Health Care Data (SPE Final project)

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
  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:
  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 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:
  gyp libc-ares2 libjs-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-builtin-modules node-builtins node-cacache
  node-call-limit node-camelcase node-caseless node-chalk node-chownr node-ci-info node-cli-boxes node-cliui 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-jsbn node-editor node-encoding node-end-of-stream node-err-code node-errno
  node-es6-promise node-escape-string-regexp node-execa 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-jsbn node-json-parse-better-errors node-json-schema node-json-schema-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 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.

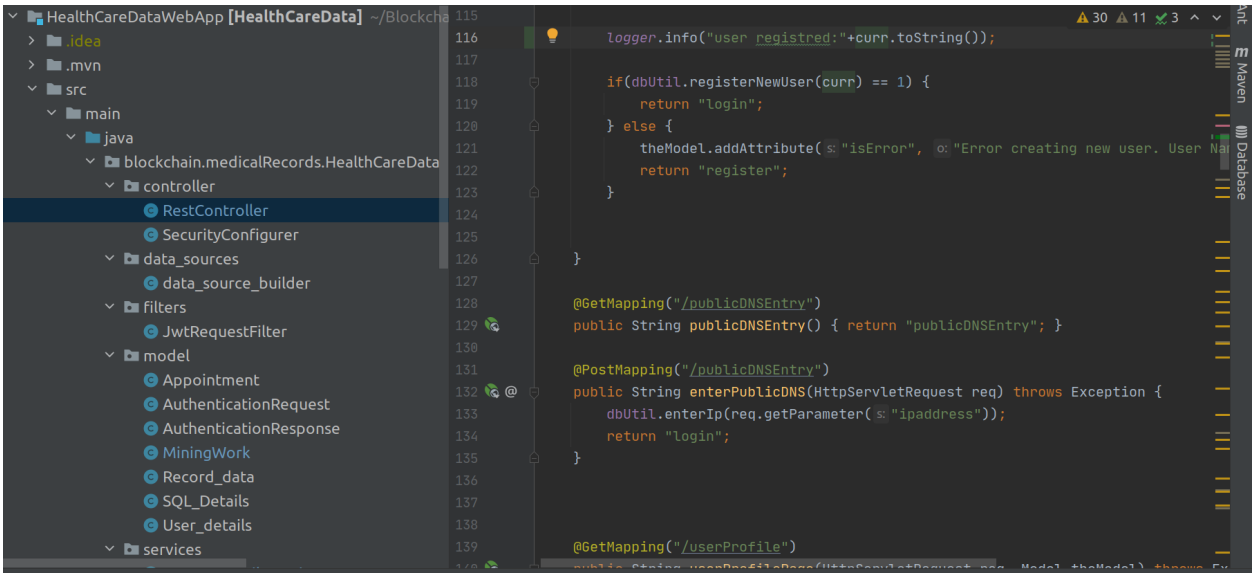
> $ sudo apt install maven

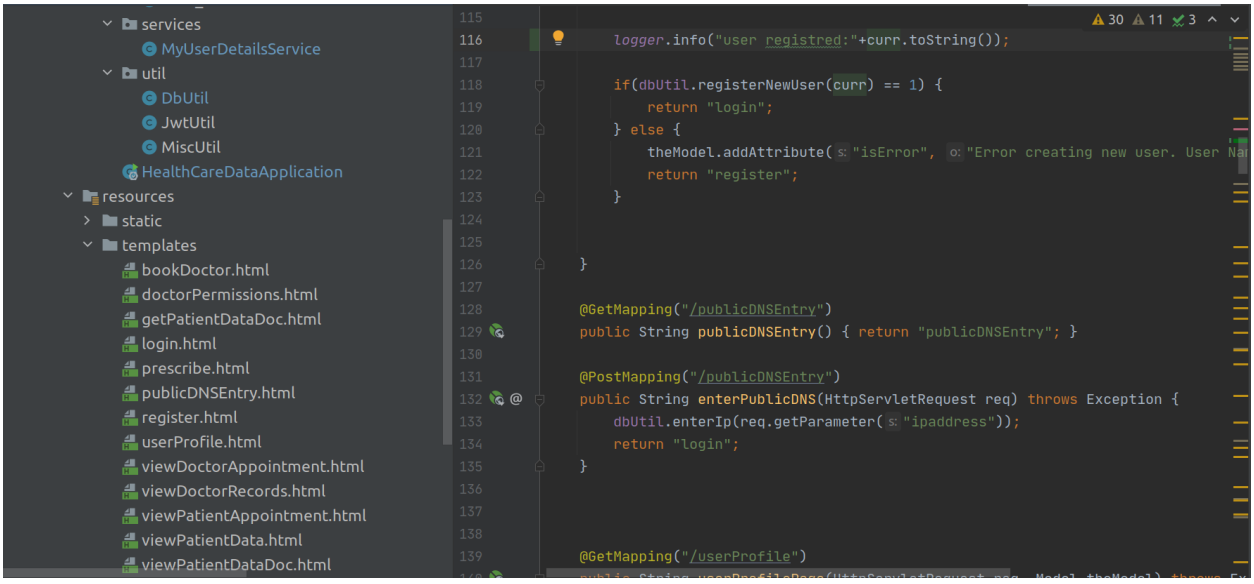
```
akshil@akshil-Inspiron-5379:~$ sudo apt install maven
Reading package lists... Done
Building dependency tree
Reading state information... Done
maven is already the newest version (3.6.3-1).
The following packages were automatically installed and are no longer required:
  blt gir1.2-gst-plugins-base-1.0 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.





