**PRACTICAL: 9.1**

**AIM:** Create Dapp (Decentralized Application) and link your client-side application with Blockchain network created in practical 5 th and 6 th .

**CODE: Event Certificate**

**Contract.js**

import web3 from "./Web3";

import EventCertificate from "./build/contracts/EventCertificate.json"; // Import ABI JSON

const contractAddress = "0x849df1B513479A5F98Faf80786192ab5869bd4cA";

const contract = new web3.eth.Contract(EventCertificate.abi, contractAddress);

export default contract;

**Web3.js**

import Web3 from "web3";

let web3;

if (window.ethereum) {

  web3 = new Web3(window.ethereum);

  window.ethereum.request({ method: "eth\_requestAccounts" });

} else {

  console.log("Please install MetaMask!");

}

export default web3;

**CertificateDisplay.js**

import React from 'react';

import web3 from '../Web3';

import certificateBackground from '../styles/certificate-bg.png';

import certificateSeal from '../styles/seal.png';

const CertificateDisplay = ({ certificate, downloadCertificate, transactionHash }) => {

  if (!certificate) return null;

  const hashDisplay = transactionHash ||

    web3.utils.sha3(`${certificate.id}\_${certificate.studentName}\_${Date.now()}`);

  return (

    <div className="certificate-wrapper">

      <div className="certificate-container">

        <div className="certificate-inner">

          <div className="certificate-content">

            <h2 className="certificate-title">Certificate of Achievement</h2>

            <div className="certificate-body">

              <p className="certificate-text">This is to certify that</p>

              <h3 className="student-name">{certificate.studentName}</h3>

              <p className="certificate-text">has successfully completed</p>

              <h4 className="course-name">{certificate.courseName}</h4>

              <div className="certificate-details">

                <p>Branch: {certificate.branch}</p>

                <p>Grade Achieved: {certificate.grade}</p>

                <p>Certificate ID: {certificate.id}</p>

              </div>

              <div className="certificate-hash">

                <small>Blockchain Transaction Hash:</small>

                <br />

                {hashDisplay}

              </div>

            </div>

            <div className="certificate-seal">

              <img src={certificateSeal} alt="Official Seal" />

            </div>

          </div>

        </div>

      </div>

      <div className="download-container">

        <button className="download-btn" onClick={downloadCertificate}>

          Download Certificate

        </button>

      </div>

    </div>

  );

};

export default CertificateDisplay;

**CertificateForm.js**

import React from 'react';

const CertificateForm = ({

setCertificateId,

setStudentName,

setBranch,

setCourseName,

setGrade,

issueCertificate,

verifyCertificate,

fetchCertificate

}) => {

return (

<div className="form-container">

<div className="input-group">

<input

type="text"

placeholder="Certificate ID"

onChange={(e) => setCertificateId(e.target.value)}

/>

<input

type="text"

placeholder="Student Name"

onChange={(e) => setStudentName(e.target.value)}

/>

<input

type="text"

placeholder="Branch"

onChange={(e) => setBranch(e.target.value)}

/>

<input

type="text"

placeholder="Course Name"

onChange={(e) => setCourseName(e.target.value)}

/>

<input

type="text"

placeholder="Grade"

onChange={(e) => setGrade(e.target.value)}

/>

</div>

<div className="button-group">

<button className="issue-btn" onClick={issueCertificate}>

Issue Certificate

</button>

<button className="verify-btn" onClick={verifyCertificate}>

Verify Certificate

</button>

<button className="fetch-btn" onClick={fetchCertificate}>

Fetch Certificate

</button>

</div>

</div>

);

};

export default CertificateForm;

**App.js**

import React, { useState } from "react";

import web3 from "./Web3";

import contract from "./contract";

import jsPDF from "jspdf";

import "./styles/Certificate.css";

import CertificateForm from "./components/CertificateForm";

import CertificateDisplay from "./components/CertificateDisplay";

function App() {

  const [certificateId, setCertificateId] = useState("");

  const [studentName, setStudentName] = useState("");

  const [branch, setBranch] = useState("");

  const [courseName, setCourseName] = useState("");

  const [grade, setGrade] = useState("");

  const [message, setMessage] = useState("");

  const [certificate, setCertificate] = useState(null);

  const issueCertificate = async () => {

    const accounts = await web3.eth.getAccounts();

    setMessage("Issuing Certificate...");

    try {

      await contract.methods

        .issueCertificate(certificateId, studentName, branch, courseName, grade)

