

Health Care Data (SPE Final project)

1. Abstract

2. Introduction

3. Overview

4. Why DevOps?

4.1 Devops Features

5. System Configuration

5.1 OPERATING SYSTEM

5.2 CUP AND RAM

5.3 LANGUAGE AND FRAMEWORK

5.4 KERNEL VERSION

5.5 DATABASE

5.6 BUILDING TOOLS

5.7 DEVOPS TOOLS

6. Software Development Lifecycle

6.1 Installation

6.1.1 Node js

6.1.2 Spring framework

6.1 Source Control Management

6.1.1 BLOCKCHAIN REPOSITORY

6.1.2 SPRING BOOT APPLICATION REPOSITORY

6.1.3 CI Pipeline

6.2 Build

6.2.1 Maven

6.2.2 NPM

6.2.3 CI Pipeline

6.3 Docker Artifact

6.3.1 CI Pipeline

6.4 Deployment

6.5 Monitor

7 Building Pipeline and Running on Host Machines

8 Experimental Setup

8.1 DESIGN DIAGRAM

8.2 Blockchain Properties

8.2.1 Distributed Network
8.2.2 Consensus Algorithm
8.2.3 Proof of Work
8.2.4 Mining

9. Result

9.1 LOGIN PAGE
9.2 REGISTER PAGE
9.3 PUBLIC DNS ENTRY
9.4 PATIENT DASHBOARD
9.5 BOOK DOCTOR
9.6 GIVE / REVOKE PERMISSION
9.7 VIEW APPOINTMENTS
9.8 DOCTOR DASHBOARD
9.9 APPOINTMENTS
9.10 PRESCRIBE
9.11 PAST RECORDS
9.12 VERIFY IN BLOCKCHAIN
9.13 RUNNING BLOCKCHAIN

10 Conclusion

1. Abstract

- Medical history of a patient holds an important and key role in prescribing and counter the current health problems of a patient. Missing out some details in history may create a huge problem. Also, many time the patient may forget his medication, his tests and even the problems he faced.
- Getting the data is essential but knowing the authenticity of the data provided is more important. The doctor should have the confidence that the data he is looking at is not tampered or incorrect. The patient also must have the trust on the system that its data will be secure and there will be no non authorized access to it since it contains sensitive medical information.
- Another major problem faced is the denial of a doctor prescribing some medication. There is a need for immutable history to which none of the parties accessing it can deny or doubt its authenticity.

2. Introduction

- This project aims at creating a non-permissible blockchain which stores the health data of a patient. The blockchain stores the timeline of a person's medical history. The main business reason behind creating this project was that the current doctor examining the patient can have access to authentic medical history of the patient. Many times, the patient forgets the medications he was given or even some problem he had in the past. So, this blockchain will have the data to overcome that problem. Also, sometimes patient does not want to reveal some information regarding health but hiding any of such details from a doctor may create a problem. So, this project also tackles it.
- Data is stored inside the blockchain after proper mining processes hence the authenticity of data can be trusted. The blockchain also tackles the problem of non-repudiation from a doctor's end. Maybe some wrong medication or wrong tests have been done or suggested by the doctor then incase if things go haywire, neither the doctor nor the hospital can deny something happening as whatever is stores on blockchain is permanent and authentic.
- The blockchain also provides APIs for 3rd party integration of a hospital so that the hospital can access and store the data over it. As a part of this project, we have also created a web application of a hospital which demonstrates the use of blockchain.

3. Overview

- Blockchain works over various nodes in a distributed environment and are also in sync with each other. New nodes can also register and contribute over the blockchain. On registration they get the latest information and are also part of the distributed system. 3rd party applications can request any service of the blockchain from any of the nodes in the network.
- The reliability of the data is also high since the system is distributed.
- Functionalities provided by blockchain:
 - Save the entry.

- Mine a block.
- Access a block.
- Access history of a patient or doctor.
- This blockchain has 4 key attributes which it maintains inside an entry. All the entries are same as a transaction in generic blockchain.
- Four attributes inside our entry are : patient id, doctor id, description and prescription. Whenever a doctor stores this information inside the blockchain after examining a patient, this information goes inside the pending entries and when the mining process is done by any of the nodes in the blockchain the entries become permanent inside a block.
- After the data is permanent both the doctor and patient can access the history.
- Patient can look at his entire history while a doctor can look at the patients, he has prescribed in the past, but he can get the history of only those patients whose data access is given to him by the patient.
- Apart from this scheduling an appointment with doctor, input of prescription, viewing history, viewing profile and other miscellaneous functionalities are provided out of blockchain as a part of hospital web application. As these features are application specific, they can be built however the 3rd party prefers.

As part of the web app in our project we have provided the following functionalities:

- View profile
- Portal for viewing history. Both patient and doctor.
- Form input for prescribing patient
- Viewing appointments.

4. Why DevOps?

Our whole approach of the project was modular, we wanted to make different sets of development modules and wanted to deploy them with every new release without any hindrance. Wanted to test the changed code with continuous integration and then continuously deploying it. So what all fills all these blanks was a culture, a philosophy DevOps. Devops provides all the tools to increase the capability to complete above set goals within minimum time and less trouble for developers. DevOps tools consist of configuration management, test and build systems, application deployment, version control and monitoring tools. Continuous integration, continuous delivery and continuous deployment require different tools.

4.1 Devops Features

- Improve deployment frequency
- Achieve faster time to market with lower failure rate
- More stable operating environments
- Improve communication and collaboration among teams

5. System Configuration

5.1 OPERATING SYSTEM

- Ubuntu 20.04.2 LTS

5.2 CUP AND RAM

- 4 core processor and RAM 4 GB (preferable 8 GB)

5.3 LANGUAGE AND FRAMEWORK

- JavaScript, Node.js
- Java, Springboot

5.4 KERNEL VERSION

- 5.8.0-53-generic

5.5 DATABASE

- MySQL 8.0.23 (MySQL Community Server - GPL)

5.6 BUILDING TOOLS

- Maven to build java application
- NPM to build node application

5.7 DEVOPS TOOLS

- Source Control Management - GitHub
- Continuous Integration - Jenkins
- Containerization - Docker
- Continuous deployment - Ansible
- Monitoring - ELK Stack (Elastic Search, Logstash, Kibana)

6. Software Development Lifecycle

6.1 Installation

6.1.1 Node js

We can install node js using following command.

```
| $sudo apt install nodejs
```

```

akshil@akshil-Inspiron-5379:~$ sudo apt install nodejs
[sudo] password for akshil:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  blt gir1.2-gst-plugins-base-1.0 gir1.2-rb-3.0 libdmapsharing-3.0-2
  libfpint-2-tod1 libgpod-common libgpod4 libllvm10 libllvm9 libsgutils2-2
  libtcl8.6 libtk8.6 tk8.6-blts
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libc-ares2 libnode64 nodejs-doc
Suggested packages:
  npm
The following NEW packages will be installed:
  libc-ares2 libnode64 nodejs nodejs-doc
0 upgraded, 4 newly installed, 0 to remove and 22 not upgraded.
Need to get 6,807 kB of archives.
After this operation, 30.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] ■

```

Also install npm package manager to handle dependencies.

```
$sudo apt install npm
```

```

akshil@akshil-Inspiron-5379:~$ sudo apt install npm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  blt gir1.2-gst-plugins-base-1.0 gir1.2-rb-3.0 libdmapsharing-3.0-2 libfpint-2-tod1 libgpod-common libgpod4 libllvm10 libllvm9
  libsgutils2-2 libtcl8.6 libtk8.6 tk8.6-blts
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  gyp libc-ares2 libbs-inherits libjs-is-typedarray libjs-psl libjs-typedarray-to-buffer libnode-dev libnode64 libuv1-dev
  node-abbrev node-ajv node-ansi node-ansi-align node-ansi-regex node-ansi-styles node-ansistyles node-aproba node-archy
  node-are-we-there-yet node-asap node-asn1 node-assert-plus node-asynckit node-aws-sign2 node-aws4 node-balanced-match
  node-bcrypt-pbkdf node-bl node-bluebird node-boxen node-brace-expansion node-builtins node-node-builtins node-cacache
  node-call-limit node-camelcase node-caseless node-chalk node-chownr node-ci-info node-cli-boxes node-clui node-clone node-co
  node-color-convert node-color-name node-colors node-columnify node-combined-stream node-concat-map node-concat-stream
  node-config-chain node-configstore node-console-control-strings node-copy-concurrently node-core-util-is node-cross-spawn
  node-crypto-random-string node-cyclist node-dashdash node-debug node-decamelize node-decompress-response node-deep-extend
  node-defaults node-define-properties node-delayed-stream node-delegates node-detect-indent node-detect-newline node-dot-prop
  node-duplexer3 node-duplexify node-ecc-jbn node-editor node-encoding node-end-of-stream node-err-code node-errno
  node-es6-promise node-escape-string-regexp node-exec node-extend node-extsprintf node-fast-deep-equal node-find-up
  node-flush-write-stream node-forever-agent node-form-data node-from2 node-fs-vacuum node-fs-write-stream-atomic
  node-fs.realpath node-function-bind node-gauge node-genfun node-get-caller-file node-get-stream node-getpass node-glob node-got
  node-graceful-fs node-gyp node-har-schema node-har-validator node-has-flag node-has-symbol-support-x node-has-to-string-tag-x
  node-has-unicode node-hosted-git-info node-http-signature node-iconv-lite node-iferr node-import-lazy node-imurmurhash
  node-inflight node-inherits node-ini node-invert-kv node-ip node-ip-regex node-is-npm node-is-obj node-is-object
  node-is-path-inside node-is-plain-obj node-is-retry-allowed node-is-stream node-is-typedarray node-isarray node-isexe
  node-isstream node-isurl node-json node-json-parse-better-errors node-json-schema node-json-traverse
  node-json-stable-stringify node-json-stringify-safe node-jsonify node-jsonparse node-jsonstream node-jsprim node-latest-version
  node-lazy-property node-lcid node-libnpx node-locate-path node-lockfile node-lodash node-lodash-packages node-lowercase-keys

