Certificate Verification Portal
A **Blockchain-Based Solution** with Admin-Controlled Features

Team Members: Abdel Taeha and Mohamed Burhan

Course: CSC196D Introduction to Blockchain (SPRING 2025)

Real-World Centralized Application:

Parchment – https://www.parchment.com/

- They issue and verify academic transcripts, certificates, diplomas online.

Table of Contents

---

1. Introduction

Our Certificate Verification Portal transforms centralized certificate management into a **decentralized application (DApp)** built on blockchain technology. Traditional systems rely on a central authority to maintain certificate records—making them vulnerable to fraud, tampering, and inefficient verification. By leveraging blockchain's **immutable ledger**, our solution provides:

- **Enhanced Security**: Certificate data is tamper-resistant.

- **Transparency**: All certificate transactions are publicly auditable.
- **Instant Verification**: Eliminates the need for third-party intermediaries.

The portal features:

1. An **admin interface** for certificate issuance and management.
2. A **public verification portal** where anyone can confirm a certificate's authenticity.

We use **Next.js** for the frontend, **Solidity** for smart contracts, and deploy on **BNB TestNet**. This seamless integration delivers a user-friendly experience underpinned by robust blockchain security.

---

2. Motivation

Traditional certificate verification faces three critical challenges:

1. **Slow Process**
   Manual checks can take days or weeks, delaying employment or academic admissions.
2. **Security Risks**
   Centralized databases are prime targets for tampering and cyberattacks.
3. **Inefficiency**
   Multiple intermediaries cause bottlenecks, increase costs, and raise the likelihood of human error.

Blockchain Benefits

- **Enhanced Security**: Immutable records prevent unauthorized modifications.
- **Streamlined Process**: Automated checks deliver near-instant certificate validity.
- **Global Accessibility**: 24/7 verification from anywhere in the world.
- **Transparency**: All updates are recorded on-chain for an auditable history. ● **Cost Efficiency**: Eliminates expensive third-party verification services.

---

3. System Actors

Our system includes three primary actor roles:

Admins

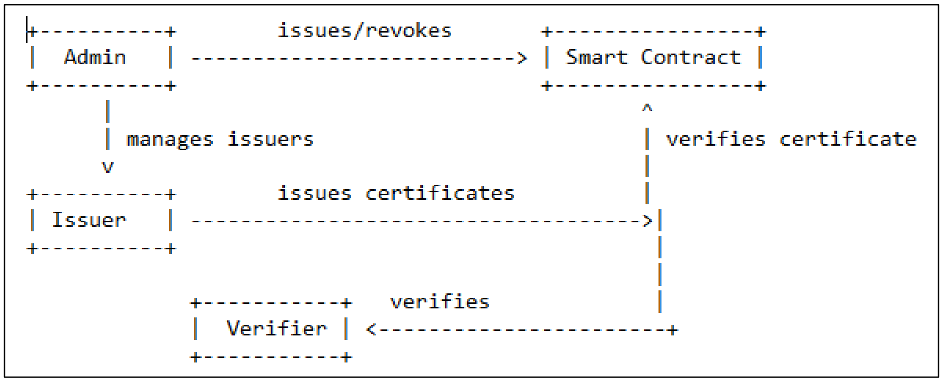
- Have full control over the entire verification platform.
- Can add or remove other administrators and issuers.
- Directly issue and revoke certificates.
- Monitor system activity through an administrative dashboard.
- Granted special privileges in the smart contract.

Issuers

- Typically educational institutions or professional certification bodies.
- Upload certificate information securely to the blockchain.
- Generate verifiable credentials and revoke them if necessary.
- Must be approved by admins; tracked in the smart contract mapping. **Verifiers**

- Employers, institutions, or anyone needing to verify a certificate.
- Access the public portal without special permissions.
- Simply enter a certificate ID to confirm authenticity.
- Smart contract provides read-only access via public view functions.

