Blood bank management system

Context:

- W3H Analysis
- Uml
- ER Diagram
- Wire Frame
- Front end
- Bank end
- Sonar Report
- Testing Report
- CI / CD

W3H ANALYSIS

Case Study 12: Blood Bank Management System

Blood bank management system automates the blood bank services. This interconnects blood banks, donors and users. It has the following features like

- Donor details management
- Blood stock management
- Blood collection management
- Blood request management
- Report generation

W3H Analysis

What	How
1. **What is the kind of user are there?** - Admin (blood hub) - Blood Bank (bank) - User - Donor - Request (For Hospital) (Without Registration) - User - Request (For individual)	1. **Registration for user, Blood Bank** M1: Separate Page for all M2: Separate page for Blood bank, User as donor and hospital login 2. **Login for all** M1: Separate page for login M2: Same page for login and enter to page based on their role 3. **Documentation upload** M1: Documentation can upload as image M2: Documentation can upload as pdf

2. Is User have specific privilege for hospital and individual?

- Yes, Hospital can request from individual and bank
- Individual can request from bank

3. Is Requested User can track the request information

Yes

4. Is Request and Donor user need special field, if required what are the details?

- Request must give basic details for authentication
- Donor must give basic details and verification documentation for blood group

5. What are the basic details given by User, Blood Bank, Hospital

- User (as Donor, Hospital) => Name, Mobile Number, Aadhar number (if user), Address, Email, password
- Bank=> Bank Name, Location, Mobile Number, Address, Bank Official Id (Government Registered Id), Document for verify the Bank.

6. Any Specific details required for hospital login as user?

- Yes, Their Documentation regarding the government registration and Registration id.

7. Who can donate their blood?

- If they register as donor, they can donate their blood by call

8**. What can Admin(hub) do?**

- They can view the bank details, user details (donor, requester), Collection camp details conducted by bank, Blood stock details by individual bank and location.
- They approve or reject registration based on their details for user (donor, hospital) and bank registration details and approve or reject request for blood.

M3: Documentation can upload as image and pdf

4. Camp Details

M1: Email sends to nearby user
M2: Web notification to user
M3: Camp Calendar shown in websites and sent email

5. Report Generation

M1: Report Generation is done by excel
M2: Report is shown in pdf
M3: Report is shown in websites

6. How is the appointment booked with the donor?

M1: Donor can choose date for appointment at the time of registration
M2: Bank can check the availability of donor when they needed and donor get called by bank

7. Document verification

M1: Done manually by admin
M2: Partial verification is done by system.

8. Blood bank can view other bank stock

M1: View by location
M2: View by blood Bank
M3: View by location and blood bank

9. Blood bank can view the details of donor

M1: Necessary details like name, blood type, contact details and location
M2: All details

10. Admin can approve and reject the user registration

M1: By verify the details and mail send to user
M2: By verify the details and notification send to websites

11. Track Request of blood

M1:By Request Id and mobile number

9. How can generate reports and what kind of report? • Bank must only view their blood donated details of their own bank and blood stock details of their own and nearby bank • Admin can view blood stock of all banks **10. what can the bank do?** • Bank can view the report of blood stock and donor details • They can donate blood to those who requested. • They can add, update, delete the blood stock • They can conduct camp • They can view local donor • They can see blood stock of another nearby bank **11. What are the details required for blood?** • Blood=> Type, Units, Location, Bank Details	M2:By Request Id Only
Why	Why not?
1. Registration for user, Blood Bank M2: Separate page for Blood bank, User as donor and hospital login R: It is easy for register and easy to maintain the page **2. Login for all** M2: Same page for login and enter to page based on their role R: It is easy to maintain the website and easy for user **3. Documentation upload** M1: Documentation can upload as image R: It is easy to view by admin 4**. Camp Details**	**1. Registration for user, Blood Bank** M1: Separate Page for all R: It is hard for user and hard maintain the page **2. Login for all** M1: Separate page for login R: It is easy but hard for user to login **3. Documentation upload** M2: Documentation can upload as pdf M3: Documentation can upload as pdf and image R: It is hard to display 4**. Camp Details** M1: Email sends to nearby user M2: Web notification to user

M3: Camp Calendar shown in websites and sent email R: It can increase the donor and pass the information to large people	R: It is not properly received by large people
5. Report Generation M3: Report is shown in websites R: It is efficient way to view the report	**5. Report Generation** M1: Report Generation is done by excel M2: Report is shown in pdf R: It is not an efficient way to view the report.
6. How is the appointment booked with the donor? M2: Bank can check the availability of donor when they needed and donor fix appointment after getting the request from bank R: It is more usable case	**6. How is the appointment booked with the donor?** M1: Donor can choose date for appointment at the time of registration R: It is less usable case
7. Document verification M1: Done manually by admin R: More trustable	**7. Document verification** M2: Partial verification is done by system. R: Less Trustable
8. Blood bank can view other bank stock M3: View by location and blood bank R: It is useful to search and find needed blood	**8. Blood bank can view other bank stock** M1: View by location M2: View by blood Bank R: It is not more useful to find blood
9. Blood bank can view the details of donor M1: Necessary details like name, blood type, contact details and location R: It helps to hide confidently details	**9. Blood bank can view the details of donor** M2: All details R: It not hide confidently details
10. Admin can approve and reject the user registration M1: By verify the details and mail send to user R: It is reachable to user	**10. Admin can approve and reject the user registration** M2: By verify the details and notification send to websites R: It is not reachable to user as it take time
11. Track Request of blood M1:By Request Id and mobile number R: It is easy and secure	**11. Track Request of blood** M2:By Request Id Only R: It is easy and not secure

UML

UML

Class Diagram

Use Case Diagram

Admin

Blood Bank

User(Donor)

User(Hospital)

User (Blood Requester)

Sequence Diagram

User (Blood Requester)

User without registration

Registration and login for User (Donor,Hospital), Blood Bank

Registration and login for User(Donor,Hospital), Blood Bank

User(Donor,Hospital) , Blood Bank

User(Donor)

User(Hospital)

Blood Bank

Admin

ER DIAGRAM

ER Diagram

WIRE FRAME

Wire Frame

Welcome Page

Login Page

Register Page (Getting Common details)

Blood Request page

Checking the Blood Availability

Incoming Donor Request (In Admin)