6.1 Source Control Management

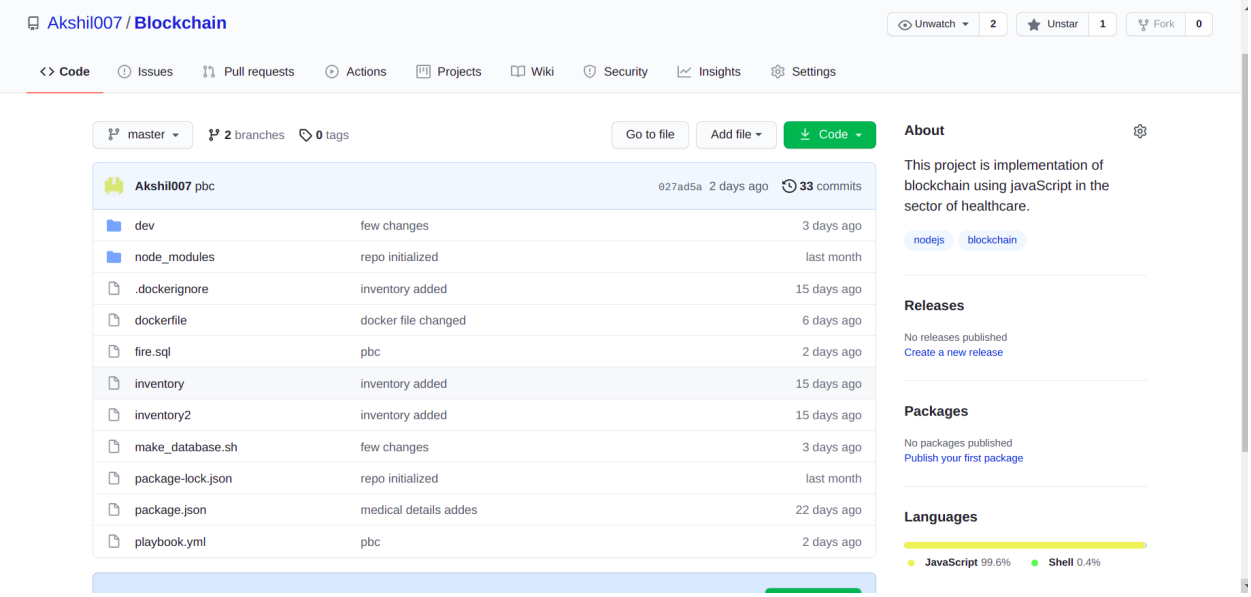
A Source Code Management (SCM) is a software tool used by programmers to manage the
source codes. For our project every team member would clone the repository from github. Create
a different branch locally on their system and then merge it with master after pulling the latest
code from git, resolving 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 reposit

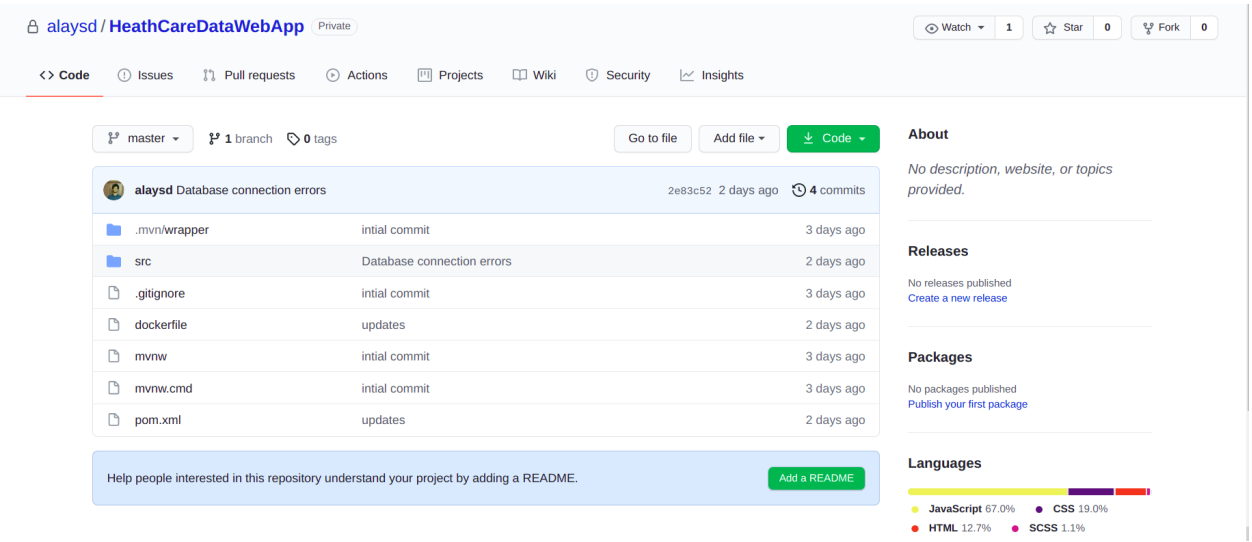
6.1.1 BLOCKCHAIN REPOSITORY

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



6.1.2 SPRING BOOT APPLICATION REPOSITORY

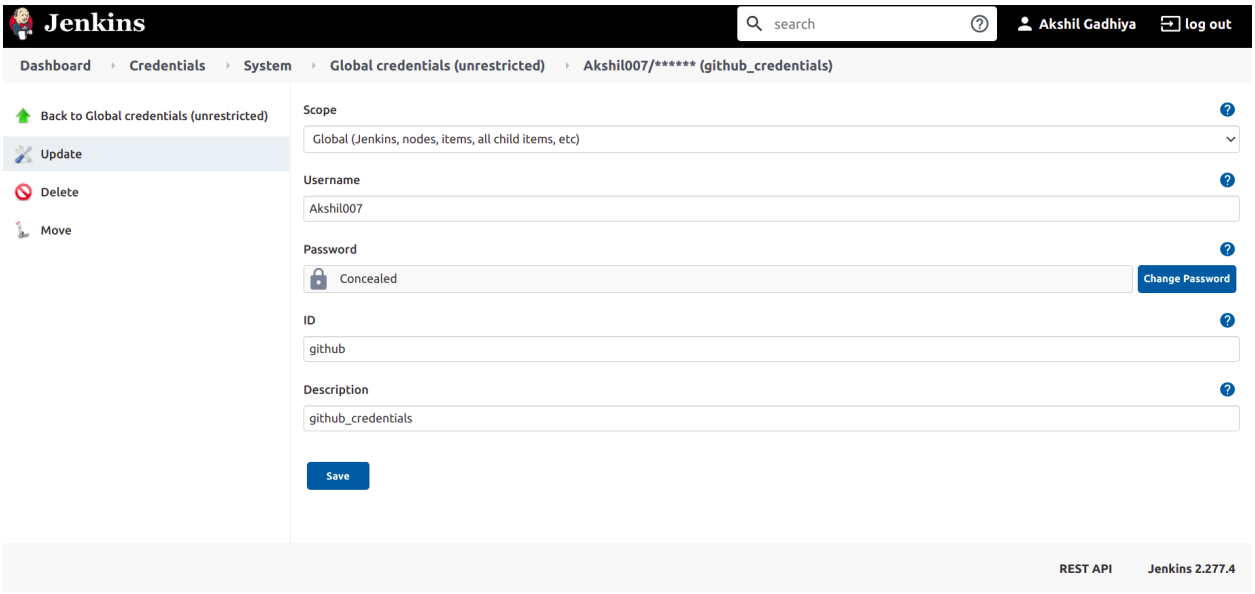
Link - https://github.com/alaysd/HeathCareDataWebApp



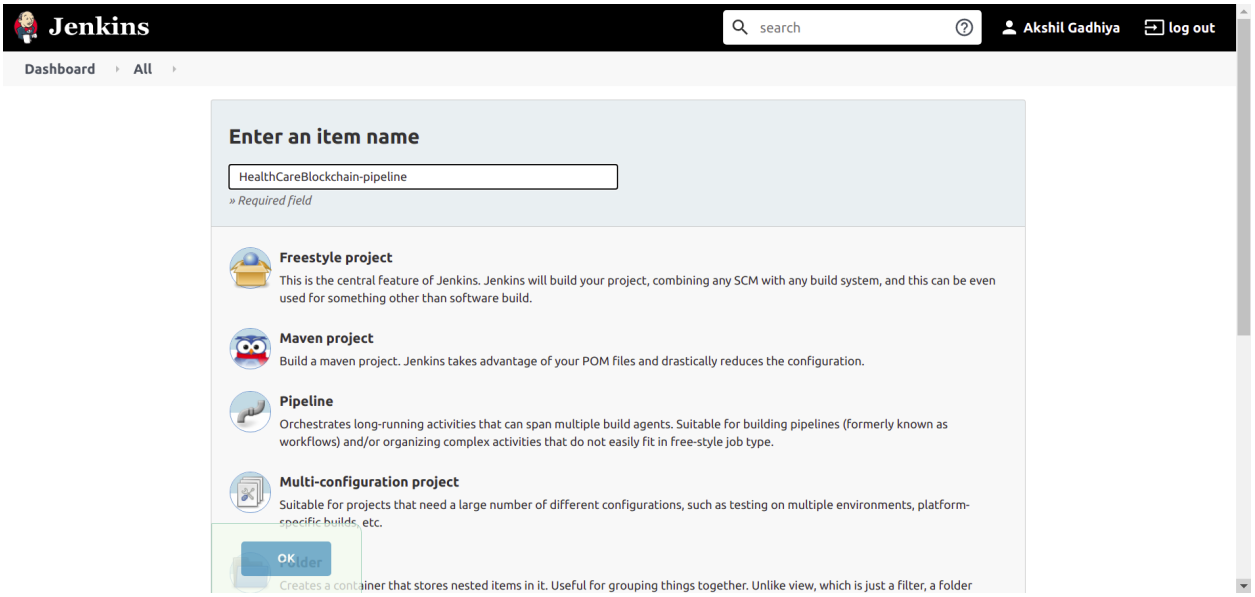
6.1.3 CI Pipeline

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.



- Then we will create new pipeline.



- Now we will clone project from our remote repository to out 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

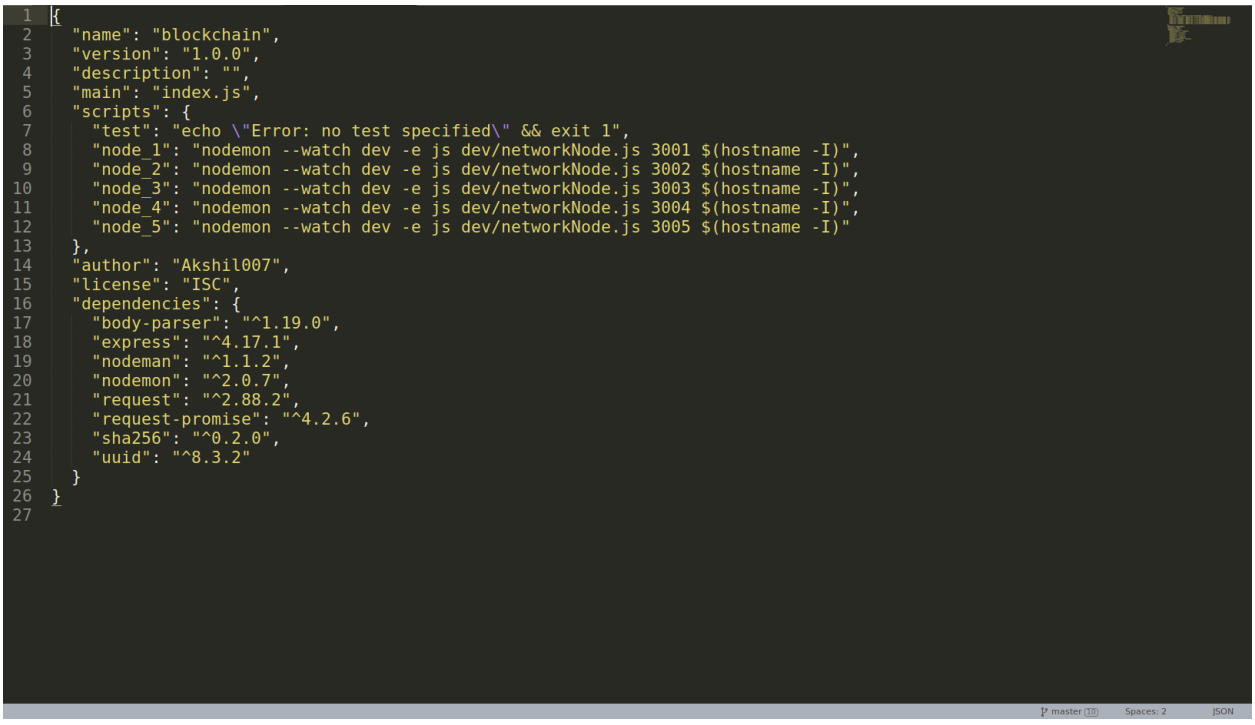
- NPM is the most important tool for working with Node applications. NPM is used to fetch any packages (JavaScript libraries) that an application needs for development, testing, and/or production, and may also be used to run tests and tools used in the development process.
- You can manually use NPM 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



