School of Computer Science Engineering and Technology
Bennett University (The Times Group)

Program- BTech/BCA 4th Semester
Course Code- CSET226/CBCA218
Year- 2026
Date- 20/02/26

Type- Sp. Core-2
Course Name-Blockchain Engineering
Semester- Even
Batch- Blockchain

Lab Assignment 6

Exp No	Name	CLO Achieved				Marks
		CO1	CO2	CO3	CO4	
6	Build a full-stack Ethereum DApp.	√	√			2

Objective: Build a full stack Dapp on Ethereum to decentralize the student credentials to do the following:

- ✓ The admin of the University can add student credential records
- ✓ Students can view their credentials
- ✓ Credentials are tamper-proof and auditable
- ✓ Each credential update is recorded through events

Outcomes: After executing this assignment, the students will be able to design, code, compile, deploy, test and run a full stack Dapp.

Hands-on Learning (40 minutes)

Student Credentials Metadata

Category	Key Fields
The Subject	`student_name`, `student_id`, `date_of_birth`, `did`
The Hostel Details	`Hostel_name, room_no`
The Fee Details	`course_Fee, fee_paid, fee_balance, hostel_fee`
The Issuer	`institution_name`, `accreditation_body`, `issuer_address`, `official_signature`
The Achievement	`degree_type`, `major`, `gpa`, `competencies`, `issue_date`, `expiry_date`
The Evidence	`transcript_link`, `portfolio_url`, `hash_of_work`

Module 1: Write a smart contract in solidity to have following features:

- ✓ Add credential
- ✓ View credential
- ✓ Update credential
- ✓ Emit events for every operation
- ✓ Admin-only access control

Module 2: Backend Dev & Deployment on Hardhat

- ✓ Compile contract
- ✓ Run local node
- ✓ Deploy contract
- ✓ Export ABI & contract address

Module 3: Frontend (React & Web3.js)

- ✓ MetaMask wallet connect
- ✓ Show current account
- ✓ Show stored credentials
- ✓ Admin dashboard for add/update

Tasks for Assessment

Task 1: Smart Contract Design & Implementation (Core Logic)

Objective: Implement the credential storage contract.

Requirements:

- Write a Solidity smart contract `UniversityCredentials.sol in` contracts directory.
- Implement:
  - `addCredential()` → Only Admin can add student credentials.
  - `getCredential()` → Students can view their credentials.
  - Role-based access control (Admin vs Student).
- Use:
  - `mapping(address => Credential)`
  - `struct Credential { string degree; string grade; uint256 year; }`
- Deploy using Hardhat.

struct `Credential { string name; string course; string hash; uint issuedOn; }`
mapping: `address => Credential[]`
Functions:

Function	Access	Purpose
addCredential(student, name, course, docHash)	Admin only	Adds new credential
getCredentials(student)	Public view	Returns student credentials
updateCredential(student, index, newHash)	Admin only	Updates credential hash

Events:

- CredentialAdded
- CredentialUpdated

Task2: Event Logging & Auditability

Objective: Ensure tamper-proof and auditable updates.

Requirements:

- Define events:
- `event CredentialAdded(address indexed student, string degree, uint256 year);`
- `event CredentialUpdated(address indexed student, uint256 timestamp);`
- Emit events on every add/update.
- Use Hardhat tests to:
  - Verify event emission
  - Verify correct parameters
- Demonstrate event retrieval using:
  - Hardhat console
  - or Ethers.js/Web3.js filters

Compile: npx hardhat compile

Task3: Unit Testing using Hardhat & Chai.

Create: `test/Credential.test.js`

Test cases:

1. Admin can add credentials
2. Non-admin cannot add credentials
3. Student can view credentials
4. Admin can update credential
5. Event emitted correctly

Run: npx hardhat test

a) Start Local Blockchain: npx hardhat node
b) Deploy to localhost: npx hardhat run scripts/deploy.js --network localhost
c) Export ABI & Address: Copy ABI from artifacts, Store it in frontend folder, Store contract address in a config file

Task4: Frontend Integration (React and web3.js/Ether.js)

Objective: Build user interface for Admin and Students.

Requirements:

- React frontend with:
  - Admin login (MetaMask)
  - Student login (wallet-based)
- Admin Panel:
  - Add credential form
- Student Panel:
  - View credentials
- Display:
  - Transaction hash
  - Event logs

Do the following:

a) Create React App

```
cd ..; mkdir frontend; cd frontend; npm create vite@latest; npm install; npm install web3; npm run dev
```

b) MetaMask Connect Page: Frontend must connect wallet, show account, show chainId, show "wrong network" warning
c) Student Dashboard- Student enters address and can fetch credentials using `getCredentials(student)and d`isplay credentials in a table
d) Admin Dashboard- Admin must be able to add credential, update credential, and see event log updates
e) Event Listener (Web3.js)- Students must listen to: CredentialAdded and CredentialUpdated

Task5: Security & Integrity Assessment

Objective: Prove credentials are tamper-proof.

Requirements:

- Implement:
  - `onlyAdmin` modifier
  - Prevent unauthorized updates
- Add:
  - Credential hashing using `keccak256 (SHA3)`
  - Store hash of credential data
- Write test cases to:
  - Attempt unauthorized modification
  - Validate hash consistency

Task6: Deploy to Sepolia Testnet- Add Alchemy/Infura RPC, Add MetaMask testnet, Verify contract

Submission Instructions:

1. Submission requires the screen shots of all the incurred steps to execute a smart contract or a video showing the whole process.
2. All these files are in single zip folder.
3. Use the naming convention: Prog_CourseCode_RollNo_LabNo.docx (Example: BTech4thSem_CSET226_ E21CSEU002_Lab1)
4. Submission is through LMS only
5. The copied assignment will become automatically zero and attract penalty of -1 mark for each copied for all whosoever assignments are ditto same.