```

6.1.2 Spring framework

Here we have used java to create the healthcare-spring code. And maven was used to build the spring java project.

```
| $ sudo apt install default-jdk
```

```
akshil@akshil-Inspiron-5379:~$ sudo apt install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  blt gir1.2-gst-plugins-base-1.0 gir1.2-rb-3.0 libdmapsharing-3.0-2 libfprint-2-tod1 libgpod-common libgpod4 libllvm10 libllvm9
  libsgutils2-2 libtcl8.6 libtk8.6 tk8.6-blt2.5
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  default-jdk-headless default-jre-headless openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre
  openjdk-11-jre-headless
Suggested packages:
  openjdk-11-demo openjdk-11-source visualvm fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei
The following NEW packages will be installed:
  default-jdk default-jdk-headless default-jre default-jre-headless openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre
  openjdk-11-jre-headless
0 upgraded, 8 newly installed, 0 to remove and 22 not upgraded.
Need to get 262 MB of archives.
After this operation, 406 MB of additional disk space will be used.
Do you want to continue? [Y/n] ■
```

We used maven as build tool for spring application.

```
| $ sudo apt install maven
```

```
akshil@akshil-Inspiron-5379:~$ sudo apt install maven
Reading package lists... Done
Building dependency tree
Reading state information... Done
maven is already the newest version (3.6.3-1).
The following packages were automatically installed and are no longer required:
  blt gir1.2-gst-plugins-base-1.0 gir1.2-rb-3.0 libdmapsharing-3.0-2 libfprint-2-tod1 libgpod-common libgpod4 libllvm10 libllvm9
  libsgutils2-2 libtcl8.6 libtk8.6 tk8.6-blt2.5
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 22 not upgraded.
```

After **mvn clean install**, once JAR file is ready you can run the application using this command.

```
| java -jar application.jar
```

Here is our Project structure.

The screenshot shows the IntelliJ IDEA interface with the project tree on the left and a code editor on the right. The project tree includes .idea, .mvn, src (main, java, data_sources, filters, model, services), and resources. The code editor displays a portion of the RestController.java file:

```

115 logger.info("user registered:" + curr.toString());
116
117 if(dbUtil.registerNewUser(curr) == 1) {
118     return "Login";
119 } else {
120     theModel.addAttribute(s: "isError", o: "Error creating new user. User Na");
121     return "register";
122 }
123
124
125
126 }
127
128 @GetMapping("/publicDNSEntry")
129 public String publicDNSEntry() { return "publicDNSEntry"; }
130
131 @PostMapping("/publicDNSEntry")
132 public String enterPublicDNS(HttpServletRequest req) throws Exception {
133     dbUtil.enterIp(req.getParameter(s: "ipaddress"));
134     return "login";
135 }
136
137
138
139 @GetMapping("/userProfile")
140 public String userProfile(HttpServletRequest req) throws Exception {
141     String userProfilePage = "userProfile.jsp";
142     Model theModel = new Model();
143     theModel.addAttribute("curr", curr);
144     theModel.addAttribute("ipaddress", req.getParameter(s: "ipaddress"));
145     theModel.addAttribute("isError", "false");
146     theModel.addAttribute("userProfilePage", userProfilePage);
147     return userProfilePage;
148 }

```

The screenshot shows the IntelliJ IDEA interface with the project tree on the left and a code editor on the right. The project tree includes services, util (DbUtil, JwtUtil, MiscUtil), HealthCareDataApplication, resources (static, templates), and a code editor for RestController.java. The code is identical to the one in the previous screenshot.

6.1 Source Control Management

A Source Code Management (SCM) is a software tool used by programmers to manage the

source codes. For our project every team member would clone the repository from github. Create

a different branch locally on their system and then merge it with master after pulling the latest

code from git, resolving any conflicts and then push the changes to git.

- `git clone <repository url>` - This command copies the entire data on the git url

- git checkout -b <branch_name> - This command creates a new branch with the name as in 'branch_name'
- git add <changed files> - This command adds changes in the working directory to the staging area
- git commit -m "message while committing" - This command is used to save your changes to the local repository with -m used to provide a concise description that helps your teammates (and yourself) understand what happened.
- git checkout master - This command switches to master branch
- git pull - This command is used to update the local version of a repository from a remote.
- git merge <branch_name> - This command is used to integrate changes from another branch.
- git push - This command will push all the latest code to the repository

6.1.1 BLOCKCHAIN REPOSITORY

Link - <https://github.com/Akshil007/Blockchain>

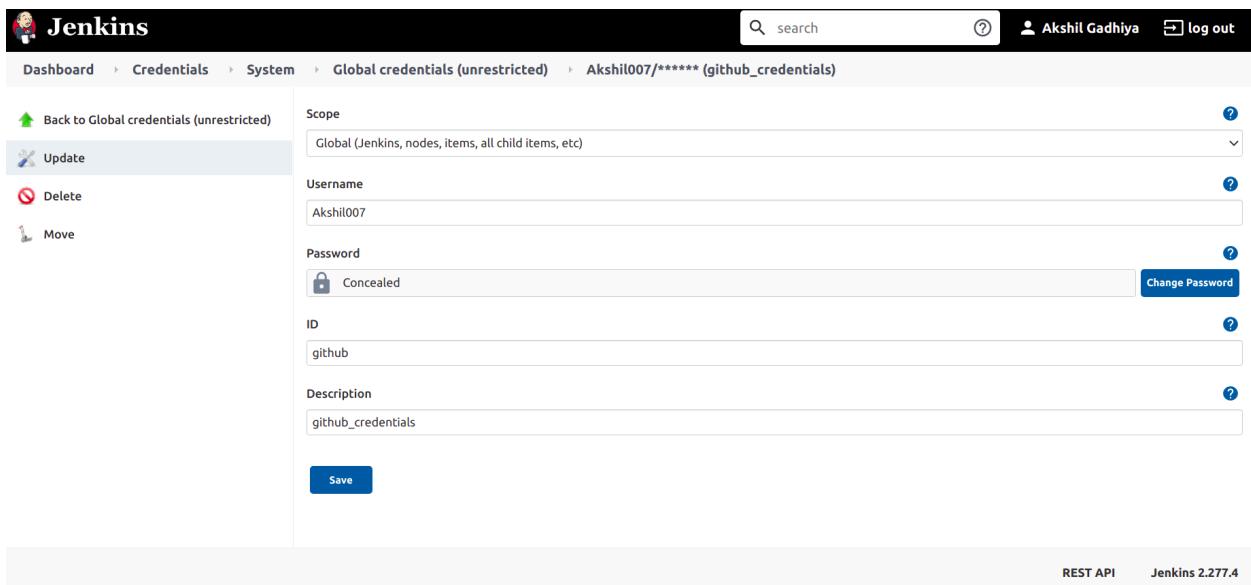
6.1.2 SPRING BOOT APPLICATION REPOSITORY

Link - <https://github.com/alaysd/HeathCareDataWebApp>

6.1.3 CI Pipeline

We used jenkins pipeline to clone these repositories from git.

- First we will add credential of github into jenkins so we can refer it later in script.