```
{
  "name": "blockchain",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "node_1": "nodemon --watch dev -e js dev/networkNode.js 3001 $(hostname -I)",
    "node_2": "nodemon --watch dev -e js dev/networkNode.js 3002 $(hostname -I)",
    "node_3": "nodemon --watch dev -e js dev/networkNode.js 3003 $(hostname -I)",
    "node_4": "nodemon --watch dev -e js dev/networkNode.js 3004 $(hostname -I)",
    "node_5": "nodemon --watch dev -e js dev/networkNode.js 3005 $(hostname -I)"
  },
  "author": "Akshil007",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "express": "^4.17.1",
    "nodeman": "^1.1.2",
    "nodemon": "^2.0.7",
    "request": "^2.88.2",
    "request-promise": "^4.2.6",
    "sha256": "^0.2.0",
    "uuid": "^8.3.2"
  }
}
```

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

```
FROM node:latest
WORKDIR ./blockchain
ADD . .
CMD npm install
CMD npm run node_1

```

Docker file of Web Application

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

```

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

> $sudo usermod –aG docker <Userame>

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

> $sudo usermod -aG docker jenkins

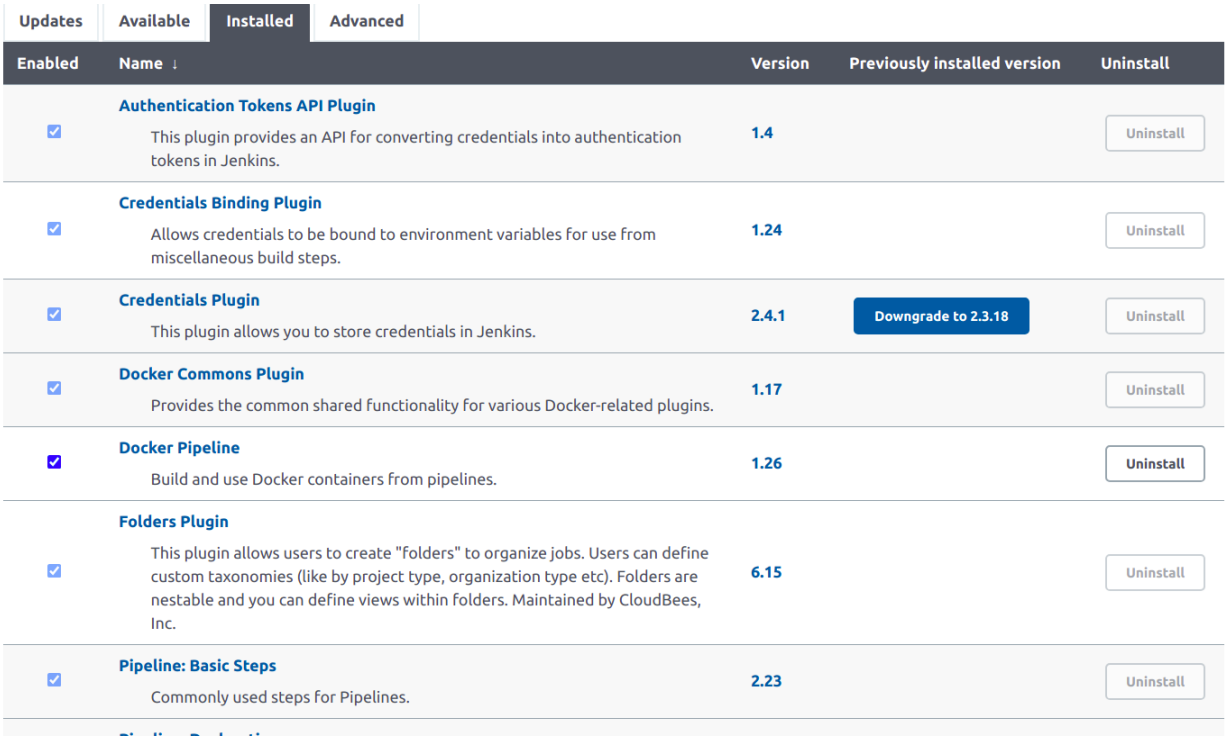
Now you can verify it using following command.

> $ sudo grep docker /etc/gshadow

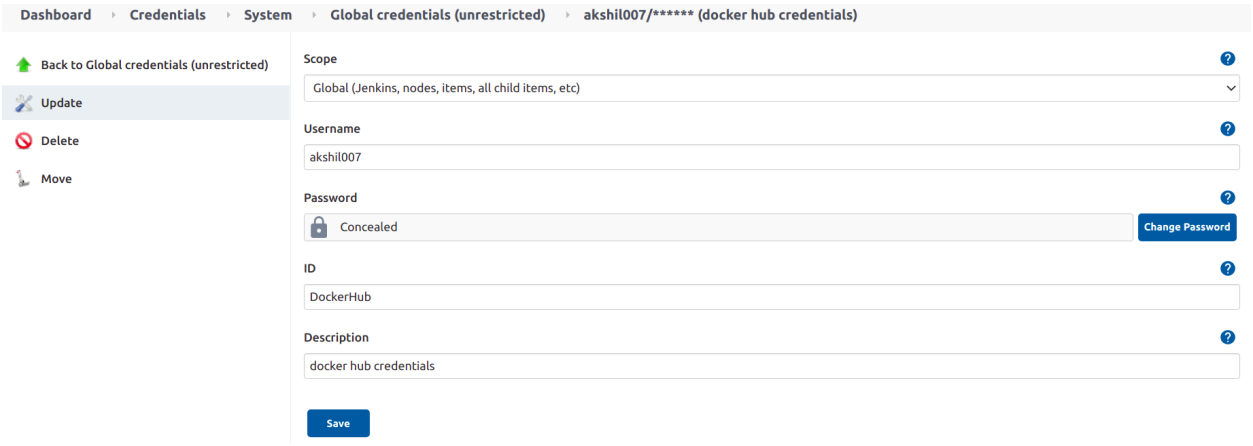
```
akshil@akshil-Inspiron-5379:~$ sudo grep docker /etc/gshadow
[sudo] password for akshil:
docker:!::jenkins,akshil
```

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.



- Add Docker Hub Credentials in Jenkins.



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

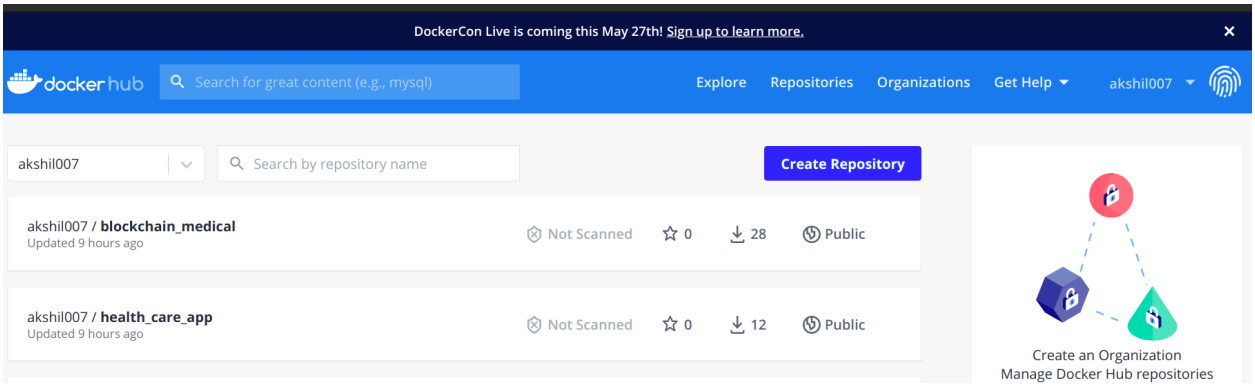
Script for building Web Application Image and deploying on docker hub in Jenkins pipeline

```
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 commnad.

> $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

```yaml
---
- name: Pull and Run docker containers
  hosts: all
  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

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


    - 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
```

Specify Hosts in Inventory file

