International Institute of Information Technology, Bangalore

E-RaktKendra

Under the guidance of Prof. B Thangaraju

Group Members

Akanksha Shukla (MT2022008)

Vikranth Bari (MT2022131)

1. Abstract

E-RaktKendra is an online blood-bank. The idea is to develop an online blood bank portal where user/patient can request/donate for any blood type online. The user can see whether the required blood type is present in the user's nearby blood banks or not. If the required blood type is the user's nearby blood bank then he can request the blood through our online portal and can visit the nearby blood bank and can claim his request. In the same way the user can donate his blood too using our platform.

The project architecture includes three layers :

- Front end
- Back end
- Database

For our E-RaktKendra application, we employed full stack development technology. React has been used as the frontend library on the frontend side. Java Spring Boot was used for the backend. We have also implemented spring security JWT. We utilized MySQL for the database.

Tech Stack :

Front End Framework - React

Back End Framework - Java with Spring Boot and Spring Security JWT

Database – MySQL

2. Introduction

2.1 Overview

Our E-RaktKendra is an online blood bank portal. The idea behind this platform is to let the users who want to donate or request blood to make information of blood type available online.

Our E-RaktKendra Platform has two actors :

- User
- FieldWorker

User :

Through our portal a user can request for any blood type from the nearby blood bank. If the blood type is available in the nearby blood bank then only his request is accepted and he can visit nearby blood bank to claim his request. If the blood type is not available then the user is notified and his request is not accepted.
If the user wants to donate blood, he can request for donation in the nearby blood bank and can donate there.

FieldWorker :

Every Blood Bank has only one fieldworker registered. FieldWorker can view all the blood requests and blood donation requests for his blood bank. A fieldworker can accept or reject the user's blood request and blood donation requests based on the user's health condition.

2.2 Features

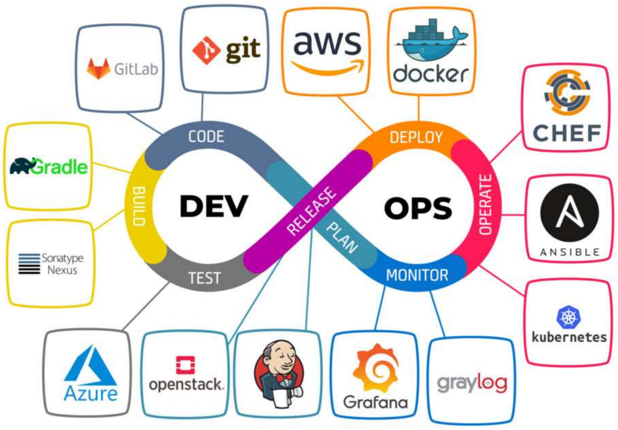
Sno.	Features	Description
1.	Register user	User can register himself
2.	User login	User can login and navigate to his dashboard
3.	User Blood Request	User can request for any blood type from any blood bank.
4.	User Blood Donation Request	User can request for donation of blood to any blood bank.
5.	Field-Worker Login	Field-Worker can login and will navigate to hi dashboard.
6.	View All Blood Requests	Field-Worker can view all blood requests for respective blood bank.
7.	View All Blood Donation Requests	Field-Worker can view all blood donation requests for respective blood bank.
8.	Accept/Reject Blood Request	Field-Worker can accept/reject the user's blood request.
9.	Accept/Reject Blood Donation Request	Field-Worker can accept/reject the user's blood donation request.

2.3 Why DEVOPS ?

Software and the Internet have transformed the world and its industries, from shopping to entertainment to banking. Software no longer merely supports a business; rather it becomes an integral component of every part of a business. Companies interact with their customers through software delivered as online services or applications and on all sorts of devices. They also use software to increase operational efficiencies by transforming every part of the value chain, such as logistics, communications, and operations. In a similar way that physical goods companies transformed how they design, build, and deliver products using industrial automation throughout the 20th century, companies in today's world must transform how they build and deliver software.
An effective DevOps pipeline is essential for every business that develops software in order to keep up with changing client expectations. The organization and concentration of the software development process is one of the pipeline's main goals. Continuous integration and continuous deployment (CI/CD) are the fundamental elements of the DevOps process. Configuration management, test and build systems, application deployment, version control, and monitoring tools are examples of DevOps tools. Different tools are needed for continuous integration, continuous delivery, and continuous deployment.

2.3.1 DevOps Features

- Continuous Integration and Deployment (CI/CD)
- Infrastructure as Code (IaC)
- Collaboration and Communication
- Agile and Lean Principles
- Monitoring and Logging
- DevOps Tools
- Scalability and Resilience



3. System Configuration

3.1 Operating System

Ubuntu 22.04

3.2 CPU and RAM

4 core processor and RAM 8 GB (preferable 16 GB)

3.3 Language

React web framework Java, Java with Spring Boot.

3.4 Database

MySQL database

3.5 Java Version

Open-jdk-11

3.6 Tools used :

- Continuous Development : Github
- Continuous Build : Maven
- Package-management toop : npm
- Continuous Integration : Jenkins
- Containerization : Docker
- Continuous Deployment : Ansible
- Continuous Monitoring : Elasticsearch-Logstash-Kibana

4. Software Development Lifecycle

4.1 Installation

4.1.1 Front End

React is a popular open-source JavaScript library for building user interfaces. Some key features of React include its component-based architecture, declarative programming model, and use of a virtual DOM for efficient updates. React allows developers to build reusable UI components, making it easier to maintain and scale complex applications. Overall, React is a powerful tool for building fast and responsive user interfaces, and has become a key technology in the web development industry.

Step 1 : Installing Npm

To install npm , open your terminal and type the following command :

- $ sudo apt install npm

To verify if the installation is completed successfully, check the npm version :

- $ npm –version

We can verify the node version through the following command :

- $ node –version

Step 2 : Install create-react-app tool

Run the following npm command to install the create-react-app utility :

- $ sudo npm -g install create-react-app

We can check the version using the following command :

- $ create-react-app –version

Now we can create our react application through the following command :

- $ create-react-app myapp

A new directory myapp is created. We can change the directory :

- $ cd myapp

To run the application, type the following command :

- $ npm start

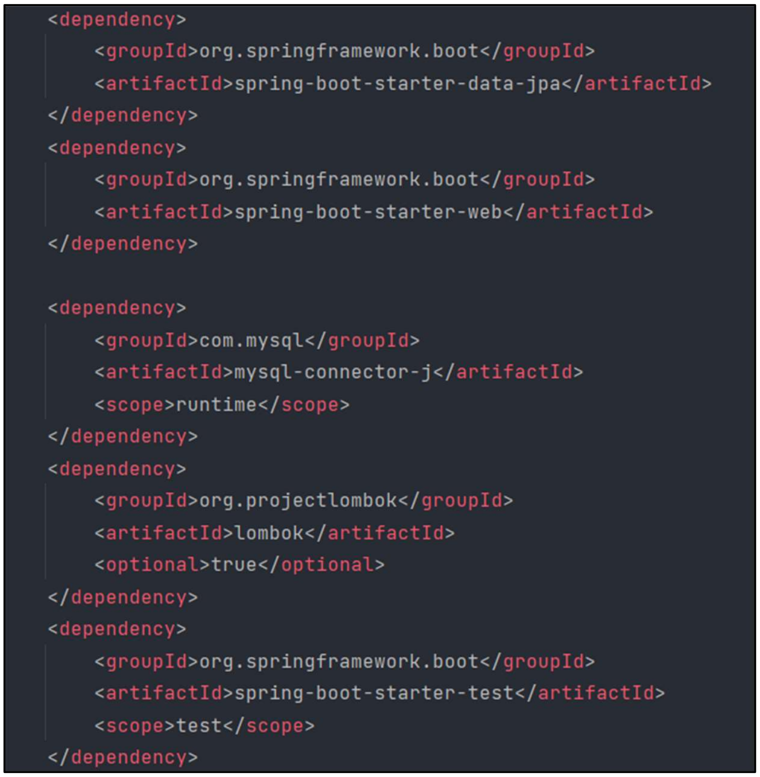
Our browser will open up and show our application and running with localhost:3000.

4.1.2 Backend

Spring Boot is a popular open-source framework for building Java-based web applications. It provides a streamlined and opinionated approach to building applications, with built-in features for configuration, dependency management, and web development. Spring Boot provides a fast and easy way to build Java-based web applications, and IntelliJ provides a powerful IDE for developing and managing your projects. Together, they provide a streamlined and efficient workflow for building high-quality applications.

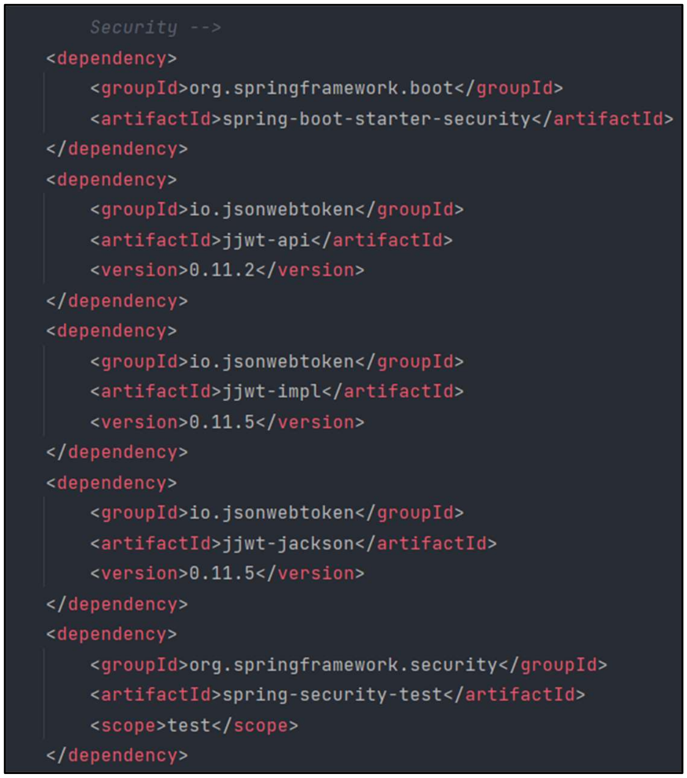
Steps for creating a new Spring Boot project in Intellij :

- Open IntelliJ and select "Create New Project".
- Select "Spring Initializr" from the left-hand menu and click "Next".
- Configure your project settings, including your project name, type, and language.
- Select the dependencies you want to include in your project, such as web, database, and security dependencies.
- Click "Finish" to create your project.
- IntelliJ will generate a new Spring Boot project structure with pre-configured files and dependencies, including a main application class, configuration files, and a pom.xml file for dependency management.
- You can now start building your application by adding new classes, controllers, and endpoints to your project.