Display Upcoming Blood Camp

FRONT END

Front End

Welcome Page

Welcome page contain blood camp and blood bank details

Login show error without entering data after login is clicked

Invalid email and password with less than 8 character show error

Invalid email error message is shown

Incorrect password error message is shown

Admin Dashboard

View Hospital details by admin

View Donor Details by admin

View blood bank details by admin

View all camp details by admin

View Incoming Hospital request by admin

View Incoming Donor request by admin

View Incoming Blood bank request by admin

View Incoming Blood Request request by admin

View all blood bank blood stock details by admin

Search by blood type

Login with pending request of account creation

Pending Page

Login donor account

Donor Home Page

View Donor Card

Blood Bank Dashboard

View Donor details by blood bank

View stock details of that bank by blood bank

Add stock by bank shows error with empty data entry

Successful message after added

Show error msg with empty data enrty

View Blood Transaction Details of that bank by blood bank

Show error while entry the empty data in add blood camp

View Blood camp details of that blood bank by blood bank

Blood stock details of all bank

View Blood request by bank they can accept or reject that request

Login with hospital details

Hospital Dashboard

View Donor details by hospital

View Blood bank details by hospital

View blood camp details by hospital

View Blood Bank with stock by hospital

View pending status of pending blood request

Show invalid id msg with invalid id

Show bank and accepted bank details with approved request id

Blood request form

Show error message with no data entry

Registration page for blood bank and hospital

Show error message with no data entry

Registration page for donor

Show error message with no data entry

Database

Account

AccountId	Name	Email	Password	PhoneNumber	Status
10bab100-d6b3-41d0-9d3b-46491cda7ca0	ABC Blood Bank	sanjaihari2002@gmail.com	ceb200a6cea4095b86c8bc4d1e801aff9812eaa...	6382202487	1
26e36c6a-0c9a-48e2-8eac-0296706b2604	ABC Hospital	smano4570@gmail.com	ceb200a6cea4095b86c8bc4d1e801aff9812eaa...	6382202586	1
3cb987fc-841f-4cd2-9c6b-46b45fe22ea6	Vasan Blood bank	keerthivasan28092001@gmail.com	ceb200a6cea4095b86c8bc4d1e801aff9812eaa...	9790953230	1
477b7d70-299c-4428-a130-7b5bf5176586	Keerthi Vasan	keerthivasan280901@gmail.com	ceb200a6cea4095b86c8bc4d1e801aff9812eaa...	9089678776	1
5f20561c-8175-4826-81a8-3f9ea108c2aa	S Lab	arunsrinivasan333@gmail.com	ceb200a6cea4095b86c8bc4d1e801aff9812eaa...	9878675432	1

Role

RoleId	RoleName
08e882cc-d50f-41c8-97e9-9ddfe1a8375d	ADMIN
1cc06485-e084-447a-a138-cd5af7751a14	DONOR
1ee71642-e3a4-4605-b53c-873624279e7b	HOSPITAL
d9157f05-ab6e-49de-b404-ab9b53b1e4e2	BLOODBANK

Account role

AccountRoleId	AccountId	RoleId
1766ba8d-742b-41d7-8e65-a3e3f6e40efe	10bab100-d6b3-41d0-9d3b-46491cda7ca0	d9157f05-ab6e-49de-b404-ab9b53b1e4e2
1a690e83-a006-4ad8-ae14-127fa682b63a	26e36c6a-0c9a-48e2-8eac-0296706b2604	1ee71642-e3a4-4605-b53c-873624279e7b
5091cb80-d24d-48bf-86f0-daaa2e78d347	62a5ec56-1022-43a8-b125-e9ada4a7ba30	08e882cc-d50f-41c8-97e9-9ddfe1a8375d
9e21a3c9-68d6-4c66-94ef-afb778846043	477b7d70-299c-4428-a130-7b5bf5176586	1cc06485-e084-447a-a138-cd5af7751a14

Address

AddressId	DoorNo	Street	Area	City	State	PostalCode
1248b1df-df9d-4ebf-8044-73728397f1bf	28	Kcet College	Vanarpettai	Tirunelveli	Tamil nadu	687300
3ab9743d-1900-4a1c-b575-cd5e19d6bdc4						
4b52532d-8e26-4d93-b6c1-aa5793b33cec	21	D street	D area	Dharmapuri	Tamilnadu	647001
6926a1fb-8e34-4ef4-b7e2-a948a60c4a5e	21/2	Salai Street,	Ktc Nagar	Tirunelveli	Tamil Nadu	627011
76ce5ff9-35d5-40ee-96de-4342a78deec5	34	K Street	Vanarpettai	Viruthunagar	TamilNadu	687300

UserDetails

UserDetailsId	Age	BloodType	Location	GovernmentId	Document	AadhaarNumber	rolestatus
09eaed17-49bc-47d3-9795-5671d3ae18b9	0		Chennai		BLOB	234598762345	0
1b60eba4-66f9-496f-a764-7afb10d7ee87	21	B+ve	Chennai		BLOB	345623458900	3
7032d920-e6bb-45e3-af07-8b40ea7575b3	0		Chennai	1245433	BLOB	0	2
75762986-e018-48c5-a001-0f3627a1fc19	0		Viruthunagar	TN00132233	BLOB	298756784566	1
a150cfd2-f3d5-4752-a391-6173ffb7f2d2	0		Chennai	1245433	BLOB	0	1