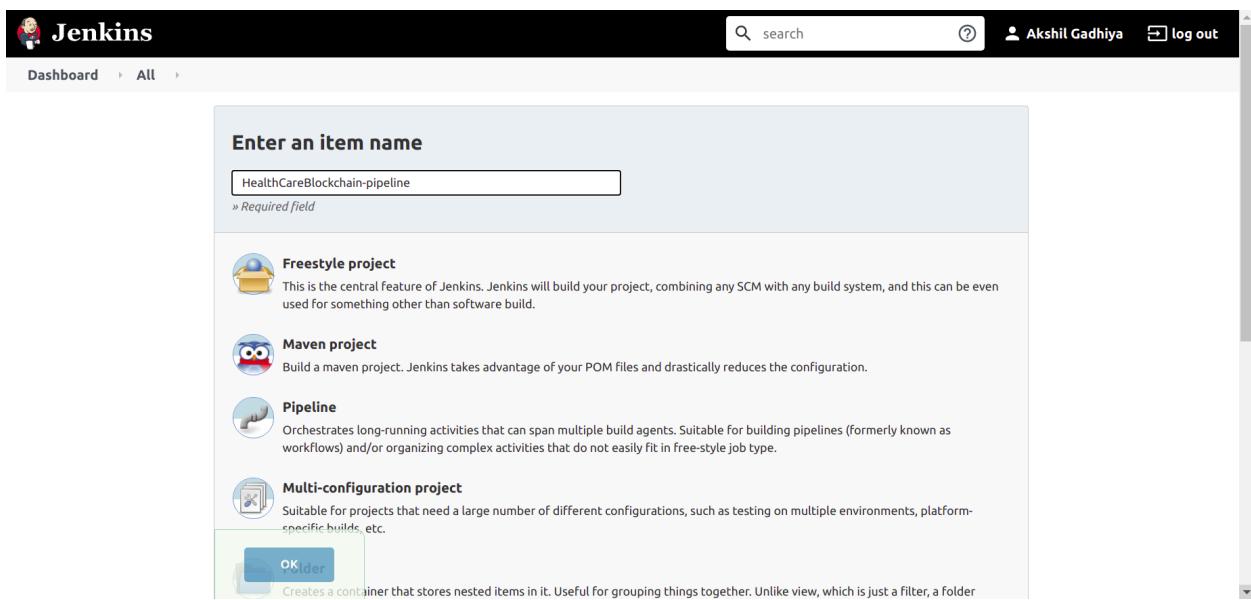


The screenshot shows the Jenkins Global credentials configuration page. The URL is `/global-credentials/unrestricted/Akshil007/***** (github_credentials)`. The page has a sidebar with options like Back to Global credentials (unrestricted), Update, Delete, and Move. The main form fields are:

- Scope:** Global (Jenkins, nodes, items, all child items, etc)
- Username:** Akshil007
- Password:** Concealed (with a Change Password button)
- ID:** github
- Description:** github_credentials

At the bottom right are links for REST API and Jenkins 2.277.4.

- Then we will create new pipeline.



The screenshot shows the Jenkins 'Enter an item name' dialog. The input field contains 'HealthCareBlockchain-pipeline'. Below it, there are several project types listed:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder

- Now we will clone project from our remote repository to our jenkins workspace using below written script.

```

stage('Git Clone WebApplication')
{
    steps{
        git branch: 'master', credentialsId: 'github', url: 'https://github.com/Akshil007/HealthCare_Blockchain.git'
    }
}

stage('Git clone blockchain') {
    steps {
        git branch: 'master', credentialsId: 'github', url: 'https://github.com/Akshil007/Blockchain.git'
    }
}

```

6.2 Build

6.2.1 Maven

Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

pom.xml -

https://github.com/Akshil007/HealthCare_Blockchain/blob/master/pom.xml

We use Pom.xml to specify all necessary dependency in your project. To use those dependency your project must have pom.xml file. Make sure you have maven installed in your machine.

6.2.2 NPM

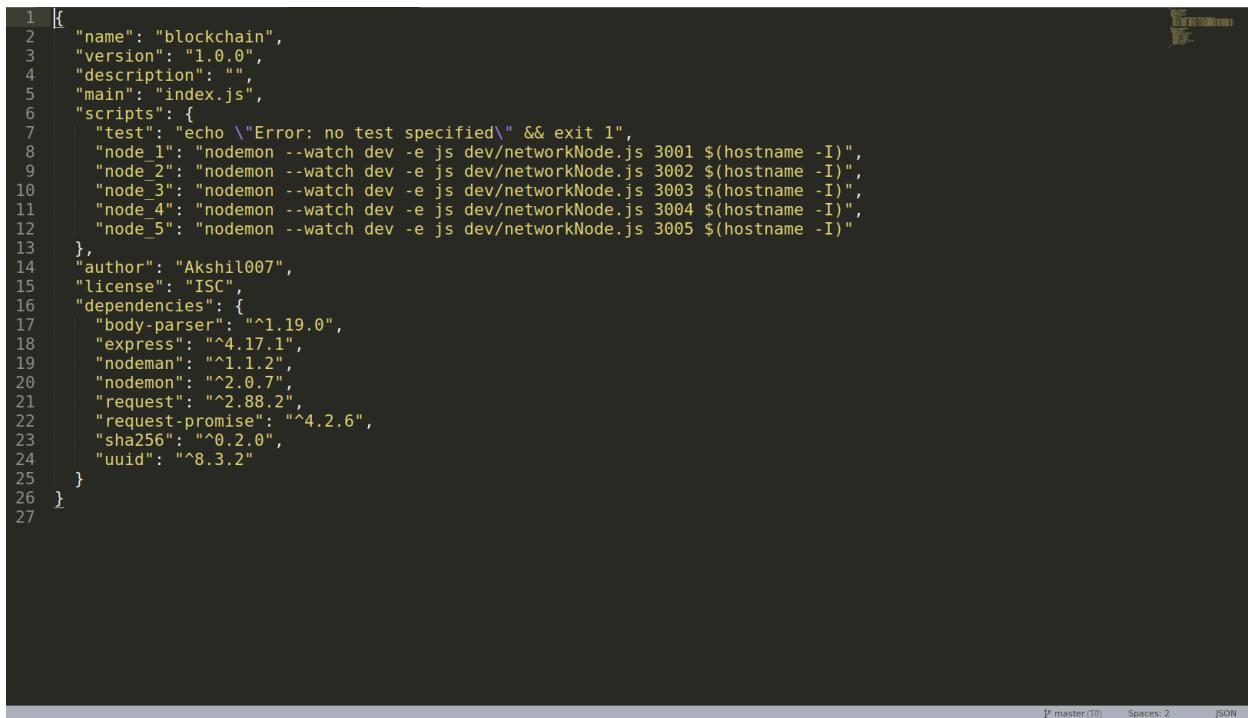
- NPM is the most important tool for working with Node applications. NPM is used to fetch any packages (JavaScript libraries) that an application needs for development, testing, and/or production, and may also be used to run tests and tools used in the development process.
- You can manually use NPM to separately fetch each needed package. Typically we instead manage dependencies using a plain-text definition file named package.json. This file lists all the dependencies for a specific JavaScript "package", including the package's name, version, description, initial file to execute, production dependencies, development dependencies,

versions of Node it can work with, etc. The package.json file should contain everything NPM needs to fetch and run your application.

- The following steps show how you can use NPM to download a package, save it into the project dependencies, and then require it in a Node application.
 - We created directory called "Blockchain"
 - Used the npm init command to create a package.json file for our application.
 - Then we installed necessary package in the blockchain directory and saved it in the dependencies list of your package.json file

i.e npm install express

Package.json file



```
1 {
2   "name": "blockchain",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1",
8     "node_1": "nodemon --watch dev -e js dev/networkNode.js 3001 $(hostname -I)",
9     "node_2": "nodemon --watch dev -e js dev/networkNode.js 3002 $(hostname -I)",
10    "node_3": "nodemon --watch dev -e js dev/networkNode.js 3003 $(hostname -I)",
11    "node_4": "nodemon --watch dev -e js dev/networkNode.js 3004 $(hostname -I)",
12    "node_5": "nodemon --watch dev -e js dev/networkNode.js 3005 $(hostname -I)"
13  },
14  "author": "Akshil007",
15  "license": "ISC",
16  "dependencies": {
17    "body-parser": "^1.19.0",
18    "express": "^4.17.1",
19    "nodeman": "^1.1.2",
20    "nodemon": "^2.0.7",
21    "request": "^2.88.2",
22    "request-promise": "^4.2.6",
23    "sha256": "^0.2.0",
24    "uuid": "^8.3.2"
25  }
26 }
27
```

6.2.3 CI Pipeline

- First we add maven integration plugin in jenkins, under Jenkins → Plugin Manager search for maven integration plugin in available, download and restart.
- Now we add mvn build stage in script file of jenkins which specifies 'mvn clean install' cmd which will build our project and generate JAR file.

```
stage('mvn build')
{
    steps{
        sh 'mvn clean install'
    }
}
```

6.3 Docker Artifact

Docker is a software platform for building applications based on containers which are small and lightweight execution environments that make shared use of the operating system kernel but otherwise run in isolation from one another.

Docker images Blockchain and WebApplication Modules are created and then pushed to docker hub, from where we can pull those images and run it at our end. Docker file is created for both projects that run when we build these docker images. So install docker on your deployment and development machines. Below are reference link and images of docker files:

- <https://github.com/Akshil007/Blockchain/blob/master/dockerfile>
- https://github.com/Akshil007/HealthCare_Blockchain/blob/master/dockerfile

Docker file of blockchain

```
1  FROM node:latest
2  WORKDIR ./blockchain
3  ADD . .
4  CMD npm install
5  CMD npm run node_1
6
```

Docker file of Web Application

```
1  FROM openjdk:8
2  ARG JAR_FILE=target/HealthCareData-0.0.1-SNAPSHOT.jar
3  COPY ${JAR_FILE} app.jar
4  ENTRYPOINT ["java","-jar","/app.jar"]
5
```

To run it as user you have to enter user to the group of docker which can be done using

```
| $sudo usermod -aG docker <Username>
```

To enter Jenkins to docker group you can use this command.

```
| $sudo usermod -aG docker jenkins
```

Now you can verify it using following command.

```
| $ sudo grep docker /etc/gshadow
```

```
akshil@akshil-Inspiron-5379:~$ sudo grep docker /etc/gshadow
[sudo] password for akshil:
docker:!::jenkins,akshil
```

6.3.1 CI Pipeline

- First add necessary plugins from Jenkins Plugin Manager. We need to install docker pipeline and all other necessary plugins as below. Search for docker plugin in available, download and restart.