        .send({ from: accounts[0] });

      setMessage(`Certificate Issued Successfully for ${studentName}`);

    } catch (error) {

      setMessage("Error issuing certificate.");

    }

  };

  const verifyCertificate = async () => {

    try {

      const verified = await contract.methods

        .verifyCertificate(certificateId, studentName, branch, courseName, grade)

        .call();

      setMessage(verified ? "✅ Certificate Verified!" : "❌ Invalid Certificate!");

    } catch (error) {

      setMessage("Error verifying certificate.");

    }

  };

  const fetchCertificate = async () => {

    try {

      const cert = await contract.methods.getCertificate(certificateId).call();

      setCertificate({

        id: cert[0],

        studentName: cert[1],

        branch: cert[2],

        courseName: cert[3],

        grade: cert[4],

      });

      setMessage("");

    } catch (error) {

      setCertificate(null);

      setMessage("Certificate not found or invalid ID.");

    }

  };

  const downloadCertificate = () => {

    // Create a new jsPDF instance with landscape orientation

    const doc = new jsPDF({

      orientation: 'landscape',

      unit: 'mm',

      format: 'a4'

    });

    // Get the certificate element to capture

    const certificateElement = document.querySelector('.certificate-container');

    if (certificateElement) {

      // Use html2canvas to capture the styled certificate (you'll need to install this package)

      import('html2canvas').then(({ default: html2canvas }) => {

        html2canvas(certificateElement, { scale: 2 }).then(canvas => {

          const imgData = canvas.toDataURL('image/png');

          // Calculate dimensions to fit the PDF

          const imgWidth = 277; // A4 landscape width in mm

          const imgHeight = (canvas.height \* imgWidth) / canvas.width;

          // Add the image to the PDF

          doc.addImage(imgData, 'PNG', 10, 10, imgWidth, imgHeight);

          doc.save(`${certificate.studentName}\_Certificate.pdf`);

        });

      });

    } else {

      // Fallback to text if element not found

      doc.text("Certificate of Completion", 80, 20);

      doc.text(`This certifies that ${certificate.studentName}`, 60, 40);

      doc.text(`Branch: ${certificate.branch}`, 60, 50);

      doc.text(`Course: ${certificate.courseName}`, 60, 60);

      doc.text(`Grade: ${certificate.grade}`, 60, 70);

      doc.save(`${certificate.studentName}\_Certificate.pdf`);

    }

  };

