Student Examination System & Advanced Distributed Systems (NPTEL)

CODING ASSIGNMENT & ONLINE COURSE ATTENDED REPORT
SUBMITTED TO

M S RAMAIAH INSTITUTE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)

SUBMITTED BY

Tejas Hegde **1MS20CS129**

As part of the Course **CSE746-Blockchain Essentials & Dapps**
SUPERVISED BY
Faculty
Dr. Parkavi. A

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

M S RAMAIAH INSTITUTE OF TECHNOLOGY

Oct-Nov 2023

Department of Computer Science and Engineering

M S Ramaiah Institute of Technology

(Autonomous Institute, Affiliated to VTU)

Bangalore – 54

CERTIFICATE

This is to certify that **Tejas Hegde 1MS20CS129** has completed the **College Examination System & Advanced Distributed Systems (NPTEL)"** as part of Coding Assignment and online course attending. I declare that the entire content embodied in this B.E, 7th Semester report contents are not copied.

Submitted by

Tejas Hegde 1MS20CS129

Guided by

Dr. Parkavi A.
(Dept of CSE, RIT)

Department of Computer Science and Engineering

M S Ramaiah Institute of Technology

(Autonomous Institute, Affiliated to VTU)

Bangalore – 54

Evaluation Sheet

USN	Name	Coding skills, Demo & Explanation (10)	Online course attended (10)	Total Marks (20)

Evaluated By

Name: Dr. Parkavi A.
Designation: Associate Professor
Department: Computer Science & Engineering, RIT
Signature:

Table of Contents

Abstract

The student examination system is a blockchain application which is aimed simulating the complex process of a student examination from the question paper perspective. Hence, it provides a useful foundation for entities like colleges which wish to integrate blockchain into education. It also demonstrates innovation applied in the design and the construction of an enterprise application.

The software is a mixture of several tools and techniques. Django, a Python web framework has been used to design the web system. Server-side rendering using Django template engine has been used to present dynamic web pages prepared using HTML, CSS and JavaScript. IPFS has been used to provide decentralized storage to the blockchain. Postgres has been used to manage the users and subjects of the examination system. And finally, Hyperledger fabric has been used to provide the blockchain to the system.

This software would prove very useful in education, enterprise softwares etc. with a modification according to the requirements. The selling point lies in the innovative approach used by this application.

Introduction:

The student examination is an important process due to its impact on the student and the society as a large. It is a very big and complex process. Right from the paper setting to its approval and the distribution of papers, there are a lot of elaborate procedures to be followed. For instance, the COE (Controller of Examination) sends a request to the HODs (head of department) of all the departments in the college. A HOD will have a select team of teachers for setting the paper for a particular subject. Once all such papers are received, the HOD will appoint another team for carefully examining the papers and approve it. Once approved, the HOD will send the papers to the COE. The COE then sends the papers to the assistants who are responsible for the printout of the papers and their disbursal.

At each step anonymity i.e., privacy has to be maintained in order to prevent malpractice. For example, a student may try to influence a teacher to leak the paper for scoring well in the exam, if the identity of the teacher is known beforehand. Also, the paper itself must be kept secret. Otherwise, it is vulnerable to leakage. For example, the papers must be kept in sealed rooms with high security. Otherwise, miscreants may try to break into the room and steal the papers.

The advent of the digital age has made almost all the process, excluding the printout of the question papers and the actual setting and physical inspection, completely digitized. Thus, a lot of the process appears to be a private network-based communication among anonymous actors. Yet, there still exists one problem; the paper must be stored with a trusted authority. If this authority is compromised for some reason, the entire examination system is compromised. This is the problem that occurs across various applications: centralization.

Problem Definition:

The problem definition is as follows: Design a decentralized student examination system. The system must simulate the original setup i.e.:

- COE sends request for setting paper to HOD
- HOD sends request for setting paper for particular subject to teacher/s.
- Teacher/s set paper and sends it to the HOD.
- HOD sends paper to the approval team.
- Approval team makes required changes and sends final paper to HOD.
- HOD sends final paper to COE.
- COE sends paper to his assistant for printout and disbursal of the question papers.

The requirements of the application are as follows:

- The system must have a web interface for ease of use.
- The user management: teachers, HODs, COEs and assistants (superintendents) must be made simple.
- Web interface must provide privacy i.e., login system.
- The papers should not be present without any protection i.e., encryption.
- Finally, the storage of the papers should be decentralized.