```
localhost ansible_user=akshil
[alay]
192.168.1.36 ansible_user=alay

```

- 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

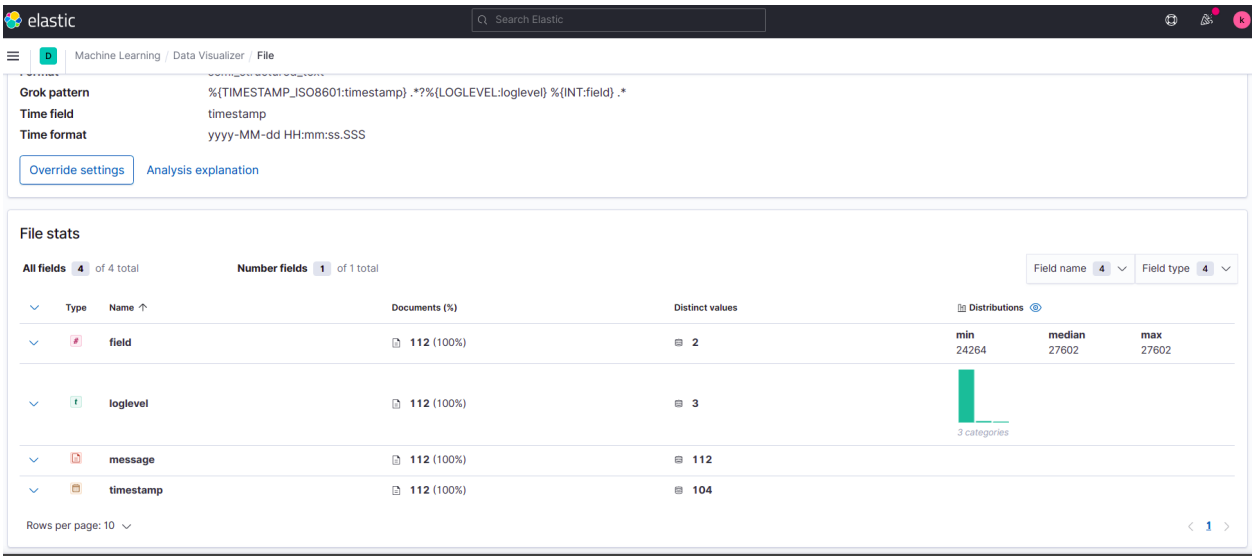
```
1  2021-05-17 22:27:59.664  INFO 24264 --- [main] b.m.H.HealthCareDataApplication          : Starting
   HealthCareDataApplication using Java 14.0.2 on akshil-Inspiron-5379 with PID 24264 (/home/akshil/
   Blockchain_WebApplication/HealthCareDataWebApp/target/classes started by akshil in /home/akshil/
   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'
```

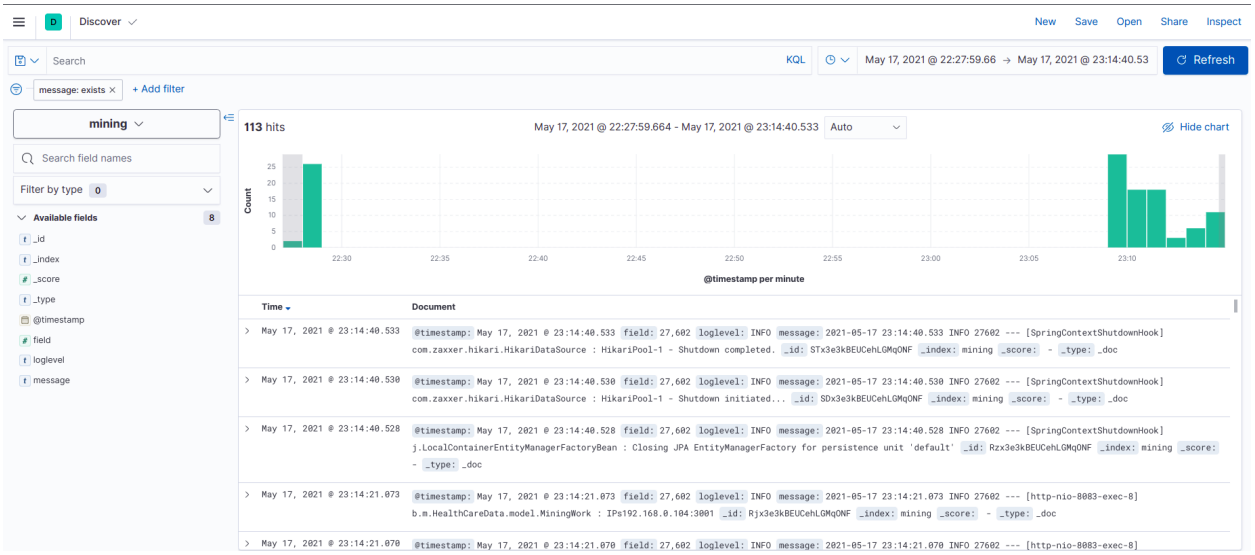
Logstash conf file feeded to logstash

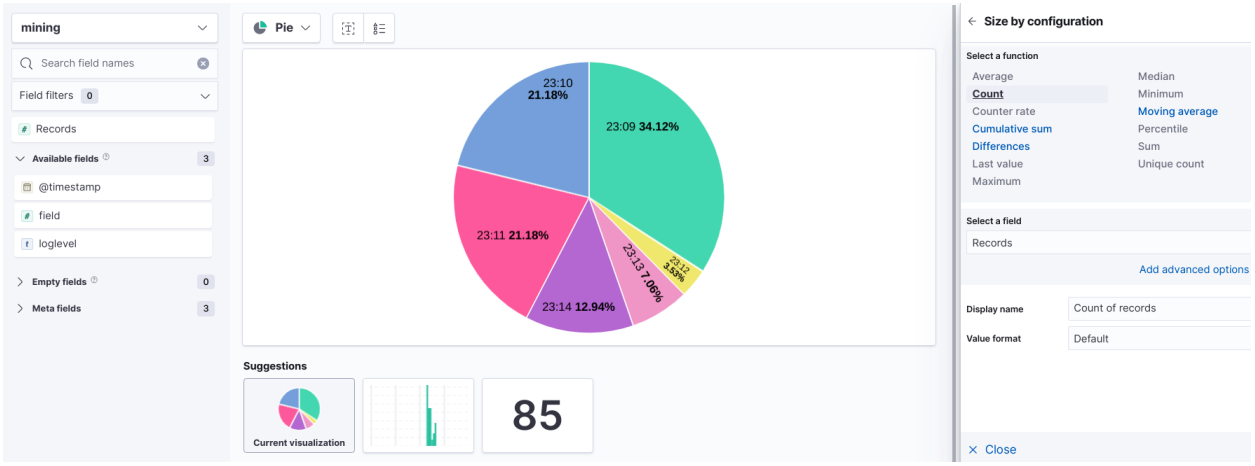
```
{
  "description": "Ingest pipeline created by text structure finder",
  "processors": [
    {
      "grok": {
        "field": "message",
        "patterns": [
          "%{TIMESTAMP_ISO8601:timestamp} .*?%{LOGLEVEL:loglevel} %{INT:field} .*"
        ]
      }
    },
    {
      "date": {
        "field": "timestamp",
        "timezone": "{{ event.timezone }}",
        "formats": [
          "yyyy-MM-dd HH:mm:ss.SSS"
        ]
      }
    },
    {
      "convert": {
        "field": "field",
        "type": "long",
        "ignore_missing": true
      }
    },
    {
      "remove": {
        "field": "timestamp"
      }
    }
  ]
}
```

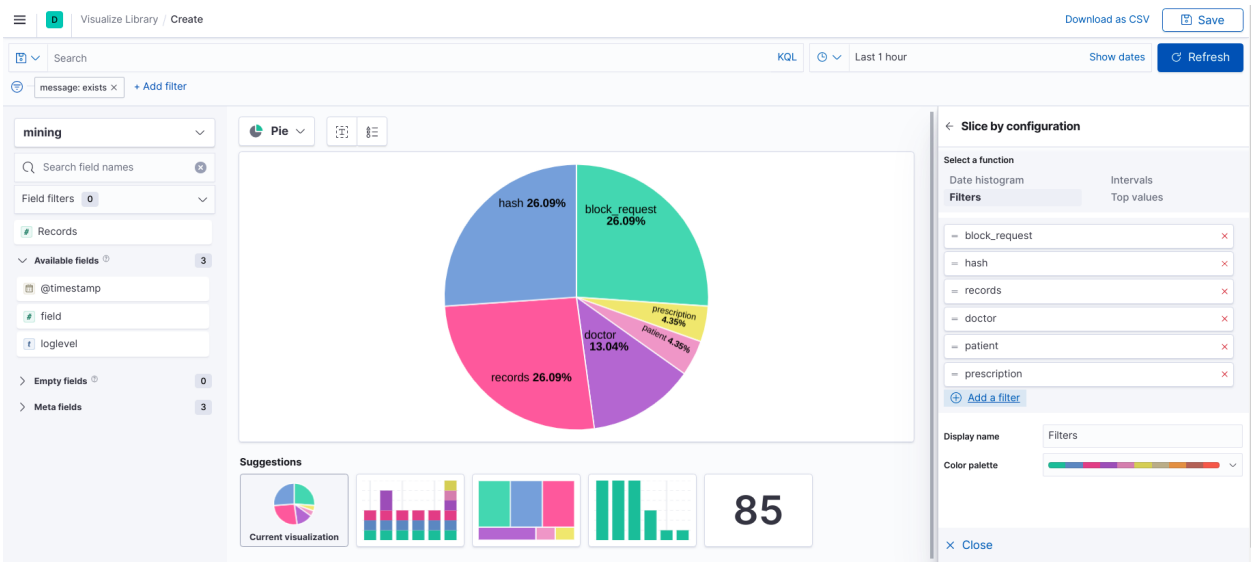
Now we can start the elasticsearch, logstash and kibana

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
Search 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
Field name 4
Field type 4
Type
Name
Documents (%)
Distinct values
Distributions
field 112 (100%) 2 min 24264 median 27602 max 27602
loglevel 112 (100%) 3 3 categories
message 112 (100%) 112
timestamp 112 (100%) 104
Rows per page: 10
1


Discover
New
Save
Open
Share
Inspect
Search
KQL
May 17, 2021 @ 22:27:59.66 → May 17, 2021 @ 23:14:40.53
Refresh
message: exists
+ Add filter
mining
Search field names
Filter by type 0
Available fields 8
_id
_index
_score
_type
@timestamp
field
loglevel
message
113 hits
May 17, 2021 @ 22:27:59.664 - May 17, 2021 @ 23:14:40.533
Auto
Hide chart
Count
@timestamp per minute
Time
Document


mining
Search field names
Field filters 0
Records
Available fields 3
@timestamp
field
loglevel
Empty fields 0
Meta fields 3
Pie
23:10 21.18%
23:09 34.12%
23:11 21.18%
23:13 7.06%
23:12 3.53%
23:14 12.94%
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
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

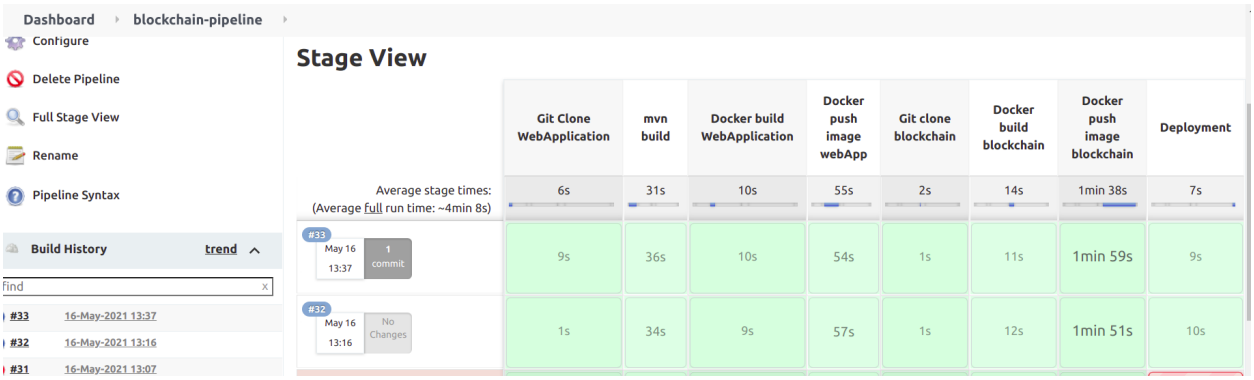
```
pipeline {
    environment{
        dockerImage = ""
    }
    agent any

