

Project Report for Plasma Donor Database Management

Team number: 384

Members:

Rohit Krishnan [20BPS1045]

Christin T Kunjumon [20BRS1005]

Shivamm Warambhey [20BPS1134]

Vyshnavi Sunil Kumar [20BPS1132]

1 INTRODUCTION

1.1 Overview

The Aim of this project is to create an application which contains and maintains the information about donors who have donated their plasma, and also a safe haven for the people in dire need of this. It has login facilities, which according to the user will provide different authorizations permissions. The information about donors is given to such users of the application. Users can also sign up for donations themselves.

1.2 Purpose

This project is aimed at the creation of an application which can act as a single stop for all the donors as well as people who are in search of donors, to come in contact with each other and save lives. This can save a lot of middlemen that are normally required to get in contact with a person who is ready to donate their plasma. Since not a lot of people are not well taught about plasma donation, as with blood donation, the number of people who sign up for this already less, and the existence of such middlemen make it more difficult for the people in need of it to find such donors. Thus this application acts as a hub for those people.

2 LITERATURE SURVEY

2.1 Existing problem

The existing problem for the current Plasma donor system is that it is very hospital centric. That is you have to consult with the hospital to get in contact with a potential donor. Yes, the process of the donation requires the help of hospitals, but the people who are in urgent need of plasma, will not have time to go through all the formalities that the hospital has to undergo, in finding a plasma donor.

2.2 Proposed solution

Thus to prevent the wastage of time that the hospital takes to find potential donor, which is also only among the limited resources and information about donors that they have, we have tried to make an application which donor centralized, that is anyone who wants to register as a donor can do so, and the people in need of plasma can browse this database and directly contact the donor, saving a lot of time and cost.

3 SOFTWARE USED

1-Java Development Kit (JDK): JDK is required to compile and run Java applications, providing the necessary tools and libraries. Download and install the latest JDK version from Oracle's website.

-DownloadJDK:

<https://www.oracle.com/java/technologies/javase-jdk11-downloads.html>

2-Integrated Development Environment (IDE): An IDE offers a comprehensive development environment for writing, debugging, and managing code. IntelliJ IDEA, Eclipse, or Visual Studio Code are popular choices for Java development.

- IntelliJ IDEA: <https://www.jetbrains.com/idea/download/>

- Eclipse: <https://www.eclipse.org/downloads/>

- Visual Studio Code: <https://code.visualstudio.com/download>

3-Spring Boot: Spring Boot simplifies Java application development by providing predefined configurations, automatic dependency management, and a streamlined development experience. Use Spring Initializr or Spring Tools for your IDE to create a Spring Boot project.

- Spring Initializr (Online): <https://start.spring.io/>

- Spring Tools 4 for Eclipse: <https://spring.io/tools>
- Spring Tools for Visual Studio Code: Install via Extensions in Visual Studio Code

4-MongoDB: MongoDB is a NoSQL document database. Install MongoDB Community Edition or use MongoDB Atlas, a cloud-based service for managing MongoDB databases.

- MongoDB Community Edition:

<https://www.mongodb.com/try/download/community>

- MongoDB Compass:

<https://www.mongodb.com/try/download/compass>

5-MongoDB Java Driver: The MongoDB Java Driver allows Java applications to connect and interact with MongoDB databases. Include this dependency to enable MongoDB integration in your Java Spring Boot project.

- Maven:

- Add the following dependency to your project's pom.xml:

xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-mongodb</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
```