Updates	Available	Installed	Advanced			
Enabled	Name			Version	Previously installed version	Uninstall
<input checked="" type="checkbox"/>	Authentication Tokens API Plugin	This plugin provides an API for converting credentials into authentication tokens in Jenkins.		1.4		Uninstall
<input checked="" type="checkbox"/>	Credentials Binding Plugin	Allows credentials to be bound to environment variables for use from miscellaneous build steps.		1.24		Uninstall
<input checked="" type="checkbox"/>	Credentials Plugin	This plugin allows you to store credentials in Jenkins.		2.4.1	Downgrade to 2.3.18	Uninstall
<input checked="" type="checkbox"/>	Docker Commons Plugin	Provides the common shared functionality for various Docker-related plugins.		1.17		Uninstall
<input checked="" type="checkbox"/>	Docker Pipeline	Build and use Docker containers from pipelines.		1.26		Uninstall
<input checked="" type="checkbox"/>	Folders Plugin	This plugin allows users to create "folders" to organize jobs. Users can define custom taxonomies (like by project type, organization type etc). Folders are nestable and you can define views within folders. Maintained by CloudBees, Inc.		6.15		Uninstall
<input checked="" type="checkbox"/>	Pipeline: Basic Steps	Commonly used steps for Pipelines.		2.23		Uninstall

- Add Docker Hub Credentials in Jenkins.

Dashboard > Credentials > System > Global credentials (unrestricted) > akshil007/******** (docker hub credentials)

Back to Global credentials (unrestricted)	Scope	<input type="text" value="Global (Jenkins, nodes, items, all child items, etc)"/>	?
Update	Username	<input type="text" value="akshil007"/>	?
Delete	Password	<input type="password" value="Concealed"/> Change Password	?
Move	ID	<input type="text" value="DockerHub"/>	?
	Description	<input type="text" value="docker hub credentials"/>	?
		Save	

- Now add script for building docker images and pushing them onto docker hub.

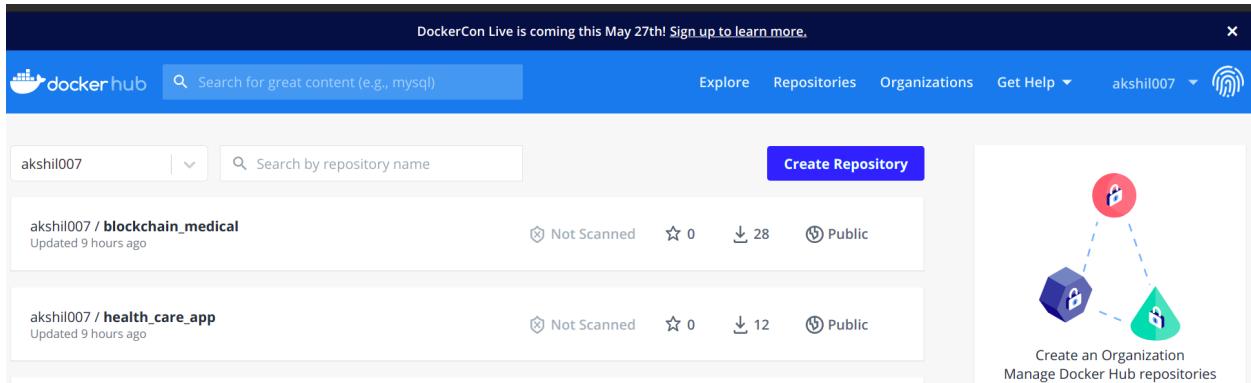
Script for building Web Application Image and deploying on docker hub in Jenkins pipeline

```
stage('Docker build WebApplication') {
    steps {
        script{
            dockerImage=docker.build "akshil007/health_care_app:latest"
        }
    }
}
stage('Docker push image webApp') {
    steps {
        script{
            docker.withRegistry(' ', 'DockerHub'){
                dockerImage.push()
            }
        }
    }
}
```

Script for building Blockchain image and deploying on docker hub in Jenkins pipeline

```
stage('Docker build blockchain') {
    steps {
        script{
            dockerImage=docker.build "akshil007/blockchain_medical:latest"
        }
    }
}
stage('Docker push image blockchain') {
    steps {
        script{
            docker.withRegistry(' ', 'DockerHub'){
                dockerImage.push()
            }
        }
    }
}
```

- After Successful Deployment we can see images on docker hub.



6.4 Deployment

Ansible is an open-source software provisioning, configuration management, and application-deployment tool enabling infrastructure as code. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows. Ansible is Agentless Automation tool. Ansible Connets to the hosts it manages using openssh.

Let's start with the setting up openssh for Ansible.

- First create ssh-key in managed host machines using following command.

```
| $ssh-key-gen -t rsa
```

```

akshil@akshil-Inspiron-5379:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/akshil/.ssh/id_rsa):
/home/akshil/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/akshil/.ssh/id_rsa
Your public key has been saved in /home/akshil/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:7Or+mS13c1/4vA5z2xjxbyZqGD/e+1WPkmCjV8gl+U4 akshil@akshil-Inspiron-5379
The key's randomart image is:
----[RSA 3072]----+
|                               |
|                               |
|                               . |
|                               o .|
|                               . =|
|                               S * E ..|
|                               . o.* . =o|
|                               o .++o+.=|
|                               ..=..+ =o=*B|
|                               .o.=+o. =o+*XB|
-----[SHA256]-----+

```

- Now copy public key of host nodes into control nodes by executing following command in control nodes.

\$ssh-copy-id <username>@<IP Address>

```

jenkins@akshil-Inspiron-5379:/home/akshil$ ssh-copy-id akshil@localhost
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/var/lib/jenkins/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
akshil@localhost's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'akshil@localhost'"
and check to make sure that only the key(s) you wanted were added.

```

- Now Ansible can access host machines.
- Now lets create Playbook file and add it into project repository so Jenkins can access it.

Link for playbook.yml file

<https://github.com/Akshil007/Blockchain/blob/master/playbook.yml>

Create Tasks in Playbook file

```

1  ---
2  - name: Pull and Run docker containers
3  hosts: all
4  tasks:
5      - name: Remove mysql existing container
6          docker_container:
7              name: healthcare-db
8              state: absent
9
10     - name: Remove Spring WebApp existing container
11         docker_container:
12             name: healthcare-webapp
13             state: absent
14
15     - name: Remove blockchain existing container
16         docker_container:
17             name: healthcare-blockchain
18             state: absent
19             |
20
21     - name: Docker pull mysql
22         docker_image:
23             name: mysql:8.0.23
24             source: pull
25
26
27     - name: Remove Previously Created Docker Network
28         docker_network:
29             name: health_net
30             state: absent
31             force: yes
32
33
34     - name: Create Docker New Network
35         docker_network:
36             name: health_net

```

Specify Hosts in Inventory file

```

1 localhost ansible_user=akshil
2 [alay]
3 192.168.1.36 ansible_user=alay
4

```

- Now let us create Jenkins script for Ansible. Go to Pipeline syntax ⇒ Select ansible playbook ⇒ Specify path of playbook and inventory file ⇒ Finally press generate script

```

stage('Deployment') {
    steps {
        ansiblePlaybook becomeUser: null, colorized: true, disableHostKeyChecking: true,
        installation: 'Ansible', inventory: 'inventory', playbook: 'playbook.yml', sudoUser:
        null
    }
}

```

6.5 Monitor

The ELK Stack is a collection of three open-source products — Elasticsearch, Logstash, and Kibana.