    stages {
        stage('Git Clone WebApplication')
        {
            steps{
                git branch: 'master', credentialsId:
                'github', url: 'https://github.com/
                Akshil007/HealthCare_Blockchain.git'
            }
        }
        stage('mvn build')
        {
            steps{
                sh 'mvn clean install'
            }
        }
        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()
                    }
                }
```

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

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

Successful Run of Jenkins Pipeline



Now Lets go step by step in Deployment Process

- First We are removing already existing containers from machines.

```
  - 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.

```
  - 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-

1. Run the docker container on port number 8085 rather than running on port number 8083.
2. 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

```
akshil@akshil-Inspiron-5379:~$ docker run --network health_net -e DATABASE_HOST=health_db -e -it --name healthcare-webapp -p 8085:8
083 akshil007/health_care_app:latest

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.5)

2021-05-16 18:13:15.511  INFO 1 --- [           main] b.m.H.HealthCareDataApplication          : Starting HealthCareDataApplication
 v0.0.1-SNAPSHOT using Java 1.8.0_282 on 328d2fddb5fc with PID 1 (/app.jar started by root in /)
2021-05-16 18:13:15.516  INFO 1 --- [           main] b.m.H.HealthCareDataApplication          : No active profile set, falling bac
k to default profiles: default
2021-05-16 18:13:16.923  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repo
sitories in DEFAULT mode.
2021-05-16 18:13:16.972  INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository sc
anning in 35 ms. Found 0 JPA repository interfaces.
2021-05-16 18:13:18.022  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8
083 (http)
2021-05-16 18:13:18.043  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-05-16 18:13:18.044  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache T
omcat/9.0.45]
2021-05-16 18:13:18.154  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebAp
plicationContext
```

- Similarly for blockchain.

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

Running Blockchain

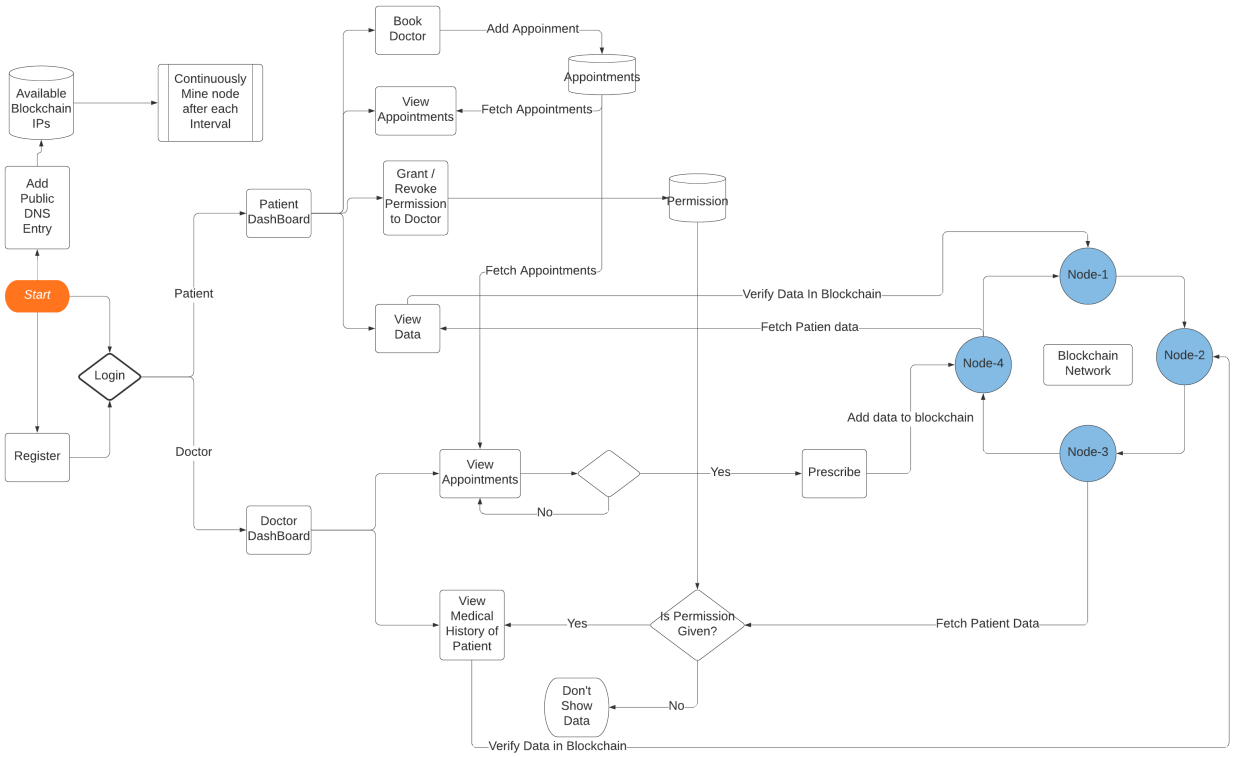
```
akshil@akshil-Inspiron-5379:~$ docker run --network health_net --name healthcare-blockchain akshil007/blockchain_medical:latest