Algorithm:

The consensus problem requires agreement among a number of processes (or agents) for a single data value. Some of the processes (agents) may fail or be unreliable in other ways, so consensus protocols must be fault tolerant or resilient. The processes must somehow put forth their candidate values, communicate with one another, and agree on a single consensus value.

Hyperledger Fabric now simply uses Crash Fault Tolerant (CFT) consensus algorithms, which is that it can't accept any malicious threat.

Crash fault tolerant is a name given the algorithms that solve the problem of nodes crashing simply. So we have a decentralized system where nodes can halt or disconnect from the network and yet we maintain the same state of truth on the system.

To solve this kind of failures, Paxos family of algorithms appeared in the 80s by Leslie Lamport. And then after years, another Crash Fault Tolerant algorithm appeared named RAFT which is built mainly for understandability. RAFT is primarily used in Hyperledger fabric which is a permissioned blockchain.

The algorithm is described as follows (since Paxos is not available in IEEE format)

1. Introduction: The Raft consensus algorithm aims to solve the problem of reaching consensus in a distributed system by electing a leader, which is responsible for making decisions and ensuring that all nodes agree on the system's state. The key design goals of Raft are understandability, safety, and availability.

2. Algorithm Description: Raft divides time into terms, where each term consists of one or more states: follower, candidate, and leader. It proceeds through the following phases:

1. Leader Election:
    a. In a new term, nodes start as followers.
    b. If a follower does not hear from a leader within a certain timeout, it becomes a candidate and starts an election.
    c. The candidate that receives votes from the majority of nodes becomes the leader.
2. Log Replication:
    a. The leader accepts client requests and appends them to its log.
    b. The leader replicates the log entries to followers.
    c. Followers apply the log entries and respond to the leader.
3. Safety Properties:
    a. Log Matching: If two logs contain an entry with the same index and term, they are identical in all preceding entries.
    b. Election Safety: At most one leader can be elected in a given term.
4. Availability:
    a. A leader is responsible for client requests, ensuring system availability.

3. Leader Election:
    1. Nodes begin as followers.
    2. If a follower receives no communication from the leader or candidate for a certain period, it becomes a candidate.
    3. The candidate requests votes from other nodes.
    4. If a candidate receives votes from the majority of nodes, it becomes the leader.
    5. If another leader is elected, the candidate returns to being a follower.
4. Log Replication:
    1. The leader receives client requests and appends them to its log.
    2. The leader replicates log entries to followers.
    3. Followers apply log entries to their state machines.
    4. Followers respond to the leader.
5. Safety Properties
    1. Log Matching: If two logs contain an entry with the same index and term, they are identical in all preceding entries.
    2. Election Safety: At most one leader can be elected in a given term.
6. Availability: The leader is responsible for handling client requests, ensuring system availability.

Implementation:

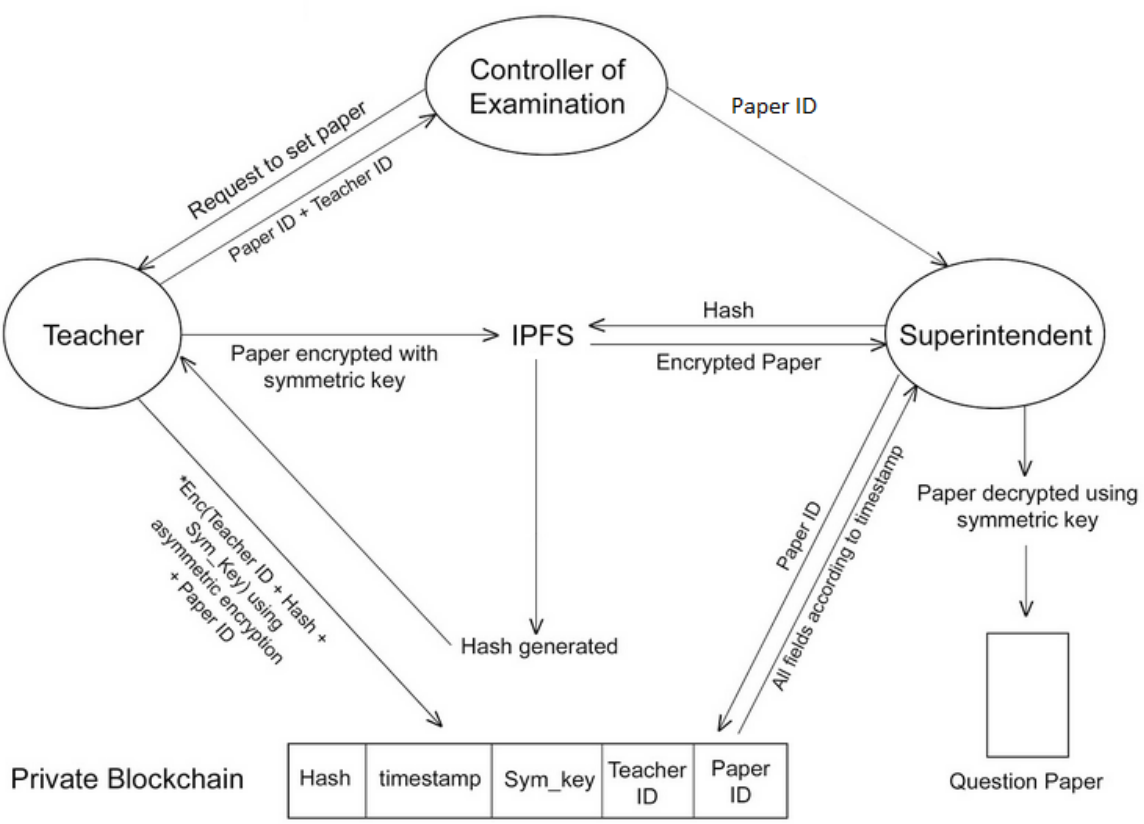

Fig. 1: Implementation Overview

The web application has been designed using Django framework in addition with PostgreSQL for managing user and subject related information.

As the picture suggests, the core of this system lies at IPFS (Interplanetary File System). IPFS is a decentralized storage which works upon a DHT (Distributed Hash Table).

In the current implementation, there are three users: COE (Controller of Examination), Teacher and the superintendent. In addition, there exists a superuser. The superuser is the one who manages the Django admin console and is responsible for the addition of new users to the system. The Django admin console can be found at the URL: http://localhost:8000/admin.

The blockchain part has been implemented using Hyperledger fabric. This is a permissioned blockchain framework which allows only trusted entities to be part of the blockchain. Hence, there is no need of a gas amount as in Ethereum since all entities are trusted. Thus, in the case of education where sensitive information such as question papers shouldn't be public,

Hyperledger fabric is a good and scalable choice for the blockchain.

The entire development was done on a Kali Linux VM with 4GB RAM and 2 CPU's. It is a resource constrained environment for the project yet the web development part and IPFS were successfully tested and integrated.

The web development part was done using Django since it allows rapid development of secure and powerful websites. The most useful feature provided by Django is the Admin console

which is at the core of the web system; new users/ subject codes are added through this. Django server-side rendering allows dynamically rendering of web pages. Postgres db as Django has special support for Postgres and it is much more scalable compared to MySQL.

The IPFS is the decentralized blockchain storage that is used in this project. The hash (UID of file) generated by IPFS was first stored in a database. To install IPFS, the binary source files were used and installed in the system's PATH environment variable. An ultralight node (for development and resource constrained purposes) was initialized. Then CORS access was enabled.

The below commands summarize this discussion:

- ***ipfs init --profile=lowpower***
- ***ipfs config --json API.HTTPHeaders.Access-Control-Allow-Origin '["*"]'***
- ***ipfs config --json API.HTTPHeaders.Access-Control-Allow-Methods '["PUT", "GET", "POST"]'***
- ***ipfs config --json API.HTTPHeaders.Access-Control-Allow-Credentials '["true"]'***

Later the IPFS server(daemon) was started using the following command: ***ipfs daemon***.

IPFS desktop can be used due to its more convenient UI but since it is too heavy for the VM it is avoided.

The original plan was to use the hyperledger fabric with hyperledger compose as an intermediator between the web system and the fabric, where the hyperledger would store the file hash with some other information. But hyperledger compose was found unsuitable because its development has been stalled, it uses python2 and its developers recommend developing smart contracts using hyperledger fabric SDK's.

The hyperledger fabric installation process is based on docker. This is a brilliant idea because we can spin up a peer-to-peer network with more than one node in a single machine.