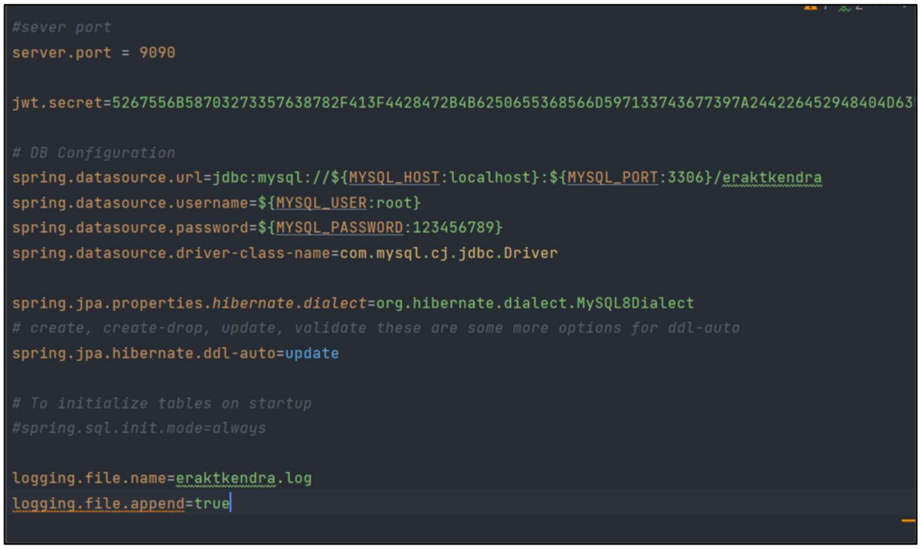
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

pom.xml



```
    Security -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-api</artifactId>
    <version>0.11.2</version>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-impl</artifactId>
    <version>0.11.5</version>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-jackson</artifactId>
    <version>0.11.5</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
```

pom.xml – jwt security dependencies



```
#sever port
server.port = 9090

jwt.secret=5267556B58703273357638782F413F4428472B4B6250655368566D597133743677397A244226452948404D63

# DB Configuration
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:${MYSQL_PORT:3306}/eraktkendra
spring.datasource.username=${MYSQL_USER:root}
spring.datasource.password=${MYSQL_PASSWORD:123456789}
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
# create, create-drop, update, validate these are some more options for ddl-auto
spring.jpa.hibernate.ddl-auto=update

# To initialize tables on startup
#spring.sql.init.mode=always

logging.file.name=eraktkendra.log
logging.file.append=true
```

application.properties

4.1.3 Database

Step 1 . Install the package of ‘mysql-server’

Execute the following command :

- $ sudo apt install mysql-server

This will install MySQL on ubuntu.

Step 2 . Verify MySQL service status

Once the installation is complete, MySQL immediately launches its service. We may use the following command to check the status of the MySQL service:

- $ sudo systemctl status mysql

Step 3 . Secure Configuration of MySQL

We will execute the security script in this stage to safeguard the installation. Some less secure features, such remote root logins, are changed when this script is run on our terminal. For secure setups, use the command:

- $ sudo mysql_secure_installation

Step 4 . Creating a dedicated MySQL user and granting privileges

- $ sudo mysql -u root -p

4.2 Source Control Management

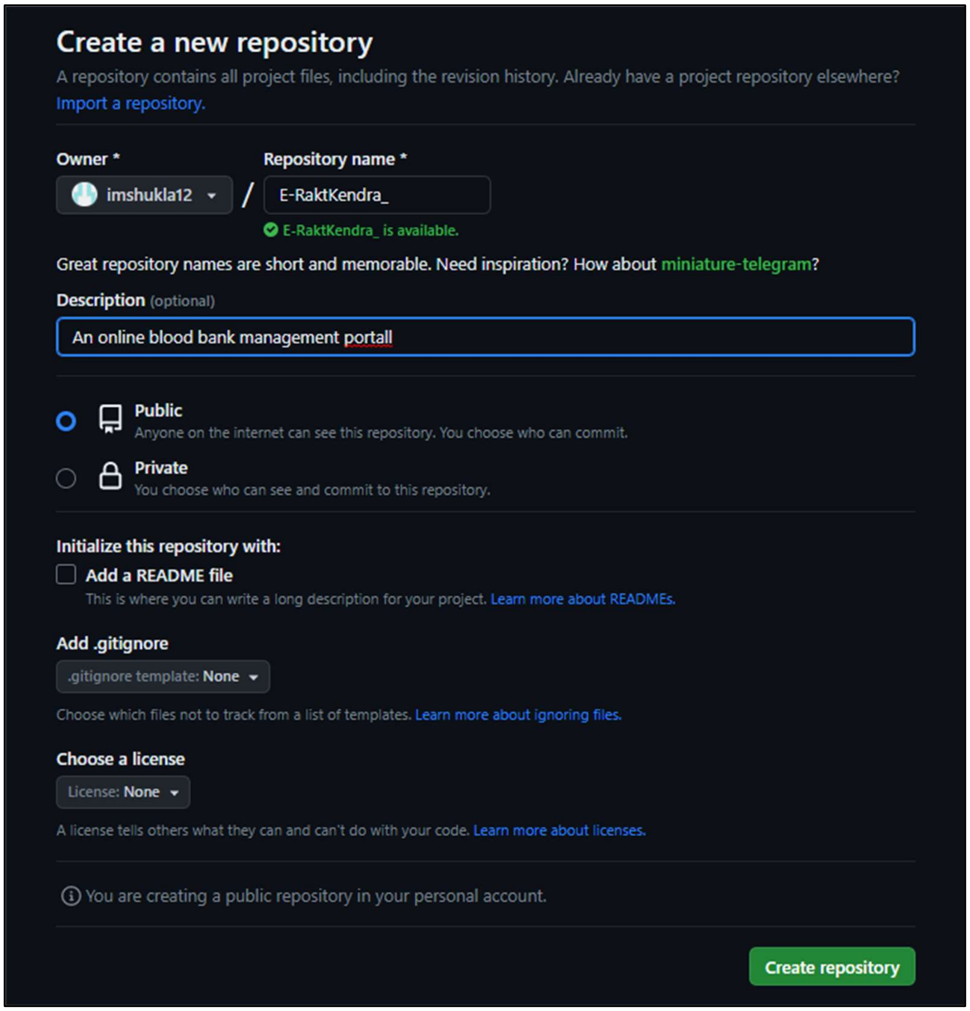
Source Code Management (SCM) is the practice of tracking changes to software code over time. It involves the use of specialized tools and processes to manage and organize the codebase, allowing developers to collaborate, share code, and maintain version control. For our project , every team member cloned the GitHub repository and worked on their respective branch and merge it with master branch.

- **git init**: initializes a new Git repository in the current directory.
- **git clone**: creates a copy of an existing Git repository.
- **git add**: adds files to the staging area for inclusion in the next commit.
- **git commit**: saves changes to the local repository, with a message describing the changes.
- **git push**: sends committed changes to a remote repository.
- **git pull**: retrieves changes from a remote repository and merges them into the local repository.
- **git status**: displays the current state of the local repository, including any changes or untracked files.
- **git log**: displays a log of all commits in the current branch, including commit messages, dates, and authors.

- **git branch**: lists all local branches in the current repository, or creates a new branch.
- **git checkout**: switches between branches, or checks out a specific commit.
- **git merge**: merges changes from one branch into another.
- **git stash**: temporarily stores changes that are not yet ready to be committed.
- **git remote**: manages connections to remote repositories, including adding, renaming, or removing connections.
- **git config**: sets configuration options for Git, such as username and email.

GitHub Repository Link

- Backend
- Frontend



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * imshukla12 / Repository name * E-RaktKendra_

E-RaktKendra_ is available.

Great repository names are short and memorable. Need inspiration? How about miniature-telegram?

Description (optional)

An online blood bank management portall

● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:

□ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.

4.3 Building

For building our project we need to install the Java Development Kit (JDK). The command for installation :

- $ sudo apt-get install openjdk-11-jdk

We can check the java version by the following command :

- $ java -version

Maven installation

- $ sudo apt install maven

Checking maven version

- $ mvn -version

Maven is a popular open-source build automation and dependency management tool used primarily for Java-based projects. It provides a simple and standardized way to manage the build process, including compiling code, running tests, and packaging artifacts. Maven uses a declarative XML file called a POM (Project Object Model) to manage the project's dependencies and build configuration. The POM file specifies the project's dependencies, including external libraries, plugins, and other dependencies required for the build process. Maven also provides a central repository of pre-built libraries and plugins called the Maven Central Repository, making it easy to find and include common dependencies in a project.

4.4 Frontend package management tool : npm

npm (Node Package Manager) is a package manager for JavaScript. It is the default package manager for the Node.js runtime environment, which is used for building server-side applications with JavaScript.

npm allows developers to easily install and manage third-party libraries and packages, called "modules," that can be used in their applications. Modules can be installed globally or locally to a specific project, and are specified in a package.json file that lists the dependencies and other metadata for the project.It is a powerful tool for managing dependencies and sharing code in the JavaScript ecosystem.

- $ npm install <package_name>

Used to install the required package in our application.

4.5 Docker

Docker is an open-source containerization platform that allows developers to package applications and their dependencies into self-contained "containers." Containers are lightweight, portable, and can run in any environment that supports Docker, making it easier to deploy and manage applications.

Docker uses a layered file system to build and manage containers, which allows developers to reuse common components across multiple applications and reduces the size of the container images. Docker also provides a central registry where developers can share and download container images, making it easy to distribute and deploy applications.

Dockerfile is created for client and server part that was run when the docker image is build.

- $ sudo apt install docker.io : used to install docker.
- $ sudo systemctl start docker : used to run docker via terminal.
- $ sudo systemctl status docker : used to view docker status.
- $ docker –version : used to check docker version.
- $ docker pull <image> : used to pull docker image.
- $ docker push <username>/<repository_name>: tagname : used to push docker image to docker hub.
- $ docker run -it <image> : used to run docker image.
- $ docker ps : used to view all the running containers.
- $ docker images : used to view all the docker images.
- $ docker build <directory>: used to build docker image from dockerfile.
- $ docker logs <container name/id>: used to view logs of a running container.
- $ docker stop <container name/id>: used to stop a running container.
- $ docker rm <container name/id>: used to remove a running container.

The instruction for creating a docker image is specified in a dockerfile.

A ‘Dockerfile’ is a text file that contains a set of instructions for building a Docker image. It is used to automate the process of creating a Docker image from a base image or a previously built image.

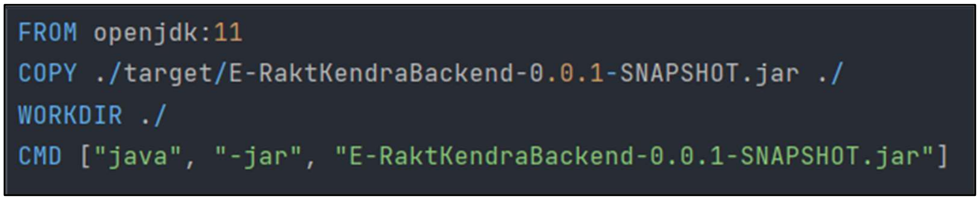
Docker - compose

Docker Compose is a tool that allows us to define and run multi-container Docker applications. It is used to define and run multiple containers as a single service. Compose is a YAML file that defines the services, networks, and volumes for our application.

Docker Compose simplifies the process of managing multi-container Docker applications. With Compose, we can define all the containers, networks, and volumes for our application in a single file, and then use a single command to start up the entire application.

