

# **Kerala Blockchain Academy (KBA)**

**An initiative of the Government of Kerala under the Kerala University of Digital Sciences, Innovation and Technology (formerly IIITM-K)**



## **Online Internship Project Report**

**on**

## **Blockchain based National Scholarship Portal**

Submitted in partial fulfillment of the requirements for Blockchain Internship Program offered by Kerala Blockchain Academy during the Academic Year

**2024-25**

by

**Asmi Rajbhar**

**Atharva Hande**

**Lifna Challissery Samu**

Project Mentor

**Lekshmi M. B** (Research Scientist, KBA)

**Vishva Prasad** (Research and Development Engineer (@KBA))

**Kerala Blockchain Academy  
(AY 2024-25)**

# **Declaration**

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

**Asmi Rajbhar**

**Atharva Hande**

**Lifna Challissery Samu**

Date: **2nd June 2025**

## INDEX

	<b>Page</b>
<b>Chapter 1: Brief overview of the project</b>	4
<b>Chapter 2: Detailed description of the tasks undertaken</b>	7
<b>Chapter 3: Technologies/tools used</b>	11
<b>Chapter 4: Key learnings and outcomes</b>	12
<b>Chapter 5: Screenshots or links to the deployed application/code repository</b>	13

## Chapter 1: Brief overview of the project

The project undertaken during the internship was titled "**Intelligent Scholarship Disbursement Module for National Scholarship Portal (NSP)**." This project was conceptualized as part of the **Digital India Initiative** by the Government of India, aimed at ensuring a **quick and efficient process for the disbursement of government scholarships** to eligible students. The existing **National Scholarship Portal (NSP)** provides a platform where students can apply for government scholarships. However, to apply for **private scholarships**, students need to approach multiple organizations separately. This dilemma often leads to confusion, delays, and administrative burdens for the students. To address this issue, the objective of this project was to design an **Intelligent Scholarship Disbursement Module** that integrates both **Government and Private Scholarship schemes** into a single, unified platform. The module automates the entire scholarship process using **Aadhaar details provided during registration**. It streamlines processes such as **document verification, income verification via PAN, and fund disbursement**, ensuring **centralized monitoring and quick processing**.

Figure 1 illustrates the overall architecture of the proposed system. The main stakeholders are the NSP, applicants, EIs, SOs (both private and public bodies), the ITD, the UIDAI, and the banks involved in fund disbursement. Internally, the Hyperledger fabric network maintains two channels: one for communication between the EIs and another for the SOs. Each of these organizations maintains a ledger and a copy of the chaincode. The membership service provider (MSP) and certificate authority (CA) are key components in the Hyperledger fabric network. The MSP manages the identities of all participating organizations, like private SO and EIs, ensuring that only verified members can access the network and perform actions. The CA issues digital certificates to these organizations, which act as their identity proof within the blockchain network. Thus, the MSP and CA create a safe and reliable environment within the Hyperledger fabric network in the NSP to efficiently manage scholarships, ensuring that only authorized institutions and providers can participate in the process.

When an applicant submits a scholarship application through a decentralized application (DApp), the request is sent over the internet to the Hyperledger fabric network established by the NSP. The request first reaches special computers called endorser peers (EPs), which check the request against a smart contract (a set of rules) to ensure its validity. If everything is in order, these peers sign off on the transaction. The signed transaction is then sent to an orderer peer (O), which organizes these transactions into a block and sends it to all relevant computers in the network, called committer peers (CPs). These peers further validate the block and permanently

record the transactions on the distributed ledger. An anchor peer (AP) helps synchronize this information across different organizations in the network.

After the scholarship deadline, the NSP initiates a smart contract to verify the credentials of the candidates through their respective EIs. These institutions authorize the candidates, and the NSP generates a verified list of eligible candidates based on the criteria defined by the SO. The SO then shortlists the candidates and initiates a smart contract to trigger the banks to disburse the funds to the selected candidates. Once the funds are transferred, the applicant and other involved entities, such as the NSP, EIs, and SOs, are notified of the transaction's status. The DApp displays the updated status to the applicant, and all related data is securely stored in a decentralized storage system, ensuring it is safe and accessible.

