



ಶಿಕ್ಷಣೇ ವಿದ್ಯಾ ನಿದಾನಂ

THE NATIONAL COLLEGE
Autonomous
Jayanagar, Bangalore-560070

PROJECT REPORT
ON
HYPERLEDGER FABRIC BLOCKCHAIN BASED
JOB APPOINTMENT STATUS APPLICATION
BY

NAYAN KUMAR YN

20NCJIB415

Under the guidance of

Prof. VARADARAJ R

FabJob project report submitted in partial fulfillment of the requirements
of

VI Semester BCA, THE NATIONAL COLLEGE JAYANAGAR



|| ಶ್ರೀಕೃಷ್ಣಾ ವಂದನಾ ನಮಃ ||

THE NATIONAL COLLEGE
Autonomous
Jayanagar, Bangalore-560070

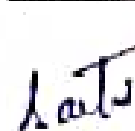
CERTIFICATE

This is to certify the project report titled "Job Appointment Status Application" is a work done by **NAYAN KUMAR YN** of THE NATIONAL COLLEGE, Jayanagar, Bengaluru, in partial fulfilment of the requirements of VI Semester BCA during the year 2022-2023.


HEAD OF THE DEPARTMENT
Head, Dept. of Computer Science
The National Degree College
(Autonomous)
Jayanagar, Bangalore - 560 070


PROJECT GUIDE

Examiners:

1. 

2.

Examination Centre

**The National College,
Jayanagar.**

Date of Examination:

ACKNOWLEDGEMENT

Job Appointment Status Application is the project of many hands from the team. Our tribute for the successful completion of the project goes to all those who helped through their constant guidance and encouragement. The satisfaction that accompanies the success would be incomplete without thanking the person who made it.

We are thankful to our beloved Principal DR.B.SURESHA, who encourages us to come with new and innovative ideas and for providing the environment with all facilities for completing the project.

We are also grateful to our Head of the Department Prof. TS ASHA Department of computer science for her valuable guidance and constant support during our project development.

We are also grateful to our project guide Prof. VARADARAJ, lecturer Department of computer science for her valuable guidance and constant support during our project development.

A special thanks to **MUTHURAM GOVINDARASU**, CEO and Founder of Indigeonous Tech Private Limited, Bangalore-32 with 10 years of experience in Blockchain for his valuable guidance and technical support for our project.

We extend our thanks to all our teaching staffs of the department of computer science. Finally, we thank one and all who helped us directly and indirectly for the completion of our project.

Table of Contents

I	Project Goal (Problem Statement)	6
II	Solution Proposed	7
III	Input Data and Verification	8
IV	Project/ Solution Design	10
V	Tools/ Technologies Used	12
VI	Project Team Members	12
VII	Referenced Documents	13
VIII	Project / Solution Setup	13
1)	Account and EC2 Instance creation and connection	13
a)	Create an AWS free account and further create an EC2 instance with Ubuntu Operating System	13
b)	Connect to the AWS EC2 Instance through "Putty" Software and open the Terminal. Change over to "root" directory with Admin permissions	13
2)	Download the required pre-requisite tools and Software into the EC2 Instance, transfer the "FabJob" Project files from Windows System to EC2 Instance using "FileZilla" and setup the project in the EC2 Instance	14
a)	Download and install all the pre-required tools and software for the Hyperledger Fabric based Blockchain set-up in the created EC2 Instance	14
b)	In the EC2 instance, clone the "fabric-samples" directory, download and install the Hyperledger Based Blockchain Binaries Version 1.4.6 and download the Hyperledger based Docker images from Docker hub.	14
c)	Transfer these program and files from Windows System to the created AWS EC2 Instance using the "FileZilla" software	14
d)	Go to "FabJob-network" directory and create Cryptographic based certificates and update the "docker-compose.yaml" file with the relevant CA certificate and save the file 14	
e)	Create a directory "channel-artifacts" and create genesis.block and channel.tx files under that directory and verify. Covert the "start.sh" file into an executable file	14
f)	Go to "fabjob" directory and covert the "startFabJob.sh" and "teardownFabJob.sh" files into executable ones	14
g)	Execute the command "npm install" and verify the creation of "node_modules" directory	14
IX	Project / Solution Execution	14
1)	In the "fabjob" directory, execute the command "_StartFabJob.sh" command and verify that the Blockchain Network is up and deployed	14
2)	Execute the command "node enrollAdmin.js" and verify that the "Admin" is enrolled successfully	17
3)	Execute the command "node registerUser.js" and verify that the "user1" is registered and enrolled successfully	18