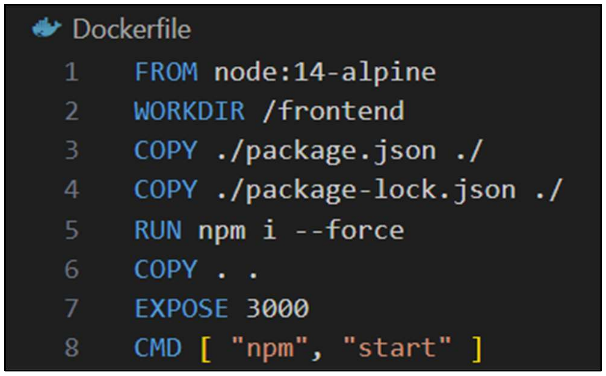
Docker hub link

- Backend
- Frontend



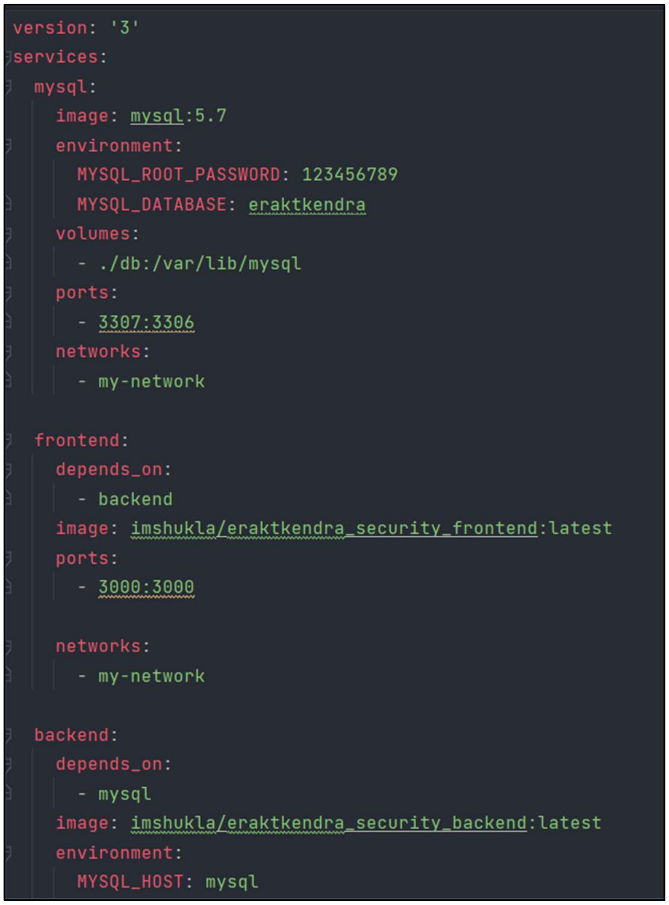
```
FROM openjdk:11
COPY ./target/E-RaktKendraBackend-0.0.1-SNAPSHOT.jar ./
WORKDIR ./
CMD ["java", "-jar", "E-RaktKendraBackend-0.0.1-SNAPSHOT.jar"]
```

Dockerfile – backend



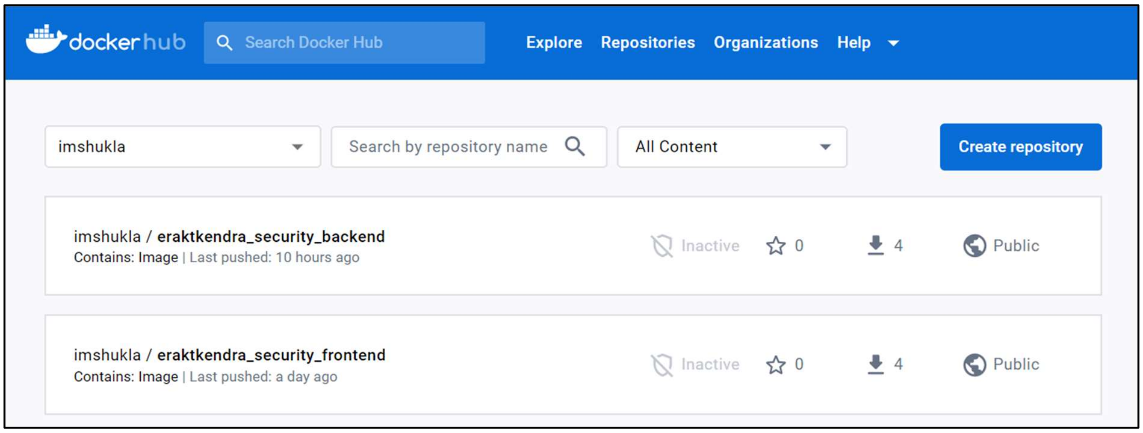
```
FROM node:14-alpine
WORKDIR /frontend
COPY ./package.json ./
COPY ./package-lock.json ./
RUN npm i --force
COPY . .
EXPOSE 3000
CMD [ "npm", "start" ]
```

Dockerfile – frontend



```
version: '3'
services:
  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: 123456789
      MYSQL_DATABASE: eraktkendra
    volumes:
      - ./db:/var/lib/mysql
    ports:
      - 3307:3306
    networks:
      - my-network

  frontend:
    depends_on:
      - backend
    image: imshukla/eraktkendra_security_frontend:latest
    ports:
      - 3000:3000

    networks:
      - my-network

  backend:
    depends_on:
      - mysql
    image: imshukla/eraktkendra_security_backend:latest
    environment:
      MYSQL_HOST: mysql
```

Docker-compose.yml file



imshukla / eraktkendra_security_backend
Contains: Image | Last pushed: 10 hours ago
Inactive ☆ 0 4 Public

imshukla / eraktkendra_security_frontend
Contains: Image | Last pushed: a day ago
Inactive ☆ 0 4 Public

Docker hub repository

4.6 Ansible Configuration and pull docker image

Ansible is an open-source automation tool designed for managing the configuration of servers, deploying applications, and orchestrating IT processes.

It uses a simple and human-readable language to define automation tasks, which makes it easy to learn and use even for non-developers.

Ansible allows us to manage multiple servers at once, using a central configuration file called a playbook. Playbooks are written in YAML format and can be used to define tasks such as installing packages, starting or stopping services, copying files, or executing custom scripts.

Ansible Playbook

An Ansible playbook is a file that contains a set of instructions that Ansible uses to automate a series of tasks on remote hosts. Playbooks are written in YAML (Yet Another Markup Language) format and consist of a series of plays, where each play is a set of tasks that need to be executed on the remote hosts.

Inventory File

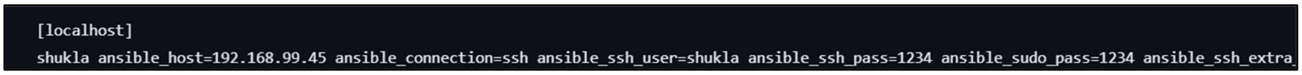
In Ansible, an inventory file is a file that contains information about the hosts or nodes that Ansible will manage. It is a simple text file that lists the hostnames or IP addresses of the target machines and groups them into logical units.

The inventory file is used by Ansible to know which machines it should communicate with and to group machines into different categories such as web servers, database servers, etc. It can also be used to define variables that can be used in the playbook.

On the controller and managed nodes, we must first install python and an ssh server before installing Ansible.

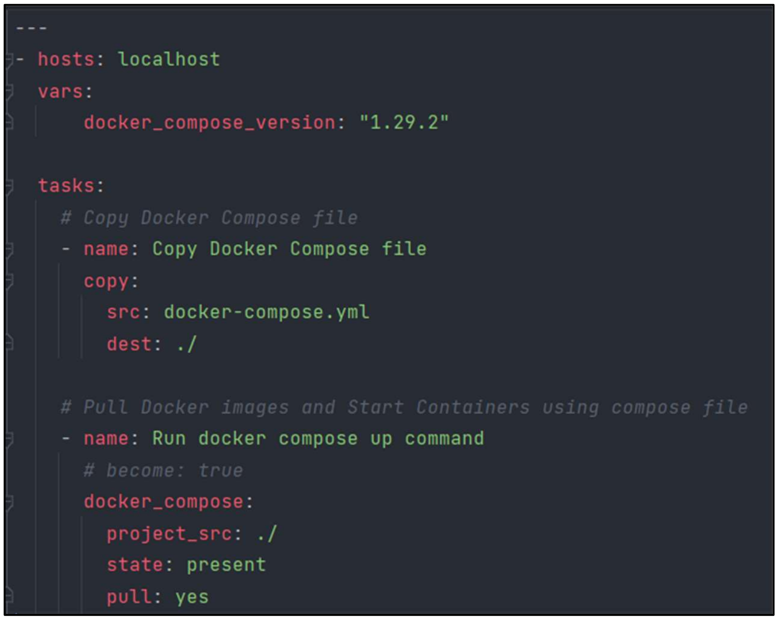
- $ sudo apt install openssh-server
- $ ssh-keygen -t rsa
- $ sudo apt install ansible
- $ ansible –version

Copy ssh key generated on remote server onto our Jenkins serve



```
[localhost]
shukla ansible_host=192.168.99.45 ansible_connection=ssh ansible_ssh_user=shukla ansible_ssh_pass=1234 ansible_sudo_pass=1234 ansible_ssh_extra_
```

Inventory file



```yaml
---
- hosts: localhost
  vars:
      docker_compose_version: "1.29.2"

  tasks:
    # Copy Docker Compose file
    - name: Copy Docker Compose file
      copy:
        src: docker-compose.yml
        dest: ./

    # Pull Docker images and Start Containers using compose file
    - name: Run docker compose up command
      # become: true
      docker_compose:
        project_src: ./
        state: present
        pull: yes
```

