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 : Build a Use Case - Tokenized Supply Chain Prototype

Objective

Create a simple prototype that tokenizes physical product batches as unique tokens (NFTs) to enable immutable tracking, ownership transfer, and status updates across supply-chain participants (Farmer → Distributor → Retailer → Consumer). Demonstrate minting, metadata storage (IPFS), transfers, and an on-chain status history.

Apparatus/Software Used:

- Solidity (Smart contract language)
- Hardhat (development & testing) or Remix for quick tests
- OpenZeppelin contracts (ERC-721)
- Ganache / Hardhat node (local blockchain)
- MetaMask (wallet testing)

Theory/Concept:

Tokenization maps a real-world product or batch to a unique on-chain token (NFT). Token metadata holds product details and an IPFS URI for richer data (certificates, photos). Smart contracts record transfers and status updates; combined with QR codes, consumers can verify history and provenance.
Benefits: immutable audit trail, secure ownership transfer, automated checks with smart contracts, tamper-evident product history.

Procedure :

Prepare environment

- Install Node.js, Hardhat, OpenZeppelin.
- Start local blockchain: npx hardhat node or Ganache.

Create contract

- Copy the above TokenizedSupplyChain.sol into contracts/.

Compile & Deploy

- Use Hardhat or Remix to compile.
- Deploy to local node. Save contract address & ABI.

Prepare product metadata

- Create JSON metadata:

```
{
  "name": "Mango Batch #M-2025-001",
  "batchId": "M-2025-001",
  "origin": "Farm A, India",
  "harvestDate": "2025-10-01",
  "certificates": ["ipfs://Qm..."],
  "image": "ipfs://Qm..."
}
```

- Upload JSON to IPFS/Pinata → get ipfs://Qm... URI.

Mint token

- Owner (manufacturer) calls mintProduct(to, batchId, ipfsURI, Stage.Manufactured, "Harvested & Packed").
- Note returned tokenId.

Simulate supply chain actions

- Distributor receives token (owner transfers token or transferFrom).
- Call updateStatus(tokenId, Stage.InTransit, "Shipped via Truck").
- Retailer receives and sets updateStatus(tokenId, Stage.ForSale, "Arrived at Warehouse").
- Optionally, consumer purchase triggers transferFrom to buyer.

Verify

- Call getHistoryCount and getHistoryEntry to view events.
- Scan QR code (links to token metadata page) → show IPFS metadata + chain history.

Observation table:

S.No	Action	On-chain Data	Status/Result	Remarks
1	Mint Product	tokenId, batchId, ipfsMetadata, initial status	✅ Token minted	Token URI points to IPFS metadata
2	Transfer to Distributor	Transfer event, history entry (Received)	✅ Ownership changed	History shows timestamp & actor
3	Update Status — InTransit	histories[tokenId] appended	✅ Status recorded immutably	Note contains shipping info
4	Transfer to Retailer	Transfer event + history	✅ Ownership	On-chain audit trail correct

ASSESSMENT

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

Signature of the Student:

Name :

Regn. No.

Signature of the Faculty: