P P SAVANI UNIVERSITY
ACADEMIC YEAR-2025-26

PRACTICAL- 6

ON

BLOCKCHAIN TECHNOLOGY(SSCS3021)

TITLE: Understanding Permissioned Blockchain and Exploring Hyperledger Fabric

BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY (BSC-IT)

SUBMITTED TO:

Name: KAUSHAL SINGH(KVS)

Designation: ASSISTANT PROFESSOR

P P Savani University

SUBMITTED BY:

Name: RAJ MO FAHIM ZAKIR

Enrollment: 23SS02IT161

BSCIT5B-Batch 2023-26

Max. Marks: 50
Marks Obtained:

Faculty Signature: _____

INSTITUTE OF COMPUTER SCIENCE AND APPLICATIONS
P P SAVANI UNIVERSITY
MANGROL, SURAT- 394125 (GUJARAT)

Student Name: RAJ MO FAHIM ZAKIR
Enrolment Number: 23SS02IT161
Subject Name: BLOCKCHAIN TECHNOLOGY
Subject Code: SSCS3021

PRACTICAL-6 **Date:17/09/2025**

Aim:Understanding Permissioned Blockchain and Exploring Hyperledger Fabric

Part A: Understanding Permissioned Blockchain

1. Definition & Characteristics

A permissioned blockchain is a private, invitation-only distributed ledger network. Unlike public blockchains where anyone can join, participants in a permissioned network must be authenticated and given explicit permission to access it. This creates a controlled environment where identities are known, and roles are defined, making it ideal for business and enterprise applications

Comparison: Permissioned vs. Permissionless Blockchain

Feature	Permissioned Blockchain (e.g., Hyperledger Fabric)	Permissionless Blockchain (e.g., Bitcoin, Ethereum)
Access Control	**Restricted & Known**. Participants require an invitation and are identified by a Certificate Authority (CA). Access rights can be finely controlled.	**Open & Anonymous**. Anyone can join the network, run a node, and participate in the consensus process. Identities are pseudonymous.
Consensus Mechanism	**Voting-based**. Uses protocols like Practical Byzantine Fault Tolerance (PBFT) or Raft. Does not require mining. Participants are known and trusted to a degree.	**Computation-based**. Uses protocols like Proof of Work (PoW) or Proof of Stake (PoS). Requires significant computational power (mining) to secure the network.
Security	Relies on **identity management** and access control policies. Threats are typically from internal	Relies on **game theory** and massive computational power (hash rate). Secured by making it economically unfeasible for a single entity to

	collusion or compromised participants rather than external anonymous attackers.	control the network (e.g., 51% attack).
Performance	**High performance**. Achieves high transaction throughput (thousands of transactions per second) and low latency because consensus is faster and involves fewer nodes.	**Low performance**. Transaction throughput is limited (e.g., Bitcoin ~7 TPS, Ethereum ~15 TPS) due to the computational overhead and block time delays of the consensus mechanism.

Use Case 1: Supply Chain Management

In a supply chain, multiple entities like manufacturers, suppliers, distributors, and retailers need to share information about goods as they move from origin to consumer.

Why Permissioned?

- A permissioned blockchain is more suitable because:
    - Privacy: Competing companies can be on the same network without seeing each other's sensitive trade secrets or pricing information. Data access can be restricted to relevant parties only.
    - Accountability: Since all participants are known and verified, it's easy to establish accountability for actions like certifying product authenticity or recording a customs clearance.
    - Performance: A supply chain can generate millions of events. A high-throughput permissioned network is necessary to handle this volume efficiently.

Use Case 2: Financial Services (Cross-Border Payments)

Banks and financial institutions need to settle transactions quickly and securely while complying with strict regulations.

Why Permissioned?

A permissioned blockchain is the better choice because:

Regulatory Compliance: Financial regulations like Know Your Customer (KYC) and Anti-Money Laundering (AML) require that all participants are identified and vetted. This is impossible on a public, anonymous network.

Confidentiality: Financial transactions are highly sensitive. A permissioned network ensures that transaction details are only visible to the involved parties.

Finality & Speed: Permissioned consensus mechanisms provide faster transaction finality, which is crucial for financial settlements, avoiding the uncertainty of probabilistic finality found in PoW blockchains.