- E stands for ElasticSearch: used for storing logs
- L stands for LogStash : used for both shipping as well as processing and storing logs
- K stands for Kibana: is a visualization tool (a web interface) which is hosted through Nginx or Apache

ELK Stack is designed to allow users to take data from any source, in any format, and to search, analyze, and visualize that data in real time. Reference figure to sample our generated log structure

```
1 2021-05-17 22:27:59.664 INFO 24264 --- [main] b.m.H.HealthCareDataApplication      : Starting
HealthCareDataApplication using Java 14.0.2 on akshil-Inspiron-5379 with PID 24264 (/home/akshil/
Blockchain/WebApplication/HealthCareDataWebApp/target/classes started by akshil in /home/akshil/
Blockchain/WebApplication/HealthCareDataWebApp)
2 2021-05-17 22:27:59.667 INFO 24264 --- [main] b.m.H.HealthCareDataApplication      : No active profile
set, falling back to default profiles: default
3 2021-05-17 22:28:00.494 INFO 24264 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping
Spring Data JPA repositories in DEFAULT mode.
4 2021-05-17 22:28:00.506 INFO 24264 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring
Data repository scanning in 5 ms. Found 0 JPA repository interfaces.
5 2021-05-17 22:28:01.021 INFO 24264 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized
with port(s): 8083 (http)
6 2021-05-17 22:28:01.027 INFO 24264 --- [main] o.apache.catalina.core.StandardService   : Starting service [
Tomcat]
7 2021-05-17 22:28:01.027 INFO 24264 --- [main] org.apache.catalina.core.StandardEngine  : Starting Servlet
engine: [Apache Tomcat/9.0.45]
8 2021-05-17 22:28:01.109 INFO 24264 --- [main] o.a.c.c.C.[Tomcat].[localhost].[]       : Initializing Spring
embedded WebApplicationContext
9 2021-05-17 22:28:01.109 INFO 24264 --- [main] w.s.c.ServletWebServerApplicationContext : Root
WebApplicationContext: initialization completed in 1395 ms
10 2021-05-17 22:28:01.265 INFO 24264 --- [main] com.zaxxer.hikari.HikariDataSource     : HikariPool-1 -
Starting...
11 2021-05-17 22:28:01.448 INFO 24264 --- [main] com.zaxxer.hikari.HikariDataSource     : HikariPool-1 -
Start completed.
12 2021-05-17 22:28:01.484 INFO 24264 --- [main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204:
Processing PersistenceUnitInfo [name: default]
13 2021-05-17 22:28:01.542 INFO 24264 --- [main] org.hibernate.Version            : HHH000412:
Hibernate ORM core version 5.4.30.Final
14 2021-05-17 22:28:01.666 INFO 24264 --- [main] o.hibernate.annotations.common.Version : HCANN000001:
Hibernate Commons Annotations {5.1.2.Final}
15 2021-05-17 22:28:01.738 INFO 24264 --- [main] org.hibernate.dialect.Dialect       : HHH000400: Using
dialect: org.hibernate.dialect.MySQL8Dialect
16 2021-05-17 22:28:01.941 INFO 24264 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator  : HHH000490: Using
JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
17 2021-05-17 22:28:01.955 INFO 24264 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA
EntityManagerFactory for persistence unit 'default'
```

Logstash conf file feeded to logstash

```

1 { "description": "Ingest pipeline created by text structure finder",
2  "processors": [
3    {
4      "grok": {
5        "field": "message",
6        "patterns": [
7          "%{TIMESTAMP_ISO8601:timestamp} .*?%{LOGLEVEL:loglevel} %{INT:field} .*"
8        ]
9      }
10     },
11   ],
12   {
13     "date": {
14       "field": "timestamp",
15       "timezone": "{{ event.timezone }}",
16       "formats": [
17         "yyyy-MM-dd HH:mm:ss.SSS"
18       ]
19     }
20   },
21   {
22     "convert": {
23       "field": "field",
24       "type": "long",
25       "ignore_missing": true
26     }
27   },
28   {
29     "remove": {
30       "field": "timestamp"
31     }
32   }
33 ]
34 ]

```

Now we can start the elasticsearch, logstash and kibana

Kibana starts at localhost:5601, after creating the index pattern, here it is the system, We discover data, and under visualization here the reference images of what we monitored.

elastic

Machine Learning / Data Visualizer / File

Grok pattern: %(TIMESTAMP_ISO8601:timestamp).*?%(LOGLEVEL:loglevel) %(INT:field).*

Time field: timestamp

Time format: yyyy-MM-dd HH:mm:ss.SSS

Override settings Analysis explanation

File stats

All fields 4 of 4 total Number fields 1 of 1 total

Type	Name	Documents (%)	Distinct values	Distributions
field	field	112 (100%)	2	min: 24264 median: 27602 max: 27602
loglevel	loglevel	112 (100%)	3	3 categories
message	message	112 (100%)	112	
timestamp	timestamp	112 (100%)	104	

Rows per page: 10 < 1 >

Discover

Search message: exists + Add filter

mining

113 hits May 17, 2021 @ 22:27:59.664 - May 17, 2021 @ 23:14:40.533 Auto

Count @timestamp per minute

Time Document

- May 17, 2021 @ 23:14:40.533 @timestamp: May 17, 2021 @ 23:14:40.533 field: 27,602 loglevel: INFO message: 2021-05-17 23:14:40.533 INFO 27602 --- [SpringContextShutdownHook] com.zaxxer.hikari.HikariDataSource : HikarPool-1 - Shutdown completed. _id: Stx3e3kBEUehLGMqNFN _index: mining _score: - _type: _doc
- May 17, 2021 @ 23:14:40.530 @timestamp: May 17, 2021 @ 23:14:40.530 field: 27,602 loglevel: INFO message: 2021-05-17 23:14:40.530 INFO 27602 --- [SpringContextShutdownHook] com.zaxxer.hikari.HikariDataSource : HikarPool-1 - Shutdown initiated... _id: Sdx3e3kBEUehLGMqNFN _index: mining _score: - _type: _doc
- May 17, 2021 @ 23:14:40.528 @timestamp: May 17, 2021 @ 23:14:40.528 field: 27,602 loglevel: INFO message: 2021-05-17 23:14:40.528 INFO 27602 --- [SpringContextShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default' _id: Rzx3e3kBEUehLGMqNFN _index: mining _score: - _type: _doc
- May 17, 2021 @ 23:14:21.073 @timestamp: May 17, 2021 @ 23:14:21.073 field: 27,602 loglevel: INFO message: 2021-05-17 23:14:21.073 INFO 27602 --- [http-nio-8083-exec-8] b.m.HealthCareData.model.MiningWork : IPS92.168.0.104:3001 _id: Rjx3e3kBEUehLGMqNFN _index: mining _score: - _type: _doc
- May 17, 2021 @ 23:14:21.070 @timestamp: May 17, 2021 @ 23:14:21.070 field: 27,602 loglevel: INFO message: 2021-05-17 23:14:21.070 INFO 27602 --- [http-nio-8083-exec-8]

mining

Search field names

Field filters 0

Available fields 3

- @timestamp
- field
- loglevel

Empty fields 0

Meta fields 3

Pie chart showing the distribution of log levels:

Category	Percentage
23:10	21.18%
23:09	34.12%
23:11	21.18%
23:14	12.94%
23:13	7.08%

Suggestions

Current visualization

85

Size by configuration

Select a function

- Average
- Count
- Counter rate
- Cumulative sum
- Differences
- Last value
- Maximum
- Median
- Minimum
- Moving average
- Percentile
- Sum
- Unique count

Select a field

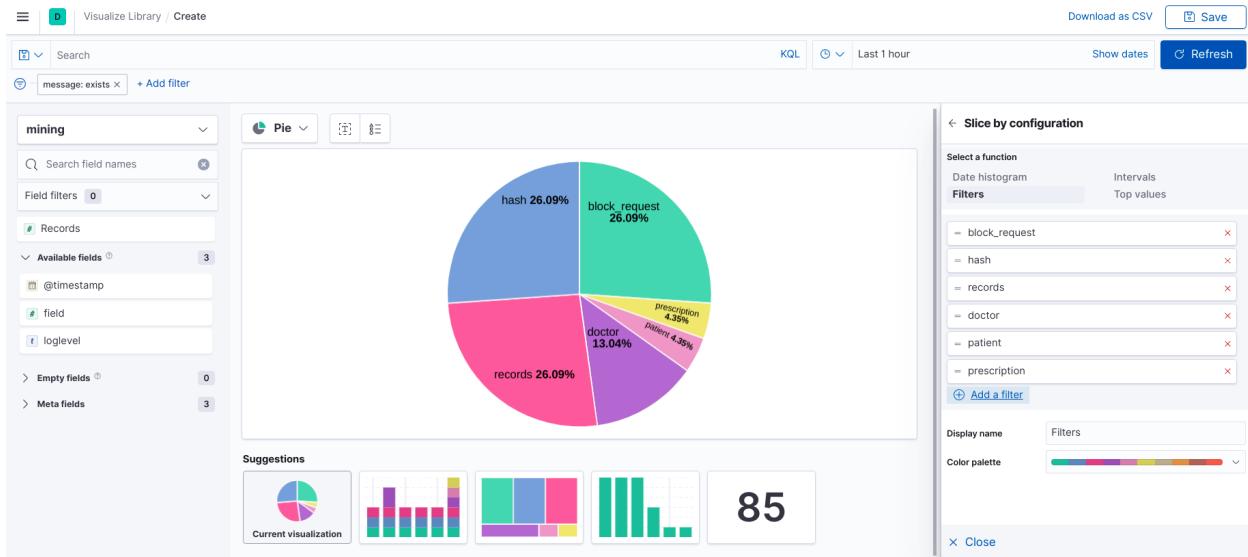
Records

Add advanced options

Display name Count of records

Value format Default

x Close



7 Building Pipeline and Running on Host Machines

From the above sections we have seen how we have integrated maven, ansible, docker and

other necessary tools with jenkins. We were able to build docker images with Jenkins. Here's the reference figure to the entire pipeline and its final full stage view after a successful run of Jenkins pipeline.

