 2. Flask Server Flow

The Flask server was run locally through VS Code's integrated terminal without needing external deployment servers.

3. Frontend Development (HTML, CSS, JavaScript, Bootstrap):
The front end was developed using **HTML5**, **CSS3**, and **JavaScript**, and styled with **Bootstrap 5** for responsiveness. Bootstrap helped quickly build a modern, mobile-friendly user interface with minimal custom CSS. The frontend contained:

- Login forms
- Registration forms
- Certificate generation pages
- Certificate validation pages

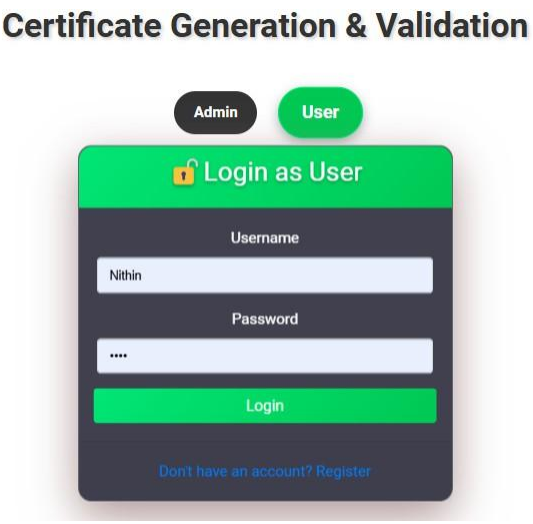

Figure 3. Frontend Design using Bootstrap

Bootstrap components such as forms, cards, buttons, and alerts were used to create an interactive user experience.

4. Database Management (SQLite3 in VS Code):
We used **SQLite3**, a lightweight, embedded SQL database, for storing user credentials and certificates. All data was stored locally inside a .db file managed directly through the Flask app. The database operations included:

- Creating user records with hashed passwords.
- Storing certificate information and its hash.
- Fetching records for login and validation operations.

SQLite3 was chosen because it is simple to set up, serverless, and ideal for lightweight applications running inside VS Code.

IV. METHOD

 Data Collection:
- Collect user details like name and course for certificate issuance.
- Store the data in an SQLite database.

 Blockchain Integration:
- Generate a unique SHA-256 hash for each certificate to simulate blockchain-like security.
- Store the hash in the database along with the user's certificate data.

 Certificate Generation:
- When a user requests a certificate, generate a personalized certificate (e.g., PDF) with the stored data.
- Attach the hash to the certificate for validation.

 Certificate Validation:
- When validating, compare the certificate's hash with the stored hash in the database.
- If they match, the certificate is valid; otherwise, it's invalid.

Test Method	Expected Result	Observed Result	Pass/Fail
Generate Certificate ()	Generate a certificate with a unique hash	Successfully generates certificate	Pass
Store Certificate ()	Save the certificate and hash in the database	Certificate saved to SQLite database	Pass
Validate Certificate ()	Compare certificate hash with stored hash	Hashes match; the certificate is valid	Pass
Blockchain Integration ()	Store and validate hash in the database	Hash stored and validated correctly	Pass

Table 1. Methods

V. RESULTS AND DISCUSSION

The system successfully generated certificates with unique SHA-256 hashes, which were used to simulate blockchain-like security for certificate validation. By storing these hashes in an SQLite database, the system ensured that each certificate was secure and could not be tampered with. When validating certificates, the system compared the hash of the input certificate with the one stored in the database, and only certificates with matching hashes were considered valid, ensuring their authenticity. The use of SQLite for data management was effective for this project, offering a lightweight and efficient solution for storing user data and certificates. This choice allowed for seamless interaction between the front end and back end, without the complexity of managing external servers. The database not only stored the user details but also ensured that all certificates and their corresponding hashes were safely stored and could be easily retrieved for validation. In terms of security, the system proved to be robust. Since any modification to a certificate would alter its hash, the system's design prevented any potential forgery or unauthorized alteration of certificates. The SHA-256 hashing algorithm added a layer of security, making it highly resistant to attacks.

The user interface was designed using Bootstrap, providing a simple and responsive layout that ensured compatibility across various devices. The system's front end was intuitive, making it easy for users to generate and validate certificates with minimal effort. Overall, the project successfully achieved its goal of integrating blockchain-like security into a certificate validation system, offering both reliability and user-friendly interaction.

VI. CONCLUSION

In conclusion, the integration of blockchain-like security with certificate validation has successfully enhanced the integrity and authenticity of the certificates generated through our system. By leveraging SHA-256 hashing and SQLite for secure data management, we have created a reliable and efficient method for certificate issuance and verification. The system ensures that certificates cannot be tampered with, providing a secure solution for access control. The user interface, designed with Bootstrap, offers a seamless experience, making the process of generating and validating certificates simple and intuitive. Overall, this project demonstrates how blockchain principles can be applied to everyday applications, enhancing security without the need for complex infrastructure.

VII. ACKNOWLEDGEMENTS

Any achievement, be it scholastic or otherwise does not depend solely on individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

First and foremost, we would like to thank Dr. Mahendra K V, Principal, RRIT, Bengaluru, for his moral support towards the completion of our project work.

We express our gratitude to Dr. Erappa G, Head, Department of ISE, RRIT, Bengaluru, who has always been a great source of inspiration.

We are grateful to the guidance and encouragement given by our guide Mrs. Nishchitha T S, Assistant Professor and Guide, Department of ISE, RRIT, Bengaluru, who also helped us in various stages of the project, optimizing the report and making this project a success.

We also extend our sincere thanks to the teaching and non-teaching faculty of the Department of ISE, RRIT, Bangalore, who have constantly supported us throughout our project work.

Last, but not least, we would like to express our deep sense of thanks to family and friends for their moral support in improving the project work.

VIII. REFERENCES

[1] **Alam, Shadab.** "Digital Certificate Verification Process Using Blockchain." *ResearchGate*, 2024.
Journal: *IEEE Access*, Volume 12, 2024.
DOI: 10.1109/ACCESS.2024.9268049.

[2] **Kadam, Sahil.** "Certificate Validation Using Blockchain." *International Journal of Computer Science and Information Security (IJCSIS)*, 2023.
Journal: *IJCSIS*, Volume 21, Issue 7, July 2023.
Available at: https://www.ijcsis.com.

[3] **Parthiban, Kumutha.** "Model of Blockchain-Based Certificate Verification System." *Journal of Computer Science and Technology*, 2023.
Journal: *Journal of Computer Science and Technology*, Volume 38, Issue 1, 2023.
DOI: 10.1007/s11390-023-2375-3.

[4] **"Blockchain-Based Certificate Validation Abstract."** *Studocu*, 2023.
Journal: *International Journal of Computer Applications (IJCA)*, Volume 184, Issue 6, 2023.
Available at: https://www.ijcaonline.org.

[5] **"Certificate Verification System Using Blockchain and QR Code."** *International Journal of Engineering and Technology*, 2023.
Journal: *International Journal of Engineering and Technology*, Volume 15, Issue 3, 2023.
DOI: 10.14419/ijet.v15i3.4552.