System Actors Diagram:



```
+----------+        issues/revokes       +----------------+
|  Admin   | --------------------------> | Smart Contract |
+----------+                             +----------------+
      |                                          ^
      | manages issuers                          | verifies certificate
      v                                          |
+----------+        issues certificates          |
| Issuer   | ----------------------------------->|
+----------+                                      |
                                                  |
             +-----------+   verifies             |
             |  Verifier | <-----------------------+
             +-----------+
```

Figure 1: Interaction between Admins, Issuers, Verifiers, and the Smart Contract

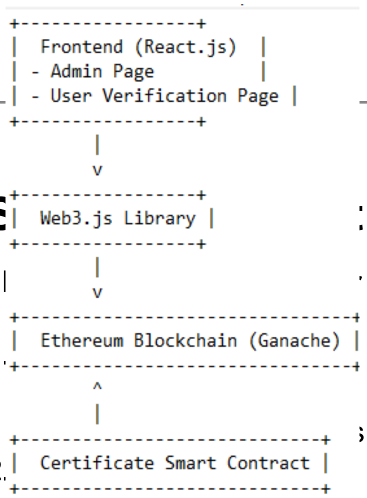
4. Application Design

The application consists of two main components:

- **Frontend Web Interface**
- **Backend Smart Contract** deployed on the Ethereum blockchain.

These components interact using **Web3.js** library to read from and write to the blockchain.

Architecture overview



```
+-----------------+
|  Frontend (React.js)  |
| - Admin Page          |
| - User Verification Page |
+-----------------+
        |
        v
+-----------------+
|  Web3.js Library |
+-----------------+
        |
        v
+--------------------------------+
|  Ethereum Blockchain (Ganache) |
+--------------------------------+
        ^
        |
+----------------------------+
|  Certificate Smart Contract |
+----------------------------+
```

- The Admin Page allows authorized users (admins and issuers) to issue, revoke, and manage certificates.

- The User Page allows anyone to verify the authenticity of a certificate.

- Web3.js handles blockchain communication from the UI.

- Ganache or a public testnet like Sepolia hosts the deployed smart contract.

- All operations (issue, revoke, verify) interact directly with the blockchain to ensure immutability and transparency.

Figure 1: Application Design and Blockchain Interaction

5. S ture

This l architecture.

1. sk for wallet authentication
2.
    - ○ Web3.js for communication with the blockchain
    - ○ Optional IPFS or similar for storing large data off-chain
3. **Blockchain Layer**
    - ○ BNB TestNet environment
    - ○ Solidity smart contracts for certificate logic
4. **Data Storage**
    - ○ Hybrid approach using on-chain for critical data
    - ○ Potential off-chain or IPFS for large file storage

Data Flow

Certificate Issuance

- Admin or Issuer authenticates via MetaMask.
- Certificate details are submitted and recorded on-chain.
- The contract emits an event confirming successful issuance.

Certificate Verification

- Verifier inputs the certificate ID in the public portal.
- The contract retrieves verification status and details.
- Results display instantly, indicating validity or revocation.

6. Smart Contract Implementation

Our **Solidity** contract enforces role-based access control and secure certificate management.

Core Data Structures

```
// Certificate structure
struct Certificate {
string recipientName;
string certificateId;
string courseName;
address issuer;     uint
issuanceDate;     bool
isValid;
}

// Mappings
mapping(string => Certificate) public certificates;
mapping(address => bool) public issuers; mapping(address
=> bool) public admins;
```

Key Functions

Administrator Management

```
function addAdmin(address _admin) public onlyAdmin {
require(!admins[_admin], "Already an admin");
    admins[_admin] = true;
    emit AdminAdded(_admin, msg.sender); }
```

Certificate Issuance

```
function issueCertificate(
string memory _recipientName,
string memory _certificateId,
string memory _courseName )
public onlyIssuerOrAdmin {
    require(bytes(certificates[_certificateId].certificateId).length == 0,
            "Certificate ID exists");

    certificates[_certificateId] = Certificate({
        recipientName: _recipientName,
certificateId: _certificateId,
courseName: _courseName,         issuer:
msg.sender,         issuanceDate:
block.timestamp,
        isValid: true
    });
    emit CertificateIssued(_certificateId, _recipientName, msg.sender); }
```

Certificate Verification

```
function verifyCertificate(string memory _certificateId)
public     view     returns (         bool exists,
        string memory recipientName,
string memory courseName,
        address issuer,
uint issuanceDate,
bool isValid
    )
{
    Certificate memory cert = certificates[_certificateId];

    if (bytes(cert.certificateId).length == 0) {
return (false, "", "", address(0), 0, false);
    }
return (
true,
        cert.recipientName,
cert.courseName,         cert.issuer,
cert.issuanceDate,
cert.isValid
```

```
    ); }
```

Security Highlights

- **Role-based modifiers** ensure only permitted addresses can add issuers or issue certificates.
- **Input validation** to prevent overwriting existing certificates.
- **Event emissions** for real-time updates and transparency.
- **Gas optimization** by storing only essential data on-chain.