Hyperledger fabric requires Go programming language hence the latest Go version was installed. It also requires some predefined environment variables to be setup.

The hyperledger has been successfully tested. Hence it was felt to use a python SDK which would allow us to integrate the hyperledger with the web system. However, this was unsuccessful because of the following:

- python sdk is not official and does not have proper tutorials.
- It requires the use of a network profile which is a JSON file that must be manually set.
- The usage of Certificate Authorities is the most difficult part. The private and public keys have to be copied from the docker containers and pasted in the configuration files of the python SDK. Otherwise, the SSL Handshake fails due to certificate verification failure. This was encountered during the development.

An innovative approach was used to solve the problem. Instead of relying on a separate SDK for connection to the Hyperledger, why not invoke the Hyperledger from the Python system itself? This is made possible using the classic system() UNIX system call which executes a shell command. The python equivalent is provided by the os module i.e., os.system(). The idea is that Hyperledger can be invoked using a set of CLI commands. These commands are language independent. Hence, there is no need to build an SDK and instead directly invoke these commands from python.

With all the above points in mind, the steps to fully configure project. Note that the project works on Linux only. Note that you must install golang, docker and hyperledger fabric:

Make sure the below lines are present in the bashrc file:

- ***export PATH=$PATH:/usr/local/go/bin/***
- ***export GOPATH=$HOME/go***
- ***export PATH=/home/kali/fabric-samples/bin:$PATH***
- ***export FABRIC_CFG_PATH=/home/kali/fabric-samples/config/***
- ***export CORE_PEER_TLS_ENABLED=true***
- ***export CORE_PEER_LOCALMSPID="Org1MSP"***
- ***export CORE_PEER_TLS_ROOTCERT_FILE=/home/kali/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt***
- ***export CORE_PEER_MSPCONFIGPATH=/home/kali/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp***
- ***export CORE_PEER_ADDRESS=localhost:7051***

Then run the following commands:

- ***ipfs daemon***
- ***sudo docker run --name some-postgres3 -p 5432:5432 -e POSTGRES_USER=admin_ems -e POSTGRES_PASSWORD=iamadmin@123 -d postgres***

Note: Use this command if above container is stopped: ***sudo docker start some-postgres3***

Install hyperledger fabric through the following command:

- ***curl -sSL http://bit.ly/2ysbOFE | sudo bash -s***

The business logic governing the blockchain is present in a program called the smart contract. In the terminology of the Hyperledger fabric, smart contract is called chain code. It is the chaincode that must be installed on all the peers in the network and invoked in the nodes.

For the purpose of configuration and development of the project, a GitHub repository was created. The URL of the repository is as follows: https://github.com/txh2020/Blockchain_Project

The smartcontract.go file present in this repo is placed instead of fabric-samples/asset-transfer-basic/chaincode-go/../smartcontract.go. After that, goto the fabric samples/test network. In the folder a terminal is opened and the following commands are typed:

- ***sudo ./network.sh up***
- ***sudo ./network.sh createChannel***
- ***sudo ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go***

Next, Goto the Blockchain_Project folder and run the following commands in a terminal.

- ***sudo su***
- ***source <bashrc_loc>/.bashrc***
- ***source /bin/activate***

Note that it is assumed that virtual environment is already setup. The steps to setup the virtual environment shall be discussed later.

Now open a browser. Open http://localhost:8000. Then open a private tab and goto http://localhost:8000/admin and enter Django superuser credentials.

Then the flow of the application is as follows:

- In admin
  - Add subject code
  - Add COE with subject code.
  - Add Teacher with subject code.
  - Add Superintendent with subject code.
- In another tab,
  - Login as COE. Submit. Send request.
  - Login as Teacher. Submit paper. ->CreateAsset
  - Login as COE. Finalize paper. ->ReadAsset
  - Login as superintendent. Download paper.

The createAsset() and readAsset functions are the ones invoking the Hyperledger.