4) Execute the command "node query-All-Jobs.js" and verify that the all the job data stored inside the blockchain are read and displayed successfully	18
5) Execute the command "node query-Job4.js" and verify that the JOB4 details are read from the blockchain and displayed successfully	19
6) Execute the command "node invoke-JOB10.js" and check that the JOB10 has got stored into the blockchain successfully	19
7) Execute the command "node query-JOB10.js" and check that the JOB10 data is read from the blockchain and displayed successfully	20
8) Execute the command "node invoke-JOB10-Status-change.js" and check that the new owner of JOB10 has got stored into the blockchain successfully	20
9) Execute the command "node query-JOB10.js" and check that the JOB10 data is read from the blockchain and displayed successfully with new Job owner	20
X) Setup and loading of "FabJob Frontend"	21
1) Open the "/fabjob-front/src/App.js" file and update the Public IP address of the FabJob EC2 Instance and save the file	21
2) From the "fabjob" directory, execute the command "node fabjob-backend.js" and verify the display of "Listening on port 4001"	24
3) Create the duplicate session of the EC2 Instance, navigate to "fabjob-front" directory in that instance, execute the command "npm run start" and verify the successful starting of the React App Development Server	25
4) Load the "fabjob-frontend" onto the Chrome browser by using the url: "http://Public IP of the Instance:3000/" and verify the successful display of the fabjob frontend	26
XI) Interaction with the FabJob Blockchain Network using FabJob Frontend from Windows System Browser	28
1) Click on "QUERY ALL" option and then "SEARCH ALL" button	28
2) Click on QUERY, enter JOB4 and then click on "SEARCH" button	28
3) Click on CREATE option, enter the new Job details and then click on "CREATE" button	Error! Bookmark not defined.
4) Click on QUERY option, enter JOB11 and click on "SEARCH" button	29
5) Click on TRANSFER option, enter JOB11 for JOB ID and Prakash for New Owner and then click on "TRANSFER" button	30
6) Click on QUERY option, enter JOB11 and click on "SEARCH" button	30
XII) Closing of "FabJob" Project	31
XIII) Project Summary	35

I) Project Goal (Problem Statement)

To design, develop, implement and verify the Hyperledger Fabric based Block chain application having the following features:

- 1) Query from the Job' data that is available as part of the Blockchain (Reading bulk data from the Blockchain)
- 2) Query any one of the Job' data that is available as part of the Blockchain (Reading individual record or data from the Blockchain)
- 3) Add a new Job data and verify that it gets added successfully into the Blockchain (Writing a new record into the blockchain and its verification)
- 4) Change the Status of existing Job data and verify (Changing [means creating new record as no modification of existing record is allowed in Blockchain] the existing data in the blockchain and its verification)
- 5) Modify the Status of the existing Job data and verify (Modifying the existing data in the blockchain and its verification). Here, also a new record is created with new owner Name and the existing record is not getting modified which is one of the main feature of Blockchain technology

Note: These verifications need to be done both in Command Line Interface (CLI) mode and also using browser based front-end mode.

D) Solution Proposed

- 1) We will design, develop, implement and verify the Hyperledger Fabric based Blockchain network namely "Job Appointment Status Application" in an Ubuntu O/S based EC2 Instance in AWS.
- 2) We will write the required Blockchain configuration and other files so as to bring-up the Blockchain network
- 3) We will write a Golang program namely "fabjob.go" in which we will implement the main logic of the Solution.
- 4) In the fabjob.go program, we will design the Job data with the Name,Status,Sex,Designation etc..

```

$ go run fabjob.go
[{"id": "1", "name": "John", "status": "Applied", "sex": "Male", "designation": "Software Engineer"}, {"id": "2", "name": "Jane", "status": "Interviewed", "sex": "Female", "designation": "Marketing Executive"}, {"id": "3", "name": "Mike", "status": "Rejected", "sex": "Male", "designation": "Sales Representative"}, {"id": "4", "name": "Sarah", "status": "Applied", "sex": "Female", "designation": "Product Manager"}, {"id": "5", "name": "David", "status": "Interviewed", "sex": "Male", "designation": "Business Development"}, {"id": "6", "name": "Emily", "status": "Rejected", "sex": "Female", "designation": "Operations Manager"}, {"id": "7", "name": "Chris", "status": "Applied", "sex": "Male", "designation": "Human Resources"}, {"id": "8", "name": "Lisa", "status": "Interviewed", "sex": "Female", "designation": "Finance Analyst"}, {"id": "9", "name": "Tom", "status": "Rejected", "sex": "Male", "designation": "Quality Assurance"}, {"id": "10", "name": "Anna", "status": "Applied", "sex": "Female", "designation": "Customer Support"}]