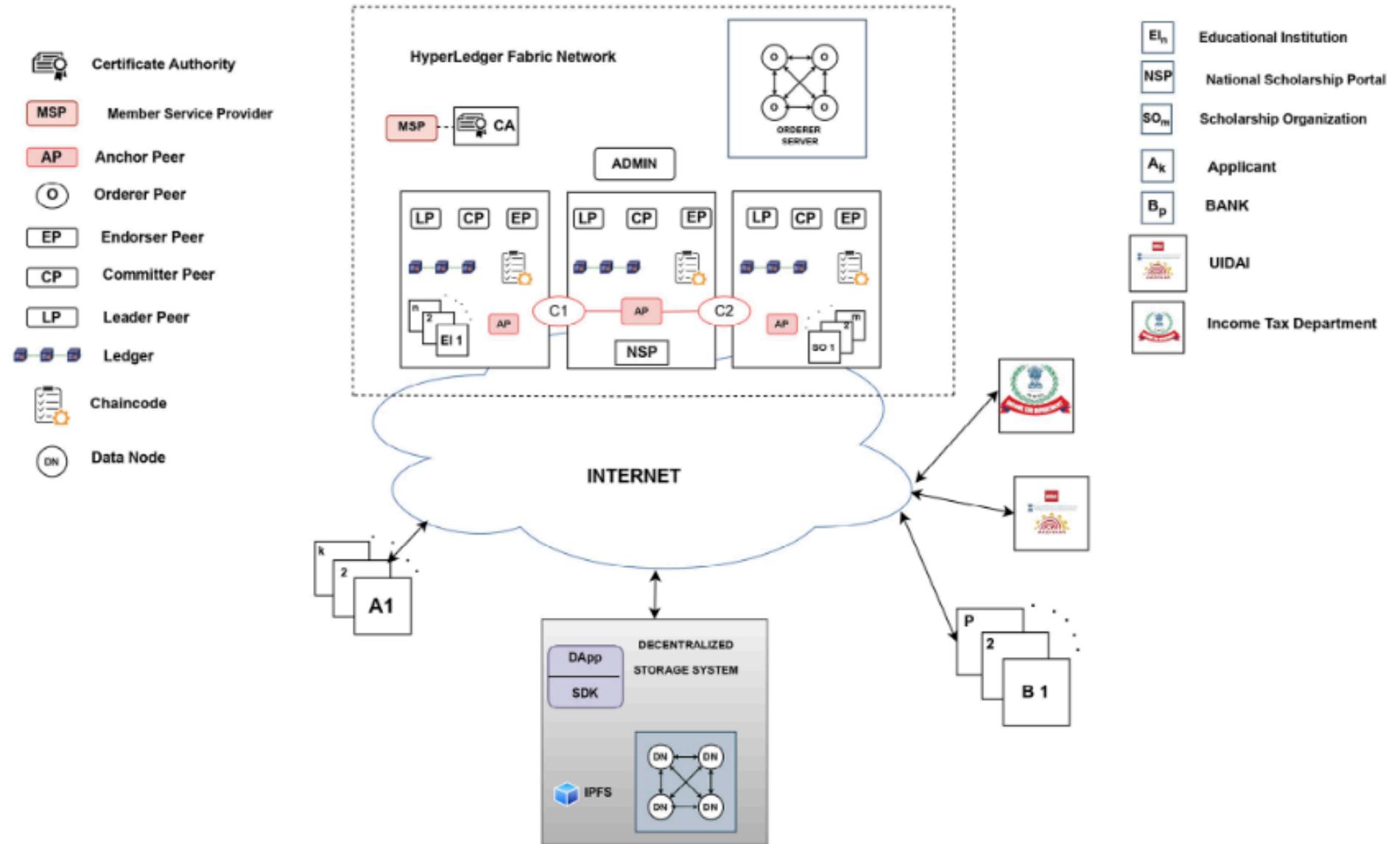


Figure - 1 : Proposed system architecture of the blockchain-based National Scholarship Portal.

## Chapter 2: Detailed description of the tasks undertaken

During the internship, the main task was to transition the Intelligent Scholarship Disbursement Module for NSP project from a static, development-only network to a production-ready Hyperledger Fabric network. Initially, a **frontend application** was prepared and connected to a static test network, which could not be used for production. To create a **production-grade blockchain network**, **Fabric samples** provided by Hyperledger Fabric were used as the foundation. We focused on modifying and configuring critical files within the Fabric samples to adapt them to the project's needs. These included:

- **compose-ca.yaml** to define and configure **certificate authorities (CAs)** necessary for managing identities of participants in the network.
- **compose-test-net.yaml** to set up and configure the **network topology**, ensuring that it correctly reflected the organizations and peers in the project.
- **registerEnroll.sh** to manage the **registration and enrollment** processes for user identities across the three key organizations in the system: **NSPOrg** (National Scholarship Portal Organization), **EIOrg** (Educational Institution Organization), and **SchOrg** (Scholarship Organization).
- **configtx.yaml** to set the **channel configurations** and **orderer system parameters**, ensuring that the network could support the required transaction processing and governance rules.

```

May 23 1:04 PM
compose-ca.yaml - DemoFolder - Visual Studio Code

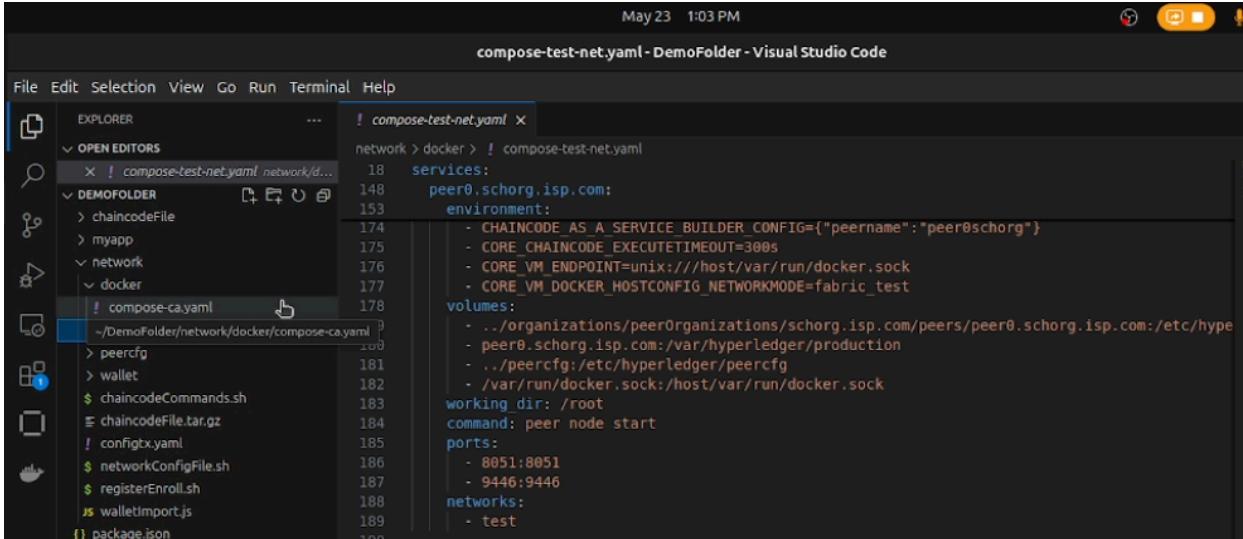
File Edit Selection View Go Run Terminal Help

EXPLORER OPEN EDITORS DEMOFOLDER compose-ca.yaml network/docker
compose-ca.yaml
compose-test-net.yaml
chaincodeFile
myapp
network
docker
chaincodeCommands.sh
chaincodeFile.tar.gz
configtx.yaml
networkConfigFile.sh
registerEnroll.sh
walletImport.js
package.json

compose-ca.yaml x
network > docker > compose-ca.yaml
12 services:
14 ca_nsport:
28 volumes:
29 - ./organizations/fabric-ca/nsport:/etc/hyperledger/fabric-ca-server
30 container_name: ca_nsport
31 networks:
32 - test
33
34 ca_eior:
35 image: hyperledger/fabric-ca:latest
36 labels:
37 service: hyperledger-fabric
38 environment:
39 - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
40 - FABRIC_CA_SERVER_CA_NAME=ca-eior
41 - FABRIC_CA_SERVER_TLS_ENABLED=true
42 - FABRIC_CA_SERVER_PORT=7054
43 - FABRIC_CA_SERVER_OPERATIONS_LISTENADDRESS=0.0.0.0:17054
44 ports:
45 - 7054:7054
46 - 17054:17054
47 command: sh -c 'fabric-ca-server start -b admin:adminpw -d'
48 volumes:
49 - ./organizations/fabric-ca/eior:/etc/hyperledger/fabric-ca-server
50 container_name: ca_eior
51 networks:
52 - test

```

Figure - 2 : Screenshot depicting the editing on file - **compose-ca.yaml**



The screenshot shows the Visual Studio Code interface with the file `compose-test-net.yaml` open in the editor. The code defines a Docker network configuration for a Hyperledger Fabric peer. Key sections include `services`, `volumes`, and `ports`. The `services` section specifies a peer named `peer0.schorg.isp.com` with various environment variables and volumes. The `ports` section maps ports 8051 and 9446 to the host.

```

May 23 1:03 PM
compose-test-net.yaml - DemoFolder - Visual Studio Code

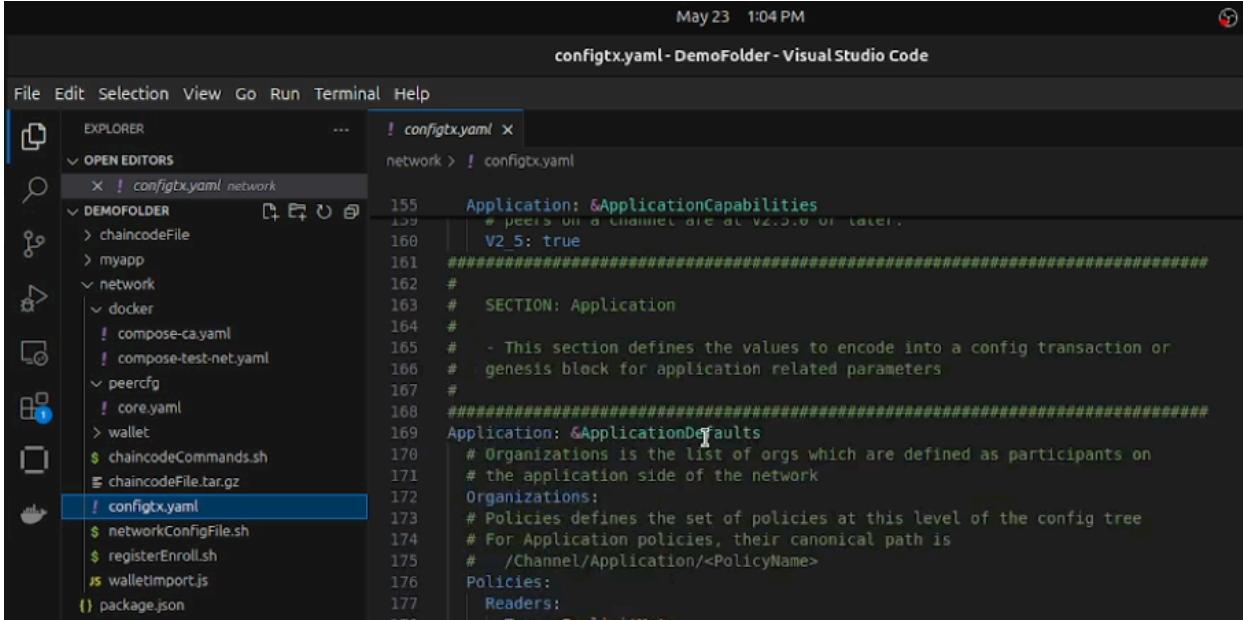
File Edit Selection View Go Run Terminal Help

EXPLORER
OPEN EDITORS
compose-test-net.yaml network/d...
DEMOFOLDER
chaincodeFile
myapp
network
docker
compose-ca.yaml
~/DemoFolder/network/docker/compose-ca.yaml
peercfg
wallet
chaincodeCommands.sh
chaincodeFile.tar.gz
configtx.yaml
networkConfigFile.sh
registerEnroll.sh
walletImport.js
package.json

compose-test-net.yaml X
network > docker > ! compose-test-net.yaml
18 services:
148 peer0.schorg.isp.com:
153 environment:
174 - CHAINCODE_AS_A_SERVICE_BUILDER_CONFIG={"peername": "peer0.schorg"}
175 - CORE_CHAINCODE_EXECUTETIMEOUT=300s
176 - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
177 - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=fabric_test
178 volumes:
180 - ../organizations/peerOrganizations/schorg.isp.com/peers/peer0.schorg.isp.com/:/etc/hyper
181 - peer0.schorg.isp.com:/var/hyperledger/production
182 - ../peercfg:/etc/hyperledger/peercfg
183 - /var/run/docker.sock:/host/var/run/docker.sock
184 working_dir: /root
185 command: peer node start
186 ports:
187 - 8051:8051
188 - 9446:9446
189 networks:
190 - test