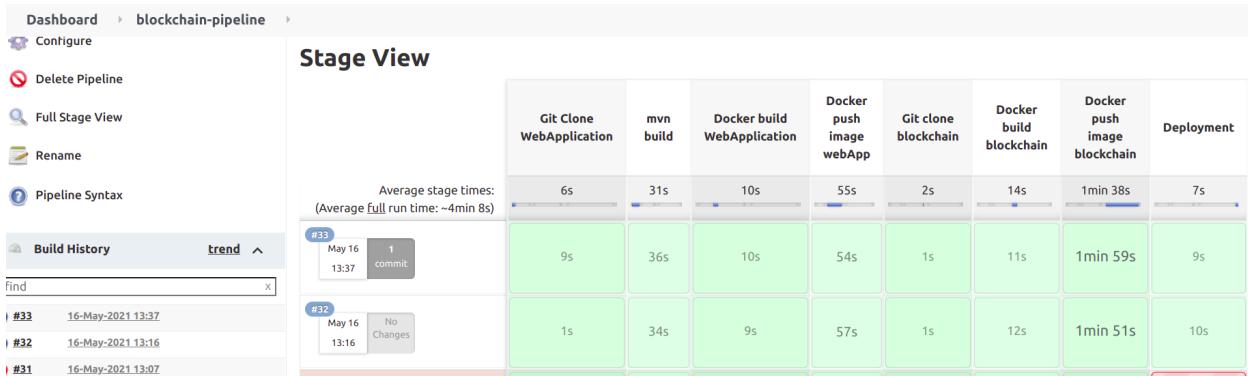
Entire Jenkins Script

```

1 pipeline {
2     environment{
3         dockerImage = ""
4     }
5     agent any
6
7     stages {
8         stage('Git Clone WebApplication')
9         {
10            steps{
11                git branch: 'master', credentialsId:
12                    'github', url: 'https://github.com/
13                    Akshil007/HealthCare_Blockchain.git'
14            }
15            stage('mvn build')
16            {
17                steps{
18                    sh 'mvn clean install'
19                }
20            }
21            stage('Docker build WebApplication') {
22                steps {
23                    script{
24                        dockerImage=docker.build
25                            "akshil007/health_care_app:latest"
26                    }
27                }
28                stage('Docker push image webApp') {
29                    steps {
30                        script{
31                            docker.withRegistry('', 'DockerHub'){
32                                dockerImage.push()
33                            }
34                    }
35                }
36            }
37            stage('Git clone blockchain') {
38                steps {
39                    git branch: 'master', credentialsId:
40                    'github', url: 'https://github.com/
41                    Akshil007/Blockchain.git'
42            }
43            stage('Docker build blockchain') {
44                steps {
45                    script{
46                        dockerImage=docker.build
47                            "akshil007/blockchain_medical:latest"
48                    }
49            }
50            stage('Docker push image blockchain') {
51                steps {
52                    script{
53                        docker.withRegistry('', 'DockerHub'){
54                            dockerImage.push()
55                        }
56                    }
57            }
58        }
59    }
60    stage('Deployment') {
61        steps {
62            ansiblePlaybook becomeUser: null,
63            colorized: true,
64            disableHostKeyChecking: true,
65            installation: 'Ansible', inventory:
66                'inventory', playbook: 'playbook.yml',
67            sudoUser: null
68        }
69    }
70 }

```

Successful Run of Jenkins Pipeline



Now Lets go step by step in Deployment Process

- First We are removing already existing containers from machines.

```
tasks:
  - name: Remove mysql existing container
    docker_container:
      name: healthcare-db
      state: absent

  - name: Remove Spring WebApp existing container
    docker_container:
      name: healthcare-webapp
      state: absent

  - name: Remove blockchain existing container
    docker_container:
      name: healthcare-blockchain
      state: absent
```

- Then We are creating docker network and deploying docker images in that docker network so that container can communicate with each other.

```
tasks:
  - name: Remove Previously Created Docker Network
    docker_network:
      name: health_net
      state: absent
      force: yes

  - name: Create Docker New Network
    docker_network:
      name: health_net
```

- Then We are pulling images from docker hub to host machines.

```
- name: Docker pull mysql
  docker_image:
    name: mysql:8.0.23
    source: pull

- name: Pull Spring WebApp image
  docker_image:
    name: akshil007/health_care_app
    source: pull

- name: Pull blockchain image
  docker_image:
    name: akshil007/blockchain_medical
    source: pull
```

- Then We are creating docker container for MYSQL.

```
- name: Create MYSQL container
  docker_container:
    name: healthcare-db
    image: mysql:8.0.23
    networks:
      - name: "health_net"
    env:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: HealthCareData
    ports:
      - "8090:3306"
```

This command will run the docker container with following specifications-

1. Run the docker container on port number 8090 rather than the local Mysql running on port number 3306.
2. It further synchronises by mysql_root password, mysql_database.
3. It specifies the network through which all the docker containers will be connected i.e Here it is "health_net"

4. Also, it specifies the docker container name and the docker image it is using i.e
mysql:8.0.23

Running MYSQL Container

```
akshil@akshil-Inspiron-5379:~$ docker container run -it -p 8090:3306 -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=HealthCareData -  
-network health_net --name healthcare-db mysql:8.0.23  
2021-05-16 18:07:54+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.23-1debian10 started.  
2021-05-16 18:07:54+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'  
2021-05-16 18:07:54+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.23-1debian10 started.  
2021-05-16 18:07:54+00:00 [Note] [Entrypoint]: Initializing database files  
2021-05-16T18:07:54.700180Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.23) initializing of server in progress as  
process 42  
2021-05-16T18:07:54.712550Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.  
2021-05-16T18:07:55.637173Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.  
2021-05-16T18:07:58.047595Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider swi-  
tching off the --initialize-insecure option.  
2021-05-16 18:08:01+00:00 [Note] [Entrypoint]: Database files initialized  
2021-05-16 18:08:01+00:00 [Note] [Entrypoint]: Starting temporary server  
mysqld will log errors to /var/lib/mysql/0baecfb58b00.err  
mysqld is running as pid 89  
2021-05-16 18:08:03+00:00 [Note] [Entrypoint]: Temporary server started.  
Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.  
Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.  
Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.  
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.  
2021-05-16 18:08:04+00:00 [Note] [Entrypoint]: Creating database HealthCareData
```

- Now We will run our web Application in Container.

```
- name: Create Web Application container
  docker_container:
    name: healthcare-webapp
    image: akshil007/health_care_app
    networks:
      - name: "health_net"
    env:
      DATABASE_HOST: healthcare_host
    ports:
      - "8085:8083"
```

This command will run the docker container with following specifications-

- Run the docker container on port number 8085 rather than running on port number 8083.
- It specifies the network through which all the docker containers will be connected i.e
Here it is "health_net"

3. Also, it specifies the docker container name and the docker image.

Running Spring Boot Application

- Similarly for blockchain.

```
- name: Run blockchain container
  docker_container:
    name: healthcare-blockchain
    image: akshil007/blockchain_medical
    networks:
      - name: "health net"
```

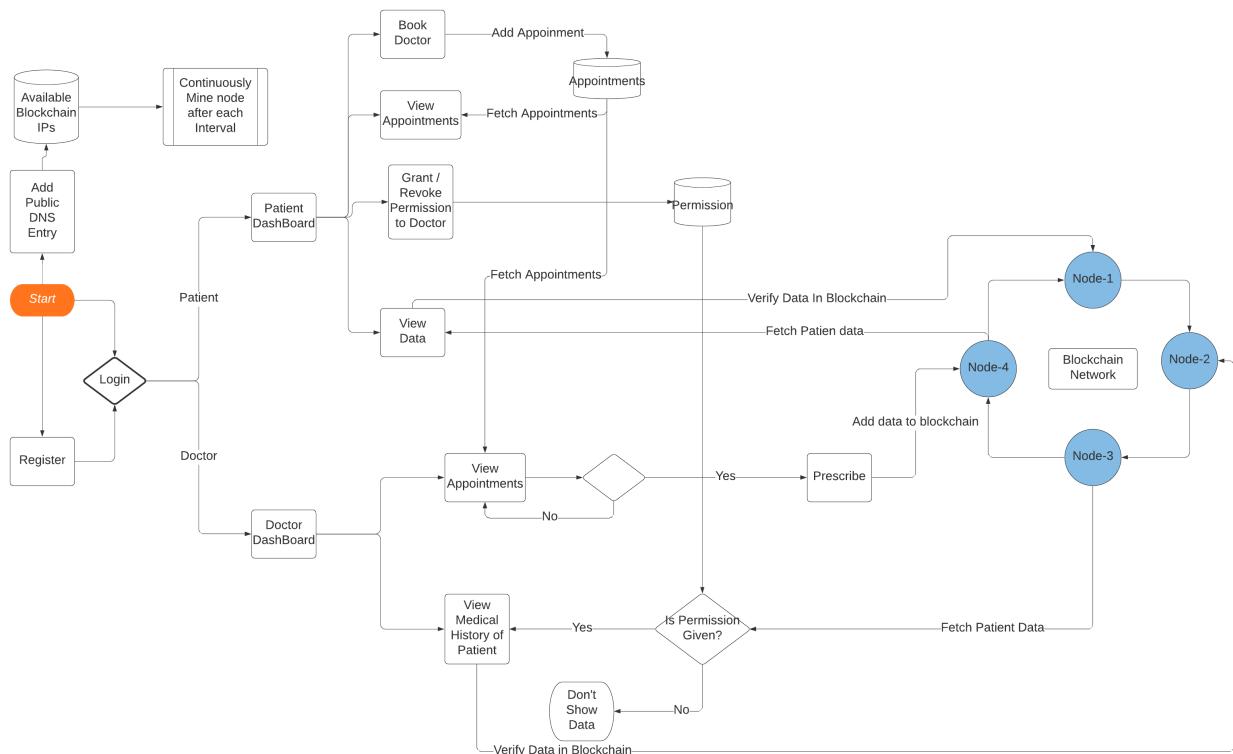
Running Blockchain

```
akshil@akshil-Inspiron-5379:~$ docker run --network health_net --name healthcare-blockchain akshil007/blockchain_medical:latest
> blockchain@1.0.0 node_1
> nodemon --watch dev -e js dev/networkNode.js 3001 $(hostname -I)

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): dev/**/*
[nodemon] watching extensions: js
[nodemon] starting `node dev/networkNode.js 3001 172.20.0.4`
http://172.20.0.4:3001
```