```

- 5) The program will have 10 Job data which will initialize the Blockchain on the starting of the Blockchain

```

$ go run fabjob.go
[{"id": "1", "name": "John", "status": "Applied", "sex": "Male", "designation": "Software Engineer"}, {"id": "2", "name": "Jane", "status": "Interviewed", "sex": "Female", "designation": "Marketing Executive"}, {"id": "3", "name": "Mike", "status": "Rejected", "sex": "Male", "designation": "Sales Representative"}, {"id": "4", "name": "Sarah", "status": "Applied", "sex": "Female", "designation": "Product Manager"}, {"id": "5", "name": "David", "status": "Interviewed", "sex": "Male", "designation": "Business Development"}, {"id": "6", "name": "Emily", "status": "Rejected", "sex": "Female", "designation": "Operations Manager"}, {"id": "7", "name": "Chris", "status": "Applied", "sex": "Male", "designation": "Human Resources"}, {"id": "8", "name": "Lisa", "status": "Interviewed", "sex": "Female", "designation": "Finance Analyst"}, {"id": "9", "name": "Tom", "status": "Rejected", "sex": "Male", "designation": "Quality Assurance"}, {"id": "10", "name": "Anna", "status": "Applied", "sex": "Female", "designation": "Customer Support"}]

```

- 6) In the program, we will create the functions queryJob, initLedger, createJob, queryAllJobs and changeJobStatus to meet the Solution requirements.


```

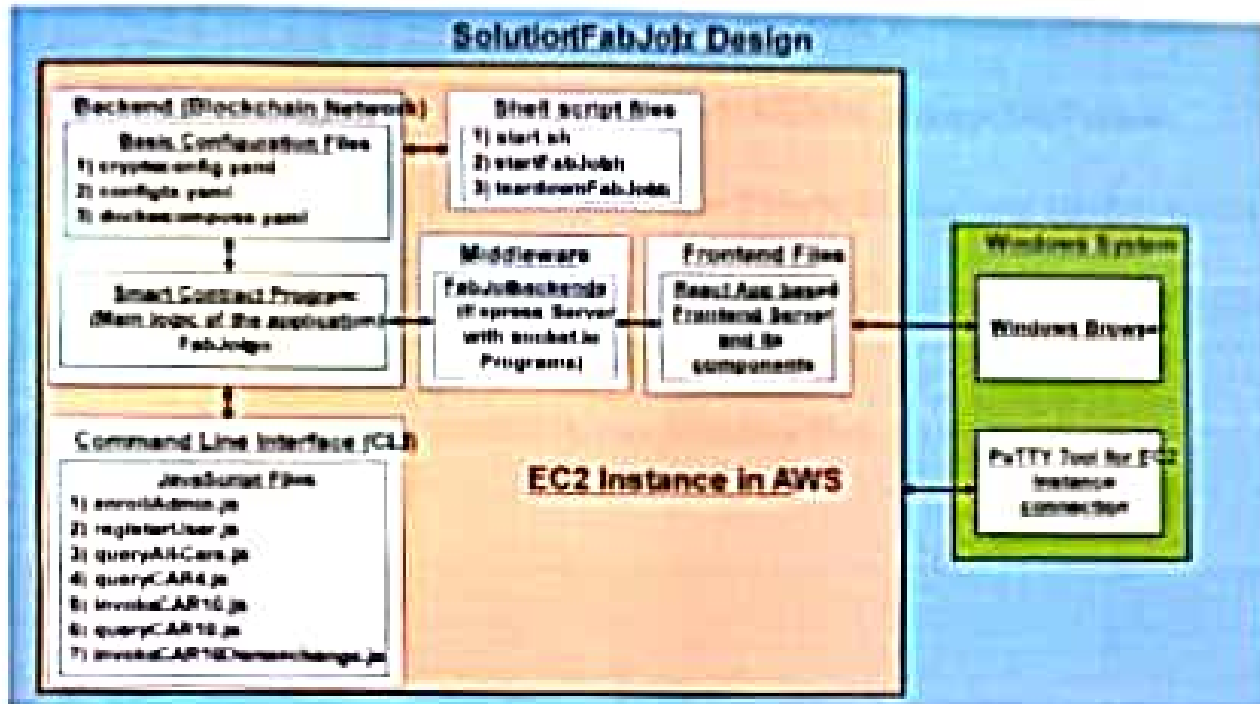
root@ip-172-31-42-255:~/fabric-samples/FabJob# node -change-job-status-Appointed.js
state path:/root/fabric-samples/FabJob/hfc-key-store
successfully loaded user1 from persistence
Assigning transaction id: b24f090724a8306569b4a020c44418511c8a4303c1220bf37ca5a4b90cf7704
Transaction proposal was good
successfully sent Proposal and received ProposalResponse: status - 200, message - **
the transaction has been committed on peer localhost:7051
send transaction promise and event listener promise have completed
successfully sent transaction to the orderer.
successfully committed the change to the ledger by the peer
root@ip-172-31-42-255:~/fabric-samples/FabJob# █