Link between userdetails , address , account

AccountUserDetailsAddressId	AccountId	UserDetailsId	AddressId
3056ac14-9ab6-4dda-abb9-41317b5b7d40	5f20561c-8175-4826-81a8-3f9ea108c2aa	75762986-e018-48c5-a001-0f3627a1fc19	76ce5ff9-35d5-40ee-96de-4342a78deec5
3d052f0d-2b70-4030-ab6e-c9c79cd2fbff	3cb987fc-841f-4cd2-9c6b-46b45fe22ea6	ed3d72fc-c081-443e-adc1-45c3c02c1a64	1248b1df-df9d-4ebf-8044-73728397f1bf
4c3b4f11-3c80-452f-91af-6a45ed029035	26e36c6a-0c9a-48e2-8eac-0296706b2604	7032d920-e6bb-45e3-af07-8b40ea7575b3	6926a1fb-8e34-4ef4-b7e2-a948a60c4a5e
50fa9fae-bfb2-4e5c-a219-2ebdc1fb4f47	477b7d70-299c-4428-a130-7b5bf5176586	1b60eba4-66f9-496f-a764-7afb10d7ee87	b6872d86-1eb1-4fca-b2ad-3f2c798440b7
c5c7ac9e-db93-4266-8137-a3042498872e	10bab100-d6b3-41d0-9d3b-46491cda7ca0	a150cfd2-f3d5-4752-a391-6173ffb7f2d2	db99c12e-3241-468d-b5d9-977d38b7c702

Blood camp

BloodCampId	BloodCampName	BloodCampLocation	Date	Time
4bd17a67-2eb9-43e7-b7fb-c76a51f8d237	Blood Sevaa	Viruthunagar	21/05/2024	11:00
791d6770-e7bd-4192-8bd2-94787b9b7507	Donate Seva	Tirunelveli	23/05/2024	10:00
NULL	NULL	NULL	NULL	NULL

Link blood camp and blood bank

BloodCampBloodBankId	BloodCampId	AccountId
4bb41fea-362f-4c49-95fd-0ba570e9d858	4bd17a67-2eb9-43e7-b7fb-c76a51f8d237	10bab100-d6b3-41d0-9d3b-46491cda7ca0
b8ab560b-03d4-4f00-a593-72d66f64510b	791d6770-e7bd-4192-8bd2-94787b9b7507	10bab100-d6b3-41d0-9d3b-46491cda7ca0
NULL	NULL	NULL

Blood Request

BloodRequestId	Name	Email	PhoneNumber	BloodType	Age	Location	AadhaarNumber	ValidTime	Status	AcceptStatus	U
4aOCW9V	Mano	keerthivasan280901@gmail.com	9790953230	B+ve	19	Tirunelveli	909090909079	03/29/2024 16:43:07	4	1	2
QcLtyYFJ	Mano	smano4570@gmail.com	9790953230	AB+ve	1	Chennai	234554320987		1	0	1
SP39NEvE	Vasan	senathipathik@gmail.com	9876543210	A-ve	35	Viruthunagar	345432433222	04/01/2024 08:40:20	2	1	3
T2937kCU	Sanjai	sanjairock85@gmail.com	9867541243	B+ve	20	Chennai	235489098762	03/29/2024 17:11:20	1	1	2

Blood transaction

BloodTransactionId	AccountId	Date	Time	BloodType	units	BloodRequestId
8bd10343-2d3c-4675-9f74-daebf35c432c	10bab100-d6b3-41d0-9d3b-46491cda7ca0	2024-02-21	10:00 Am	A-ve	1	4aOCW9V
cbb7f7ec-2ebf-45b3-af88-12729cb4fa41	10bab100-d6b3-41d0-9d3b-46491cda7ca0	2024-01-21	1:00 Pm	B+ve	2	T2937kCU
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Blood stock

BloodStockId	BloodType	Units	AccountId
2e219639-4946-465b-a58e-59b62e2eb46f	A+ve	1	10bab100-d6b3-41d0-9d3b-46491cda7ca0
411c93d3-2c48-47ce-86eb-51af420ebbce	AB+ve	2	10bab100-d6b3-41d0-9d3b-46491cda7ca0
52143f5a-e264-4a0e-82cd-3f7d8f12cdd2	B+ve	2	10bab100-d6b3-41d0-9d3b-46491cda7ca0
d4c1e8fd-7acd-41a7-a2c9-1a40c21849c6	A-ve	96	10bab100-d6b3-41d0-9d3b-46491cda7ca0