> 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
listining to 3001 port...
```

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.

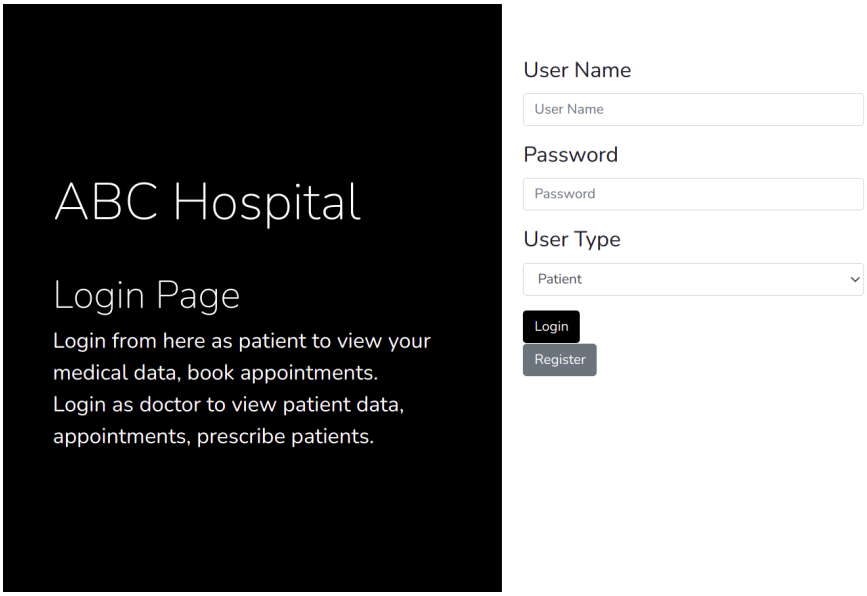
8.2.4 Mining

Mining is basically verifying transactions. All new transactions are waiting in pending records untill they get mined by some miner. Once Records get mined, they are added in blockchain. we uses 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

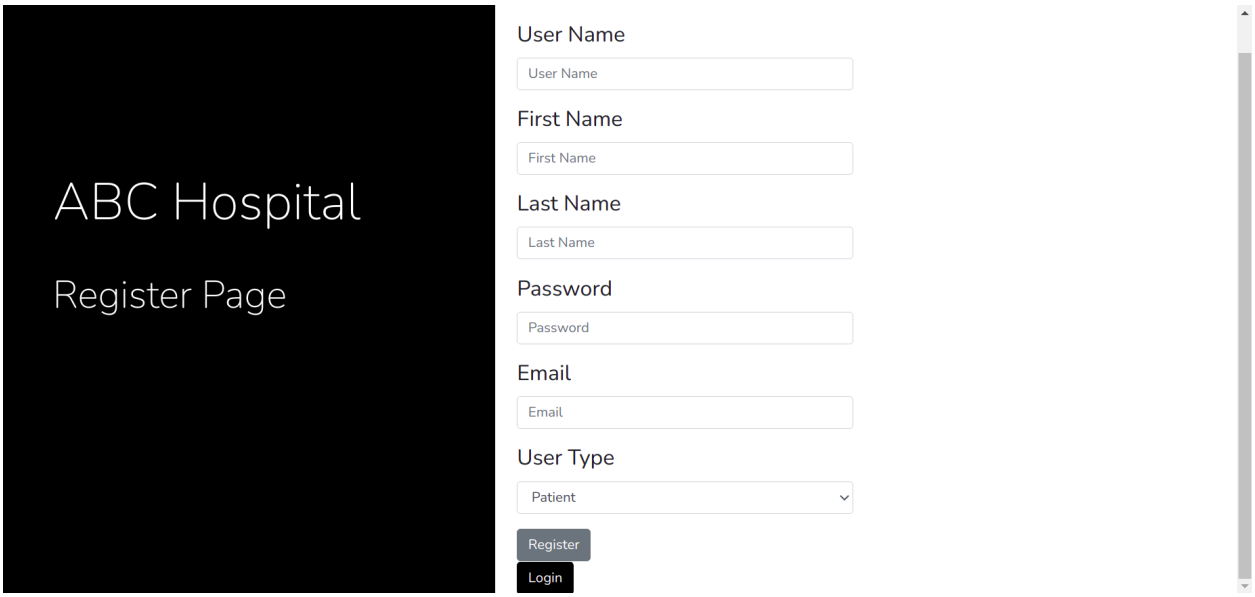
9.1 LOGIN PAGE

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



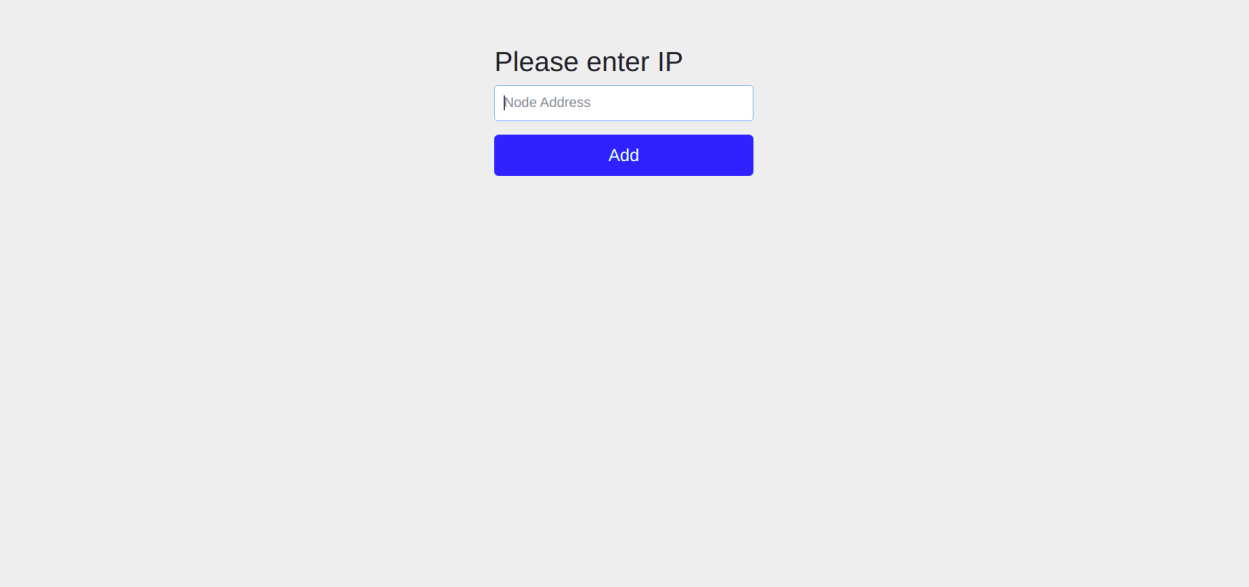
9.2 REGISTER PAGE