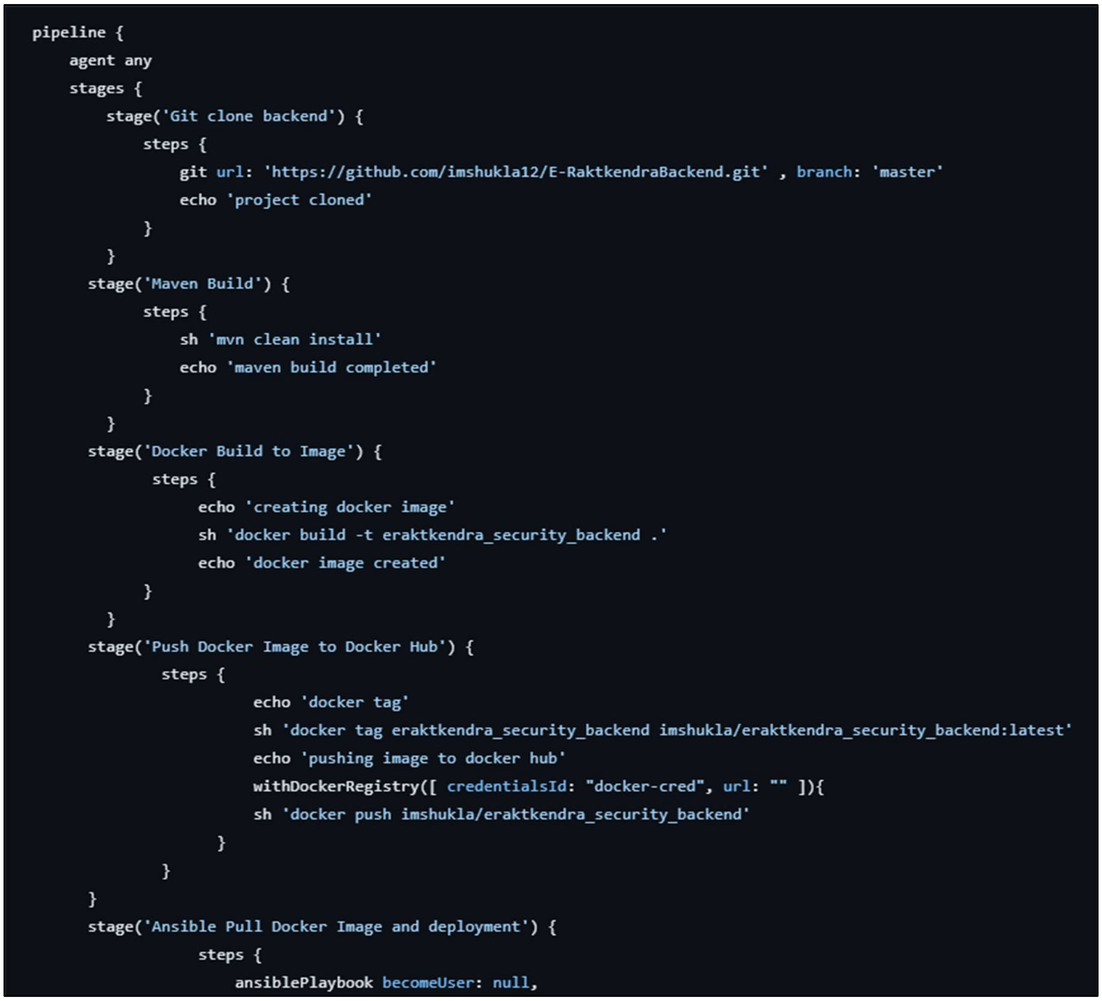
ansible playbook

4.7 Jenkins Pipeline

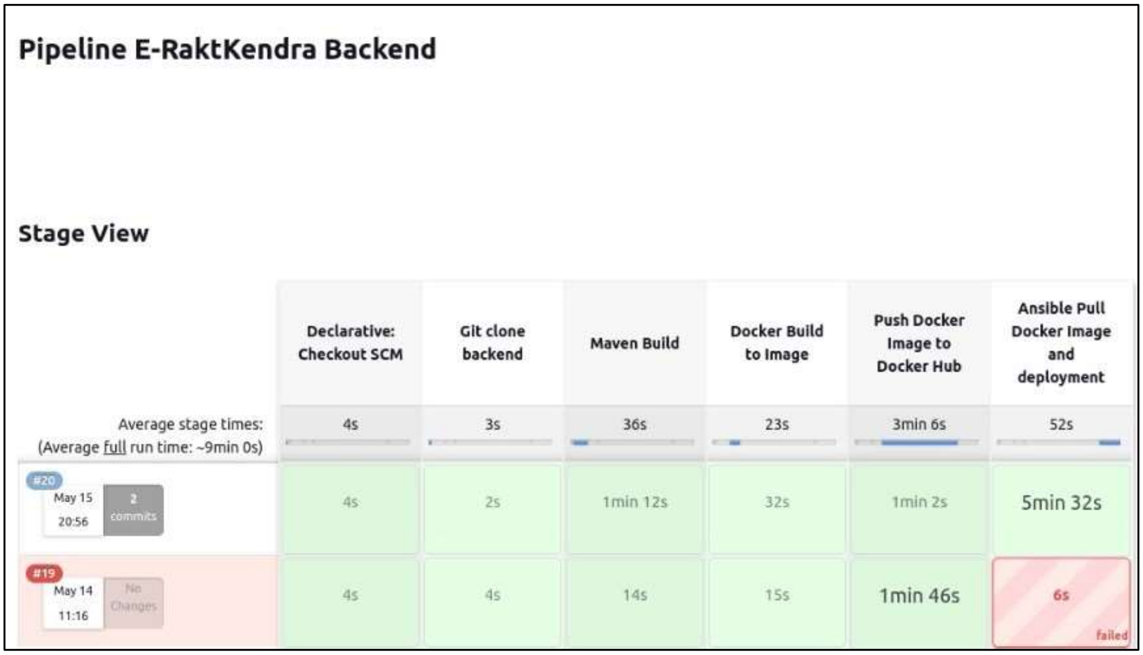
Jenkins is a popular open-source automation server used for continuous integration and continuous delivery (CI/CD) of software projects. It provides a web-based interface to manage, automate, and monitor the software development processes. With Jenkins, developers can automate the building, testing, and deployment of their software projects, allowing them to release new features and updates more quickly and efficiently.

Jenkins Pipeline is a suite of plugins that allows users to define and automate their software delivery pipelines as code. This means that the pipeline is defined using a Jenkinsfile, which is a text file that contains the stages and steps of the pipeline. With Jenkins Pipeline, developers can define the entire delivery process in code, from building and testing to deploying and releasing the software.

Jenkins Pipeline – backend



```
pipeline {
    agent any
    stages {
        stage('Git clone backend') {
            steps {
                git url: 'https://github.com/imshukla12/E-RaktkendraBackend.git' , branch: 'master'
                echo 'project cloned'
            }
        }
      stage('Maven Build') {
            steps {
                sh 'mvn clean install'
                echo 'maven build completed'
            }
        }
      stage('Docker Build to Image') {
             steps {
                  echo 'creating docker image'
                  sh 'docker build -t eraktkendra_security_backend .'
                  echo 'docker image created'
            }
        }
      stage('Push Docker Image to Docker Hub') {
              steps {
                        echo 'docker tag'
                        sh 'docker tag eraktkendra_security_backend imshukla/eraktkendra_security_backend:latest'
                        echo 'pushing image to docker hub'
                        withDockerRegistry([ credentialsId: "docker-cred", url: "" ]){
                        sh 'docker push imshukla/eraktkendra_security_backend'
                    }
              }
      }
      stage('Ansible Pull Docker Image and deployment') {
                  steps {
                      ansiblePlaybook becomeUser: null,
```

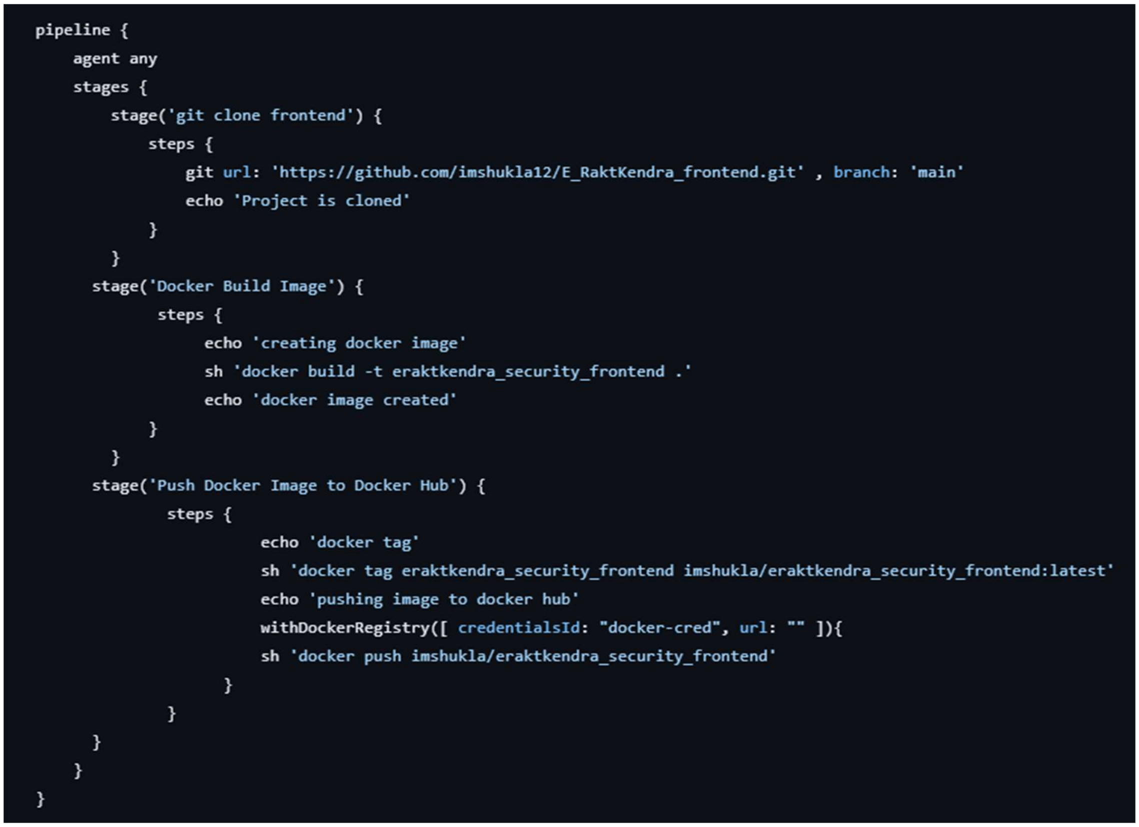


Pipeline E-RaktKendra Backend

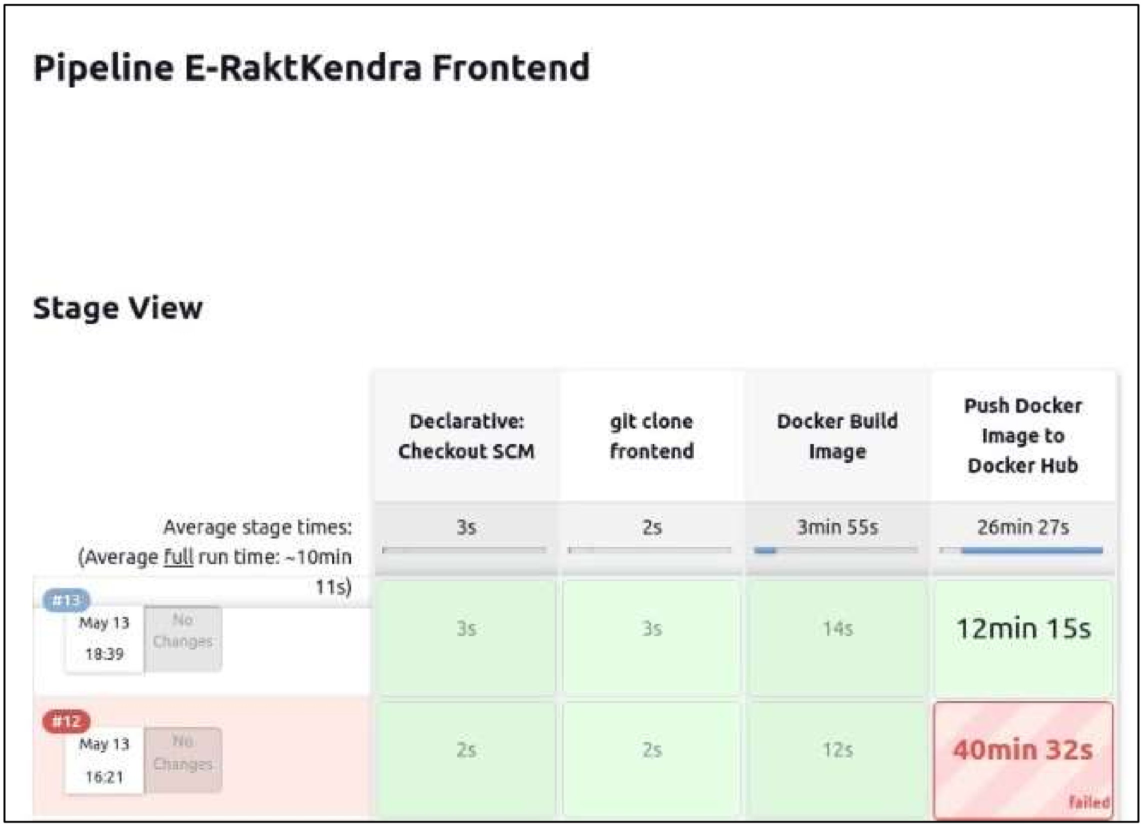
Stage View

	Declarative: Checkout SCM	Git clone backend	Maven Build	Docker Build to Image	Push Docker Image to Docker Hub	Ansible Pull Docker Image and deployment
Average stage times: (Average full run time: ~9min 0s)	4s	3s	36s	23s	3min 6s	52s
#20 May 15 20:56 2 commits	4s	2s	1min 12s	32s	1min 2s	5min 32s
#19 May 14 11:16 No Changes	4s	4s	14s	15s	1min 46s	6s failed