```

Figure - 3 : Screenshot depicting the editing on file - **compose-test-net.yaml**



The screenshot shows the Visual Studio Code interface with the file `configtx.yaml` open in the editor. The code defines a configuration transaction for a Hyperledger Fabric network. It includes sections for `Application` and `Policies`. The `Application` section defines the application capabilities and organizations. The `Policies` section defines the set of policies for the network.

```

May 23 1:04 PM
configtx.yaml - DemoFolder - Visual Studio Code

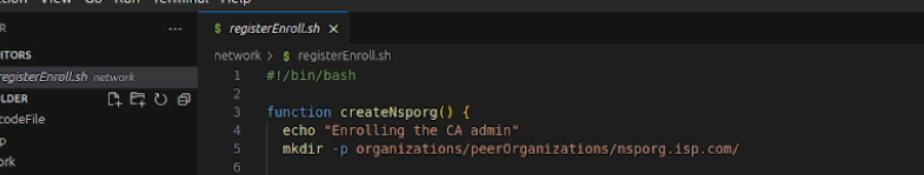
File Edit Selection View Go Run Terminal Help

EXPLORER
OPEN EDITORS
configtx.yaml X
DEMOFOLDER
chaincodeFile
myapp
network
compose-ca.yaml
compose-test-net.yaml
peercfg
core.yaml
wallet
chaincodeCommands.sh
chaincodeFile.tar.gz
configtx.yaml
networkConfigFile.sh
registerEnroll.sh
walletImport.js
package.json

configtx.yaml X
network > ! configtx.yaml
155 Application: &ApplicationCapabilities
156 # peers on a channel are at v2.0.0 or later.
157 V2_5: true
158 #####
159 #
160 #
161 # SECTION: Application
162 #
163 #
164 #
165 # - This section defines the values to encode into a config transaction or
166 # - genesis block for application related parameters
167 #
168 #####
169 Application: &ApplicationDefaults
170 # Organizations is the list of orgs which are defined as participants on
171 # the application side of the network
172 Organizations:
173 # Policies defines the set of policies at this level of the config tree
174 # For Application policies, their canonical path is
175 # /Channel/Application/<PolicyName>
176 Policies:
177 Readers:
178 Type: ImplicitMeta