```

        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </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>
</dependencies>

```

Make sure to download and install the appropriate versions based on your system and project requirements. With these prerequisites in place, you'll be ready to start building your Java Spring Boot project with both MySQL or MongoDB as the database.

Role Based Access:-

Roles of Admin and User can be defined for book a doctor application:

1-Admin Role:

- **Responsibilities:** The admin role has full control and administrative privileges over the system.
- **Permissions:**
 - Manage donors: Admins can add, edit, and delete donor information.
 - Manage users: Admins can create, edit, and delete user accounts, as well as manage their roles and permissions.
 - Manage the donations information: Admins have access to information about the donors, and the specifics of the donation. This can help them alter if any donor wants to donate, but the specifics of the previous donations does not approve for the current one.

2-User Role:

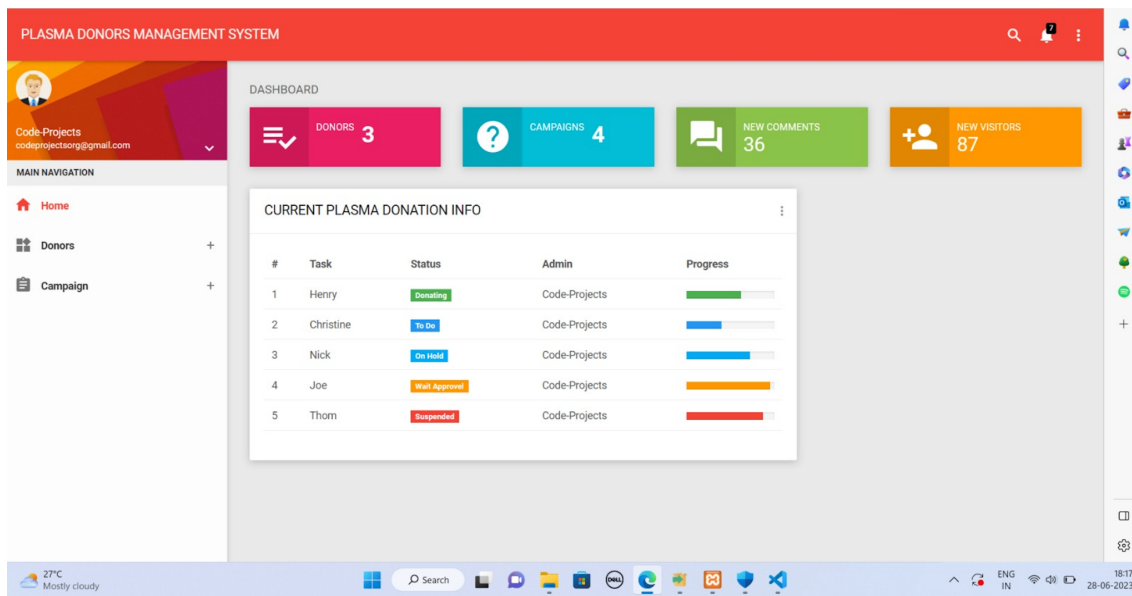
- **Responsibilities:** Users are the people who don't have the administrative permissions, but are able to track about the donations they have made, or to search and contact anyone available for donations.
- **Permissions:**
 - View donors: Users can search for donors, view their contact details, Plasma type and other details.
 - Contact donors: With the contact section of the donors, we are able to provide the people in need of a donor, to directly contact the donors.
 - View personal profile: Users can view and edit their own profile information, such as contact details, address, and other personal information.

Project Flow:

- Frontend Development
- Backend Development
- Integration
- Containerization

Milestone 1: Frontend Development:

Frontend development involves building the user interface (UI) and implementing the visual elements of the plasma donor application. It focuses on creating an intuitive and engaging user experience that allows users to interact with the application seamlessly. The following activities are part of the frontend development process:



Activity 1: UI Design and Layout

- Design the overall user interface (UI) for the plasma donors management system.
- Create wireframes and mockups to visualize the layout and structure of the application.
- Determine the color scheme, typography, and overall visual style.
- Implement responsive design to ensure the application is compatible with different devices.

Activity 2: ADD , View , Edit and Remove Donor Information

- Develop the donor search functionality, allowing users to search for plasma donors on criteria such as blood group.
- Implement the donor listing page, displaying relevant donor options with details such as group and their details.

The screenshot displays the 'ADD DONOR' form within the 'PLASMA DONORS MANAGEMENT SYSTEM'. The interface includes a sidebar with navigation options: Home, Donors (with sub-options: Add Donors, View Donors, Edit Donors, Remove Donors), and Campaigns. The main form area contains input fields for Full Name, Gender, Date of birth (dd-mm-yyyy), Weight, Contact, and Blood Type, followed by a 'Submit' button. The system is running on a Windows 10 desktop environment, as indicated by the taskbar and system tray.

Activity 3: User Authentication and Profile UI

- Develop the campaign search functionality, allowing users to search for plasma donors campaign.
- Implement the campaign listing page, displaying relevant plasma campaign with details such as dates , location and their details.

The screenshot displays the 'DASHBOARD' of the 'PLASMA DONORS MANAGEMENT SYSTEM'. The dashboard features a sidebar with navigation options: Home, Donors, and Campaign (with sub-options: Create Campaign, Edit Campaign Details, View Campaign, Remove Campaign). The main content area includes a 'DASHBOARD' section with four summary cards: DONORS (3), CAMPAIGNS (4), NEW COMMENTS (36), and NEW VISITORS (87). Below these cards is a 'CURRENT PLASMA DONATION INFO' table.

#	Task	Status	Admin	Progress
1	Henry	Donating	Code-Projects	<div><div></div></div>
2	Christine	To do	Code-Projects	<div><div></div></div>
3	Nick	On Hold	Code-Projects	<div><div></div></div>
4	Joe	Wait Approval	Code-Projects	<div><div></div></div>
5	Thom	Suspended	Code-Projects	<div><div></div></div>

Milestone 2: Backend Development:

Backend development involves building the server-side components and logic of the doctor booking application. It focuses on handling the business logic, processing requests from the frontend, and interacting with the database. The following activities are part of the backend development process:

Activity 1: Server Setup and API Development

- Set up the server environment and choose a suitable backend framework such as Java Spring Boot.
- Develop the RESTful APIs for doctor search, booking management, user authentication, and profile management.

```
if(credentialsrepository.findByemail(email)!=null && credentialsrepository.findBypassword(password)!=null)
{
    if (credentialsrepository.findByemail(email).user.id.equalsIgnoreCase(credentialsrepository.findBypassword(password).user.id) )
    {
        user_global=credentialsrepository.findByemail(email).user;
        System.out.println("You have successfully logged in!!");
    }
    else
    {
        System.out.println("Email or Password is invalid. ");
        {
            System.out.println("Please do retry!!");
        }
    }
}
```

```
else
{
    System.out.println("No such login found.");
    System.out.println("Do you want to register yourself in the website(Y/N)?");
    String choice=obj.readLine();

    if(choice.equalsIgnoreCase("Y"))
    {
        {
            System.out.print("Enter the First Name: ");
            firstname=obj.readLine();
            System.out.print("Enter the Last Name: ");
            lastname=obj.readLine();
            System.out.print("Enter the Blood Type: ");
            bloodtype=obj.readLine();
            role="u";
            System.out.print("Enter the Address: ");
            address=obj.readLine();
            System.out.print("Enter the Phone Number: ");
            ph_no=obj.readLine();
            System.out.print("Enter the Email: ");
            email=obj.readLine();
            System.out.print("Enter the Password: ");
            password=obj.readLine();
            User user=new User(firstname, lastname/*,role*/,bloodtype);
            userrepository.save(user);
            Contact contact=new Contact(user,address,ph_no);
            contactrepository.save(contact);
            Credentials credentials=new Credentials(user, email, password);
            credentialsrepository.save(credentials);
            userAuthority userauthority=new userAuthority(user,role);
            userauthorityrepository.save(userauthority);

            user_global=user;
        }
    }
}
```



```

if (userauthorityrepository.findByuser(user_global).role.equalsIgnoreCase("a"))
{
    System.out.println("What is your next operation:");
    System.out.println("1. Add a user");
    System.out.println("2. Print all the users");
    System.out.println("3. Get info on a particular user with the first name");
    System.out.println("4. Get info on a particular user with the last name");
    System.out.println("5. Get info on a particular user with the blood type");
    System.out.println("6. Delete the record of a user");
    System.out.println("7. Change the role of a user");
    System.out.println("8. Add yourself as a donor");
    System.out.println("9. Exit");

    selection=Integer.parseInt(obj.readLine());
}

```

```

switch(selection)
{
    case 1:
        if (userauthorityrepository.findByuser(user_global).role.equalsIgnoreCase("a"))
        {
            System.out.print("Enter the first name: ");
            firstname=obj.readLine();
            System.out.print("Enter the last name: ");
            lastname=obj.readLine();
            System.out.print("Enter the blood type: ");
            bloodtype=obj.readLine();
            System.out.print("Enter the role (a=admin, u=normal user): ");
            role=obj.readLine();
            System.out.print("Enter the address: ");
            address=obj.readLine();
            System.out.print("Enter the phone number: ");
            ph_no=obj.readLine();
            User user=new User(firstname, lastname+"/"+role+"/"+bloodtype);
            userrepository.save(user);
            Contact contact=new Contact(user,address,ph_no);
            contactrepository.save(contact);
            userAuthority userauthority=new userAuthority(user,role);
            userauthorityrepository.save(userauthority);
        }
        else
        {
            System.out.println("Error: No permission to access!");
            System.out.println();
            continue;
        }
        break;
}

```

```

case 2:
    if (userauthorityrepository.findByuser(user_global).role.equalsIgnoreCase("a"))
    {
        System.out.println("Customers found with findAll():");
        System.out.println("-----");
        for (User user : userrepository.findAll())
        {
            System.out.println(user);
            System.out.println(contactrepository.findByuser(user));
            System.out.println(userauthorityrepository.findByuser(user));
        }

        System.out.println();
    }
    else
    {
        System.out.println("Error: No permission to access!");
        System.out.println();
        continue;
    }
    break;
}

```

```

case 3:
    if (userauthorityrepository.findByuser(user_global).role.equalsIgnoreCase("a"))
    {
        System.out.print("Enter the first name: ");
        firstname=obj.readLine();
        System.out.println("Customer found with findByFirstName("+firstname+")");
        System.out.println("-----");
        for (User user: userrepository.findByFirstName(firstname))
        {
            System.out.println(user);
            System.out.println(contactrepository.findByuser(user));
        }

        System.out.println();
    }
    else
    {
        System.out.println("Error: No permission to access!");
        System.out.println();
        continue;
    }
    break;
}

```

```

case 4:
if (userauthorityrepository.findByuser(user_global).role.equalsIgnoreCase("a"))
{
    System.out.print("Enter the last name: ");
    lastname=obj.readLine();
    System.out.println("Customers found with findByLastName(\"+lastname+\"):");
    System.out.println("-----");
    for (User user : userrepository.findBylastName(lastname))
    {
        System.out.println(user);
        System.out.println(contactrepository.findByuser(user));
    }
}
else
{
    System.out.println("Error: No permission to access!");
    System.out.println();
    continue;
}
break;

```

```

case 5:
if (userauthorityrepository.findByuser(user_global).role.equalsIgnoreCase("a"))
{
    System.out.print("Enter the blood type: ");
    bloodtype=obj.readLine();
    System.out.println("Urs found with the blood type: "+bloodtype);
    System.out.println("-----");
    for (User user: userrepository.findBybloodType(bloodtype))
    {
        System.out.println(user);
        System.out.println(contactrepository.findByuser(user));
    }
    System.out.println();
}
else
{
    System.out.println("Error: No permission to access!");
    System.out.println();
    continue;
}
break;

```

```

case 6:
if (userauthorityrepository.findByuser(user_global).role.equalsIgnoreCase("a"))
{
    System.out.print("Enter the first name: ");
    firstname=obj.readLine();

    for (User user: userrepository.findByfirstName(firstname))
    {
        userrepository.deleteByfirstName(firstname);
        contactrepository.deleteByuser(user);
    }
}
else
{
    System.out.println("Error: No permission to access!");
    System.out.println();
    continue;
}
break;
case 7:
if (userauthorityrepository.findByuser(user_global).role.equalsIgnoreCase("a"))
{
    System.out.println("Enter your First Name: ");
    firstname=obj.readLine();

    for (User user: userrepository.findByfirstName(firstname))
    {
        System.out.println("User found: "+user);
        System.out.println("With the user authority: "+userauthorityrepository.findByuser(user));
        System.out.println("What do you want to change it to (a=admin, u=user)? : ");
        String role_choice=obj.readLine();
        userauthorityrepository.findByuser(user).role=role_choice;
    }
}
else
{
    System.out.println("Error: No permission to access!");
    System.out.println();
    continue;
}
break;

```

```

case 8:
{
    System.out.println("Enter your Plasma Type: ");
    String plasmatype=obj.readLine();

    Donor donor=new Donor(user_global, plasmatype,"",0);

    donorrepository.save(donor);
}
break;
case 9:
iterate=false;
break;
default:
System.out.println("Wrong input.....Please try again");
continue;

```