Jenkins Pipeline – frontend



```
pipeline {
    agent any
    stages {
        stage('git clone frontend') {
            steps {
                git url: 'https://github.com/imshukla12/E_RaktKendra_frontend.git' , branch: 'main'
                echo 'Project is cloned'
            }
        }
      stage('Docker Build Image') {
             steps {
                  echo 'creating docker image'
                  sh 'docker build -t eraktkendra_security_frontend .'
                  echo 'docker image created'
            }
        }
      stage('Push Docker Image to Docker Hub') {
              steps {
                        echo 'docker tag'
                        sh 'docker tag eraktkendra_security_frontend imshukla/eraktkendra_security_frontend:latest'
                        echo 'pushing image to docker hub'
                        withDockerRegistry([ credentialsId: "docker-cred", url: "" ]){
                        sh 'docker push imshukla/eraktkendra_security_frontend'
                    }
              }
      }
    }
}
```



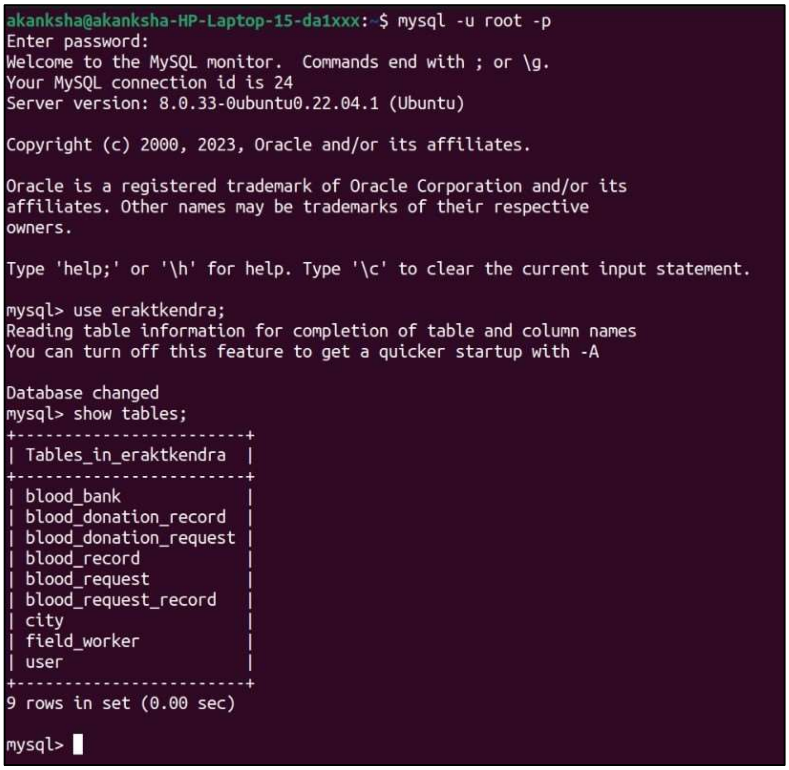
Pipeline E-RaktKendra Frontend

Stage View

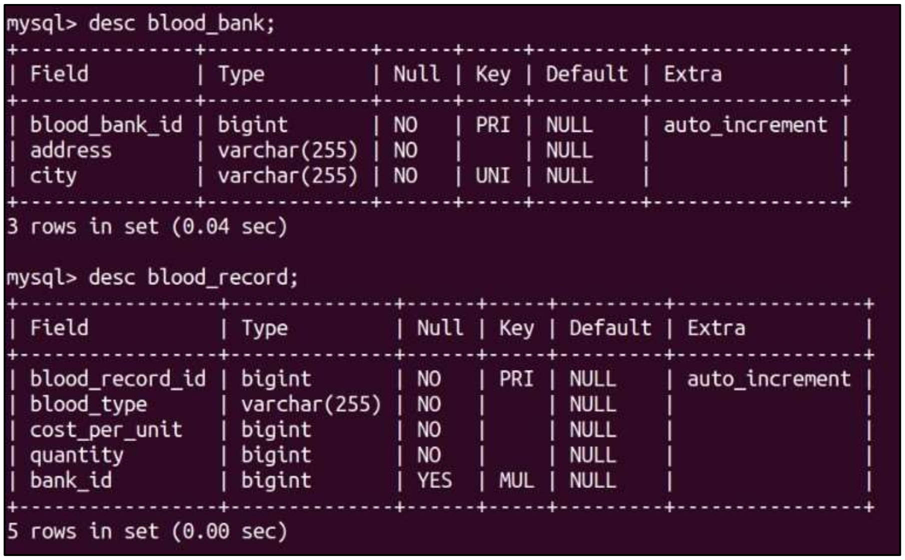
	Declarative: Checkout SCM	git clone frontend	Docker Build Image	Push Docker Image to Docker Hub
Average stage times: (Average full run time: ~10min 11s)	3s	2s	3min 55s	26min 27s
#13 May 13 18:39 No Changes	3s	3s	14s	12min 15s
#12 May 13 16:21 No Changes	2s	2s	12s	40min 32s failed

4.8 Database

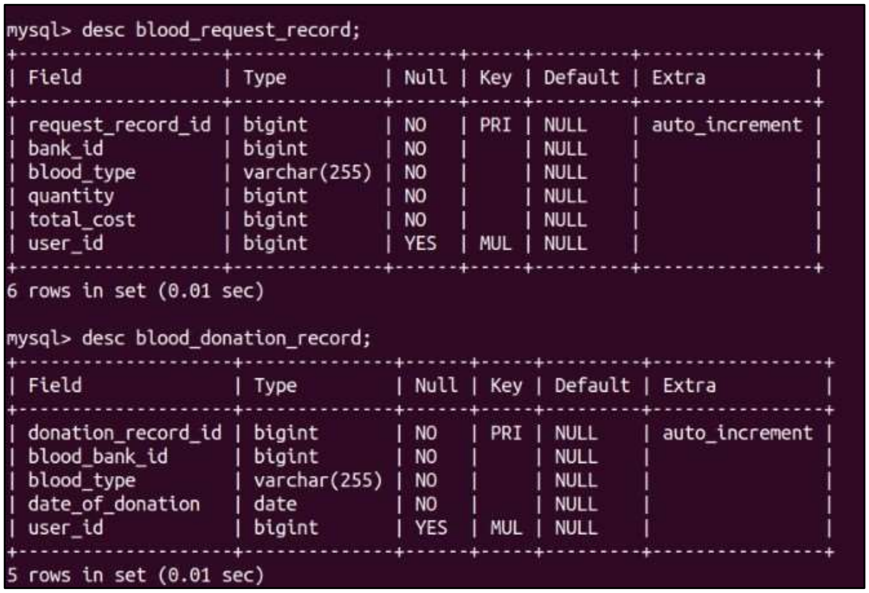
Our E-RaktKendra is using MySQL database. MySQL is an open-source relational database management system. It is widely used in web applications and is known for its high performance, reliability, and ease of use. MySQL is compatible with many operating systems and programming languages, and it provides various tools for database management and administration.



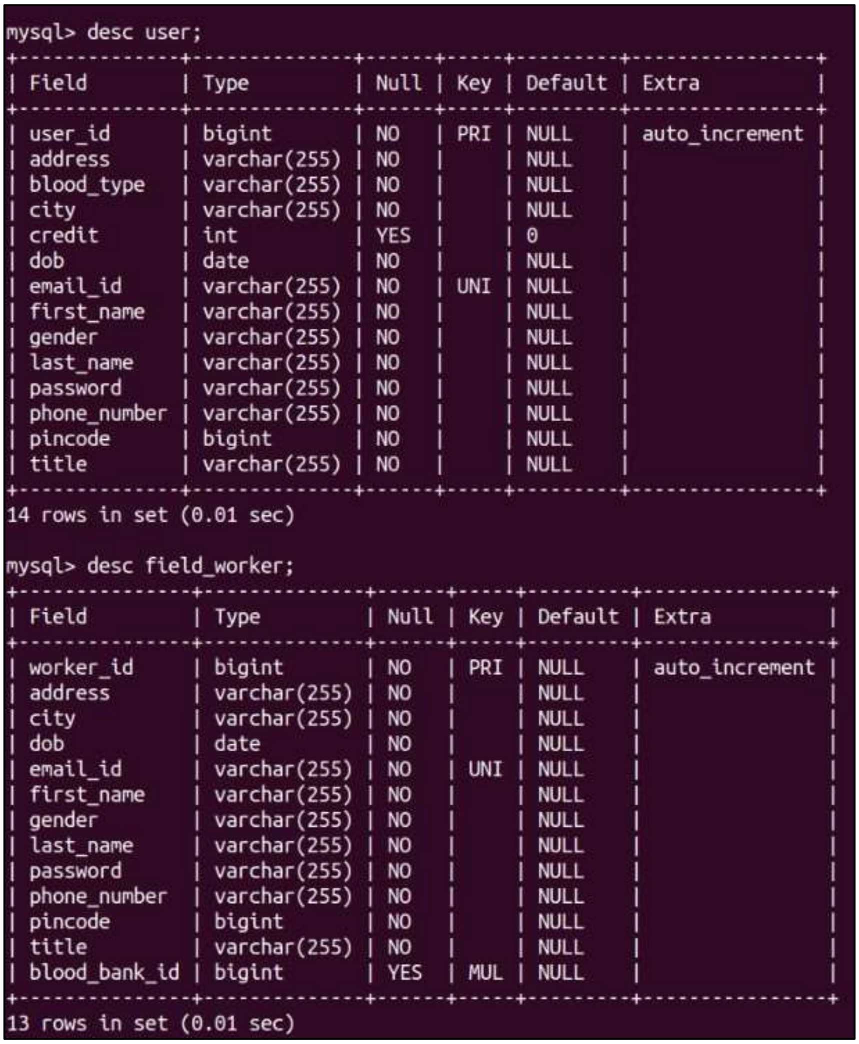
```
akanksha@akanksha-HP-Laptop-15-da1xxx:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 24
Server version: 8.0.33-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use eraktkendra;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+------------------------+
| Tables_in_eraktkendra  |
+------------------------+
| blood_bank             |
| blood_donation_record  |
| blood_donation_request |
| blood_record           |
| blood_request          |
| blood_request_record   |
| city                   |
| field_worker           |
| user                   |
+------------------------+
9 rows in set (0.00 sec)

mysql> 
```



```
mysql> desc blood_bank;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| blood_bank_id | bigint       | NO   | PRI | NULL    | auto_increment |
| address       | varchar(255) | NO   |     | NULL    |                |
| city          | varchar(255) | NO   | UNI | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
3 rows in set (0.04 sec)

mysql> desc blood_record;
+-----------------+--------------+------+-----+---------+----------------+
| Field           | Type         | Null | Key | Default | Extra          |
+-----------------+--------------+------+-----+---------+----------------+
| blood_record_id | bigint       | NO   | PRI | NULL    | auto_increment |
| blood_type      | varchar(255) | NO   |     | NULL    |                |
| cost_per_unit   | bigint       | NO   |     | NULL    |                |
| quantity        | bigint       | NO   |     | NULL    |                |
| bank_id         | bigint       | YES  | MUL | NULL    |                |
+-----------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```



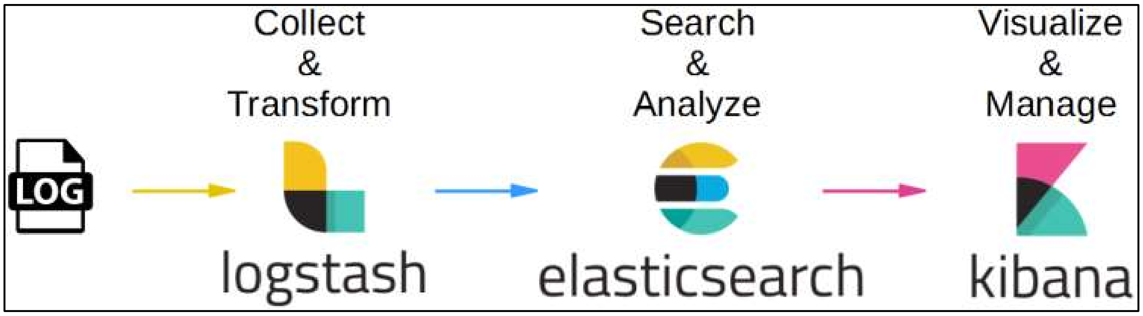
```
mysql> desc blood_request_record;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| request_record_id | bigint       | NO   | PRI | NULL    | auto_increment |
| bank_id           | bigint       | NO   |     | NULL    |                |
| blood_type        | varchar(255) | NO   |     | NULL    |                |
| quantity          | bigint       | NO   |     | NULL    |                |
| total_cost        | bigint       | NO   |     | NULL    |                |
| user_id           | bigint       | YES  | MUL | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)

