**Student Certificate Validation DApp Documentation**

The Student Certificate Validation DApp is a decentralized application that enables educational authorities (admins) to upload student certificates to IPFS and the Ethereum blockchain, and allows institutions to verify the authenticity of those certificates. This system ensures secure, immutable, and transparent verification using blockchain and IPFS technology.

**Technology Stack**

* **Frontend:** React.js with TypeScript and Tailwind CSS
* **Backend:** Node.js with Express
* **Blockchain:** Ethereum (Ganache for local development)
* **Smart Contract:** Solidity (Truffle Framework)
* **Decentralized Storage:** IPFS (via Pinata)
* **Database:** MySQL or PostgreSQL (via Sequelize ORM)

**Application Roles**

**Admin**

* Can log in using hardcoded or stored credentials.
* Uploads student certificates with details:
  + Student Name
  + Degree Name
  + Graduation Date
  + Institution Name
  + Certificate File (PDF/Image)
* Certificate data is:
  + Uploaded to IPFS (file)
  + Stored on Ethereum blockchain (metadata)
  + Stored in SQL database for querying

**Institution**

* Can sign up and log in securely.
* Can verify certificates using an IPFS hash.
* Receives status and details of certificate if valid.

**How the App Works**

**1. Admin Login and Upload**

1. Admin logs in from /admin-login.
2. JWT token is returned and stored in frontend.
3. Admin uploads certificate (via form + file upload).
4. Certificate is sent to backend:
   * File is uploaded to IPFS
   * Metadata stored on blockchain via smart contract
   * Record saved in SQL database

**2. Institution Signup and Verification**

1. Institution signs up at /institution-auth.
2. JWT token is issued after login.
3. On verification page:
   * Institution enters an IPFS hash
   * Backend verifies hash by reading blockchain
   * Returns student details if matched, else error

**Testing the Backend API**

Use **Postman** or **cURL** to test the following endpoints:

**1. Signup (Institution)**

* **POST** /api/auth/signup

{

"name": "Test College",

"email": "college@example.com",

"password": "test123"

}

**2. Login (Admin/Institution)**

* **POST** /api/auth/login

{

"email": "admin1234@gmail.com",

"password": "12345"

}

* Response: JWT token

**3. Upload Certificate (Admin Only)**

* **POST** /api/certificates/upload
* Headers:
  + Authorization: Bearer <admin\_token>
  + Content-Type: multipart/form-data
* Body:
  + studentName: string
  + degreeName: string
  + graduationDate: string
  + institutionName: string
  + certificate: file (PDF/Image)

**4. Verify Certificate (Institution Only)**

* **POST** /api/certificates/verify

{

"ipfsHash": "Qm..."

}

* Headers:
  + Authorization: Bearer <institution\_token>

**5. Check Server is Running**

* Open browser or use: curl http://localhost:5000
* Should respond or redirect

**Developer Setup**

**1. Smart Contract**

cd smart-contract

truffle migrate --reset --network development

**2. Backend**

cd backend

npm install

cp .env.example .env # Edit your credentials

node server.js

**3. Frontend**

cd cert-validation-frontend

npm install

npm start

**Environment Variables**

Create .env in backend/:

PORT=5000

JWT\_SECRET=your\_jwt\_secret\_here

DB\_HOST=localhost

DB\_USER=root

DB\_PASSWORD=yourpassword

DB\_NAME=certificatevalidationsystem

PINATA\_API\_KEY=your\_key

PINATA\_API\_SECRET=your\_secret

GANACHE\_URL=http://127.0.0.1:7545

**Notes**

* Ensure Ganache is running before backend
* Always use correct contract ABI in backend
* Use IPFS hash from upload for verification
* Admin token required for upload, institution token for verification