  return (

    <div className="container">

      <h1 className="dapp-title">Blockchain Certificate Management</h1>

      <CertificateForm

        setCertificateId={setCertificateId}

        setStudentName={setStudentName}

        setBranch={setBranch}

        setCourseName={setCourseName}

        setGrade={setGrade}

        issueCertificate={issueCertificate}

        verifyCertificate={verifyCertificate}

        fetchCertificate={fetchCertificate}

      />

      {message && <div className="message">{message}</div>}

      <CertificateDisplay

        certificate={certificate}

        downloadCertificate={downloadCertificate}

      />

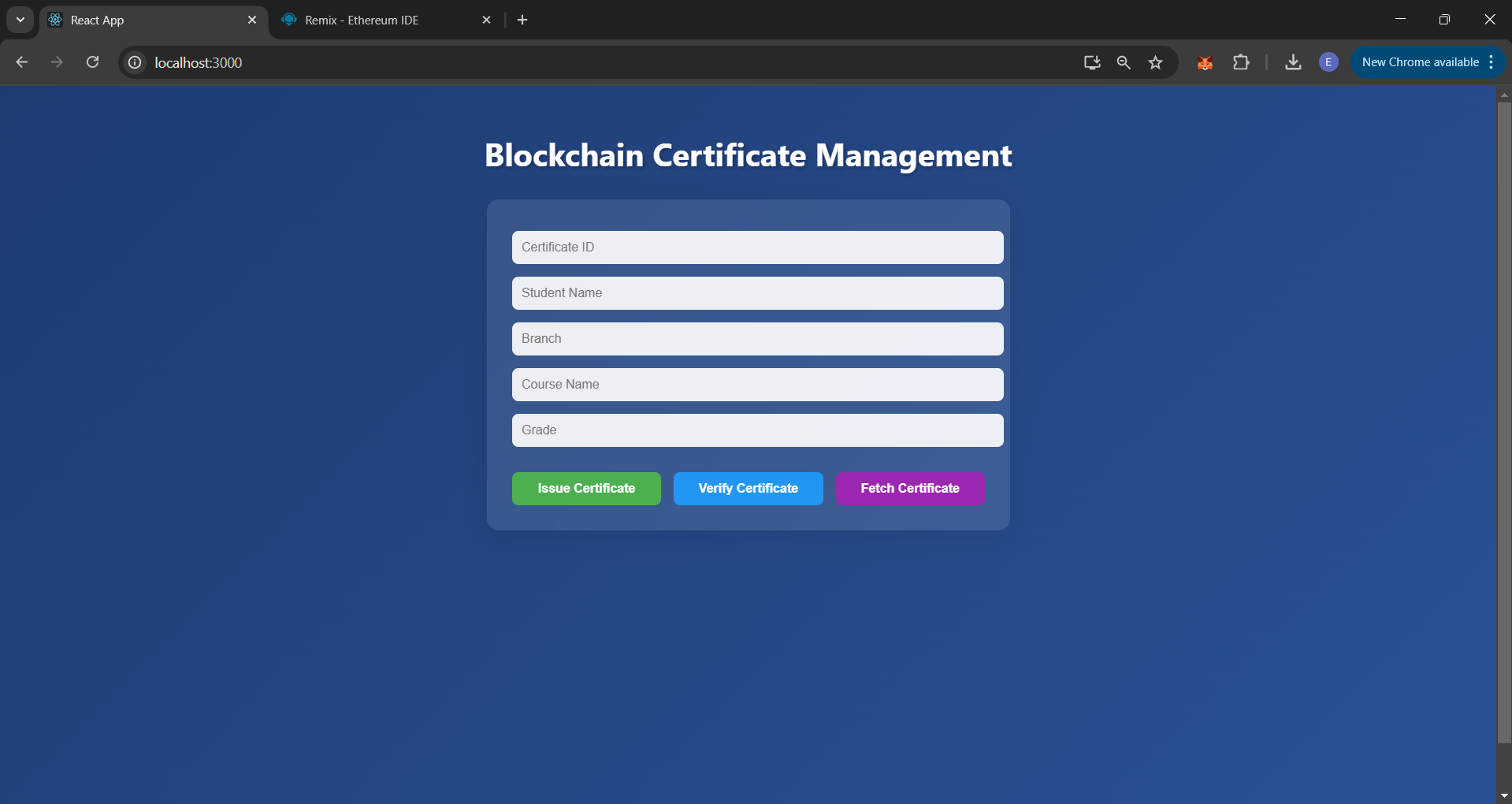
    </div>

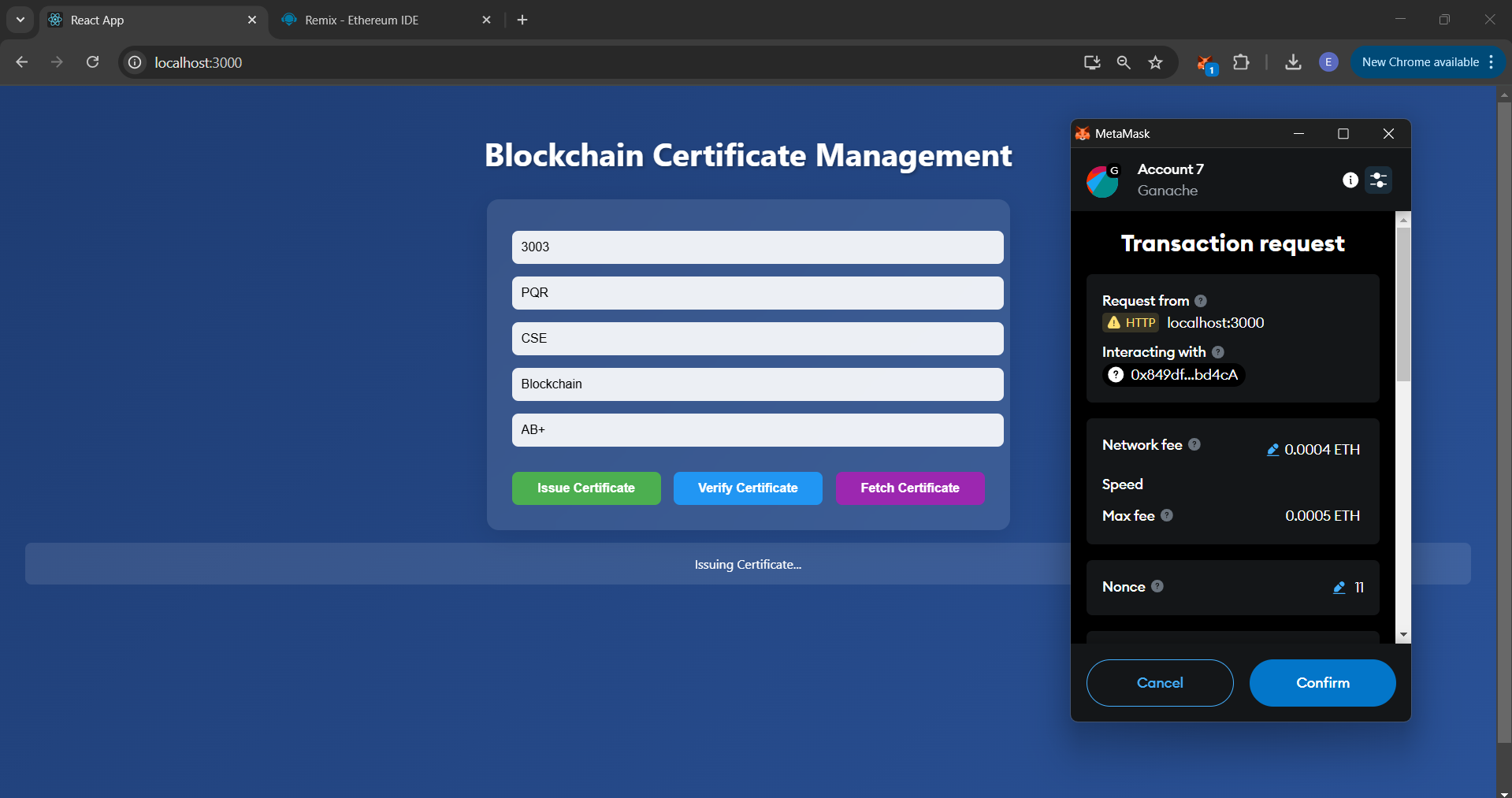
  );