---

7. UI Implementation

We provide two distinct UIs:

Admin Dashboard

- **Restricted Access**: Requires MetaMask authentication.
- **Certificate Issuance**: Simple form to enter certificate data.
- **Revocation & Management**: Admin can revoke certificates and manage issuers.
- **Admin Controls**: Add or remove other admins.

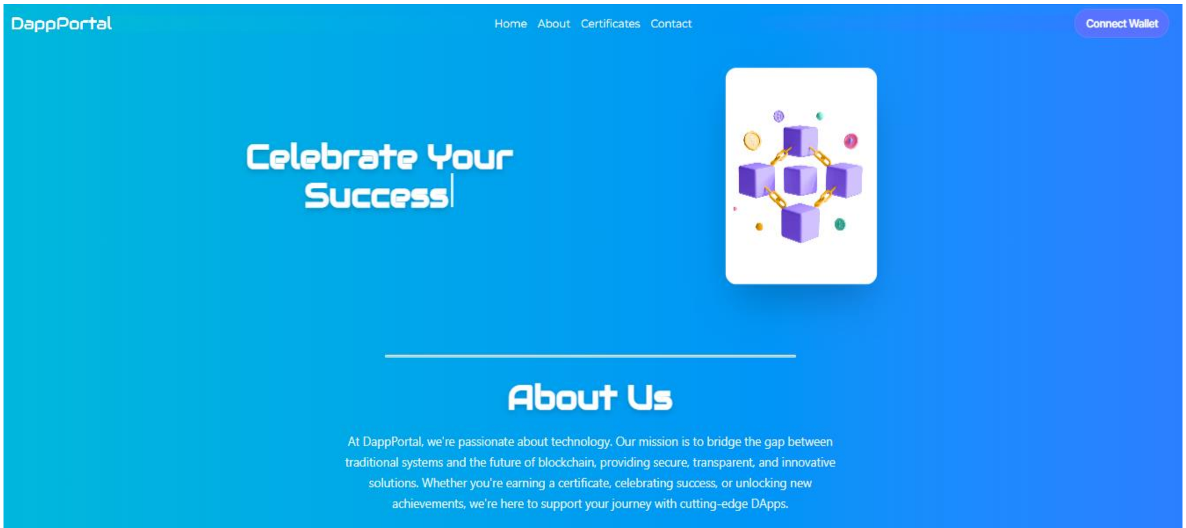
Verification Portal

- **Public Access**: No wallet needed.
- **Certificate Lookup**: Users enter the certificate ID.
- **Instant Status**: Displays certificate details and validity.
- **ByteCode Sharing**: Optional mechanism for secure, shareable credentials.

UX Priorities

- **Responsiveness** across multiple device types and screen sizes.
- **Intuitive Navigation** with clear labels.
- **Error Handling** to guide users if something goes wrong.
- **Simplicity** so non-technical users can easily verify certificates.