```

else
{
    System.out.println("What is your next operation:");
    System.out.println("1. Show profile");
    System.out.println("2. Reset Password");
    System.out.println("3. Donate");
    System.out.println("4. Look for donors by Plasma Type");
    System.out.println("5. Exit");

    selection=Integer.parseInt(obj.readLine());

    switch(selection)
    {
        case 1:
            System.out.println(user_global);
            break;
        case 2:
            {
                System.out.println("Enter your password: ");
                password=obj.readLine();
                credentialrepository.findByuser(user_global).setPaassword(password);
            }
        }
        break;
    }
}

```

```

        case 3:
        {
            System.out.println("Enter your Plasma Type: ");
            String plasmatype=obj.readLine();
            System.out.println("Enter the date: ");
            String visited = obj.readLine();
            System.out.println("Enter the amount donated: ");
            int amtdonated=Integer.parseInt(obj.readLine());

            Donor donor=new Donor(user_global, plasmatype,visited,amtdonated);

            donorrepository.save(donor);

        }
        break;
        case 4:
        {
            System.out.println("Enter the plasma type: ");
            String plasmatype=obj.readLine();
            for (Donor donor: donorrepository.findByPlasmaType(plasmatype))
            {
                System.out.println(donor);
                System.out.println(contactrepository.findByuser(donor.user));
            }
        }
        break;
        case 5:
            System.exit(0);
            break;
        default:
            System.out.println("Invalid input!!");
    }
}

```

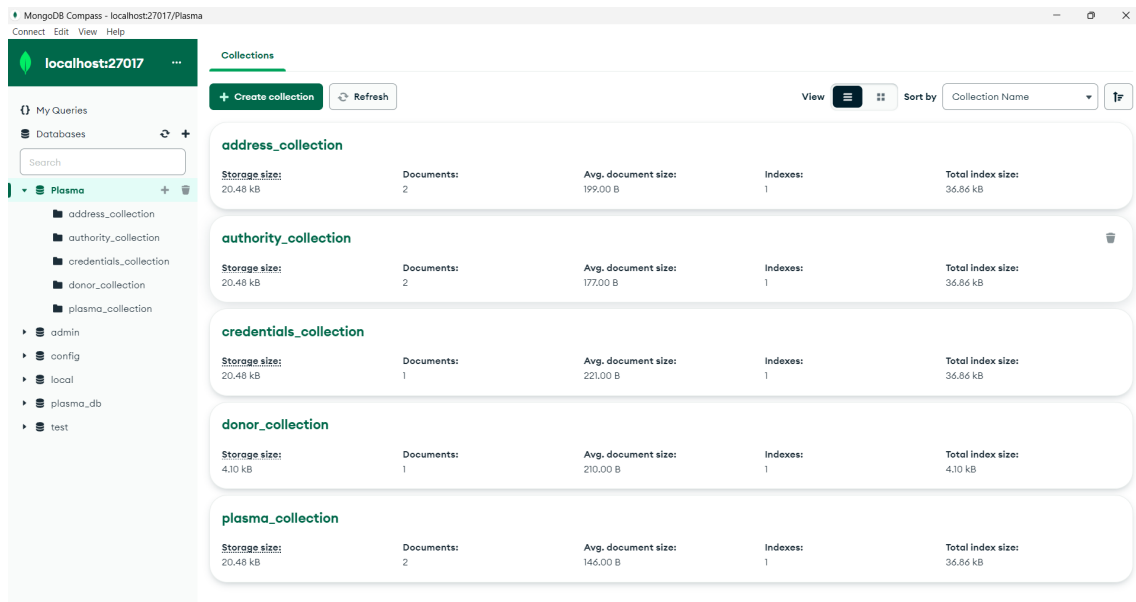
```