mysql> desc blood_donation_record;
+--------------------+--------------+------+-----+---------+----------------+
| Field              | Type         | Null | Key | Default | Extra          |
+--------------------+--------------+------+-----+---------+----------------+
| donation_record_id | bigint       | NO   | PRI | NULL    | auto_increment |
| blood_bank_id      | bigint       | NO   |     | NULL    |                |
| blood_type         | varchar(255) | NO   |     | NULL    |                |
| date_of_donation   | date         | NO   |     | NULL    |                |
| user_id            | bigint       | YES  | MUL | NULL    |                |
+--------------------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```



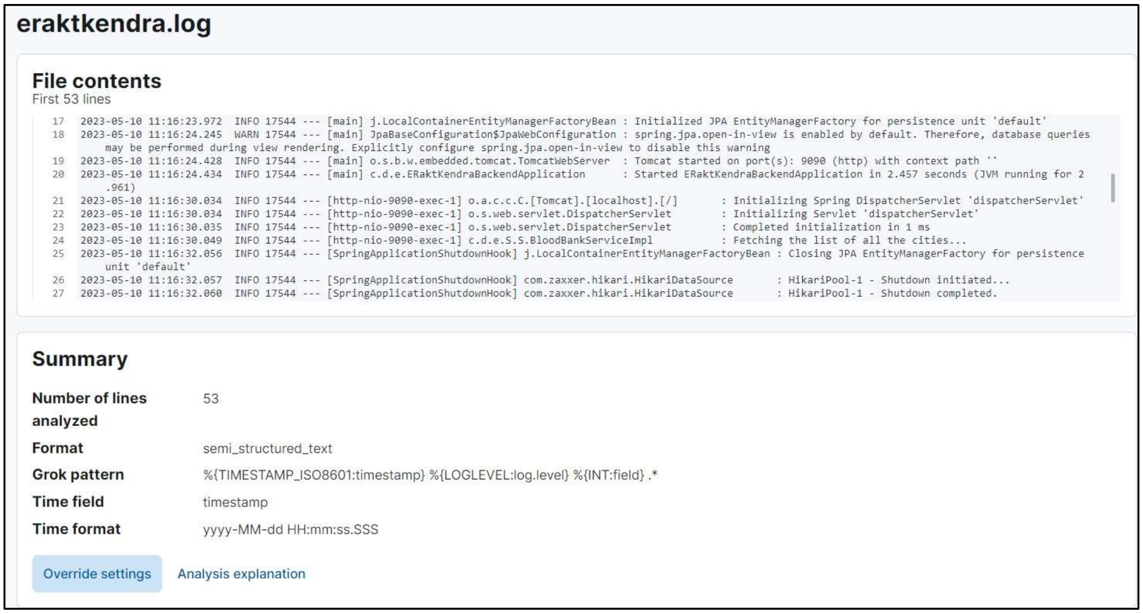
```
mysql> desc user;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| user_id      | bigint       | NO   | PRI | NULL    | auto_increment |
| address      | varchar(255) | NO   |     | NULL    |                |
| blood_type   | varchar(255) | NO   |     | NULL    |                |
| city         | varchar(255) | NO   |     | NULL    |                |
| credit       | int          | YES  |     | 0       |                |
| dob          | date         | NO   |     | NULL    |                |
| email_id     | varchar(255) | NO   | UNI | NULL    |                |
| first_name   | varchar(255) | NO   |     | NULL    |                |
| gender       | varchar(255) | NO   |     | NULL    |                |
| last_name    | varchar(255) | NO   |     | NULL    |                |
| password     | varchar(255) | NO   |     | NULL    |                |
| phone_number | varchar(255) | NO   |     | NULL    |                |
| pincode      | bigint       | NO   |     | NULL    |                |
| title        | varchar(255) | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
14 rows in set (0.01 sec)

mysql> desc field_worker;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| worker_id     | bigint       | NO   | PRI | NULL    | auto_increment |
| address       | varchar(255) | NO   |     | NULL    |                |
| city          | varchar(255) | NO   |     | NULL    |                |
| dob           | date         | NO   |     | NULL    |                |
| email_id      | varchar(255) | NO   | UNI | NULL    |                |
| first_name    | varchar(255) | NO   |     | NULL    |                |
| gender        | varchar(255) | NO   |     | NULL    |                |
| last_name     | varchar(255) | NO   |     | NULL    |                |
| password      | varchar(255) | NO   |     | NULL    |                |
| phone_number  | varchar(255) | NO   |     | NULL    |                |
| pincode       | bigint       | NO   |     | NULL    |                |
| title         | varchar(255) | NO   |     | NULL    |                |
| blood_bank_id | bigint       | YES  | MUL | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
13 rows in set (0.01 sec)
```

4.9 ELK

ELK is an acronym that stands for Elasticsearch, Logstash, and Kibana. These three tools are often used together as a unified platform for managing and analyzing large amounts of data, particularly log data. Elasticsearch is a search and analytics engine that is used to store and index data, while Logstash is a tool used to collect, process, and transform data from various sources, such as logs, metrics, and other events. Kibana is a data visualization and exploration tool that provides a web interface for searching, analyzing, and visualizing data stored in Elasticsearch. Together, ELK provides a powerful solution for managing and analyzing data at scale.