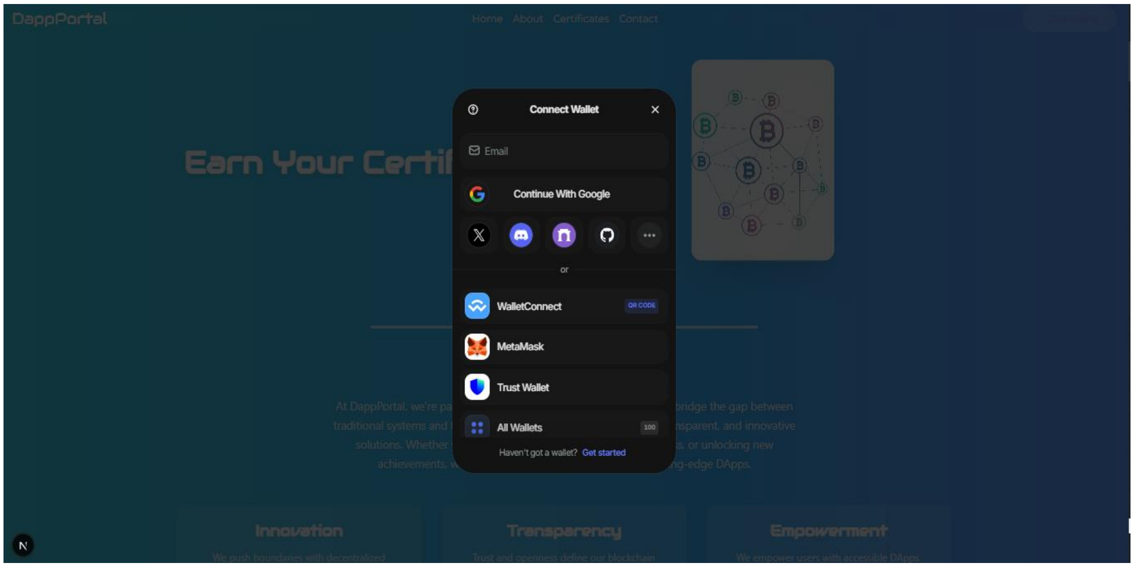
<u>Screenshots</u>



Celebrate Your Success

About Us

At DappPortal, we're passionate about technology. Our mission is to bridge the gap between traditional systems and the future of blockchain, providing secure, transparent, and innovative solutions. Whether you're earning a certificate, celebrating success, or unlocking new achievements, we're here to support your journey with cutting-edge DApps.



Earn Your Certif

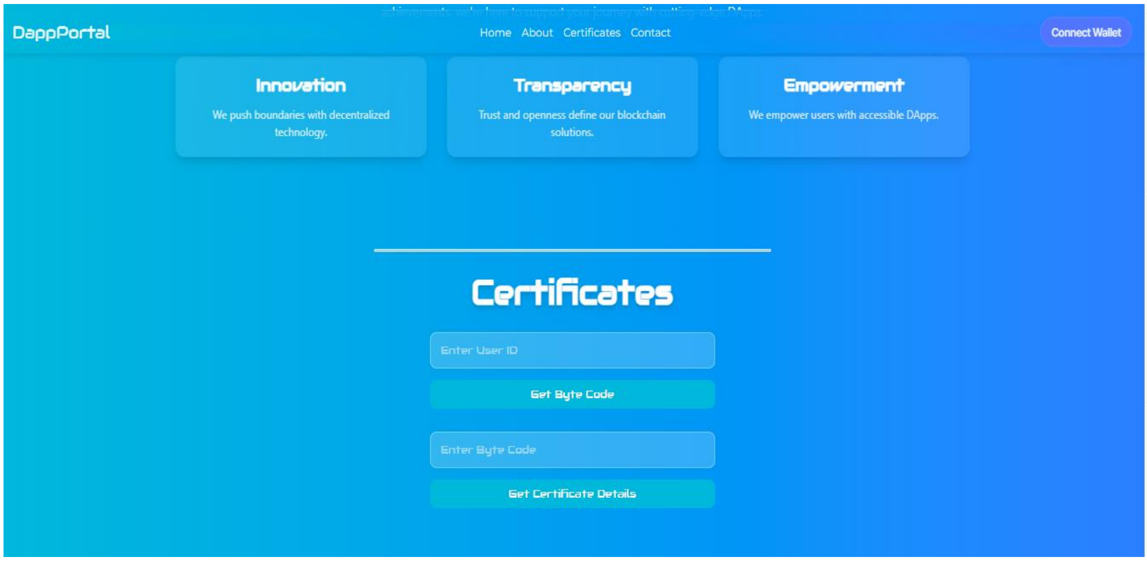
Haven't got a wallet? Get started

At DappPortal, we're pa bridge the gap between traditional systems and t nsparent, and innovative solutions. Whether s, or unlocking new achievements, w ng-edge DApps.

We push boundaries with decentralized

Trust and openness define our blockchain

We empower users with accessible DApps.



achievements, we're here to support your journey with cutting-edge DApps.

Innovation

We push boundaries with decentralized technology.