8 Experimental Setup

8.1 DESIGN DIAGRAM



- Here the reference design diagram where all the different modules interact with each other.
- After successful login of patient, patient has four options either he can book new appointment , he can view previous appointment, allow or revoke permission to doctor or he can view his past data. If Patient choose to book appointment then he will redirected to page where he can specify his condition/problem and he can choose doctor from available doctors. After doctor has prescribed and this data got mined then patient can view and verify his past data from blockchain.

- After successful login as a doctor, doctor can either prescribe to available appointments or he can cancel the appointments. After he submit the prescription, Data goes into pending records of blockchain until our predefined background process will mine it. If patient has given the permission, then doctor can see full medical history of patient which will help him to diagnose patient more accurately.

8.2 Blockchain Properties

8.2.1 Distributed Network

We have maintained distributed network of nodes here. All nodes have the same exact data. Once node gets the new data then it distributes that data to all other nodes. We are maintaining list of all nodes in the network and every node contains this list. Whenever new node enters into network, the receiver node distribute the address of new node to all other nodes and new node gets the address of all other nodes.

8.2.2 Consensus Algorithm

There is no central authority present to validate and verify the transactions, yet every transaction in the Blockchain is considered to be completely secured and verified. This is possible only because of the presence of the consensus protocol which is a core part of any Blockchain network.

Using Consensus algorithm we can also find out integrity of blockchain's data. For that we have used longest chain rule. We will consider the longest blockchain as a valid blockchain.

A consensus algorithm is a procedure through which all the peers of the Blockchain network reach a common agreement about the present state of the distributed ledger. In this way, consensus algorithms achieve reliability in the Blockchain network and establish trust between unknown peers in a distributed computing environment. Essentially, the consensus protocol makes sure that every new block that is added to the Blockchain is the one and only version of the truth that is agreed upon by all the nodes in the Blockchain. There are many consensus algorithm is available but we used Proof of Work.

8.2.3 Proof of Work

This consensus algorithm is used to select a miner for the next block generation. Bitcoin uses this PoW consensus algorithm. The central idea behind this algorithm is to solve a complex mathematical puzzle and easily give out a solution. This mathematical puzzle requires a lot of computational power and thus, the node who solves the puzzle as soon as possible gets to mine the next block.

8.2.4 Mining

Mining is basically verifying transactions. All new transactions are waiting in pending records until they get mined by some miner. Once Records get mined, they are added in blockchain. we use proof of work method for mining. In our case all we are not giving any rewards or actually there are no miner hence we have created one background process which will mine nodes after each specific interval.

9. Result

9.1 LOGIN PAGE

This is our login page. Here you can either login as a Patient or Doctor.

ABC Hospital

Login Page

Login from here as patient to view your medical data, book appointments.
Login as doctor to view patient data, appointments, prescribe patients.

User Name

Password

User Type

9.2 REGISTER PAGE

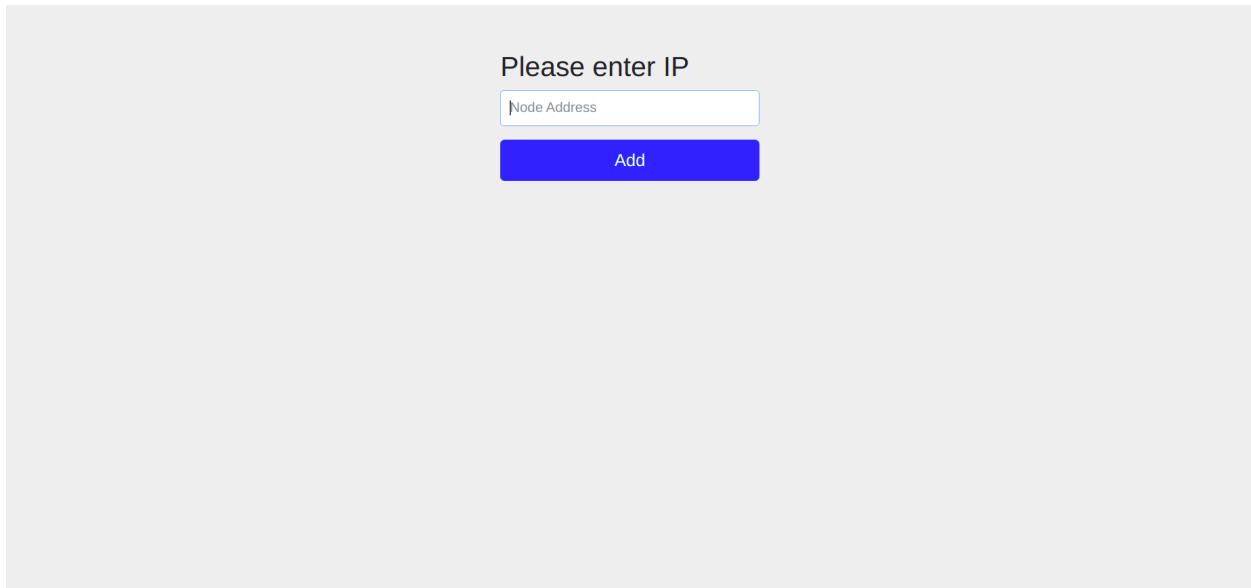
Using Register Page you can sign up in our application as a Patient or as a Doctor.

The screenshot shows a registration form titled "Register Page" on the left. The right side contains the following fields:

- User Name: An input field labeled "User Name".
- First Name: An input field labeled "First Name".
- Last Name: An input field labeled "Last Name".
- Password: An input field labeled "Password".
- Email: An input field labeled "Email".
- User Type: A dropdown menu set to "Patient".
- Buttons: Two buttons, "Register" and "Login".

9.3 PUBLIC DNS ENTRY

Here you can register this application with one of the running blockchain node.



9.4 PATIENT DASHBOARD

After successful login as a patient you can see following options in Dashboard page.

ABC Hospital

pt pt

≡

Profile

patient

Name pt pt

User name pt

Email pt@pt.com

- View Profile
- View Data
- Book Doctor
- Give / Revoke Doctor Permission
- View Appointments

9.5 BOOK DOCTOR

Here you can book an appointment with any listed doctor below.

The screenshot shows a user interface for booking a doctor's appointment. On the left is a dark sidebar with the title "ABC Hospital" and two user icons ("pt pt"). Below this are several menu items with icons: View Profile, View Data, Book Doctor (selected), Give / Revoke Doctor Permission, and View Appointments. The main content area has a header "Book Appointment". It contains a "Form" section with a "Doctor ID" input field containing the value "2". Below it is a "Description" input field containing the value "covid". At the bottom is a blue "Submit" button. Underneath the form is a section titled "Available Doctors" with a table:

Doctor ID	First name	Last name	User name	Email
2	doc	doc	doc	doc@doc.com

9.6 GIVE / REVOKE PERMISSION

Here you can give permission to doctor to see your past medical history. You can also revoke back this permission at any time.