Steps to setup virtual environment with other perquisites:

1. Install Postgres. Create user "admin_ems" with password "iamadmin@123"
2. Install ipfs desktop.
3. Make sure postgres runs on port 5432 and ipfs runs on port 5001
4. Clone this repository. Goto the cloned folder. Create a python virtual environment using ***python -m venv virtualvenv***

5. Activate the virtual environment using ***virtualenv\Scripts\activate*** in Windows and ***source virtualenv/bin/activate*** in Linux
6. Install all packages using ***pip install -r requirements.txt***
7. Then follow the following steps:

- ***python manage.py makemigrations***
- ***python manage.py migrate***
- ***python manage.py createsuperuser***
- ***python manage.py runserver*** to start the server

Results:

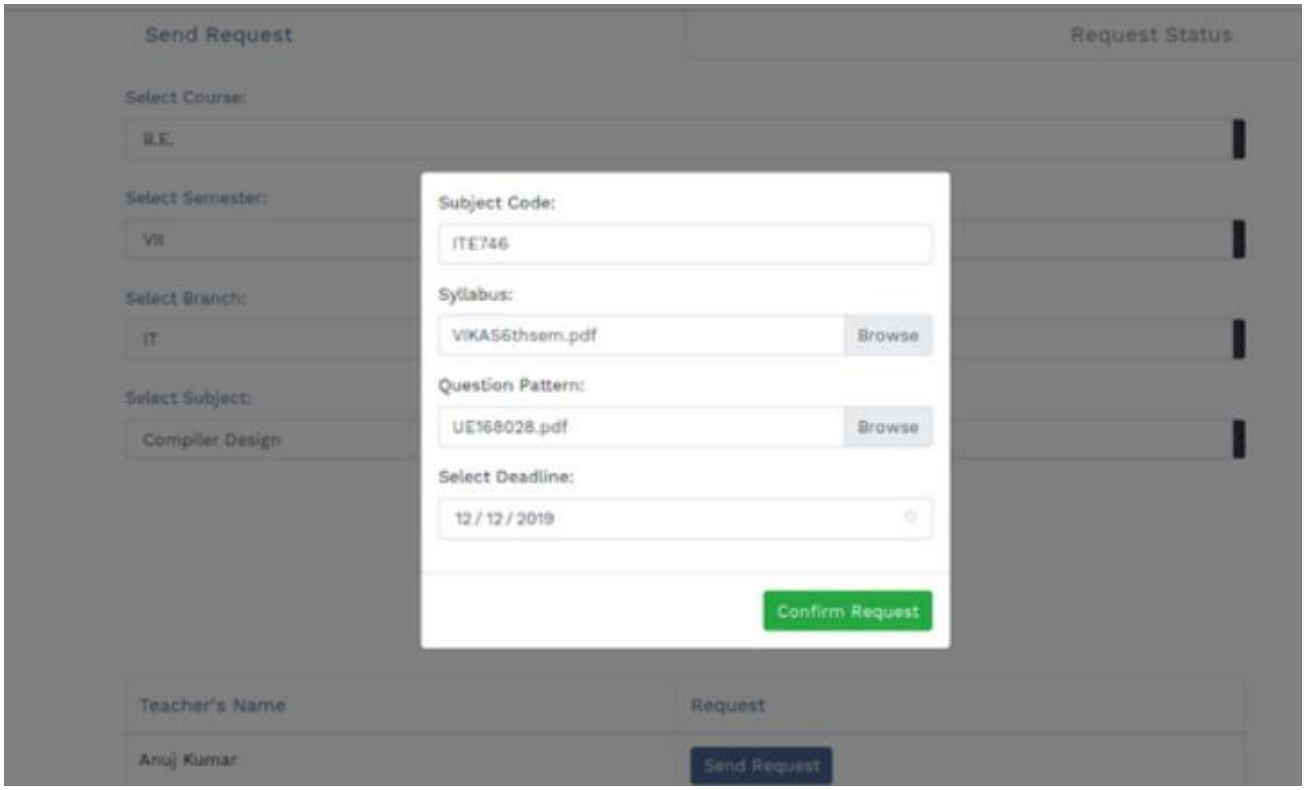

Fig. 2: COE Dashboard

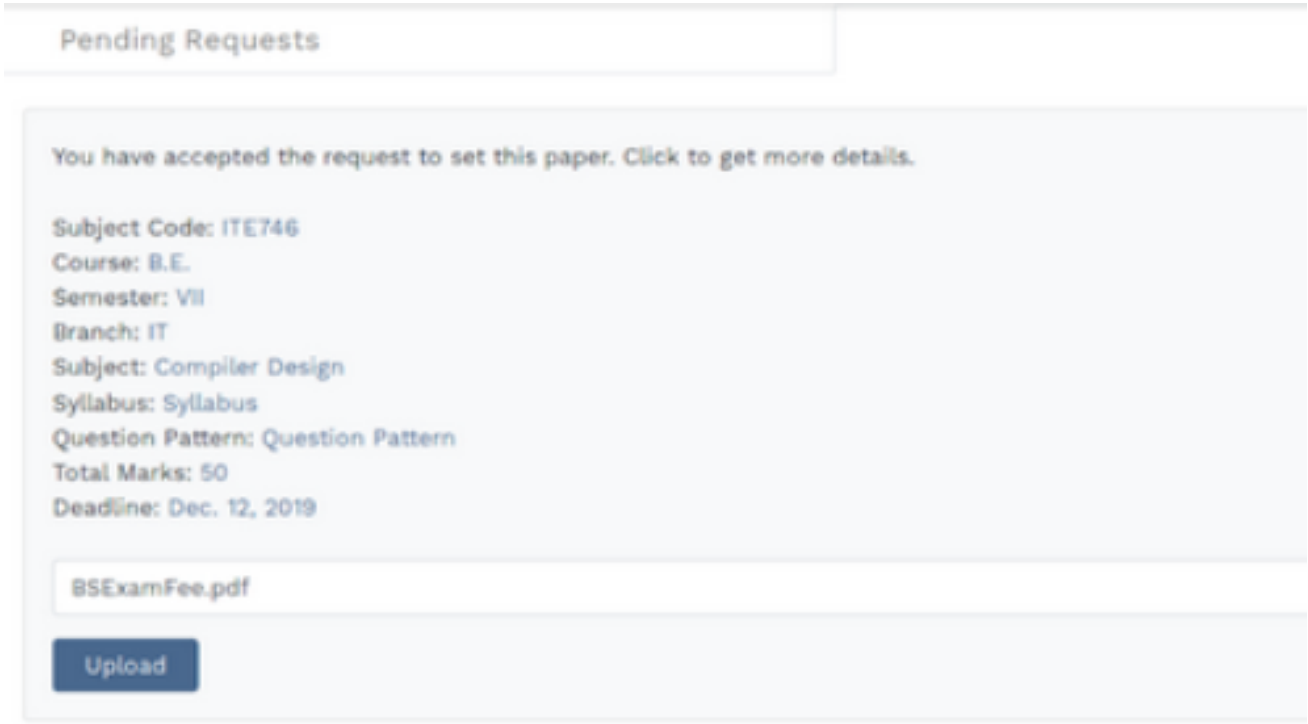

Fig. 3: Teacher Dashboard

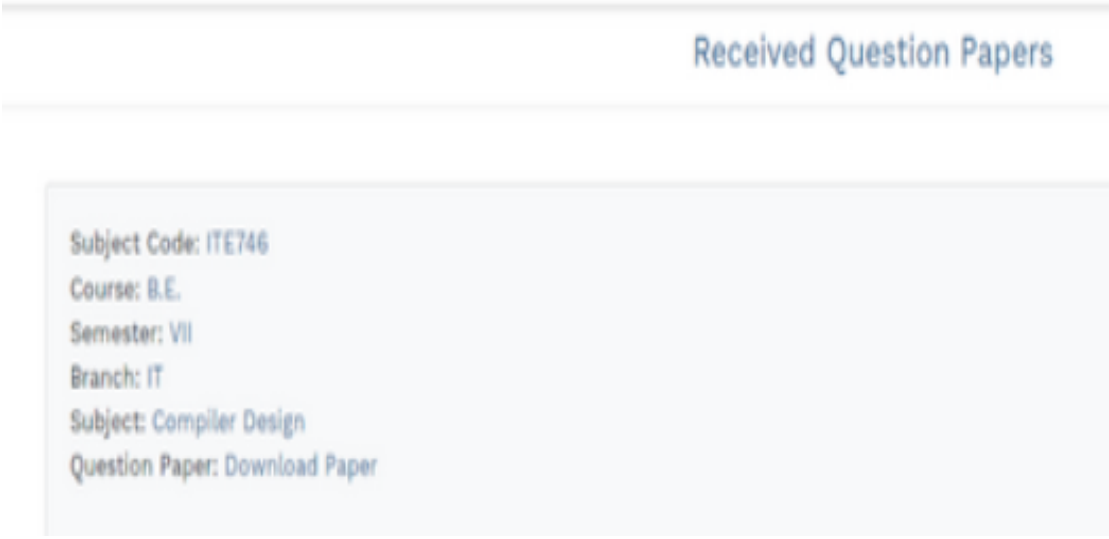

Fig. 4: Assistant (Superintendent) Dashboard

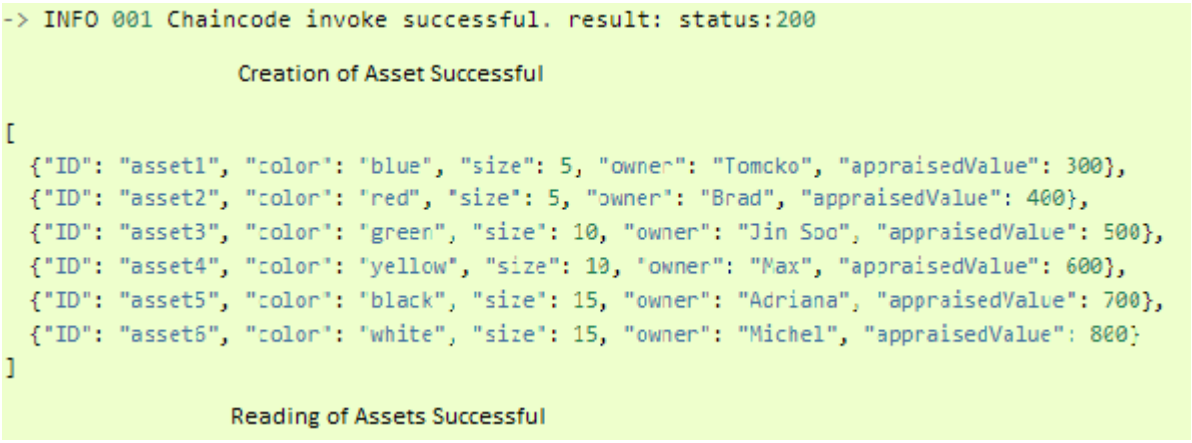