```

- 7) Also, we will create relevant JavaScript files to interact with the established Blockchain network in CLI mode, so as to meet the solution requirements.
- 8) We will create the required Shell script files so as to automate the starting and closing of the Blockchain network.
- 9) We will also create a frontend (React App based) for the solution so that you can interact with the Blockchain network from your Windows system.

III) Input Data and Verification

Sl.No.	Car ID	Make	Model	Colour	Owner	Remarks	Program Used
Initial 10 Cars' records to be updated in "fabcar.go" program							
1	CAR0	Toyota	Prius	blue	Tomoko	These 10 Cars' records data will get stored in a block of the Blockchain on starting of the Blockchain	fabcar.go
2	CAR1	Ford	Mustang	red	Brad		
3	CAR2	Hyundai	Tucson	green	Jin Soo		
4	CAR3	Volkswagen	Passat	yellow	Max		
5	CAR4	Tesla	S	black	Adriana		
6	CAR5	Peugeot	205	purple	Michel		
7	CAR6	Chery	S22L	white	Aarav		
8	CAR7	Fiat	Punto	violet	Pari		
9	CAR8	Tata	Nano	indigo	Valeria		
10	CAR9	Holden	Barina	brown	Shotaro		
Query and display all the 10 records stored in the block of the Blockchain in CLI Mode						Read and display of stored bulk records from Blockchain	query-All-Cars.js
Query the Car Record, having Car ID as CAR4 and display it in CLI Mode						Read and display of a single Car record (CAR4) from Blockchain	query-CAR4.js
Creation of New Car Record in the Blockchain using JavaScript File in CLI Mode						Creation of new Car (CAR10) record in a block of the Blockchain	invoke-CAR10.js
11	CAR10	Honda	Accord	Black	Tom	Read and verify the successful creation of the New Car (CAR10) record	query-CAR10.js
Verification of the created New Record in the Blockchain using JavaScript File in CLI Mode							
11	CAR10	Honda	Accord	Black	Tom	Modify the existing blockchain record (Here, the existing record is immutable). So, a new record will be created in the Blockchain reflecting the change	invoke-CAR10-Owner-change.js
Change of Owner of the Car (CAR10) using JavaScript File in CLI Mode							
11	CAR10	Honda	Accord	Black	Nike		
Query and verification of (CAR10) data with changed Owner name using JavaScript File in CLI Mode							
11	CAR10	Honda	Accord	Black	Nike	Creation of new Car (CAR11) record in a block of the Blockchain	Using frontend
Creation and verification of New Car (CAR11) record using Frontend							
12	CAR11	Mahindra	XUV500	Silver	Ramu	Read and verify the successful creation of the New Car (CAR11) record	Using frontend
Change and verification of changed Owner of Car (CAR11) using Frontend							
12	CAR11	Mahindra	XUV500	Silver	Prakash		



