Centurion University
Shaping Lives... Empowering Communities...

School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement : Decentralized Identity – DID and Credential Demo

Objective/Aim:

To understand and demonstrate **Decentralized Identity (DID)** and **Verifiable Credentials (VCs)** using blockchain to enable user-controlled digital identity management.

.

Apparatus/Software Used:

1. MetaMask wallet
2. DID/VC platform (e.g., SpruceID, Ceramic, Polygon ID)
3. Node.js / Hardhat / Remix IDE
4. JSON-LD (for credential structure)
5. Blockchain network (e.g., Ethereum, Polygon ID testnet)

Theory:

Decentralized Identity (DID) allows users to own and control their digital identities without relying on centralized authorities.

Each DID is a **unique identifier** stored on blockchain, linking to cryptographically verifiable credentials (VCs) such as certificates, licenses, or memberships.

Key Concepts:

- **DID (Decentralized Identifier):**
  A unique, blockchain-based identifier owned by a user's wallet.
  Example: did:ethr:0xabc123...
- **Verifiable Credentials (VC):**
  Digital proofs issued by trusted entities, signed cryptographically (e.g., proof of student status, KYC verification).
- **Verifier:**
  Entity that checks authenticity of a credential without needing a central database.

Procedure:

Step 1: Key Generation (Identity Creation)

Tools used: `Node.js`, `elliptic`, `crypto`

- Every decentralized identity (DID) begins with a **public/private key pair**.
- The private key stays **secure** with the identity owner (you).
- The public key is **published** in a DID Document so others can verify your credentials.
- Example:
- `DID: did:web:priko_rx`
- `Public Key: <visible in did.json>`
- `Private Key: <kept secret locally>`

The `elliptic` and `crypto` libraries generate these keys in `keys.js`.

Step 2: DID Document Creation

Files involved: `.well-known/did.json`

- The **DID Document** defines your decentralized identity — who you are and how others can v your signatures.
- It contains:
  - The **DID** (e.g., `did:web:priko_rx`)
  - **Verification method** (your public key)
  - **Service endpoints** (optional – for contact or credential services)

Example snippet from `did.json`:

```
{
  "id": "did:web:priko_rx",
  "verificationMethod": [{
    "id": "did:web:priko_rx#owner",
    "type": "EcdsaSecp256k1VerificationKey2019",
    "publicKeyJwk": { "kty": "EC", "crv": "secp256k1", "x": "...", "y": "..." }
  }],
  "assertionMethod": ["did:web:priko_rx#owner"]
}
```

- You host this file under your web domain:
  `https://priko_rx/.well-known/did.json`
  so anyone can "resolve" your DID using **did-resolver** or **web-did-resolver**.

- You host this file under your web domain:
  `https://priko_rx/.well-known/did.json`
  so anyone can "resolve" your DID using **did-resolver** or **web-did-resolver**.

Step 3: JWT Token Creation (Digital Signature)

Tools used: `did-jwt`, `crypto`

- Using your private key, you can **sign a JSON Web Token (JWT)**.
- This proves that *you* (the DID owner) sent the message or claim.
- Others can verify the signature using your **public key** in the DID document.

Example usage:

```
npm run jwt
```

Output: A **signed JWT** proving identity ownership.

Step 4: Verifiable Credential (VC) Issuance

Tools used: `did-jwt-vc`

- A **credential** is like a digital certificate that proves something — for example: *"Priko_RX has completed Blockchain Developer Certification"*
- The **issuer (university or organization)** signs this credential with their private key.
- The **holder (you)** stores it in a wallet or identity system.
- The **verifier (employer, platform)** can check it using the DID document.

Example usage:

```
npm run vc
```

Output: A **Verifiable Credential (VC)** and optionally a **Verifiable Presentation (VP)**.

Step 5: Verification & Trust

Tools used: `did-jwt`, `did-jwt-vc`, `did-resolver`

- When someone (a verifier) receives your credential:
  - They check your **DID document** to fetch your public key.
  - They verify that the **credential signature** matches your DID.
  - If valid → your credential is authentic and untampered.
  - If not → it's rejected as fake or revoked.

How the Entire Flow Connects

```
Key Generation → DID Creation → Credential Issuance → Credential Verification
      ▼              ▼               ▼                      ▼
   (crypto)       (did:web)      (did-jwt-vc)          (did-resolver)
```

No central authority (like Google or Government)
Full ownership of your identity
Tamper-proof credentials that can be verified anywhere

Real-World Application Example

Role	Example
DID Owner	`did:web:priko_rx` (you)
Issuer	University or Organization issuing a certificate
Verifier	Employer or DApp validating your credential

Observation:

1. Successfully generated a **Decentralized Identifier (DID)** linked to a wallet address.
2. Issued a **Verifiable Credential (VC)** with cryptographic proof.
3. Verified credentials without using a central authority.
4. Demonstrated privacy-preserving user identity control.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

**As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.