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



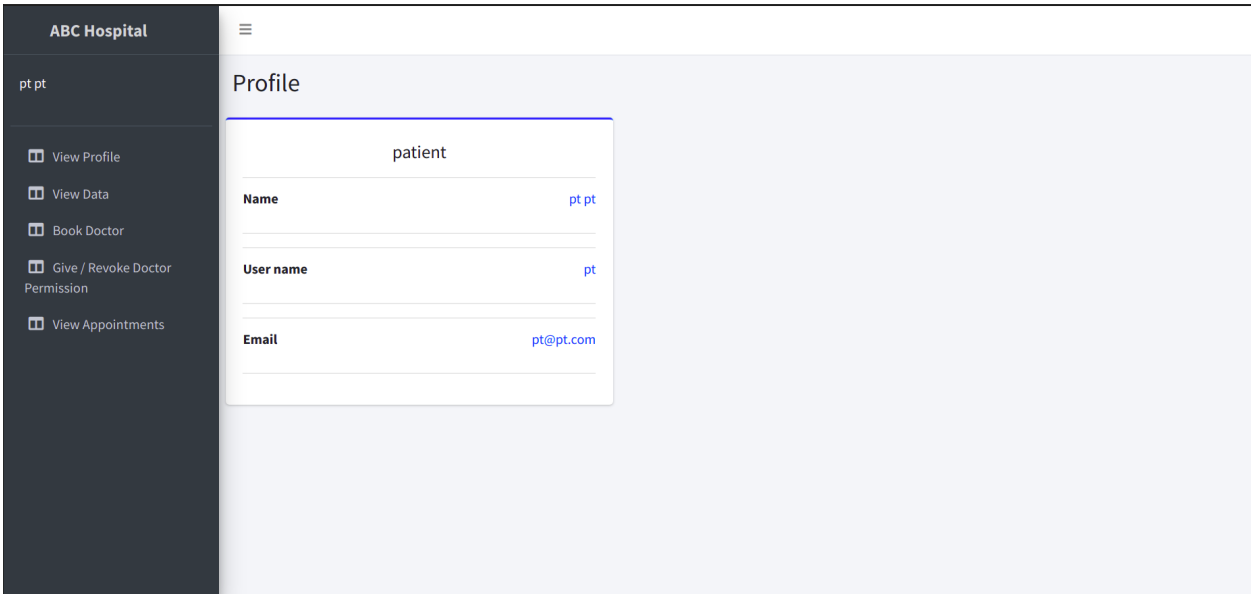
9.3 PUBLIC DNS ENTRY

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



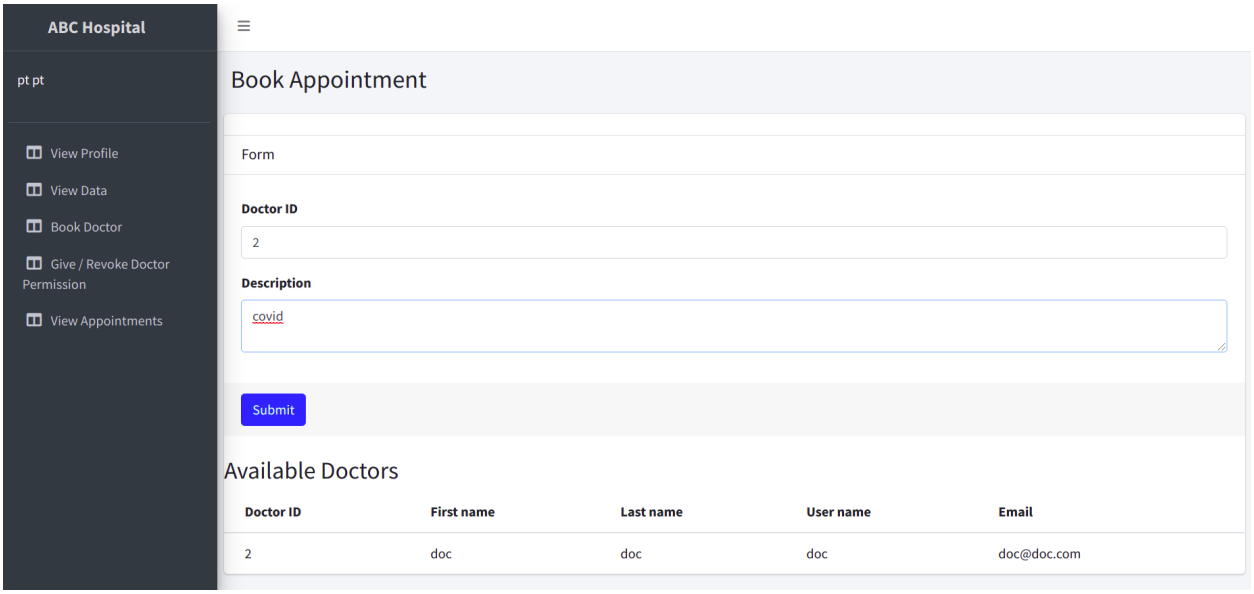
9.4 PATIENT DASHBOARD

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



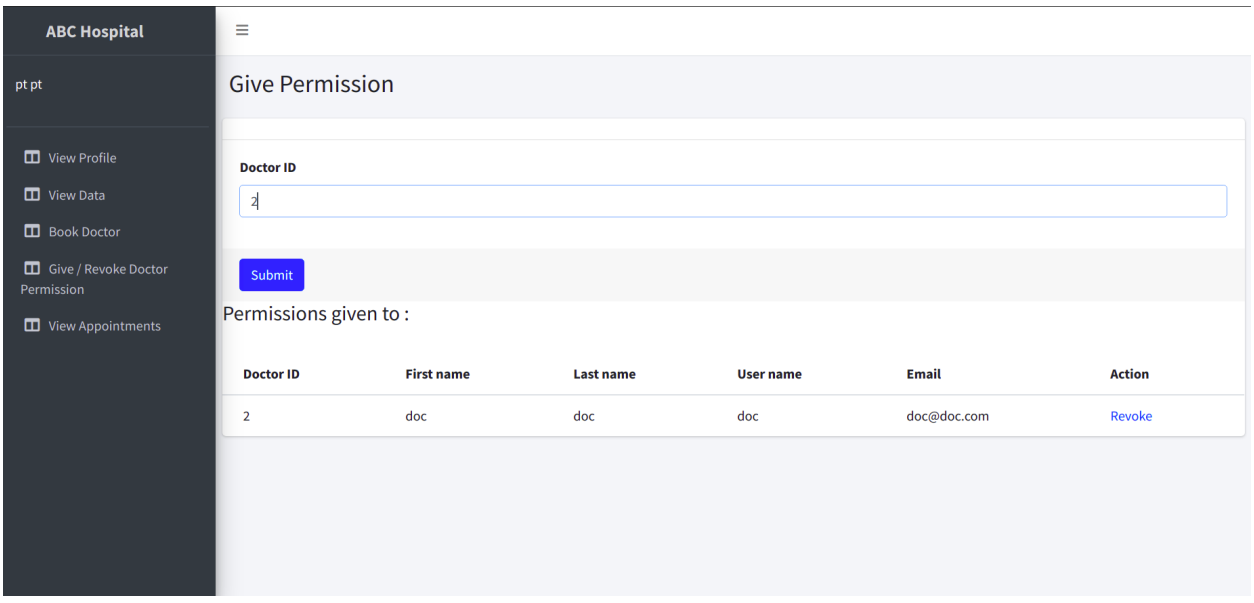
9.5 BOOK DOCTOR

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



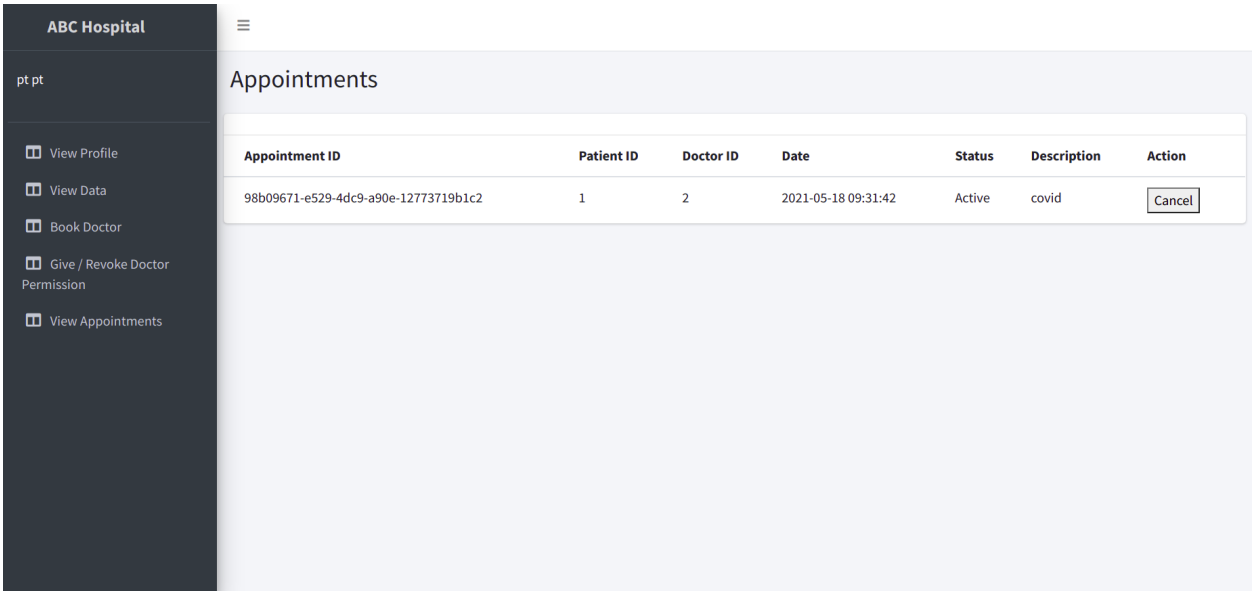
9.6 GIVE / REVOKE PERMISSION

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



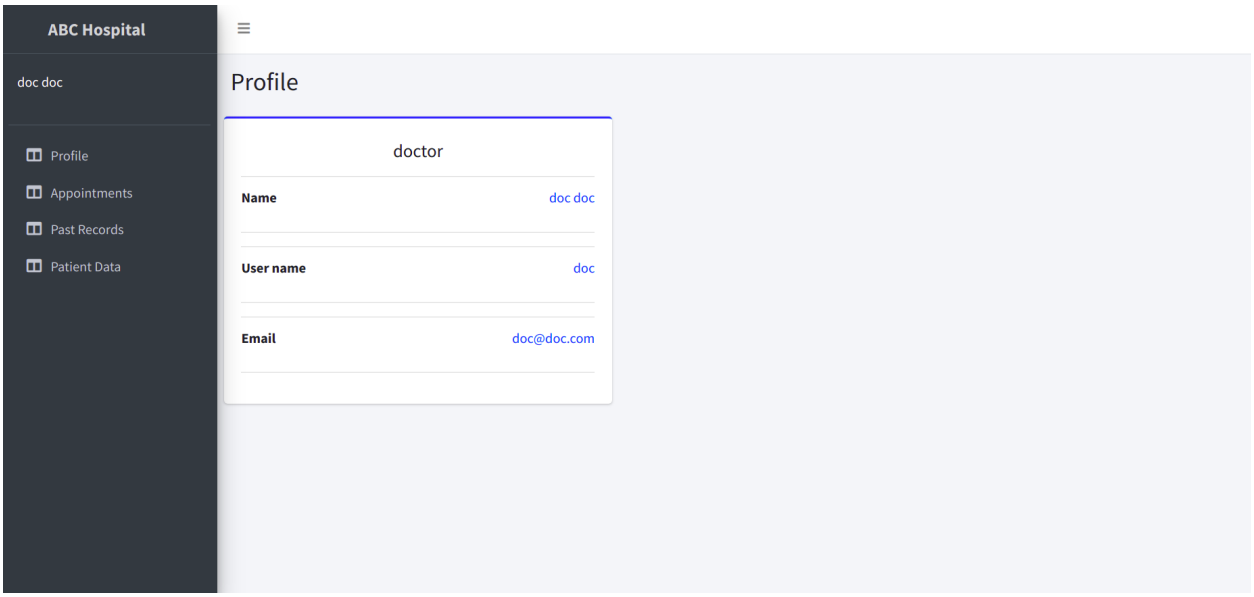
9.7 VIEW APPOINTMENTS