2023-06-29T18:02:24.029+05:30 INFO 5072 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2023-06-29T18:02:24.926+05:30 INFO 5072 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-06-29T18:02:24.940+05:30 INFO 5072 --- [ restartedMain] c.e.s.SpringMongo2Application : Started SpringMongo2Application in 4.127 seconds (process running for 0.001s)
Enter your email ID: christintheekank@gmail.com
Enter the password:
blah
true
You have successfully logged in!!
What is your next operation:
1. Add a user
2. Print all the users
3. Get info on a perticular user with the first name
4. Get info on a perticular user with the last name
5. Get info on a perticular user with the blood type
6. Delete the record of a user
7. Change the role of a user
8. Add yourself as a donor
9. Exit
2
Customers found with findAll():
=====
User info[id=649a965e9705af434d55f4cf, firstName='Christin', lastName='T Kunjumon', bloodType='O-positive']
User contact info[address='Kerala', phone number='8330818863']
User authority info[user first name='Christin', role='a']
User info[id=649c4bfdb383304b3dbab425, firstName='Masashi', lastName='Kishimoto', bloodType='AB-positive']
User contact info[address='Japan', phone number='997654210']
User authority info[user first name='Masashi', role='u']

```

Activity 2: Database Integration and ORM

- Set up the database server (e.g., MySQL or MongoDB) and establish the necessary database connections.
- Design the database schema based on the application requirements, including tables/entities and their relationships.
- Implement object-relational mapping (ORM) techniques (e.g., Hibernate or Spring Data) to interact with the database.
- Develop database queries and CRUD operations for doctor data, user information, and bookings.



MongoDB Compass - localhost:27017/Plasma

Connect Edit View Help

localhost:27017

My Queries

Databases

Search

Plasma

- address_collection
- authority_collection
- credentials_collection
- donor_collection
- plasma_collection

admin

config

local

plasma_db

test

Collections

+ Create collection Refresh

View [icon] [icon] Sort by Collection Name [icon]

Collection Name	Storage size	Documents	Avg. document size	Indexes	Total index size
address_collection	20.48 kB	2	199.00 B	1	36.86 kB
authority_collection	20.48 kB	2	177.00 B	1	36.86 kB
credentials_collection	20.48 kB	1	221.00 B	1	36.86 kB
donor_collection	4.10 kB	1	210.00 B	1	4.10 kB
plasma_collection	20.48 kB	2	146.00 B	1	36.86 kB

MongoDB Compass - localhost:27017/Plasma.plasma_collection

Connect Edit View Collection Help

localhost:27017 Documents Plasma.plasma_co...

My Queries Databases Search

Plasma address_collection authority_collection credentials_collection donor_collection **plasma_collection** admin config local plasma_db test

Plasma.plasma_collection

2 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find Options

ADD DATA EXPORT DATA 1 - 2 of 2

```
{ "_id": ObjectId("649a965e9785af434d55f4cf"), "firstName": "Christin", "lastName": "Kunjumon", "bloodType": "0-positive", "_class": "com.example.Spring_mongo_2.model.User" }
```

```
{ "_id": ObjectId("649a98f71f95928a6650b4ff"), "firstName": "Masashi", "lastName": "Kishimoto", "bloodType": "0-positive", "_class": "com.example.Spring_mongo_2.model.User" }
```

MongoDB Compass - localhost:27017/Plasma.donor_collection

Connect Edit View Collection Help

localhost:27017 Documents Plasma.donor_co...

My Queries Databases Search

Plasma address_collection authority_collection credentials_collection **donor_collection** plasma_collection admin config local plasma_db test

Plasma.donor_collection

1 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find Options

ADD DATA EXPORT DATA 1 - 1 of 1

```
{ "_id": ObjectId("649a98f71f95928a6650b4ff"), "plasmaType": "0", "lastVisited": "", "antiDonated": 0, "_class": "com.example.Spring_mongo_2.model.Donor" }
```

MongoDB Compass - localhost:27017/Plasma.credentials_collection

Connect Edit View Collection Help

localhost:27017 Documents Plasma.credentia...

My Queries Databases Search

Plasma address_collection authority_collection **credentials_collection** donor_collection plasma_collection admin config local plasma_db test

Plasma.credentials_collection

1 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find Options

ADD DATA EXPORT DATA 1 - 1 of 1