High Level:

- 1) Create the following in the EC2 Instance of an AWS Account:
 - a) Golang program with required data and functions
 - b) FabJob Blockchain basic configuration files
 - c) Required Shell Script files
 - d) CLI based JavaScript files in the EC2 Instance
 - e) Middleware JavaScript file for the frontend interaction
 - f) React App based frontend files
- 2) Download and install the following in your Windows System:
 - a) PuTTY.exe along with PuTTYgen
 - b) Filezilla software tool

Details:

- 1) Write a goLang program namely "fabJob.go" with the below given info:
 - a) Job Data Structure (make, model, colour, owner)
 - b) Functions (queryJob, createJob, queryAllJobs, changeJobStatus)
 - c) Data for 10 Jobs
- 2) Write the below given basic Hyperledger Fabric Blockchain configuration files
 - a) crypto-config.html
 - b) configtx.html
 - c) docker-compose.html
- 3) Write the below given JavaScript application files
 - a) enrollAdmin.js
 - b) registerUser.js
 - c) query-All-Jobs.js
 - d) query-Job4.js
 - e) invoke-Job10.js
 - f) query-Job10.js
 - g) invoke-Job10-Status-change.js
- 4) Write the below given Shell script files
 - a) start.sh
 - b) startFabJob.sh
 - c) teardownFabJob.sh
- 5) Write the middleware JavaScript file namely "fabjob-backend.js"
- 6) Create the required React App based frontend files

Note: Organise these files as given in the below given directory structure. We need to transfer this folder to the EC2 Instance using "Filezilla" tool



V) Tools/ Technologies Used

- AWS EC2 Instance: AWS EC2 Instance with O/S Ubuntu AMI
- Nodejs and NPM
- Golang
- Docker Container and Docker Compose
- Hyperledger Fabric based Blockchain Binaries Version 1.4.6
- Hyperledger based Docker Images
- JavaScript
- Shellscript
- React App

VI) Project Team Member

- 1) Nayan Kumar Y N

VI) Referenced Documents:

- 1) **Appendix-A: 02-AWS-Free-Account-and-Instance-Creation-Steps-25-Aug-2022.docx**
- 2) **Appendix-B: 03-Setup-Hyperledger-Fabric-based-Blockchain-Network-and-Setup-Environment-for-FabJob-Project-25-Aug-2022.docx**
- 3) **Appendix-C: 04-List-of-Programs-and-Files-Used-in-FabJob-Project-25-Aug-2022.docx**

VII) Project / Solution Setup

- 1) **Account and EC2 Instance creation and connection**
 - a) **Create an AWS free account and further create an EC2 instance with Ubuntu Operating System**
 - b) **Connect to the AWS EC2 Instance through "Putty" Software and open the Terminal. Change over to "root" directory with Admin permissions**

(For the above given Steps-1 and 2, please refer the Document at "Appendix-A" to this report)

- 2) Download the required pre-requisite tools and Software into the EC2 Instance, transfer the "FabJob" Project files from Windows System to EC2 Instance using "FileZilla" and setup the project in the EC2 Instance
- Download and install all the pre-required tools and software for the Hyperledger Fabric based Blockchain set-up in the created EC2 Instance
 - In the EC2 Instance, clone the "fabric-samples" directory, download and install the Hyperledger Based Blockchain Binaries Version 1.4.6 and download the Hyperledger based Docker images from Docker hub.
 - Transfer these program and files from Windows System to the created AWS EC2 Instance using the "FileZilla" software
 - Go to "FabJob-network" directory and create Cryptographic based certificates and update the "docker-compose.yaml" file with the relevant CA certificate and save the file
 - Create a directory "channel-artifacts" and create genesis.block and channel.tx files under that directory and verify. Convert the "start.sh" file into an executable file
 - Go to "FabJob" directory and convert the "startFabJob.sh" and "teardownFabJob.sh" files into executable ones
 - Execute the command "npm install" and verify the creation of "node_modules" directory

(For the above given Steps-a to g, please refer the Document at "Appendix-B" to this report)

(For referring to the Programs used in this project, please refer the Document at "Appendix-C" to this report)

3) Project / Solution Execution

- In the "FabJob" directory, execute the command "./StartFabJob.sh" command and verify that the Blockchain Network is up and deployed