Fig. 5: Figure showing invocation of Hyperledger fabric

Conclusion:

The Student Examination System taught various concepts useful in building practical web applications such as:

- Linux OS: Useful for handling backend
- IPFS: Useful for enabling decentralized storage.
- Logging: Useful for debugging applications.
- Frontend tools such as server-side rendering.
- Python libraries Django and cryptography.
- Enterprise blockchain framework Hyperledger fabric

The Student Examination System is a useful application in terms of the potential it has for being used across various verticals in the IT industry. As stated earlier in the introduction, this software would prove very useful in education, enterprise softwares etc. with a modification according to the requirements.

The Student Examination System becomes a textbook example of how web application can and should be built. Most of the enterprise web applications today rely on the technologies that have been used in The Student Examination System such as Linux OS, IPFS, Hyperledger Fabric and Logging. Hence this software is a treasure trove of practical knowledge.

IEWT also taught various non-technical skills such as time management, logical reasoning and last but not the least the ability to dare to think outside the box without which would it would not have been possible to build this application.

References:

[1] Md Rahat Ibne Sattar etal, "An advanced and secure framework for conducting online examination using blockchain method", Cyber Security and Applications, Volume 1, 2023, 100005, ISSN 2772-9184, https://doi.org/10.1016/j.csa.2022.100005 .

[2] Tehseen, Rabia & Omer, Uzma & Farooq, Shoaib. (2021). Blockchain based Online Examination Assessment models for Educational Institutes. VFAST Transactions on Software Engineering. 9. 57-67. 10.21015/vtse.v9i3.707.

[3] Jain, Apoorv & Tripathi, Arun & Chandra, Naresh & Ponnusamy, Chinnasamy. (2021). Smart Contract enabled Online Examination System Based in Blockchain Network. 1-7. 10.1109/ICCCI50826.2021.9402420.