Here you can see your active appointments with doctor.



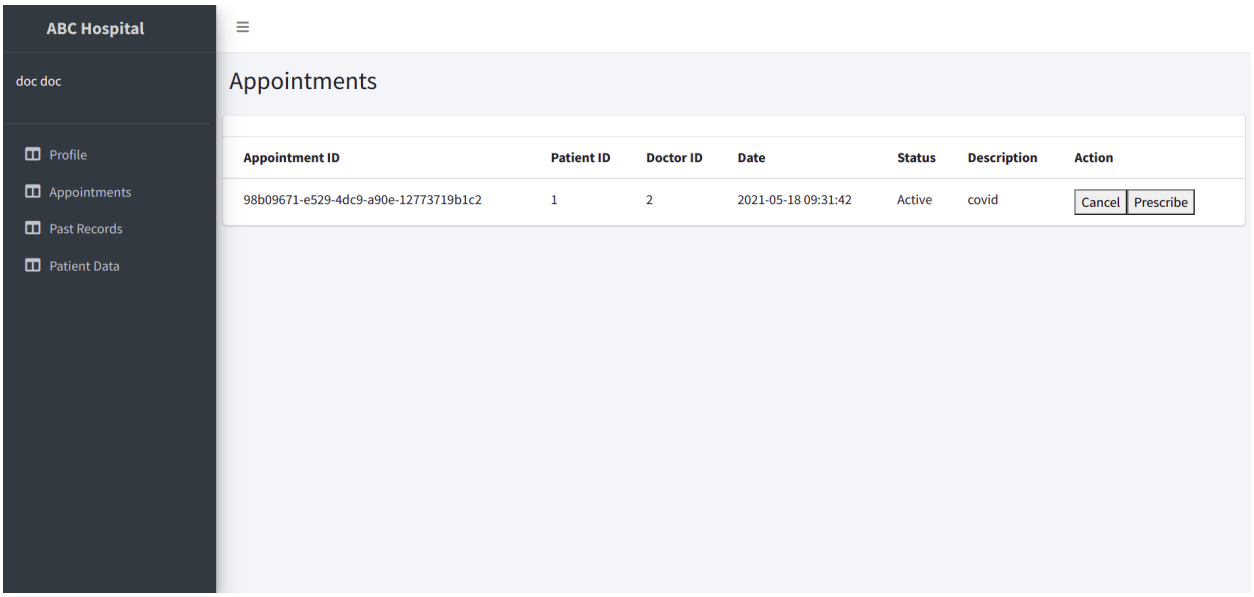
9.8 DOCTOR DASHBOARD

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



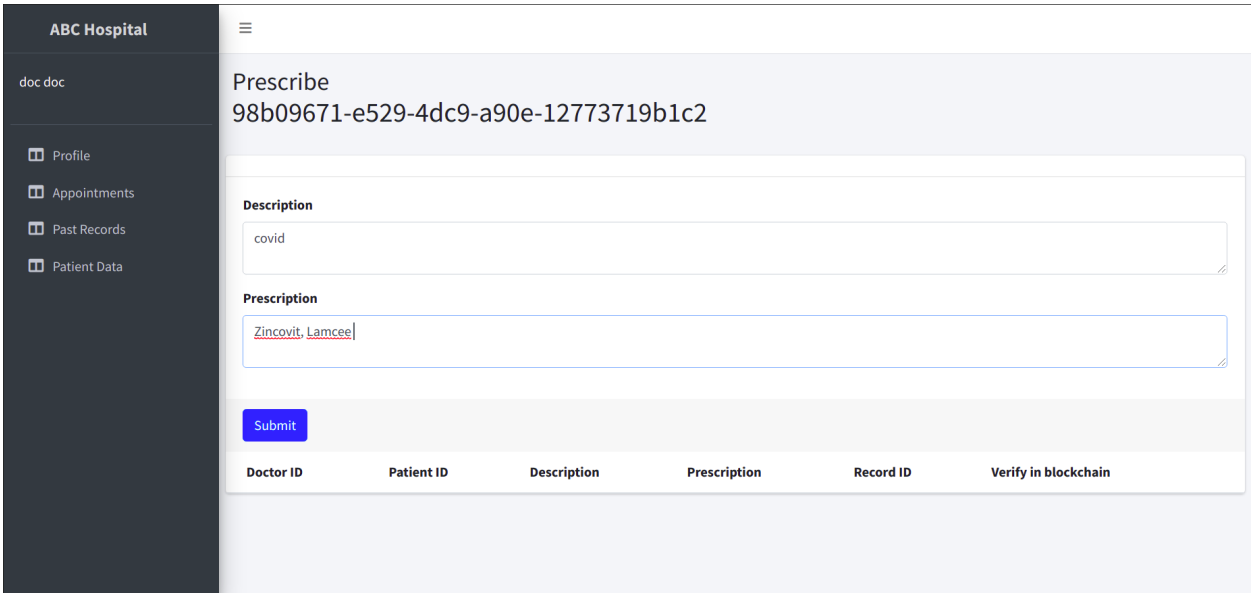
9.9 APPOINTMENTS

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



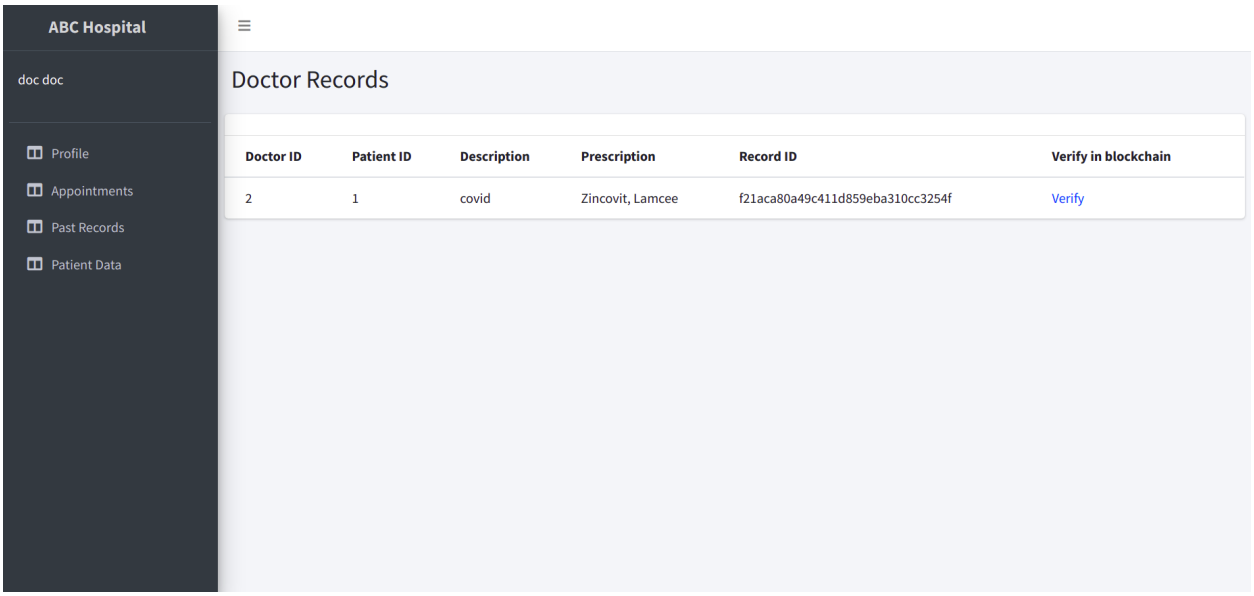
9.10 PRESCRIBE

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.

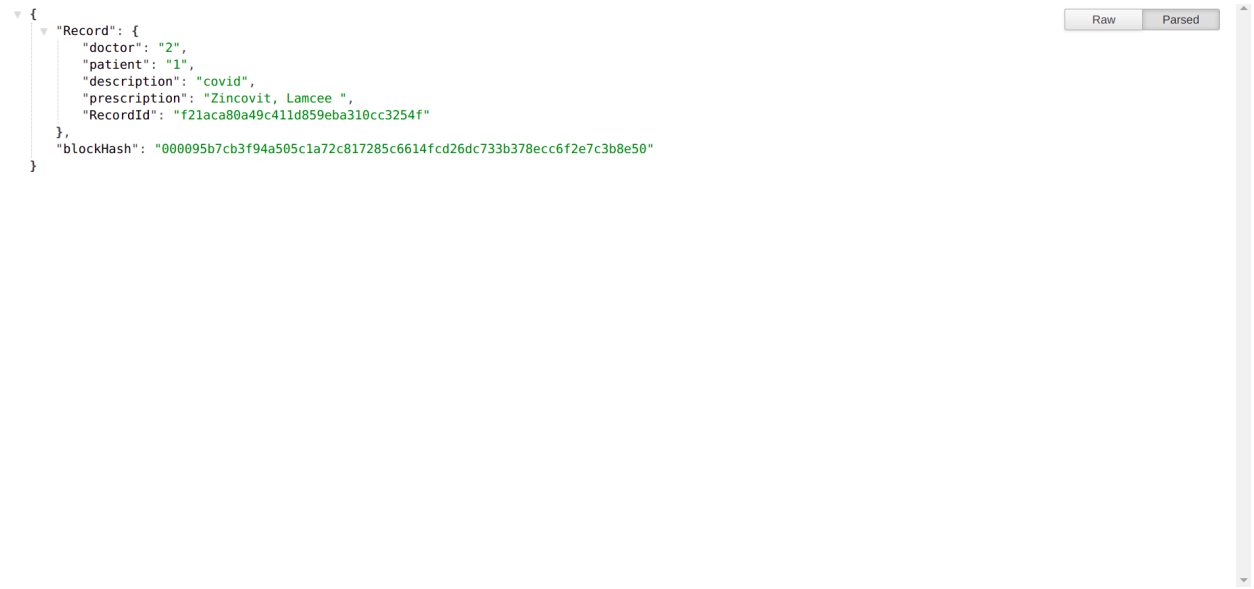


9.11 PAST RECORDS

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.



9.12 VERIFY IN BLOCKCHAIN



9.13 RUNNING BLOCKCHAIN

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.