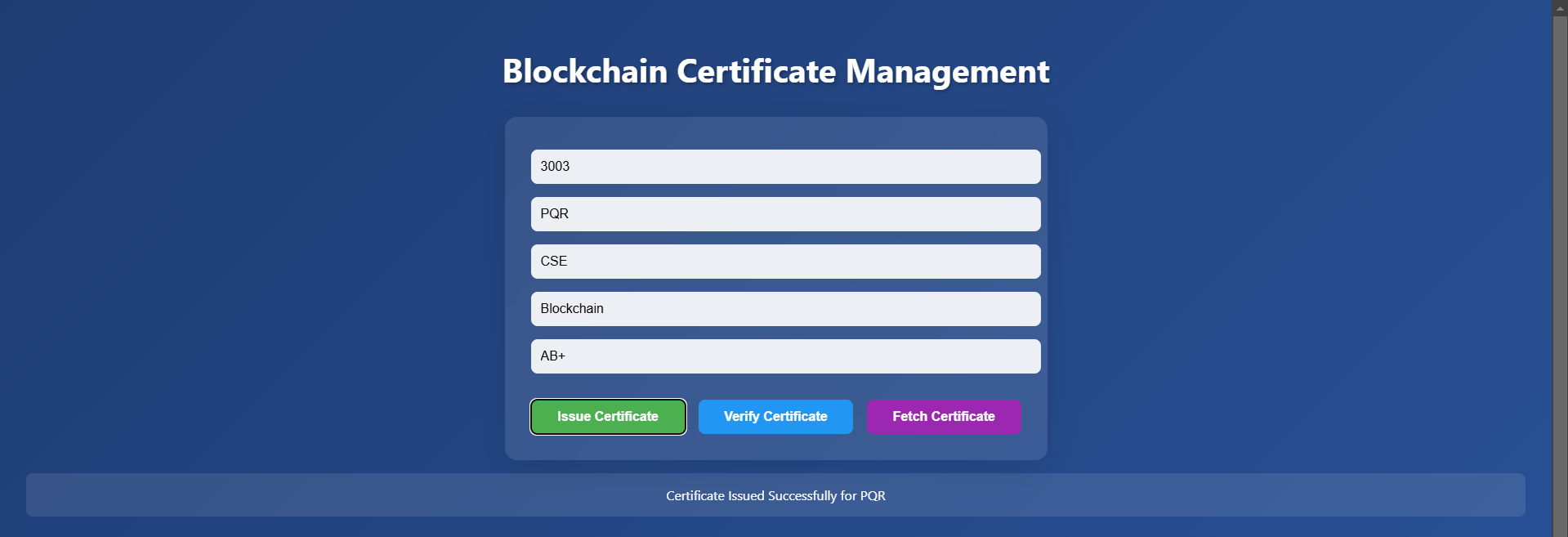
}

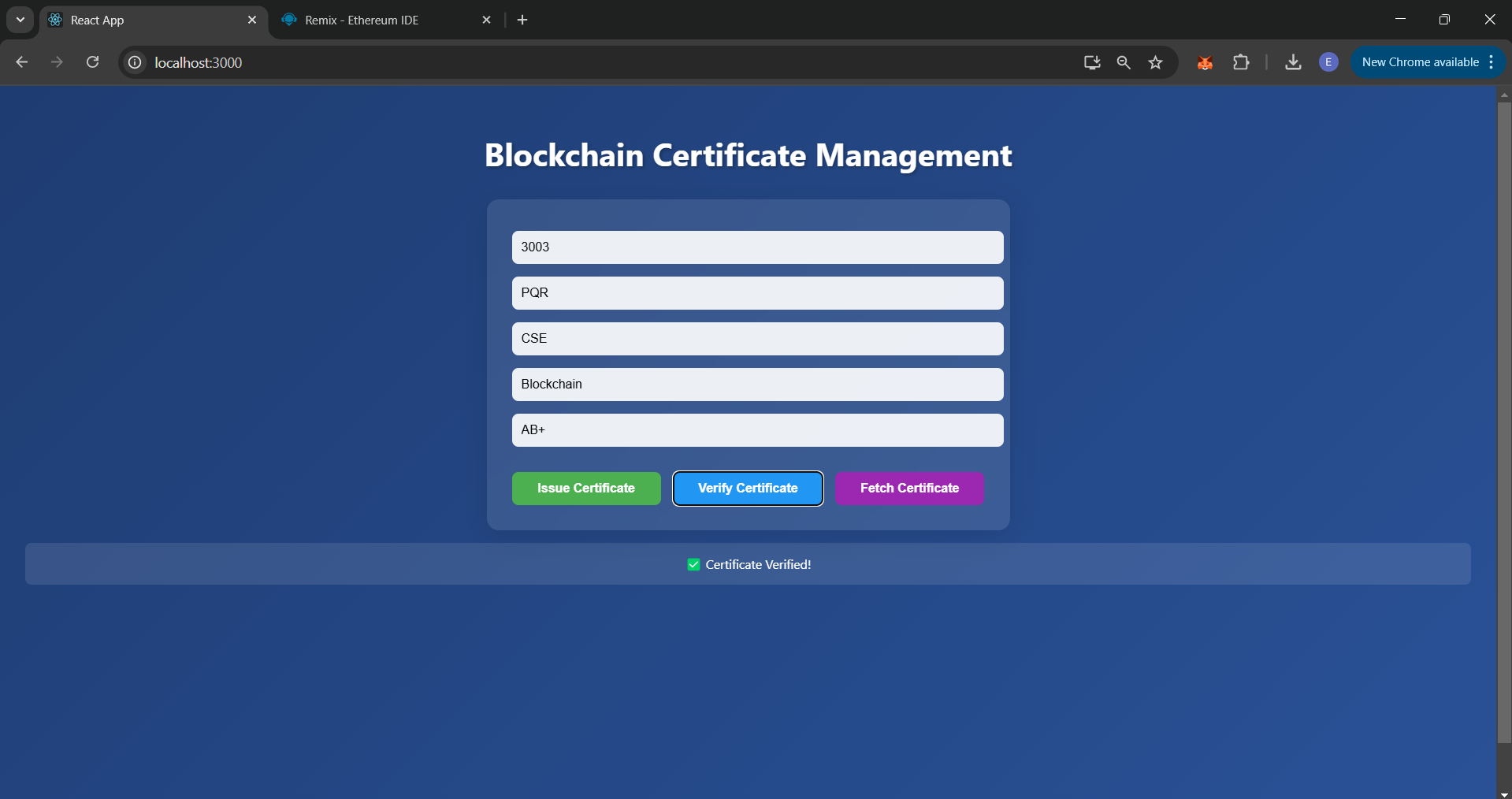
export default App;