BANK END

Backend report

1.Coding

2. Output in swagger (screenshot)

Coding :

DataContext :

AppDbContext.cs

```
using Microsoft.EntityFrameworkCore;
using BloodBankManagementWebapi.Model;
namespace BloodBankManagementWebapi.DataContext
{
   public class AppDbContext:DbContext
   {
      public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }
      public DbSet<Account> Account{ get; set; }
      public DbSet<Role> Role { get; set; }
      public DbSet<AccountRole>  AccountRole { get; set; }
      public DbSet<Address> Address { get; set; }
      public DbSet<UserDetails> UserDetails { get; set; }
      public DbSet<BloodCamp> BloodCamp { get; set; }
      public DbSet<BloodStock> BloodStock { get; set; }
      public DbSet<BloodTransaction> BloodTransaction { get; set; }
      public DbSet<BloodRequest> BloodRequest { get; set; }
      public DbSet<BloodStockRequester> bloodStockRequesters { get; set; }
      public DbSet<AccountUserDetailsAddress> AccountUserDetailsAddress { get; set; }
      public DbSet<BloodCampBloodBank> BloodCampBloodBank { get; set; }
   }
```

```
}
```

Appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection":
"server=localhost;user=root;password=admin123;database=BloodBankManagementSystem1.0;port=33
06"
  },
  "JWT": {
    "ValidIssuer": "*",
    "ValidAudience": "*",
    "SecretKey": "bd1a1ccAS09R37f361a4Dd351e7c0de65f0776bfc278ea8d312c763bb6c#3ac!a=="
  }
}
```

Program.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.Text;

var builder = WebApplication.CreateBuilder(args);
```

```
var ConnectionString = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<AppDbContext>(options => options.UseMySql(ConnectionString, new
MySqlServerVersion(new Version())));

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

#region CORS setting for API

builder.Services.AddCors(options =>

{

   options.AddPolicy(name: "_myAllowSpecificOrigins",

   policy =>

   {

      policy.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod().AllowAnyMethod();

   }

   );

});

#endregion

//start jwt configuration

builder.Services.AddAuthentication(options =>

{

   options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;

   options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

   options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

   options.SaveToken = true;

   options.RequireHttpsMetadata = false;

   options.TokenValidationParameters = new TokenValidationParameters()
```

```
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidIssuer = builder.Configuration["JWT:ValidIssuer"],
        ValidAudience = builder.Configuration["JWT:ValidAudience"],
        IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["JWT:SecretKey"]!))
    };
});
//end
var app = builder.Build();
    app.UseSwagger();
    app.UseSwaggerUI();
app.UseCors("_myAllowSpecificOrigins");
app.UseHttpsRedirection();
app.UseAuthentication();
app.UseRouting();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

OtherOperation

SHA256Encrypt.cs

```
using System.Text;
using System.Security.Cryptography;
namespace BloodBankManagementWebapi.OtherOperation
{
    public static class SHA256Encrypt
```

```
  {
    public static string ComputePasswordToSha256Hash(string plainText)
    {
                using (SHA256 sha256Hash = SHA256.Create())
      {
          byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(plainText));
   StringBuilder stringbuilder = new StringBuilder();
          for (int i = 0; i < bytes.Length; i++)
          {
             stringbuilder.Append(bytes[i].ToString("x2"));
          }
          return stringbuilder.ToString();
      }
    }
  }
}
```

EmailGenerator.cs

```
using BloodBankManagementWebapi.Model;
using Google.Protobuf;
using Microsoft.AspNetCore.Http.HttpResults;
using Org.BouncyCastle.Asn1.Ocsp;
using System.Net;
using System.Net.Mail;
using System.Xml.Linq;

namespace BloodBankManagementWebapi.OtherOperation
{
  public static class EmailGenerator
```

