Blockchain Internship Assignment Report

This report documents the design and implementation of a blockchain-based asset management system developed using Hyperledger Fabric. The system allows a financial institution to manage assets with attributes such as Dealer ID, MSISDN, MPIN, Balance, Status, Transaction Amount, Transaction Type, and Remarks.

System Architecture

The solution is composed of two major components: 1. **Chaincode (Smart Contract in Go):** Implements core asset operations like create, read, update, delete, transfer amount, and fetch history. The chaincode is deployed on the Hyperledger Fabric network. 2. **REST API (Node.js):** Provides an HTTP interface to interact with the blockchain. It uses Fabric Gateway SDK to connect to the network and invoke chaincode functions. This REST service is containerized with Docker.

Chaincode Design

The chaincode is written in Go and defines an Asset structure with fields: - DEALERID (string) - MSISDN (string) - MPIN (string) - BALANCE (float) - STATUS (string) - TRANSAMOUNT (float) - TRANSTYPE (string) - REMARKS (string) Functions implemented: - CreateAsset - ReadAsset - UpdateAsset - DeleteAsset - GetAllAssets - GetAssetHistory - TransferAmount The TransferAmount function allows secure transfer of balance between accounts while recording transaction type and remarks.

REST API Design

The REST API is implemented in Node.js with Express. It provides endpoints to call the chaincode functions. Endpoints include: - POST /api/asset → Create an asset - GET /api/asset/:id → Read an asset - PUT /api/asset/:id → Update asset details - DELETE /api/asset/:id → Delete an asset - GET /api/assets → List all assets - GET /api/asset/:id/history → Get history of an asset - POST /api/asset/:id/transfer → Transfer balance between assets

Deployment Instructions

1. Setup Hyperledger Fabric test network from fabric-samples/test-network. 2. Deploy the Go chaincode using Fabric lifecycle commands. 3. Place connection profile (connection-org1.json) and wallet identities in the REST API folder. 4. Run REST API: - npm install - node server.js 5. Optionally, build Docker image using provided Dockerfile: - docker build -t asset-rest-api . - docker run -p 3000:3000 asset-rest-api

Test Cases

Test Case	Input	Expected Output
Create Asset	POST /api/asset with asset details	Asset successfully created
Read Asset	GET /api/asset/asset1	Returns JSON with asset details
Update Asset	PUT /api/asset/asset1 with new data	Asset updated successfully
Delete Asset	DELETE /api/asset/asset1	Asset removed from ledger

Get All Assets	GET /api/assets	List of all assets
Transfer Amount	POST /api/asset/asset1/transfer with toID and amount	Balances updated for both accounts

This project demonstrates the use of blockchain technologies to implement a secure and transparent asset management system. The Hyperledger Fabric network ensures immutability and distributed consensus, while the REST API provides ease of integration for external applications.