```

Figure - 4 : Screenshot depicting the editing on file - **configtx.yaml**

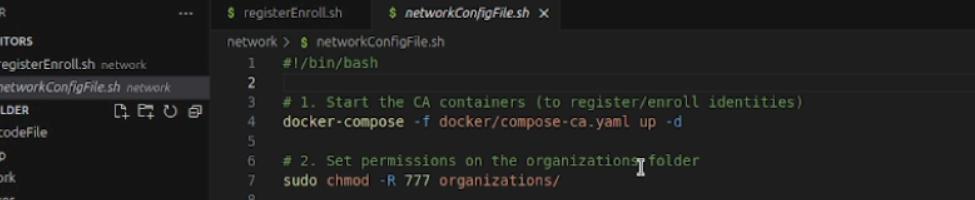


The screenshot shows the Visual Studio Code interface with the title bar "registerEnroll.sh - DemoFolder - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer sidebar on the left lists files and folders related to a "DEMOFOLDER" project, including "chaincodeFile", "myapp", "network", "docker", "compose-ca.yaml", "compose-test-net.yaml", "perfcfg", "core.yaml", "wallet", "chaincodeCommands.sh", "chaincodeFile.tar.gz", "configtx.yaml", "networkConfigFile.sh", "registerEnroll.sh" (which is selected and highlighted in blue), "walletImport.js", and "package.json". The main code editor shows the content of the "registerEnroll.sh" script:

```
$ registerEnroll.sh
network > $ registerEnroll.sh
1  #!/bin/bash
2
3  function createNsorg() {
4      echo "Enrolling the CA admin"
5      mkdir -p organizations/peerOrganizations/nsorg.isp.com/
6
7      export FABRIC_CA_CLIENT_HOME=${PWD}/organizations/peerOrganizations/nsorg.isp.com/
8
9      set -x
10     fabric-ca-client enroll -u https://admin:adminpw@localhost:6054 --caname ca-nsorg --tls.certfile ./ca-nsorg/tls/ca/server.pem
11     { set +x; } 2>/dev/null
12
13     echo 'NodeOUs:
14     Enable: true
15     ClientOUIdentifier:
16         Certificate: cacerts/localhost-6054-ca-nsorg.pem
17         OrganizationalUnitIdentifier: client
18     PeerOUIdentifier:
19         Certificate: cacerts/localhost-6054-ca-nsorg.pem
20         OrganizationalUnitIdentifier: peer
21     AdminOUIdentifier:
22         Certificate: cacerts/localhost-6054-ca-nsorg.pem
```

Figure - 5 : Screenshot depicting the editing on file - **registerEnroll.sh**

After these changes, a custom script called **networkConfigFile.sh** was created to streamline and automate the setup and configuration of the network based on the modified files. This script ensured that the network components, including the **peers**, **orderers**, and **CAs**, were deployed correctly. Once the network was operational, the focus shifted to deploying the **chaincode (smart contract)**, which defines the logic for the scholarship disbursement process. A second script, **chaincodeCommands.sh**, was created to automate the commands required to **install, approve, and commit the chaincode** to the production network. The result was a **fully functional, production-ready Hyperledger Fabric network** with a deployed chaincode, ready to support the **Intelligent Scholarship Disbursement Module** project.



```
#!/bin/bash
# 1. Start the CA containers (to register/enroll identities)
docker-compose -f docker/compose-ca.yaml up -d
# 2. Set permissions on the organizations folder
sudo chmod -R 777 organizations/
# 3. Make sure the script is executable
chmod +x registerEnroll.sh
# 4. Run your script to register/enroll identities
./registerEnroll.sh
# 5. After all identities and crypto material are generated,
# start the main Fabric network (orderer, peers, etc.)
docker-compose -f docker/compose-test-net.yaml up -d
docker ps -a
export PATH=${PWD}/../bin:$PATH
export FABRIC_CFG_PATH=${PWD}/configtx
#####
echo "Generate the genesis block of the channel"
configtxgen -profile ChannelUsingRaft -outputBlock ./channel-artifacts/isp-channel.block -chan
```

Figure - 6 : Screenshot depicting the editing on file - **networkConfigFile.sh**

```
$ networkConfigFile.sh $ chaincodeCommands.sh
network > $ chaincodeCommands.sh
1 #!/bin/bash
2
3 export FABRIC_CFG_PATH=$PWD/peercfg
4
5 #####Deploy Smart Contract to the channel#####
6 peer channel getinfo -c isp-channel
7
8 #####peer lifecycle chaincode package command#####
9 peer lifecycle chaincode package chaincodeFile.tar.gz --path ../chaincodeFile --lang golang --label
10
11 ######Install the chaincode package#####
12
13
14 #####NSPorg#####
15 export CORE_PEER_TLS_ENABLED=true
16 export CORE_PEER_LOCALMSPID="NspOrgMSP"
17 export CORE_PEER_TLS_ROOTCERT_FILE=$PWD/organizations/peerOrganizations/nsporg.isp.com/peers/peer0.nsporg.isp.com/tls/ca.crt
18 export CORE_PEER_MSPCONFIGPATH=$PWD/organizations/peerOrganizations/nsporg.isp.com/users/Admin@nsporg.isp.com/msp
19 export CORE_PEER_ADDRESS=localhost:7051
20
21 peer lifecycle chaincode install chaincodeFile.tar.gz
22
23
24 #####ETOrg#####
25 export CORE_PEER_TLS_ENABLED=true
26 export CORE_PEER_LOCALMSPID="EiOrgMSP"
27 export CORE_PEER_TLS_ROOTCERT_FILE=$PWD/organizations/peerOrganizations/eiorg.isp.com/peers/peer0.eiorg.isp.com/tls/ca.crt
```

Figure - 7 : Screenshot depicting the editing on file - **chaincodeCommands.sh**

## Chapter 3: Technologies/tools used

The experimental environment was configured on an Intel Core i3-2350 M CPU @ 2.30 GHz × 4 with 8GB of main memory, running on Ubuntu 20.04 LTS. Docker v28.1.0 was utilized for containerization, alongside Docker Compose v1.29.2 to manage multi-container applications. Node.js v22.12.0 and npm v10.9.0 were used for package management. The smart contracts were written in Go v1.20, and Hyperledger Fabric v2.5.4<sup>[1]</sup> components were deployed as the blockchain framework, facilitating the execution of the experiment. The entire documentation of the setup along with smart contracts code and related documents are available at the Github repository<sup>[2]</sup> for the project. The following technologies were used for the implementation of the Internship Project.