```
{
    public static bool SendEmail(string Username, string Mail, string Password)
    {

        string fromMail = "smano8312@gmail.com";
        string senderPass = "mktt mzmx pasy gdgl";
        MailMessage message = new MailMessage();
        message.From = new MailAddress(fromMail);
        message.To.Add(Mail);
        message.Subject = $"Confidential! New Password for Your {Mail} Accounts";

        message.Body = $"Dear {Username},\r\n\r\nThis is a notification from the management. Your new
password for {Mail} accounts has been generated. Please take a moment to update your
password.\r\n\r\nNew Password: {Password}\r\n\r\nRemember to change your password promptly for
security reasons.\r\n\r\nThank you, Management Team";
        SmtpClient smtpClient = new SmtpClient("smtp.gmail.com");
        smtpClient.Port = 587;
        smtpClient.UseDefaultCredentials = false;
        smtpClient.Credentials = new NetworkCredential(fromMail, senderPass);
        smtpClient.EnableSsl = true;
        try
        {
            smtpClient.Send(message);
            return true;
        }
        catch (Exception ex)
        {
            return false;
        }
    }
```

```
public static bool SendEmail(string Username, string Mail, string Message,string RequestId,string ValidDate)
{
    string fromMail = "smano8312@gmail.com";
    string senderPass = "mktt mzmx pasy gdgl";
    MailMessage message = new MailMessage();
    message.From = new MailAddress(fromMail);
    message.To.Add(Mail);
    if (Message == "BloodRequestApprove")
    {
        message.Subject = $"Important! Blood Request is approved by management";
        message.Body = $"Dear {Username},\r\n\r\nThis is a notification from the management. Your Blood Request for Id {RequestId} is approved by management.Please check the website and it is valid for date {ValidDate}.\r\n\r\nThank you, Management Team";
    }
    else if (Message == "BloodRequestReject")
    {
        message.Subject = $"Important! Blood Request is rejected by management";
        message.Body = $"Dear {Username},\r\n\r\nThis is a notification from the management. Your Blood Request for Id {RequestId} is rejected by management.Please apply again with correct information.\r\n\r\nThank you, Management Team";
    }



    SmtpClient smtpClient = new SmtpClient("smtp.gmail.com");
    smtpClient.Port = 587;
    smtpClient.UseDefaultCredentials = false;
    smtpClient.Credentials = new NetworkCredential(fromMail, senderPass);
    smtpClient.EnableSsl = true;
    try
```

```
        {
            smtpClient.Send(message);
            return true;
        }
        catch (Exception ex)
        {
            return false;
        }
    }
    public static bool SendEmailForAccountApproval(string Name, string Email, string ApprovalStatus)
    {
        string fromMail = "smano8312@gmail.com";
        string senderPass = "mktt mzmx pasy gdgl";
        MailMessage message = new MailMessage();
        message.From = new MailAddress(fromMail);
        message.To.Add(Email);
        if (ApprovalStatus == "AccountApprove")
        {
            message.Subject = $"Important! Account is approved by management";

            message.Body = $"Dear {Name},\r\n\r\nThis is a notification from the management. Your Account is approved by management.Please check the website .\r\n\r\nThank you, Management Team";

        }
        else if (ApprovalStatus == "AccountReject")
        {
            message.Subject = $"Important! Account is rejected by management";
```

```
        message.Body = $"Dear {Name},\r\n\r\nThis is a notification from the management. Your Account is rejected by management.Please check the website .\r\n\r\nThank you, Management Team";


      }



      SmtpClient smtpClient = new SmtpClient("smtp.gmail.com");

      smtpClient.Port = 587;

      smtpClient.UseDefaultCredentials = false;

      smtpClient.Credentials = new NetworkCredential(fromMail, senderPass);

      smtpClient.EnableSsl = true;

      try

      {

        smtpClient.Send(message);

        return true;

      }

      catch (Exception ex)

      {

        return false;

      }



    }

    public static void SendEmailForAcceptRequest(string Email, string BloodRequestId, Account account, Address address,UserDetails userdetails)

    {

      string fromMail = "smano8312@gmail.com";

      string senderPass = "mktt mzmx pasy gdgl";