[4] Samanta, A.K., Sarkar, B.B. & Chaki, N. A Blockchain-Based Smart Contract Towards Developing Secured University Examination System. J. of Data, Inf. and Manag. 3, 237–249 (2021). https://doi.org/10.1007/s42488-021-00056-0

[5] Anu Bansal, “Secured-Blockchain-Based-Examination-Management-System”, https://github.com/anubansal17/Secured-Blockchain-Based-Examination-Management-System, 2017

[6] Devesh D, “Examination Management System”, https://github.com/deveshd2k/clgproject, 2019

[7] Hyperledger fabric, https://www.hyperledger.org/projects/fabric

[8] Hyperledger fabric tutorials, https://hyperledger-fabric.readthedocs.io/en/release-2.5/

[9] IPFS, https://ipfs.tech/

[10] Hyperledger fabric samples, https://github.com/hyperledger/fabric-samples

Online course attended:

NPTEL Advanced Distributed Systems:

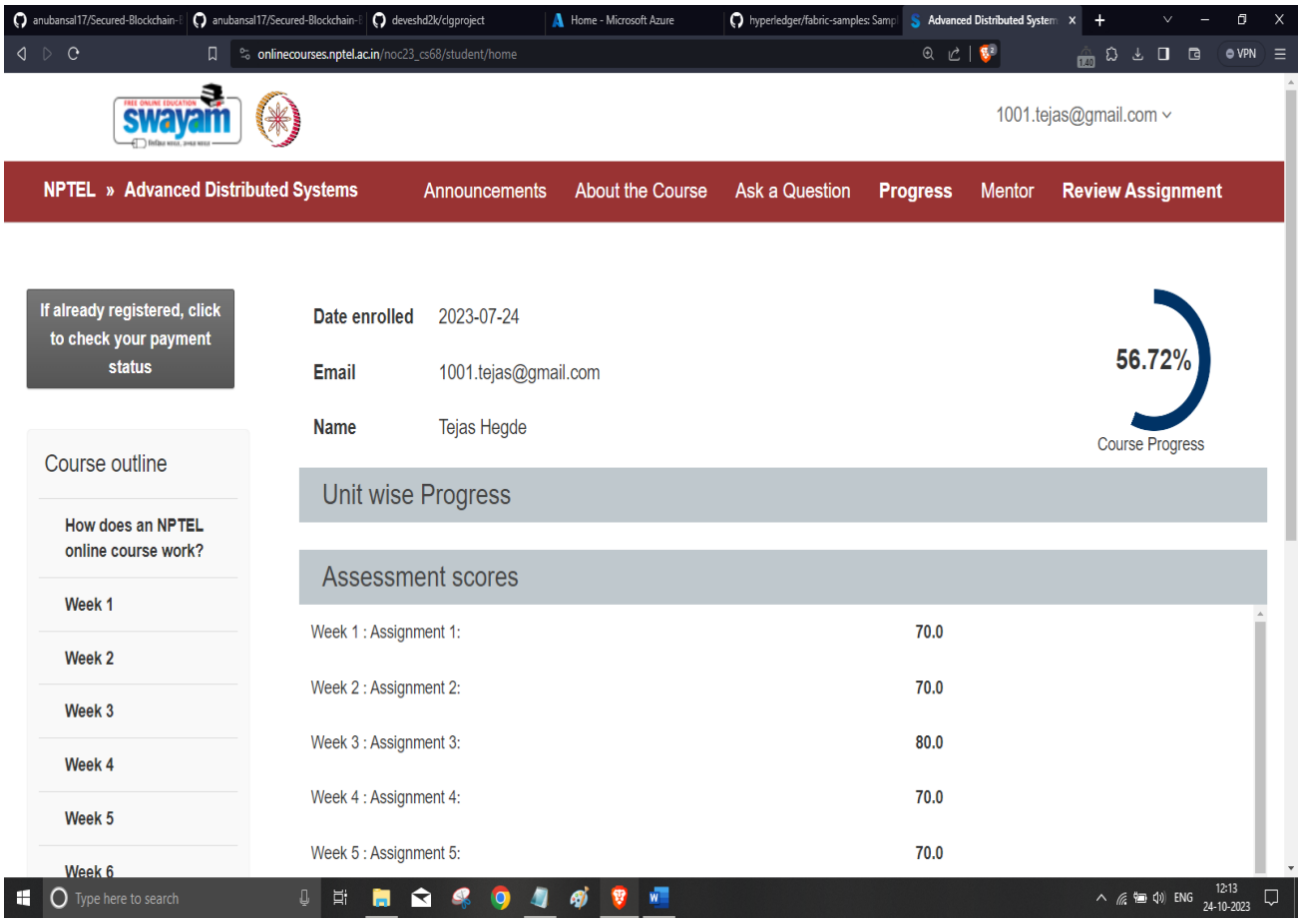

Fig. 6: Course attended and assignment proof