- **Hyperledger Fabric:** for creating and managing the blockchain network.
- **Docker:** for running network components such as peers, orderers, and certificate authorities in isolated containers.
- **Hyperledger Fabric Samples:** provided sample files and templates that were customized for the project
- **Shell Scripts (Bash):** for automating the network configuration (**networkConfigFile.sh**) and chaincode deployment (**chaincodeCommands.sh**).
- **YAML Configuration Files:** such as **docker-ca.yaml**, **docker-test-net.yaml**, and **configtx.yaml** for defining the structure and behavior of the network.
- **Node.js / React.js:** used for building the frontend application (based on what you mentioned in earlier discussions).

## Chapter 4: Key learnings and outcomes

The following are the outcomes of the Internship Project :

- **Learned how to configure and customize a Hyperledger Fabric network** for production use by modifying core files like **docker-ca.yaml** and **configtx.yaml**.
- **Understood the structure of a Fabric network** and the roles of different organizations (**NSPOrg**, **EIOrg**, **SchOrg**) in the scholarship system.
- **Gained hands-on experience in creating and executing shell scripts** to automate network setup and chaincode deployment.
- **Learned to integrate a production-ready blockchain network** with an existing frontend application for real-world use.
- **Developed problem-solving skills** in configuring and troubleshooting network components and chaincode deployment.
- Successfully **deployed a production-grade network** and **integrated chaincode logic** into the Intelligent Scholarship Disbursement Module project.

## Chapter 5: Screenshots or links to the deployed code repository

The script file mentioned in chapter 2 were executed successfully and the screenshots are as follows:

```
May 23 1:07 PM
networkConfigFile.sh - DemoFolder - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS
$ registerEnroll.sh $ networkConfigFile.sh
DEMOFOLDER
> chaincodeFile
> myapp
network
> docker
! compose-ca.yaml
! compose-test-net.yaml
> organizations
> peercfg
! core.yaml
> wallet
$ chaincodeCommands.sh
$ chaincodeFile.tar.gz
! configtx.yaml
$ networkConfigFile.sh
$ registerEnroll.sh
$ walletImport.js
{} package.json

TERMINAL
asmi-rajbhar@ubuntu:~/DemoFolder/network$ ./networkConfigFile.sh
Creating network "fabric_test" with the default driver
Creating ca_nsorg ... done
Creating ca_schorg ... done
Creating ca_orderer ... done
Creating ca_eiorg ... done
[sudo] password for asmi-rajbhar: [REDACTED]
```

Figure - 8(a) : Depicts the execution of **networkConfigFile.sh** file creating the peer network

```
May 23 1:07 PM
networkConfigFile.sh - DemoFolder - Visual Studio Code

File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS
$ registerEnroll.sh $ networkConfigFile.sh
DEMOFOLDER
> chaincodeFile
> myapp
network
> channel-artifacts
> docker
! compose-ca.yaml
! compose-test-net.yaml
> organizations
> peercfg
! core.yaml
> wallet
$ chaincodeCommands.sh
$ chaincodeFile.tar.gz
! configtx.yaml
$ networkConfigFile.sh
$ registerEnroll.sh
$ walletImport.js
{} package.json

TERMINAL
Generate the genesis block of the channel
2025-05-23 13:07:41.699 IST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2025-05-23 13:07:41.707 IST 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> ord
r type: etcdraft
2025-05-23 13:07:41.707 IST 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Ord
r.Etcdraft.Options.unset, setting to tick_interval:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_
cks:5 snapshot_interval_size:16777216
2025-05-23 13:07:41.707 IST 0004 INFO [common.tools.configtxgen.localconfig] load -> Loaded configuration:
configtx.yaml
2025-05-23 13:07:41.709 IST 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2025-05-23 13:07:41.709 IST 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application cha
l genesis block
2025-05-23 13:07:41.710 IST 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
Create the application channel name isp-channel
Status: 201
{
  "name": "isp-channel",
  "url": "/participation/v1/channels/isp-channel",
  "consensusRelation": "consenter",
  "status": "active",
  "height": 1
}
List channels on an orderer
Status: 200
```

Figure - 8(b) : Depicts the execution of **networkConfigFile.sh** file where the Genesis block is generated for the channel

```

May 23 1:08 PM
networkConfigFile.sh - DemoFolder - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
OPEN EDITORS
$ registerEnroll.sh network
$ networkConfigFile.sh network
DEMOFOLDER
> chaincodeFile
> myapp
> network
> channel-artifacts
> docker
! compose-ca.yaml
! compose-test-net.yaml
> organizations
> peercfg
! core.yaml
> wallet
$ chaincodeCommands.sh
$ networkConfigFile.sh
$ registerEnroll.sh
$ walletImport.js
{} package.json

TERMINAL
Join peers to the channel
For Peer of NspOrg
2025-05-23 13:07:48.969 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-23 13:07:49.001 IST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
For Peer of EIOrg
2025-05-23 13:07:51.084 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-23 13:07:51.118 IST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
For Peer of SchOrg
2025-05-23 13:07:53.202 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-23 13:07:53.232 IST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
Set anchor peer
Peer from NspOrg
2025-05-23 13:07:55.316 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-23 13:07:55.328 IST 0002 INFO [cli.common] readBlock -> Received block: 0
2025-05-23 13:07:55.329 IST 0003 INFO [channelCmd] fetch -> Retrieving last config block: 0
2025-05-23 13:07:55.322 IST 0004 INFO [cli.common] readBlock -> Received block: 0
2025-05-23 13:07:55.602 IST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-05-23 13:07:55.621 IST 0002 INFO [channelCmd] update -> Successfully submitted channel update
Peer from EIorg