      MailMessage message = new MailMessage();
```

```
        message.From = new MailAddress(fromMail);
        message.To.Add(Email);
          message.Subject = $"Important! Your Blood Request is accept by {account.Name}";


          message.Body = $"Dear {account.Name},\r\n\r\nYour request is accepted by {account.Name}-
{userdetails.Location}.Please check the bank and collect the blood.
\nAddress :\n{address.DoorNo},\n{address.Street},\n{address.Area},\n{address.City},\n{address.State}-
{address.PostalCode}.\r\n\r\nThank you, Management Team";



        SmtpClient smtpClient = new SmtpClient("smtp.gmail.com");
        smtpClient.Port = 587;
        smtpClient.UseDefaultCredentials = false;
        smtpClient.Credentials = new NetworkCredential(fromMail, senderPass);
        smtpClient.EnableSsl = true;
        try
        {
          smtpClient.Send(message);
          return;
        }
        catch (Exception ex)
        {
          return;
        }


      }
    }
}
```

RandomPasswordGenerator.cs

```
using System.Text;


namespace BloodBankManagementWebapi.OtherOperation
{
   public static class RandomPasswordGeneration
   {
      public static string RandomPasswordGenerator()
      {
         Random random = new Random();
         int passwordLength = random.Next(8, 12);
         const string validCharacters =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890@#$&!?";
         StringBuilder password = new StringBuilder();
         for (int i = 0; i < passwordLength; i++)
         {
            int randomcode = random.Next(0, validCharacters.Length-1);
            password.Append(validCharacters[randomcode]);
         }


         return password.ToString();
      }
   }
}
```

Model

Account.cs

```
using System.ComponentModel;


namespace BloodBankManagementWebapi.Model
```

```
{
    public class Account
    {

        public string? AccountId { get; set; }
        public string? Name { get; set; }
        public string? Email { get; set; }
        public string? Password { get; set; }
        public long PhoneNumber { get; set; }
        [DefaultValue(0)]
        public int Status { get; set; } = 0;
            public BloodTransaction? BloodTransaction { get; }
        public BloodCampBloodBank? BloodCampBloodBank { get; }
        public BloodBankBloodStock? BloodBankBloodStock { get; }
    }
}
```

AccountRole.cs

```
namespace BloodBankManagementWebapi.Model
{
    public class AccountRole
    {
        public string? AccountRoleId { get; set; }
        public Account? Account { get; set; }
        public Role? Role { get; set; }
    }

}
```

AccountUserDetailsAddress.cs

```
namespace BloodBankManagementWebapi.Model
{
   public class AccountUserDetailsAddress
   {
      public string? AccountUserDetailsAddressId {  get; set; }
      public Account? Account {  get; set; }
      public UserDetails? UserDetails { get; set; }
      public Address? Address { get; set; }
   }
}
```

Address.cs

```
namespace BloodBankManagementWebapi.Model
{
   public class Address
   {
      public string? AddressId { get; set; }
      public string? DoorNo { get; set; }
      public string? Street { get; set; }
      public string? Area { get; set; }
      public string? City { get; set; }
      public string? State { get; set; }
      public string? PostalCode { get; set; }
    }
}
```

BloodCamp.cs

```
namespace BloodBankManagementWebapi.Model
{
   public class BloodCamp
   {

      public string? BloodCampId { get; set; }
      public string? BloodCampName { get; set; }
      public string? BloodCampLocation { get; set; }
      public string? Date { get; set; }
      public string? Time { get; set; }
      public BloodCampBloodBank? BloodCampBloodBank {  get; }
   }
}
```

BloodCampBloodBank.cs

```
namespace BloodBankManagementWebapi.Model
{
   public class BloodCampBloodBank
   {
      public string? BloodCampBloodBankId { get; set; }
      public BloodCamp? BloodCamp { get; set; }
      public Account? Account { get; set; }
   }
}
```

BloodRequest.cs

```
using System.ComponentModel;
namespace BloodBankManagementWebapi.Model
{
```

```
    public class BloodRequest
    {
        public string? BloodRequestId { get; set; }
        public string? Name { get; set; }
        public string? Email { get; set; }
        public int Units { get; set; }
        public long PhoneNumber { get; set; }
        public string? BloodType { get; set; }
        public int Age { get; set; }
        public string? Location { get; set; }
        public long AadhaarNumber { get; set; }
        [DefaultValue(null)]
        public string ValidTime { get; set; }
        [DefaultValue(0)]
        public int Status { get; set; }
        public int AcceptStatus { get; set; }
        public BloodTransaction BloodTransaction { get;  }= null;
    }
}
```

BloodStock.cs