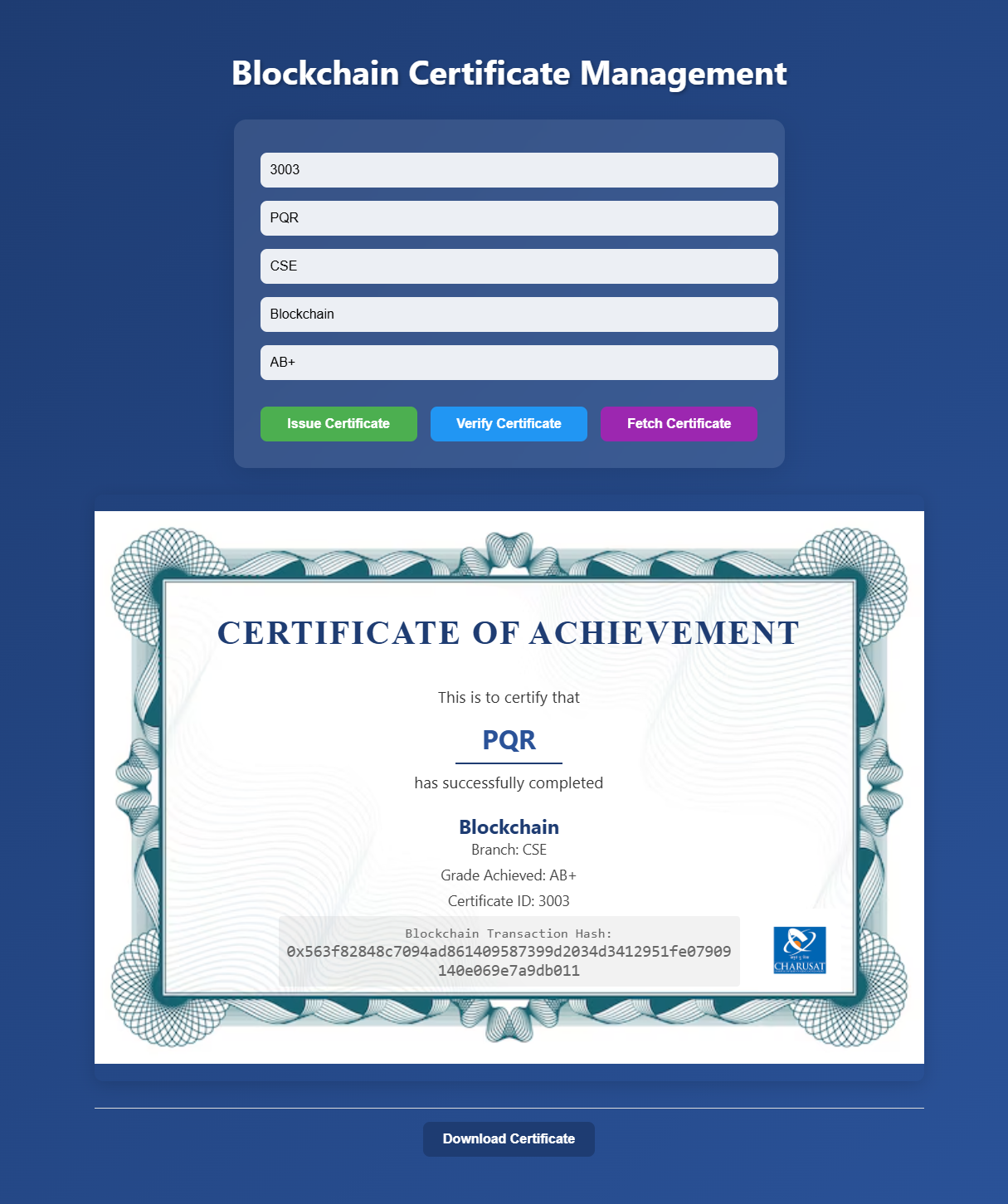
**OUTPUT:**

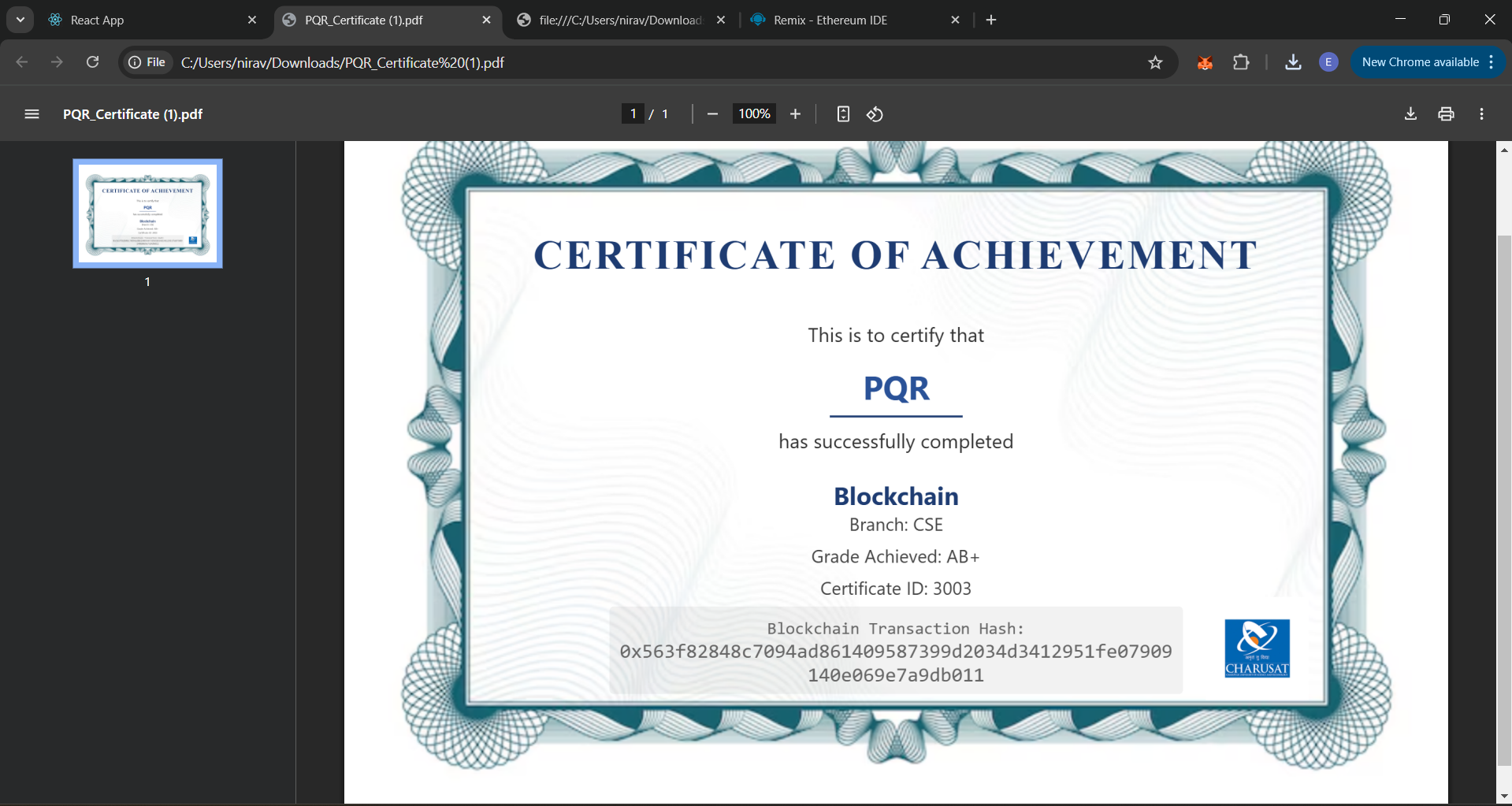












**LEARNING OUTCOME:**

A Decentralized Application (DApp) is built by developing both the client-side application (front-end) and back-end integration with smart contracts. This includes using Metamask for user authentication and performing secure blockchain transactions. The architecture of DApps is studied, understanding how they interact with blockchain networks and smart contracts. Real-world use cases for DApps, such as decentralized finance (DeFi) or identity management, are also explored.