The screenshot shows a user interface for giving or revoking doctor permissions. The sidebar is identical to the previous screenshot, showing "ABC Hospital" and the "Book Doctor" menu item. The main content area has a header "Give Permission". It contains a "Doctor ID" input field with the value "2". At the bottom is a blue "Submit" button. Below the form is a section titled "Permissions given to :" with a table:

Doctor ID	First name	Last name	User name	Email	Action
2	doc	doc	doc	doc@doc.com	Revoke

9.7 VIEW APPOINTMENTS

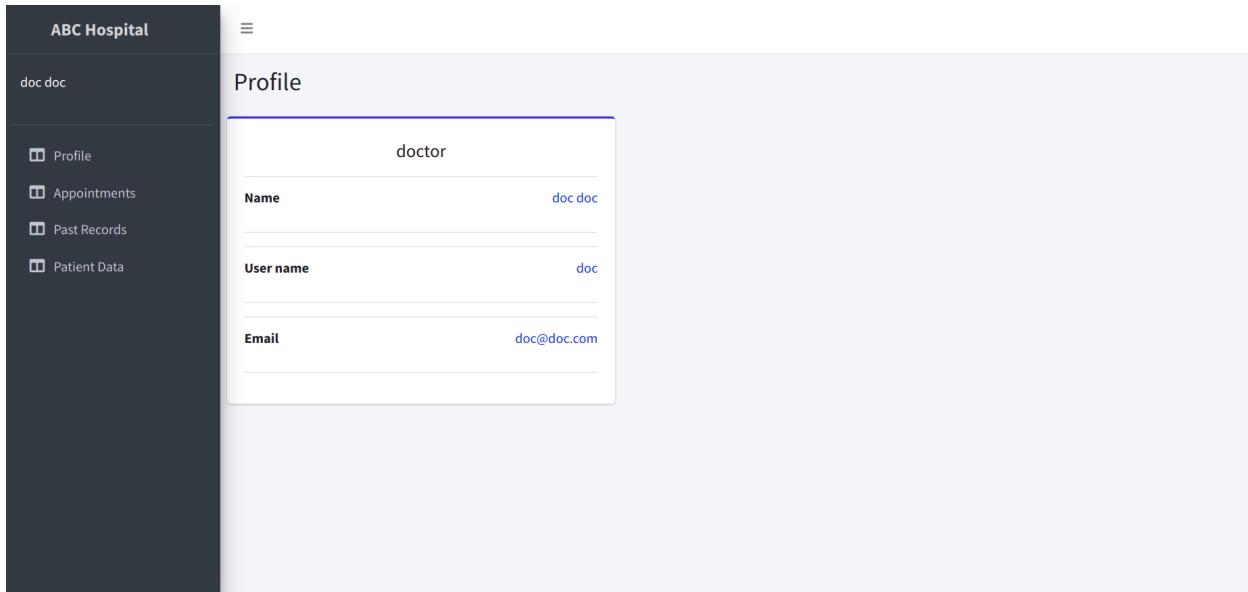
Here you can see your active appointments with doctor.

The screenshot shows a user interface for managing appointments. On the left is a sidebar with the title 'ABC Hospital' and a user icon 'pt pt'. Below this are several menu items with icons: 'View Profile', 'View Data', 'Book Doctor', 'Give / Revoke Doctor Permission', and 'View Appointments'. The main content area is titled 'Appointments' and contains a table with one row of data. The table has columns for Appointment ID, Patient ID, Doctor ID, Date, Status, Description, and Action. The data in the table is as follows:

Appointment ID	Patient ID	Doctor ID	Date	Status	Description	Action
98b09671-e529-4dc9-a90e-12773719b1c2	1	2	2021-05-18 09:31:42	Active	covid	<button>Cancel</button>

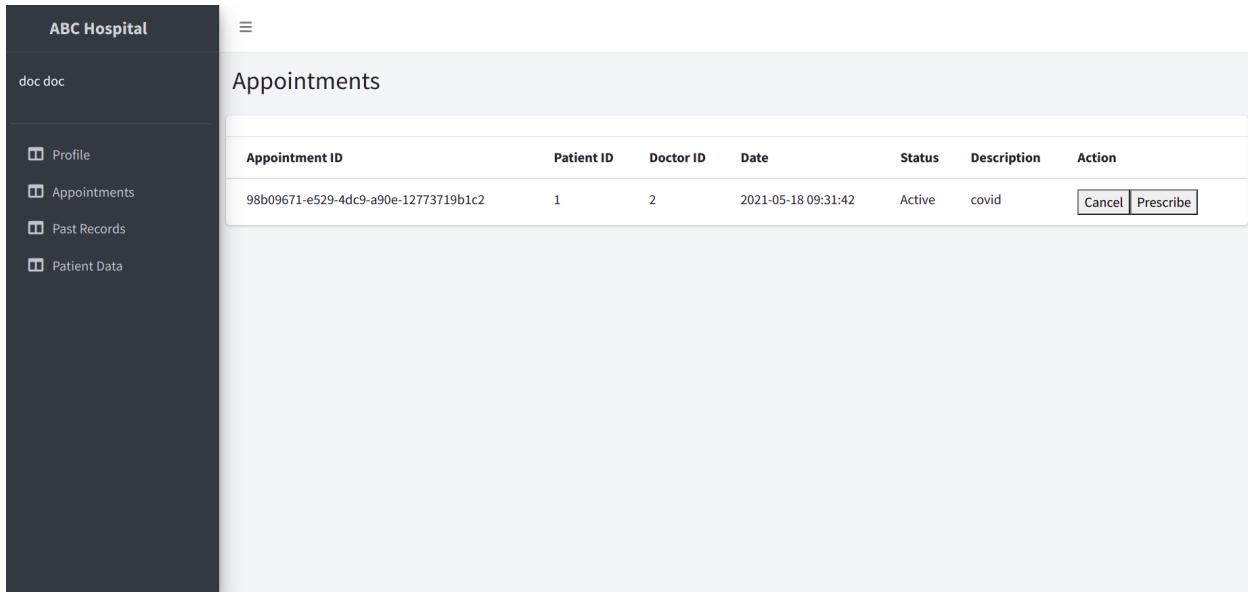
9.8 DOCTOR DASHBOARD

When you login as a doctor, you can see dashboard with following options.



9.9 APPOINTMENTS

Here Doctor has two options either prescribe or cancel the Appointments.



9.10 PRESCRIBE

Here Doctor can prescribe to patient and when doctor press submit, data goes into pending records of blockchain through Node that we registered through public DNS Entry page.

The screenshot shows a user interface for a medical application. On the left, there's a sidebar with the text "ABC Hospital" and "doc doc". Below that is a list of navigation items: "Profile", "Appointments", "Past Records", and "Patient Data". The main content area has a title "Prescribe" and a unique record identifier "98b09671-e529-4dc9-a90e-12773719b1c2". Under this, there are two input fields: "Description" containing "covid" and "Prescription" containing "Zincovit, Lamcee". At the bottom of this section is a blue "Submit" button. Below the submission area is a horizontal row of buttons labeled "Doctor ID", "Patient ID", "Description", "Prescription", "Record ID", and "Verify in blockchain".

9.11 PAST RECORDS

Once the data is mined by one of the nodes from blockchain and if Patient has given the permission to doctor to see past records then doctor can see all the past records of that patient here. Also Doctor can verify this data in blockchain.

Doctor ID	Patient ID	Description	Prescription	Record ID	Verify in blockchain
2	1	covid	Zincovit, Lamcee	f21aca80a49c411d859eba310cc3254f	Verify

9.12 VERIFY IN BLOCKCHAIN

```
{ "Record": { "doctor": "2", "patient": "1", "description": "covid", "prescription": "Zincovit, Lamcee ", "RecordId": "f21aca80a49c411d859eba310cc3254f" }, "blockHash": "000095b7cb3f94a505c1a72c817285c6614fc..."}  
Raw Parsed
```

9.13 RUNNING BLOCKCHAIN

```

{
  "chain": [
    {
      "index": 1,
      "timeStamp": "2021-05-18T09:26:04.293Z",
      "records": [],
      "nonce": 100,
      "hash": "0",
      "previousBlockHash": "0"
    },
    {
      "index": 2,
      "timeStamp": "2021-05-18T09:31:04.830Z",
      "records": [],
      "nonce": 22352,
      "hash": "00003c8aa79d8ef20ebbd3cf6adfd67a4c247288655809242bc799e0813ee1",
      "previousBlockHash": "0"
    },
    {
      "index": 3,
      "timeStamp": "2021-05-18T09:32:05.408Z",
      "records": [],
      "nonce": 69668,
      "hash": "0000598f31f82944e29998a04836f618655b6bb87efddb48c40f9e05322de91e",
      "previousBlockHash": "00003c8aa79d8ef20ebbd3cf6adfd67a4c247288655809242bc799e0813ee1"
    },
    {
      "index": 4,
      "timeStamp": "2021-05-18T09:33:05.477Z",
      "records": [],
      "nonce": 3667,
      "hash": "0000d8303d827e138b60d422ad97107b7a8cb873aa997fd61f977522b3ae4aea",
      "previousBlockHash": "0000598f31f82944e29998a04836f618655b6bb87efddb48c40f9e05322de91e"
    }
  ]
}

```

10 Conclusion

From the project we have demonstrated the use of blockchain to store data so that its integrity and authenticity is maintained along with non repudiation of the sources providing data.

We have also shown how a blockchain can be build providing 3rd party APIs so that other web applications can access the data

Since core medical data is stored on all the distributed nodes of the network, availability of the data is also high since all the nodes have APIs so the external web applications can request or store data on any of the nodes.