Part B: Exploring Hyperledger Fabric

Step 1: Environment Setup

1. Install Docker & Docker Compose
   - From the Website of Docker

output of successfully install:

2. Download Fabric binaries & samples

```
PS C:\Users\hp> curl -sSL https://bit.ly/2ysbOFE | bash -s
```

Output:

```
Cloning into 'fabric-samples'...
===> Cloning hyperledger/fabric-samples repo
===> Downloading version 2.5.0 platform specific fabric binaries
===> Extracting hyperledger-fabric-windows-amd64-2.5.0.tar.gz
===> Binary tools placed into ./bin
```

3. Verify installation

```
PS C:\Users\hp>export PATH=${PWD}/bin:$PATH
peer version
...
peer:
 Version: 2.5.0
 Commit SHA: abc123...
 Go version: go1.20
 OS/Arch: windows/amd64
 Chaincode package format: code
```

Step 2: Running First Network

1. Navigate to test network

```
PS C:\Users\hp>cd fabric-samples/test-network
PS C:\Users\hp\fabric-samples/test-network>
```

2. Start the network

```
PS C:\Users\hp\fabric-samples/test-network>./network.sh up
Creating network "test" with default driver
Creating peer0.org1.example.com ...
Creating peer0.org2.example.com ...
Creating orderer.example.com ...
Creating ca_org1 ...
Creating ca_org2 ...
Creating cli ...
```

3. Create a channel

```
PS C:\Users\hp\fabric-samples/test-network>./network.sh create Channel -c my channel
Creating channel 'mychannel'.
peer0.org1.example.com joined the channel
peer0.org2.example.com joined the channel
```

Step 3: Deploying Chaincode (Smart Contract)

1. Deploy chaincode

```
PS C:\Users\hp\fabric-samples/test-network>./network.sh deployCC -ccn basic -ccp ../asset-transfer-
basic/chaincode-go -ccl go
Deploying chaincode 'basic' on channel 'mychannel'
Chaincode definition committed successfully
```

Step 4: Interact with Chaincode

1. Create Asset

```
PS C:\Users\hp\fabric-samples/test-network>peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com \
  --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-
cert.pem \
  -C mychannel -n basic -c '{"Args":["CreateAsset","asset1","blue","20","Tom","100"]}'
....
INFO  Created asset asset1 successfully
```

2. Read Asset

```
PS C:\Users\hp\fabric-samples/test-network>peer chaincode query -C mychannel -n basic -c
'{"Args":["ReadAsset","asset1"]}'

{"ID":"asset1","Color":"blue","Size":20,"Owner":"Tom","AppraisedValue":100}
PS C:\Users\hp\fabric-samples/test-network>
```

3. Update Asset

```
PS C:\Users\hp\fabric-samples/test-network>peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com \
  --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-
cert.pem \
  -C mychannel -n basic -c '{"Args":["UpdateAsset","asset1","blue","25","Jerry","150"]}'
INFO  Updated asset asset1 successfully
PS C:\Users\hp\fabric-samples/test-network>
```

4. Delete Asset

```
PS C:\Users\hp\fabric-samples/test-network>peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com \
  --tls --cafile
${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-
cert.pem \
  -C mychannel -n basic -c '{"Args":["DeleteAsset","asset1"]}'

INFO  Deleted asset asset1 successfully
PS C:\Users\hp\fabric-samples/test-network>
```

Step 4: Observations

1. Interaction Between Components

- **Clients** → submit transaction proposals to peers.
- **Peers** → simulate the transaction by executing chaincode (smart contract) and return endorsements (signed responses).
- **Orderer** → collects endorsed transactions from clients, orders them chronologically into blocks, and delivers them to all peers.
- **Peers** → validate endorsements & commit the transaction to the ledger (state database + blockchain).
- **Chaincode (Smart Contract)** → defines the business logic (e.g., CreateAsset, UpdateAsset) that runs inside a peer's container.

Key Observation:

Transactions are **not immediately committed** when invoked. They first need **endorsement + ordering + validation**, ensuring trust even in a permissioned blockchain.

2. Endorsement Policy in Hyperledger Fabric

- An **endorsement policy** defines **which peers must approve (endorse)** a transaction before it is considered valid.
- Example:
  - Policy = `AND('Org1.peer','Org2.peer')`

    → both Org1 and Org2 peers must endorse the transaction.
  - Policy = `OR('Org1.peer','Org2.peer')`

    → either Org1 or Org2 peer's endorsement is enough.
- Endorsements are **digital signatures** attached by peers after simulating a transaction.
- At the commit stage, peers **validate** that the required endorsements match the policy. If not → transaction is **discarded**.

Why important?

It ensures:

- **Trust** among consortium members.
- **Consistency** of ledger state.
- **Security** against tampering (only endorsed transactions are committed)