```
{ "_id": ObjectId("649a98f71f95928a6650b4ff"), "email": "christinthekekank@gmail.com", "password": "blah", "_class": "com.example.Spring_mongo_2.model.Credentials" }
```

MongoDB Compass - localhost:27017/Plasma.authority_collection

Connect Edit View Collection Help

localhost:27017 Documents Plasma.authority...

My Queries Databases Search

Plasma address_collection **authority_collection** credentials_collection donor_collection plasma_collection admin config local plasma_db test

Plasma.authority_collection

2 DOCUMENTS 1 INDEXES

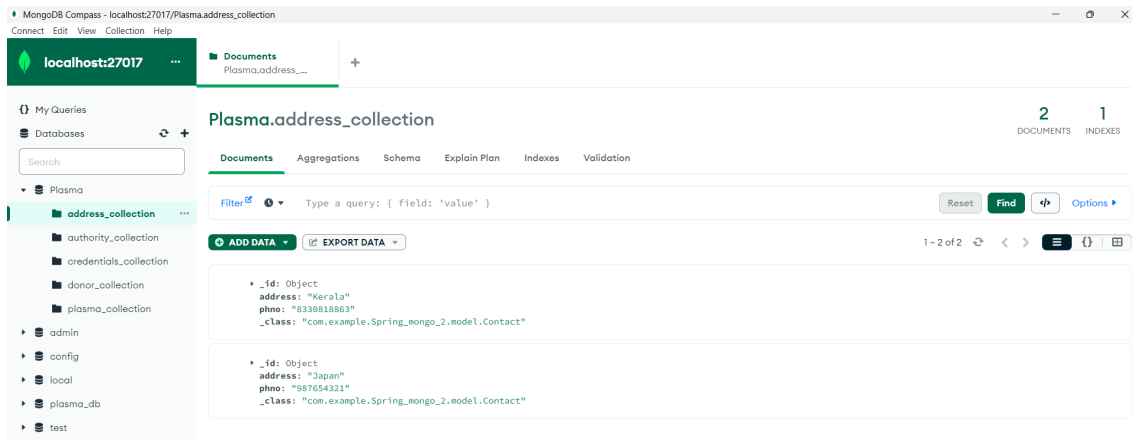
Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find Options

ADD DATA EXPORT DATA 1 - 2 of 2

```
{ "_id": ObjectId("649a98f71f95928a6650b4ff"), "role": "a", "_class": "com.example.Spring_mongo_2.model.userAuthority" }
```

```
{ "_id": ObjectId("649a98f71f95928a6650b4ff"), "role": "u", "_class": "com.example.Spring_mongo_2.model.userAuthority" }
```



Milestone 2: Integration

The back-end made using spring boot and mongo db with java is integrated with the front end made using html , css and js

Milestone 3: Containerization

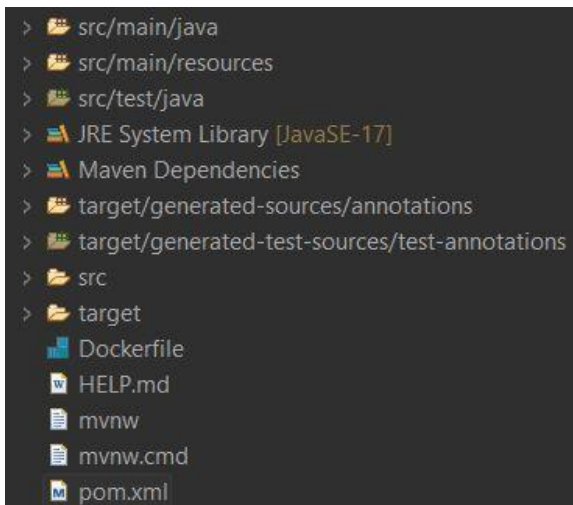
Containerization has been implemented using an application called docker here the application make use of docker command to containerize the specific spring boot and html project creating a container or docker image.

```
C:\Users\Christin T K\workspace\springboot_docker>docker build -t springboot_docker_image .
[+] Building 0.1s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 140B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:8
=> [internal] load build context
=> => transferring context: 93B
=> [1/2] FROM docker.io/library/openjdk:8
=> CACHED [2/2] COPY target/springboot_docker-0.0.1-SNAPSHOT.jar app.jar
=> exporting to image
=> => exporting layers
=> => writing image sha256:3b62139281ace6f1395498da5aa79ad5562062acc25e9467f72342cee404cbe6
=> => naming to docker.io/library/springboot_docker_image