```

Figure - 8(c) : Depicts the execution of **networkConfigFile.sh** file where the peers are added to the channel

```

chaincodeCommands.sh - DemoFolder - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
OPEN EDITORS
$ networkConfigFile.sh net...
$ chaincodeCommands.sh
DEMOFOLDER
> chaincodeFile
> myapp
> network
> channel-artifacts
> connection-profile
> docker
> organizations
> peercfg
! core.yaml
> wallet
$ chaincodeCommands.sh
$ chaincodeFile.tar.gz
! configtx.yaml
$ networkConfigFile.sh
$ registerEnroll.sh
$ walletImport.js
> node_modules
{} package-lock.json
{} package.json

TERMINAL
2025-06-02 17:36:18.849 IST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nChaincode_1.0:f231dc4492a904fdfaf788fdd4853376b76e55006c6b3b3c0fcfd9f786a5414\\022\\rchaincode_1.0"
>
2025-06-02 17:36:18.849 IST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: chaincode_1.0:f231dc4492a904fdfaf788fdd4853376b76e55006c6b3b3c0fcfd9f786a5414
2025-06-02 17:36:19.343 IST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nChaincode_1.0:f231dc4492a904fdfaf788fdd4853376b76e55006c6b3b3c0fcfd9f786a5414\\022\\rchaincode_1.0"
>
2025-06-02 17:36:19.343 IST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: chaincode_1.0:f231dc4492a904fdfaf788fdd4853376b76e55006c6b3b3c0fcfd9f786a5414
2025-06-02 17:36:19.343 IST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nChaincode_1.0:f231dc4492a904fdfaf788fdd4853376b76e55006c6b3b3c0fcfd9f786a5414\\022\\rchaincode_1.0"
>
2025-06-02 17:36:19.855 IST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: chaincode_1.0:f231dc4492a904fdfaf788fdd4853376b76e55006c6b3b3c0fcfd9f786a5414
Installed chaincodes on peer:
Package ID: chaincode_1.0:f231dc4492a904fdfaf788fdd4853376b76e55006c6b3b3c0fcfd9f786a5414, Label: chaincode_1.0
2025-06-02 17:36:22.038 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [008b43af02513acf3265133e08fcbb960e71aed691031a4d568d2fdbad6e8] committed with status (VALID) at localhost:7051
2025-06-02 17:36:24.118 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [84bb5b2228f7a0110ec91502fdead35caba73472dbfed08746a1faa3f4423a95] committed with status (VALID) at localhost:9051
2025-06-02 17:36:26.183 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [7ebf49d5bff09e77731dba24b925f8126dcef00cfef209ec9467a0b10a0b6a165] committed with status (VALID) at localhost:8051
{
    "approvals": [
        "EiOrgMSP": true,
        "NspOrgMSP": true,
        "SchOrgMSP": true
    ]
}

```

Figure - 9(a) : Executing the **chaincodeCommands.sh** file for installing the chaincode on to the peers remotely.

```

chaincodeCommands.sh - DemoFolder - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER          $ networkConfigFile.sh    $ chaincodeCommands.sh
OPEN EDITORS       $ chaincodeCommands.sh net...
DEMOPARTER        > chaincodeFile
                  > myapp
                  > network
                  > channel-artifacts
                  > connection-profile
                  > docker
                  > organizations
                  > peercfg
                  ! core.yaml
                  > wallet
                  > chaincodeCommands.sh
                  > chaincodeFile.tar.gz
                  ! configtx.yaml
                  > networkConfigFile.sh
                  $ registerEnroll.sh
                  JS walletImport.js
                  {} package-lock.json
                  {} package.json
PROBLEMS          OUTPUT      DEBUG CONSOLE   TERMINAL
TERMINAL
status:200 payload:"\nNchaincode_1.0:f231dc4492a904fdaf788fdd4853376b76e55006c6b3b3c0fc1d9f786a5414\\022\rchaincode_1.0"
>
2025-06-02 17:36:19.855 IST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: chaincode_1.0:f231dc4492a904fdaf788fdd4853376b76e55006c6b3b3c0fc1d9f786a5414, Label: chaincode_1.0
2025-06-02 17:36:22.038 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [008b43af02513aaac9f3265133e08fc8bb960e71aed69103
1a4d568d2fdb46de8] committed with status (VALID) at localhost:7051
2025-06-02 17:36:24.110 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [84bb5b2228f7a0110ec91502fdead35cab73472dbfed08
746a1faa3f4423a95] committed with status (VALID) at localhost:9051
2025-06-02 17:36:26.183 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [7ebf49d5bfff09e77731dba24b925f8126cef00cfce209ec
9467a0b10a0b6a165] committed with status (VALID) at localhost:8051
{
    "approvals": {
        "EiOrgMSP": true,
        "NspOrgMSP": true,
        "SchOrgMSP": true
    }
}
2025-06-02 17:36:28.417 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [0cae3d36ab456aa0b76e58f585c63c851b44a239e52efe9
3f23180f4e1475b4e] committed with status (VALID) at localhost:9051
2025-06-02 17:36:28.451 IST 0002 INFO [chaincodeCmd] ClientWait -> txid [0cae3d36ab456aa0b76e58f585c63c851b44a239e52efe9
3f23180f4e1475b4e] committed with status (VALID) at localhost:7051
2025-06-02 17:36:28.455 IST 0003 INFO [chaincodeCmd] ClientWait -> txid [0cae3d36ab456aa0b76e58f585c63c851b44a239e52efe9
3f23180f4e1475b4e] committed with status (VALID) at localhost:8051
Committed chaincode definition for chaincode 'chaincodeFile' on channel 'isp-channel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [EiOrgMSP: true, NspOrgMSP: true, SchOrgMSP: true]
asmi-rajbhar@ubuntu:~/DemoFolder/network$ 
```