ELK stack architecture



eraktkendra.log

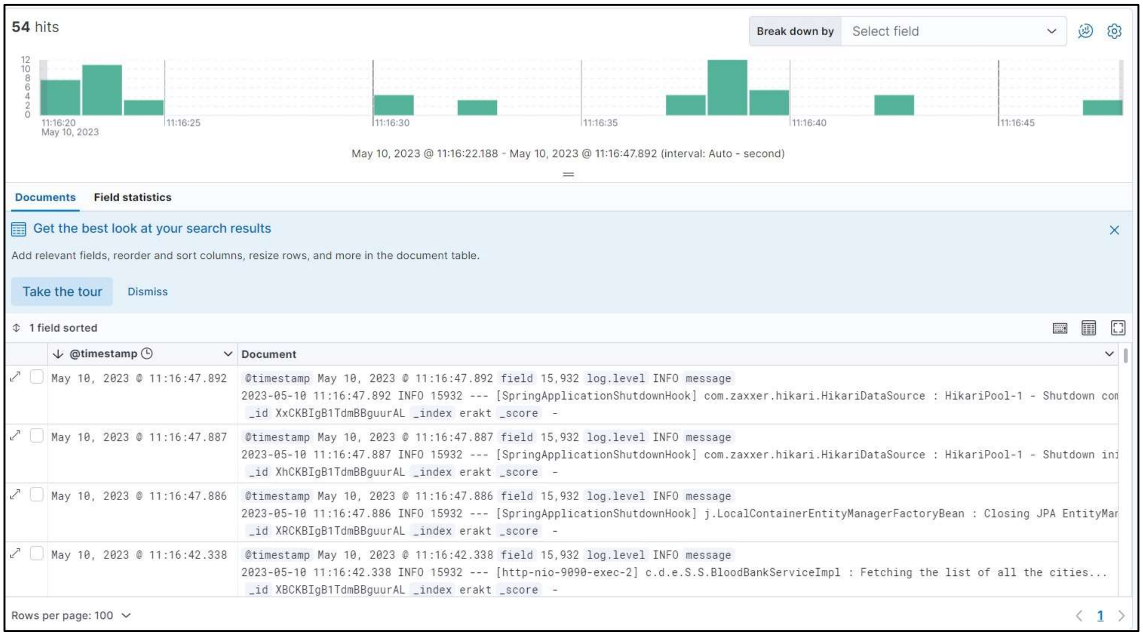
File contents

First 53 lines

```
2023-05-10 11:16:23.972  INFO 17544 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-05-10 11:16:24.245  WARN 17544 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries
    may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2023-05-10 11:16:24.428  INFO 17544 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 9090 (http) with context path ''
2023-05-10 11:16:24.434  INFO 17544 --- [main] c.d.e.ERaktKendraBackendApplication      : Started ERaktKendraBackendApplication in 2.457 seconds (JVM running for 2
    .961)
2023-05-10 11:16:30.034  INFO 17544 --- [http-nio-9090-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-05-10 11:16:30.034  INFO 17544 --- [http-nio-9090-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2023-05-10 11:16:30.035  INFO 17544 --- [http-nio-9090-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2023-05-10 11:16:30.049  INFO 17544 --- [http-nio-9090-exec-1] c.d.e.S.S.BloodBankServiceImpl           : Fetching the list of all the cities...
2023-05-10 11:16:32.056  INFO 17544 --- [SpringApplicationShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence
    unit 'default'
2023-05-10 11:16:32.057  INFO 17544 --- [SpringApplicationShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
2023-05-10 11:16:32.060  INFO 17544 --- [SpringApplicationShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Summary

Number of lines analyzed	53
Format	semi_structured_text
Grok pattern	%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:log.level} %{INT:field} .*
Time field	timestamp
Time format	yyyy-MM-dd HH:mm:ss.SSS



May 10, 2023 @ 11:16:22.188 - May 10, 2023 @ 11:16:47.892 (interval: Auto - second)

Get the best look at your search results

Add relevant fields, reorder and sort columns, resize rows, and more in the document table.

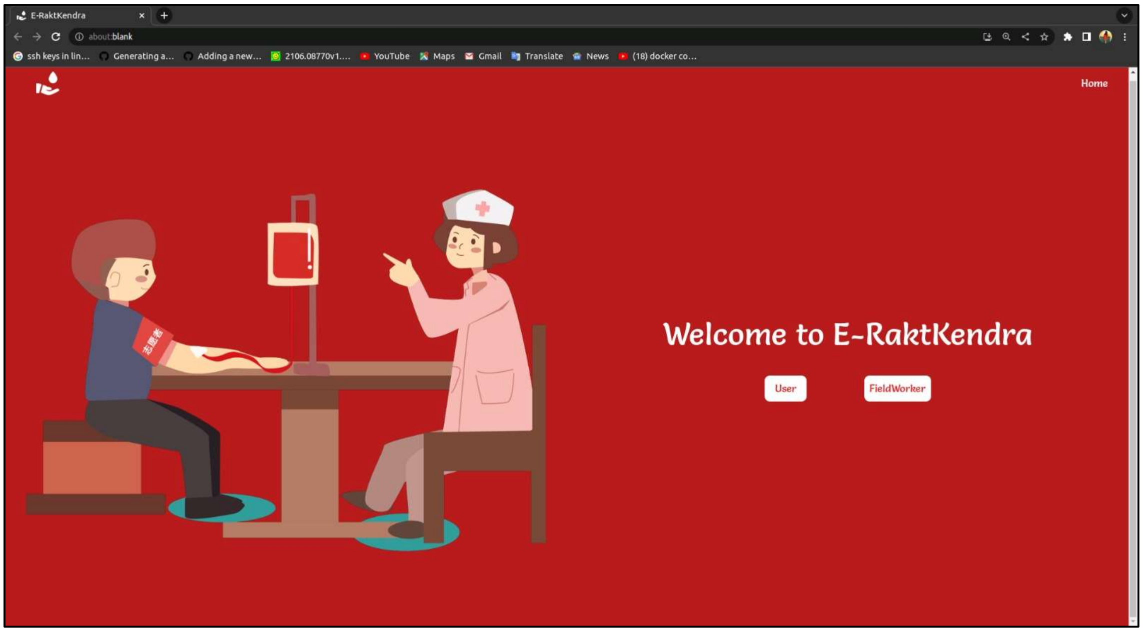
1 field sorted

@timestamp	Document
May 10, 2023 @ 11:16:47.892	@timestamp May 10, 2023 @ 11:16:47.892 field 15,932 log.level INFO message 2023-05-10 11:16:47.892 INFO 15932 --- [SpringApplicationShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown com _id XxCKBIgB1TdmBBguurAL _index erakt _score -
May 10, 2023 @ 11:16:47.887	@timestamp May 10, 2023 @ 11:16:47.887 field 15,932 log.level INFO message 2023-05-10 11:16:47.887 INFO 15932 --- [SpringApplicationShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown ini _id XhCKBIgB1TdmBBguurAL _index erakt _score -
May 10, 2023 @ 11:16:47.886	@timestamp May 10, 2023 @ 11:16:47.886 field 15,932 log.level INFO message 2023-05-10 11:16:47.886 INFO 15932 --- [SpringApplicationShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityMar _id XRCKBIgB1TdmBBguurAL _index erakt _score -
May 10, 2023 @ 11:16:42.338	@timestamp May 10, 2023 @ 11:16:42.338 field 15,932 log.level INFO message 2023-05-10 11:16:42.338 INFO 15932 --- [http-nio-9090-exec-2] c.d.e.S.S.BloodBankServiceImpl : Fetching the list of all the cities... _id XBCKBIgB1TdmBBguurAL _index erakt _score -

5 . Our Application : E-RaktKendra

Project screenshots and working :

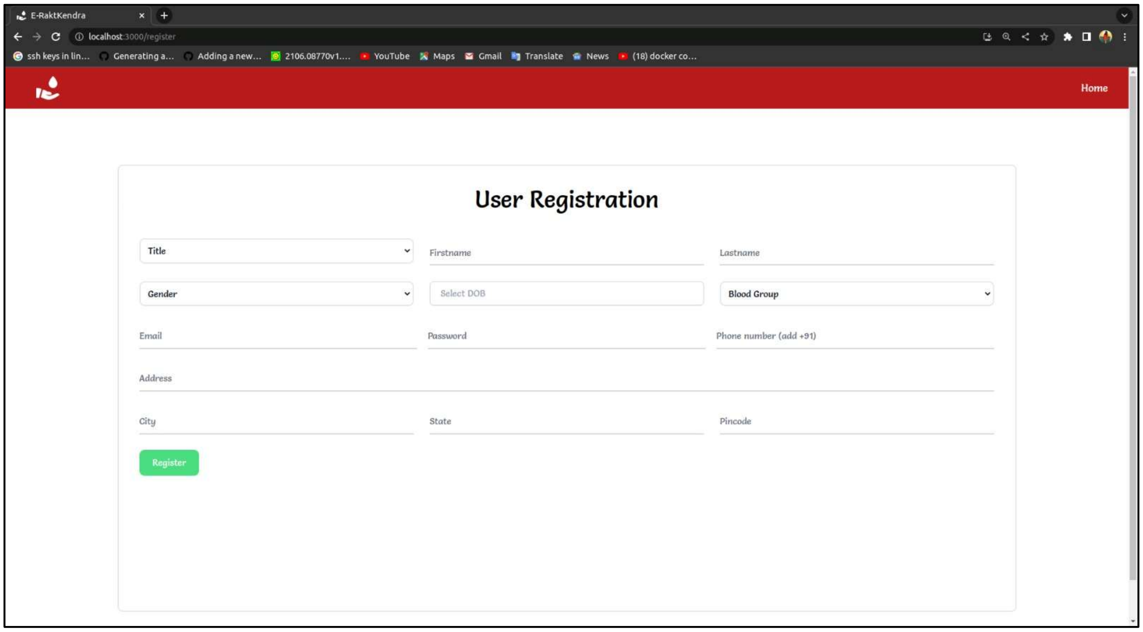
HomePage:



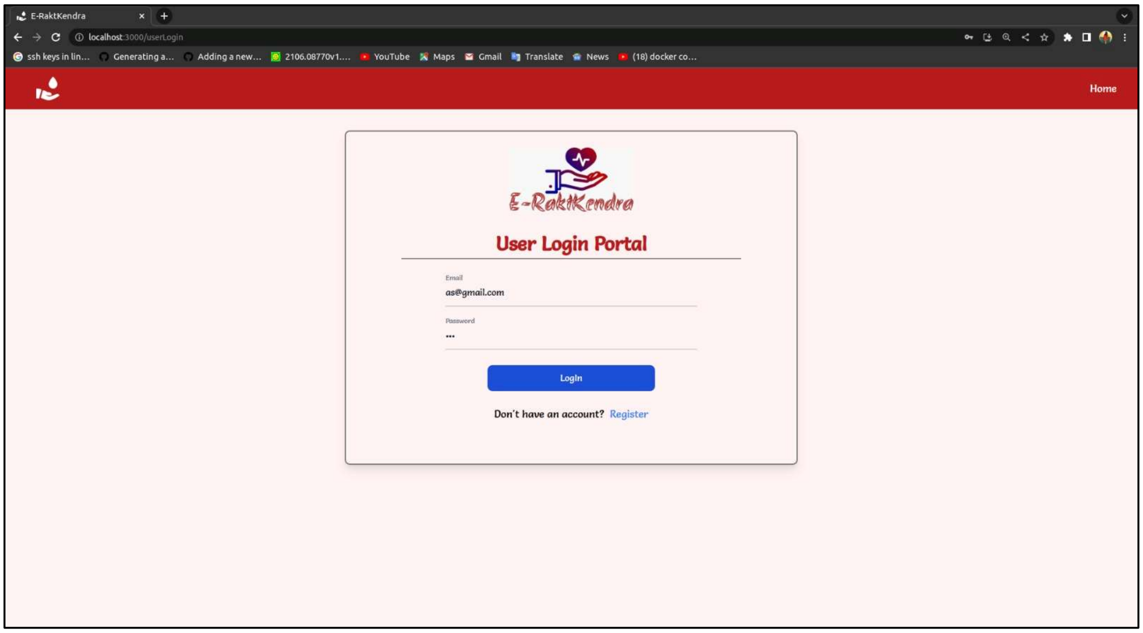
Welcome to E-RaktKendra

User :

User Registration :



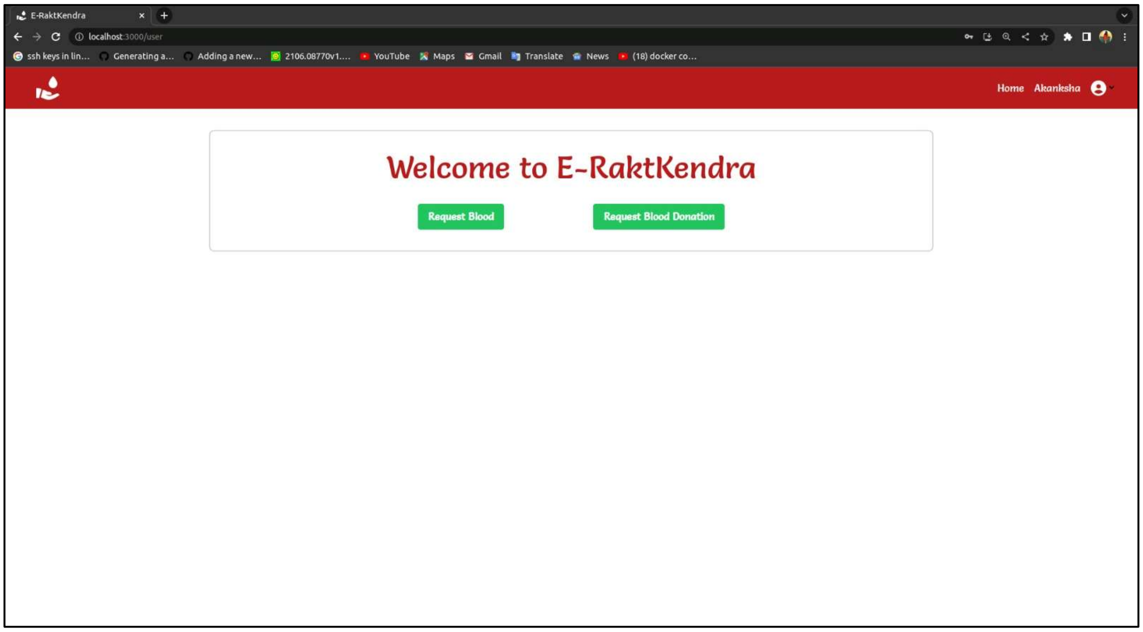
User Login :



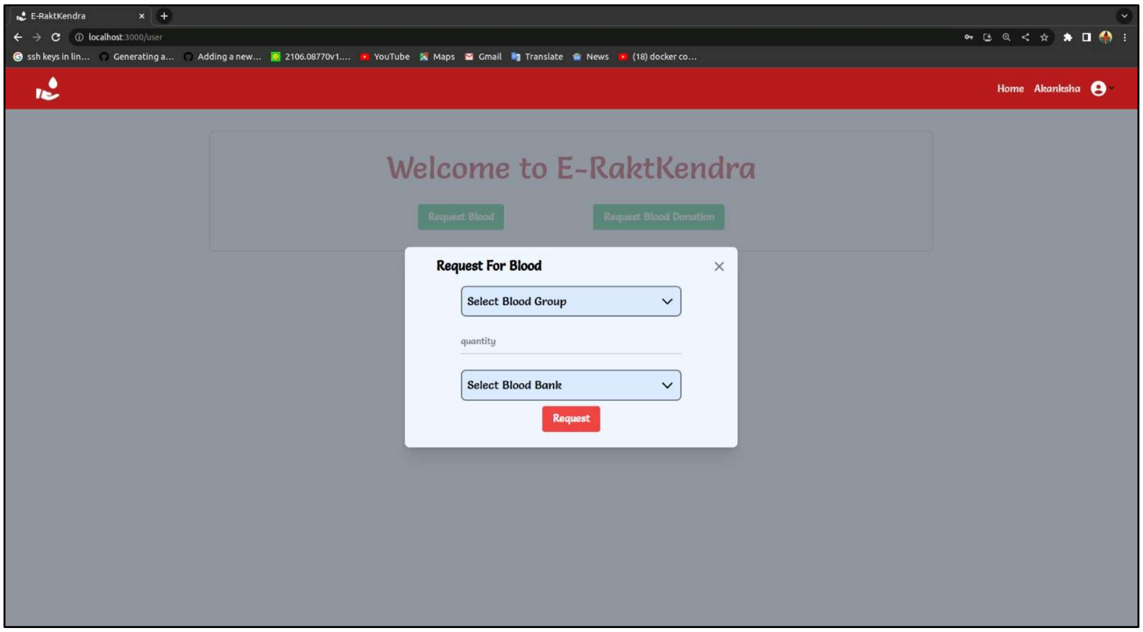
Email
as@gmail.com

Password
•••