C:\Users\Christin T K\workspace\springboot_docker>docker image ls
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
springboot_docker_image latest      3b62139281ac  2 hours ago  557MB
springboot_docker.jar latest      7ba44e73dfa8  18 hours ago  557MB
welcome-to-docker   latest     2b6b1e61838a  9 days ago   270MB
mongo               latest     7e32c3979b02  12 days ago  653MB
docker/welcome-to-docker latest     912b66cfd46e  12 days ago  13.4MB
sanienski/mongo-express-docker-extension 1.0.2      1053c1983266  4 months ago 14.9MB
openjdk             8          b273004037cc  11 months ago 526MB
alpine              3.10      e7b300aee9f9  2 years ago  5.58MB

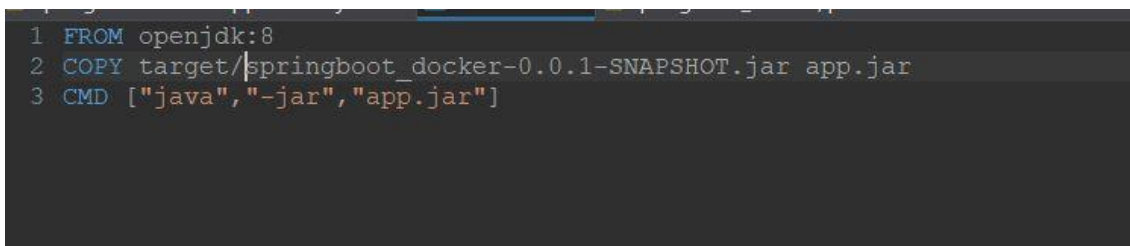
C:\Users\Christin T K\workspace\springboot_docker>
```

The requirement for building such a docker image includes having a docker file named as it is “Dockerfile” in the main directory of the project. This has been depicted in the image below for the given project.



The content of the docker file is shown in the image above . It has to have the following command

- FROM command which indicate the project to use openjdk 8
- COPY command which copies the contents of the jar file of the project to a separate file called “app.jar”
- CMD command which includes the parts of the cmd command that should be used /run in the cmd window or terminal in the device.



After executing all of these a docker image will be successfully made , and with a command “docker image ls”will show the newly created docker image along with the previously created one. This proves the successful creation of the container image of the specific project .The container image can be effortlessly used to isolate this resources and other requirements from those needed the other processes running on the device.

5 APPLICATIONS

The Plasma Donor System is a robust and efficient application built using Java Spring Boot, aimed at streamlining the process of plasma donation and facilitating the matching of donors with recipients. This user-friendly application provides a seamless experience for both donors and recipients, allowing them to create profiles, provide essential information, and search for potential matches. Using the interface, the Users can enter their details and can search for possible donors. The administrator has the ability to view all the donors as

well as edit/delete them if needed. This application leverages the power of Java Spring Boot's framework to deliver a scalable and reliable solution, enabling the healthcare community to effectively manage and coordinate plasma donations for those in need.

6 CONCLUSION

In conclusion, the Plasma Donor System project, developed using HTML, CSS, JavaScript, MongoDB, and Java SpringBoot, has proven to be a significant and impactful contribution to the healthcare community. By leveraging the power of web technologies and a robust backend framework, this system has effectively addressed the critical need for plasma donation coordination and management. Through its user-friendly interface and seamless integration with a reliable database, the project has streamlined the process of connecting plasma donors with recipients in need, thereby saving lives and making a positive difference. This project exemplifies the potential of modern software development technologies to bring about meaningful change in the healthcare industry, underscoring the importance of innovation and collaboration in addressing real-world challenges.

7 FUTURE SCOPE

The Plasma Donor System project holds great potential for future expansion and enhancement. One of the key areas of future scope lies in incorporating more advanced features and functionalities. For instance, integrating machine learning algorithms could help in predicting donor availability based on historical data and optimizing the matching process between donors and recipients. Additionally, the project could be extended to include a mobile application, enabling users to access the system on their smartphones and facilitating real-time notifications and updates. Moreover, incorporating a social media integration could encourage wider participation and reach, as it would allow users to share their donation experiences and spread awareness through various online platforms. Furthermore, collaborating with blood banks, hospitals, and other healthcare organizations could enable seamless data exchange, ensuring accurate and up-to-date information on plasma availability. Overall, the future scope of this project holds immense potential for further development, innovation, and impact in the field of plasma donation coordination and management.