Figure - 9(b) : Executing the **chaincodeCommands.sh** file where the chaincode is approved by the peers remotely.

```

asmi-rajbhar@ubuntu:~/DemoFolder/network$ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.isp.com --channelID
isp-channel --name chaincodeFile --version 1.0 --sequence 1 --tls --cafile "${PWD}/organizations/ordererOrganizations/isp.com/orderers/orderer.isp.co
m/msp/tlscacerts/tlsca.isp.com-cert.pem" --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/nsporg.isp.com/pe
ers/peer0.nsporg.isp.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/eiorg.isp.com/peers/pee
r0.eiorg.isp.com/tls/ca.crt" --peerAddresses localhost:8051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/schorg.isp.com/peers/peer0.sch
org.isp.com/tls/ca.crt"
2025-05-22 13:35:59.905 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [5731746de673a11dfbb5c6ca8387bc980e451128e82533851994d6adafe2e706c] committed
with status (VALID) at localhost:8051
2025-05-22 13:35:59.905 IST 0002 INFO [chaincodeCmd] ClientWait -> txid [5731746de673a11dfbb5c6ca8387bc980e451128e82533851994d6adafe2e706c] committed
with status (VALID) at localhost:9051
2025-05-22 13:35:59.908 IST 0003 INFO [chaincodeCmd] ClientWait -> txid [5731746de673a11dfbb5c6ca8387bc980e451128e82533851994d6adafe2e706c] committed
with status (VALID) at localhost:7051
asmi-rajbhar@ubuntu:~/DemoFolder/network$ peer lifecycle chaincode querycommitted --channelID isp-channel --name chaincodeFile --cafile "${PWD}/organ
izations/ordererOrganizations/isp.com/orderers/orderer.isp.com/msp/tlscacerts/tlsca.isp.com-cert.pem"
Committed chaincode definition for chaincode 'chaincodeFile' on channel 'isp-channel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [EiOrgMSP: true, NspOrgMSP: true, SchOrgMSP: true]
asmi-rajbhar@ubuntu:~/DemoFolder/network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.isp.com --tls --cafile "${PWD}/
organizations/ordererOrganizations/isp.com/orderers/orderer.isp.com/msp/tlscacerts/tlsca.isp.com-cert.pem" -C isp-channel -n chaincodeFile --peerAddres
ses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/nsporg.isp.com/peers/peer0.nsporg.isp.com/tls/ca.crt" --peerAddresse
s localhost:8051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/eiorg.isp.com/peers/peer0.eiorg.isp.com/tls/ca.crt" --peerAddresse
s localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/schorg.isp.com/peers/peer0.schorg.isp.com/tls/ca.crt" -c '{"function": "EIContract
CreateEI", "Args": [{"HEIID123": "7023", "78786478761", "989578398785", "98987874880", "visit@ves.ac.in", "VESIT", "Chembur", "Mumbai", "Maharashtra"}]}'
2025-05-22 13:45:15.223 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
asmi-rajbhar@ubuntu:~/DemoFolder/network$ peer chaincode query -C isp-channel -n chaincodeFile -c '{"Args": ["EIContract:EIEExists", "7023"]}' true
asmi-rajbhar@ubuntu:~/DemoFolder/network$ 
```

Figure - 10 : Depicts the invocation of chaincode for making an EIOrg entry into the ledger of the network.

## Code Repository : Blockchain based National Scholarship Portal—Github Repository.

<https://github.com/LifnaJos/Blockchain-based-National-Scholarship-Portal>.

## References

- Hyperledger fabric documentation.  
[https://hyperledger-fabric.readthedocs.io/en/release-2.2/getting\\_started.html](https://hyperledger-fabric.readthedocs.io/en/release-2.2/getting_started.html)
- Hyperledger GitLab from KBA  
[https://gitlab.com/CHF\\_KBA/kba\\_chf\\_vb14](https://gitlab.com/CHF_KBA/kba_chf_vb14)