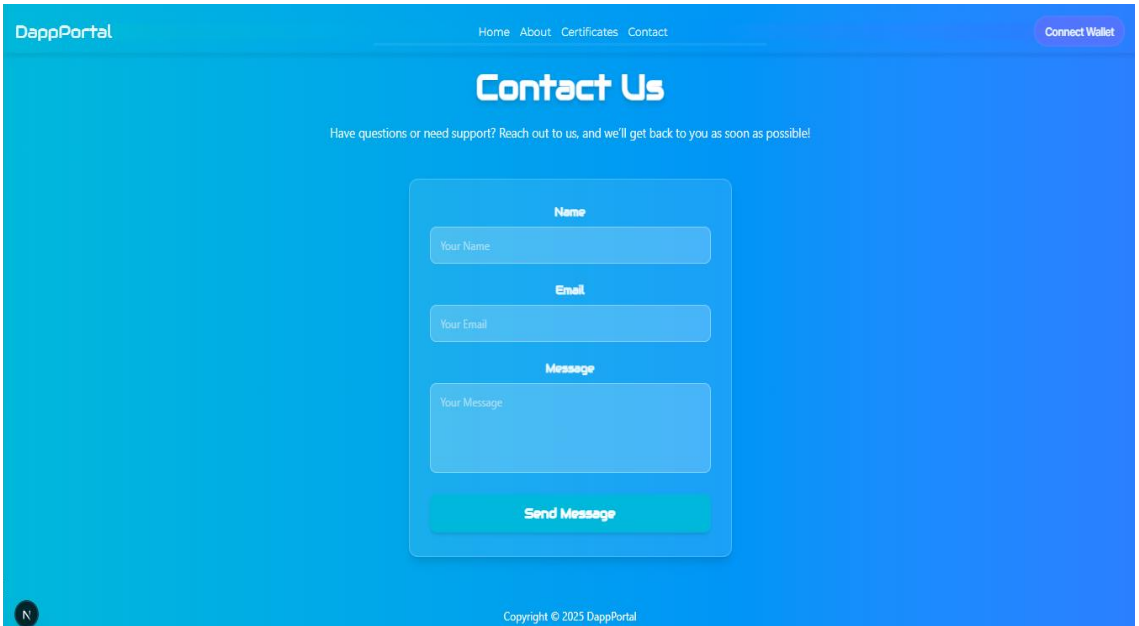
Transparency

Trust and openness define our blockchain solutions.

Empowerment

We empower users with accessible DApps.

Certificates



Contact Us

Have questions or need support? Reach out to us, and we'll get back to you as soon as possible!

Name

Email

Message

Copyright © 2025 DappPortal

8. Project Timeline

A structured 8-week schedule guided our development:

Week	Dates	Milestone	Status
1	Mar 12–18	Project Proposal	Completed
2	Mar 19–22	Technical Planning	Completed
3–4	Mar 23–27	Design & Initial Impl.	Completed
5–6	Mar 28–Apr 7	Full Implementation	Completed
7	Apr 8–12	Testing & Optimization	Completed
8	Apr 13–17	Documentation & Presentation	Completed

This phased approach ensured an organized workflow, ample testing, and timely deliverables.

9. Performance Analysis

We ran both **manual and automated** tests to measure efficiency:

Gas Consumption

- **Issuance**: 196,253 gas ($0.98 at 5 Gwei)
- **Revocation**: 31,876 gas ($0.16 at 5 Gwei)
- **Verification**: View function (no gas cost for user)

Transaction Times (BNB TestNet averages)

- **Issuance**: ~4.8 seconds
- **Revocation**: ~3.6 seconds

- **Verification**: ~1.2 seconds

Scalability

- Handled up to 500 certificates with minimal performance impact.
- Verification times stayed under 1.5 seconds, even at high volumes.

10. Roles and Responsibilities

Team Members:

- **Abdel Taeha**
  - Frontend Development
  - Website Functionality
  - User Interface (UI) Design
  - Web3.js Integration
  - User Experience Testing
  - Application Deployment
- **Mohamed Burhan**
  - Smart Contract Development
  - Solidity Programming
  - Contract Security Implementation
  - Gas Optimization
  - Blockchain Testing
  - Testnet Deployment

11. Links & Steps to Run

The complete source code, smart contracts, and project files are available at:

GitHub Repository:
https://github.com/abdeltaehass/CSC196D.git

Steps to run(provided in the GitHub repo readme but will also include below):

1. Clone the Repository
   git clone https://github.com/abdeltaehass/CSC196D.git

2. Install Dependencies
   cd CSC196D

```
npm install
```

3. Start Local Blockchain (Ganache CLI)
   ```
   ganache-cli
   ```

4. Compile and Deploy Smart Contracts
   ```
   truffle compile
   truffle migrate
   ```

5. Run the Frontend
   ```
   npm run dev
   ```

---

12. Conclusion

Our **Certificate Verification Portal** demonstrates the power of blockchain in creating a **secure**, **transparent**, and **user-friendly** system for credential management. Key takeaways include:

1. **Tamper-Proof Certificates**: Stored on an immutable ledger.
2. **Instant Verification**: Eliminates delays associated with manual checks.
3. **Role-Based Access**: Protects system integrity by assigning privileges.
4. **User-Centric Design**: Hides blockchain complexity, providing a straightforward interface.
5. **Cost-Effective Deployment**: Low gas fees and minimal infrastructure overhead.

Overall, this DApp showcases how decentralized technology can replace traditional, centralized approaches, providing faster, safer, and more reliable certificate verification for institutions and individuals worldwide.