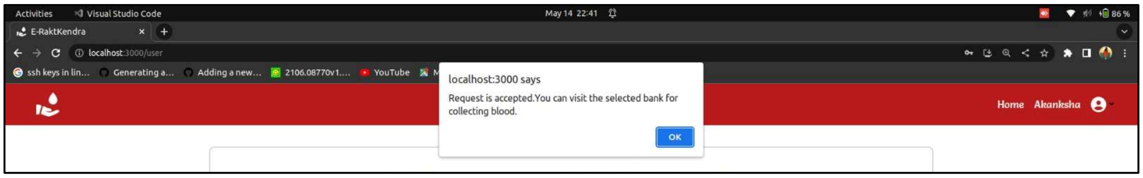
User Dashboard :



User Request for blood type :



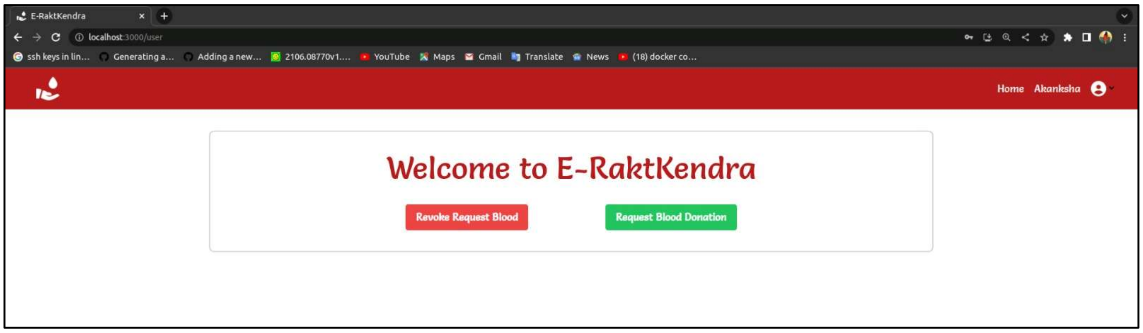
User Request Approved:



localhost:3000 says

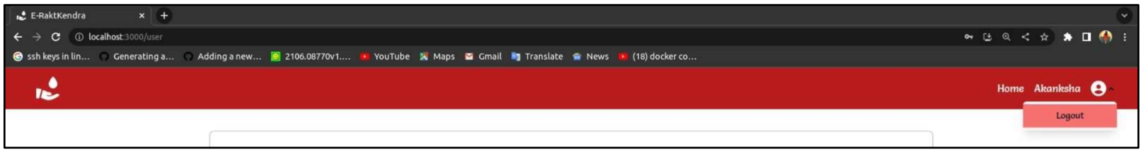
Request is accepted.You can visit the selected bank for collecting blood.

User cannot make more than one request at a time. First he has to revoke his previous request then only he can make any new requests.



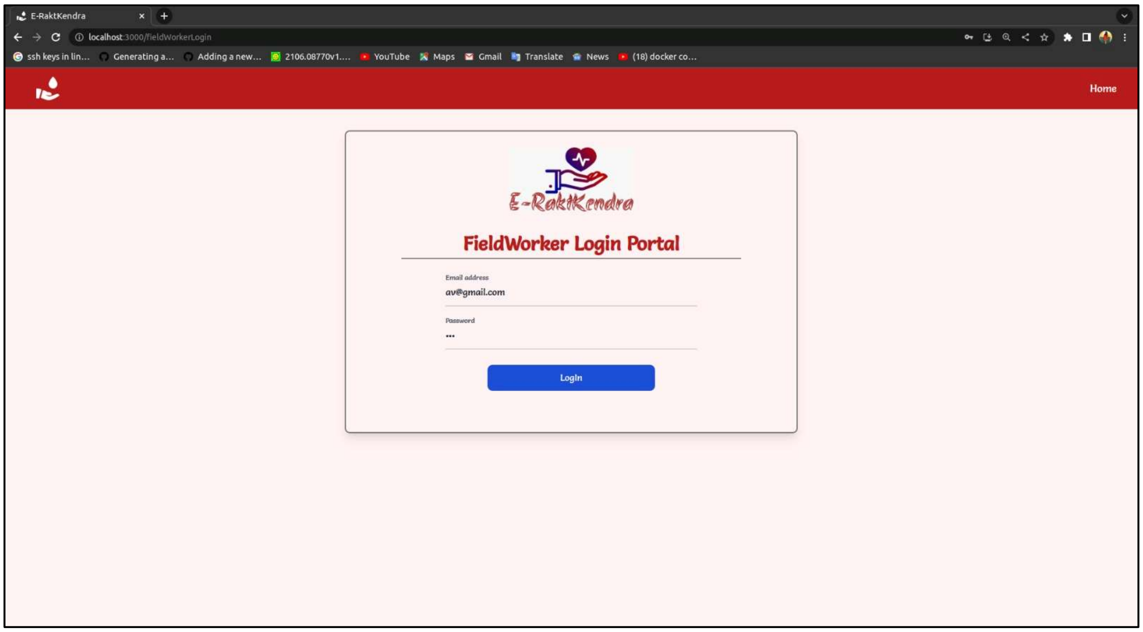
Same way user blood donation request work too.

User logout :



Fieldworker :

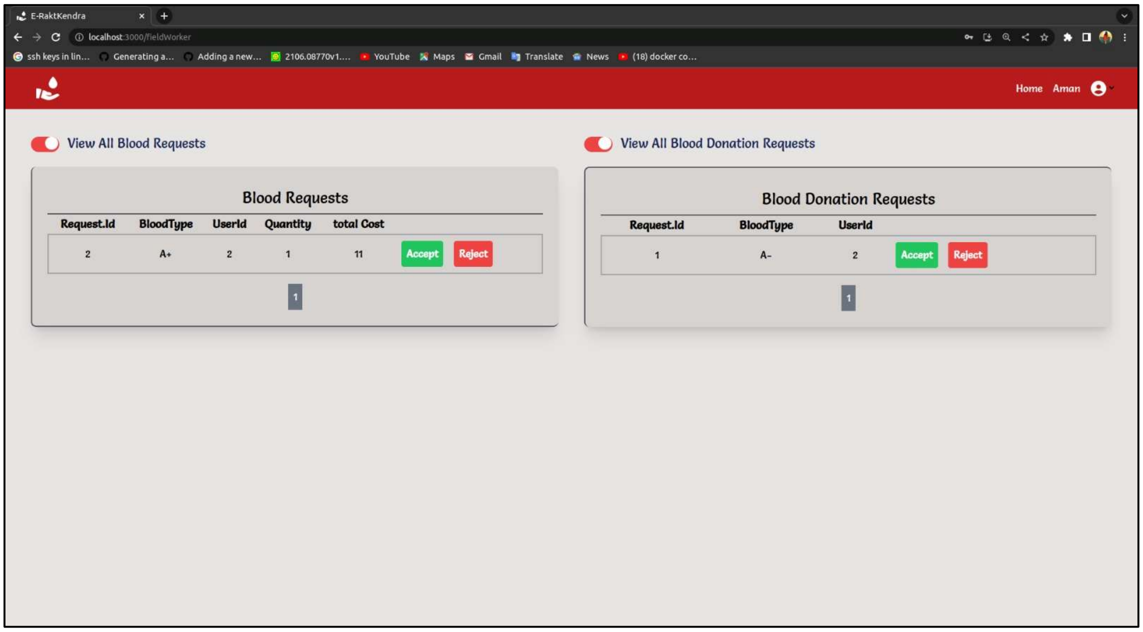
Fieldworker Login :



Email address
av@gmail.com

Password
•••

Fieldworker Dashboard : he can view all blood request/blood donation request for his respective bank and can accept/reject a request.



View All Blood Requests

Blood Requests

Request.Id	BloodType	UserId	Quantity	total Cost		
2	A+	2	1	11	Accept	Reject

1

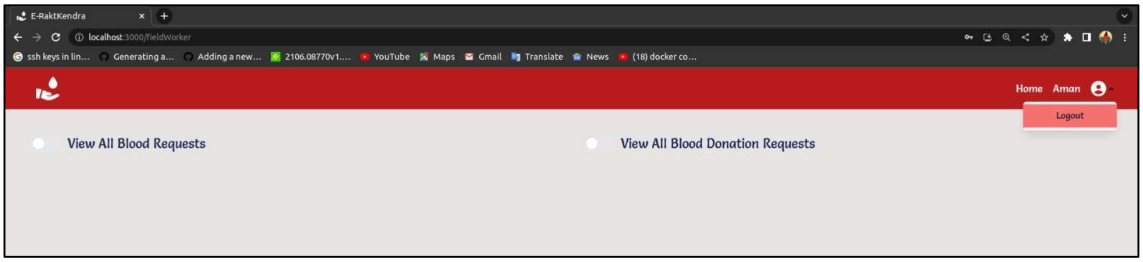
View All Blood Donation Requests

Blood Donation Requests

Request.Id	BloodType	UserId		
1	A-	2	Accept	Reject

1

Fieldworker logout :



View All Blood Requests

View All Blood Donation Requests

6. Scope and Future Work

Our project is equipped with spring JWT security and has very few functionalities. In future we will add more functionalities like user will get a receipt in the form of pdf after making request. And will add more functionalities on fieldworker side.

We are generating ELK logs manually. We will try to generate logs automatically.

Due to our system configuration we were unable to do Kubernates. We will try to do that too.

And we need to work more on our devops pipeline.

7. Conclusion

Our E-RaktKendra is built successfully with JWT security, in which user can request for blood type and make request for blood donation.

Tools used : GitHub , Jenkins , Docker , Ansible.

Our API Documentation is provided below :

API Documentation

End Point	HTTP method	Input	Description
/getCitiesAndBankId	GET	None	Gives the list of all cities and their ids.
/bloodDonationRequest	POST	bankId, blood type, userId	Generates a request for blood donation in respective blood bank
/revokeBloodDonationRequest/{userId}	DELETE	userId	Deletes user's blood donation request
/checkBloodDonationRequest/{userId}	GET	userId	Checks if any blood donation request exists for the user or not
/acceptBloodDonationRequest/{donationRequestId}	DELETE	DonationRequestId	Deletes donation request and adds it to the donation record table
/getAllBloodDonationRequests/{bankId}	GET	bankId	Get list of all the blood donation request for the particular bank
/getBloodDonationHistory/{userId}	GET	userId	Get the list of all the blood donation user has done.
/bloodRequestByUser	POST	bloodbankId, userId, bloodType, quantity	Generates a request for blood from the particular bloodBank
/checkBloodRequest/{userId}	GET	userId	Check if blood request exists or not for the particular user
/revokeBloodRequest/{userId}	DELETE	userId	Deletes user bloodRequest
/acceptBloodRequest/{bloodRequestId}	DELETE	bloodRequestId	Accepts blood request and deletes it from blood request table

/getAllBloodRequests/{bankId}	GET	bankId	Gives the list of all blood request of a particular bank
/registerFieldWorker	POST	fieldworker details	Register the field worker
/generateToken	GET	Username, password, role	Generates JWT token and gives the detail according to the role.
/adduser	POST	UserDetails	Register a user.