```
namespace BloodBankManagementWebapi.Model
{
    public class BloodStock
    {
        public string? BloodStockId { get; set; }
        public string? BloodType { get; set; }
        public int Units { get; set; }
       public Account? Account { get; set; }
```

```
    }
}
```

BloodStockRequester.cs

```
namespace BloodBankManagementWebapi.Model
{
    public class BloodStockRequester
    {
        public string? BloodStockRequesterId {  get; set; }


        public BloodRequest BloodRequest { get; set; }
         public Account Account { get; set; }
        public UserDetails UserDetails { get; set; }
        public Address Address { get; set; }
    }
}
```

BloodTransaction.cs

```
namespace BloodBankManagementWebapi.Model
{
    public class BloodTransaction
    {
        public string? BloodTransactionId { get; set; }
        public string? AccountId { get; set; }
        public Account Account { get; set; }
        public string? BloodRequestId { get; set; }
        public BloodRequest BloodRequest { get; set; }
        public string? BloodType { get; set; }
        public int units { get; set; }
```

```
        public string? Date { get; set; }
        public string? Time { get; set; }
    }
}
```

Role.cs

```
namespace BloodBankManagementWebapi.Model
{
    public class Role
    {
        public string? RoleId { get; set; }
        public string? RoleName { get; set; }
       }
}
```

UserDetails.cs

```
namespace BloodBankManagementWebapi.Model
{
    public class UserDetails
    {
        public string? UserDetailsId { get; set; }
        public int Age { get; set; }
        public string? BloodType { get; set; }
        public string? Location {  get; set; }
        public string? GovernmentId {  get; set; }
        public byte[] Document { get; set; }
  public long AadhaarNumber { get; set; }
        public int rolestatus { get; set; }
    }
```

```
}
```

Controller

AcceptRequesterByBankController.cs

```
using BloodBankManagementWebapi.ApiModel;
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.Model;
using BloodBankManagementWebapi.OtherOperation;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
namespace BloodBankManagementWebapi.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class AcceptRequesterByBankController : ControllerBase
  {
    private readonly AppDbContext _context;
    public AcceptRequesterByBankController(AppDbContext context)
    {
      _context = context;
    }

    [HttpPost]
    public IActionResult Post([FromBody] AcceptRequester acceptRequester)
    {
      var requester = _context.BloodRequest.Find(acceptRequester.RequesterId);
      requester.AcceptStatus = 1;
      _context.BloodRequest.Update(requester);
```

```
        _context.SaveChanges();


        var account = _context.Account.Find(acceptRequester.AccountId);

        var accountuserdetailsaddress = _context.AccountUserDetailsAddress.Include(x =>
x.Account).Include(x => x.Address).Include(x => x.UserDetails).Where(x => x.Account ==
account).FirstOrDefault();



        var userdetails =accountuserdetailsaddress.UserDetails;

        var address =accountuserdetailsaddress.Address;

        BloodStockRequester bloodStockRequester = new BloodStockRequester()

        {

           BloodStockRequesterId=Guid.NewGuid().ToString(),

           Account=account,

           UserDetails=userdetails,

           Address=address,

           BloodRequest= requester

        };

        _context.bloodStockRequesters.Add(bloodStockRequester);

        _context.SaveChanges();

        EmailGenerator.SendEmailForAcceptRequest(requester.Email, requester.BloodRequestId,
account, address, userdetails);

        return Ok();

      }

   }

}
```

AdminDashboardController.cs

```
using BloodBankManagementWebapi.DataContext;

using Microsoft.AspNetCore.Http;
```

```
using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using BloodBankManagementWebapi.ApiModel;

namespace BloodBankManagementWebapi.Controllers

{

   [Route("api/[controller]/[action]")]

   [ApiController]

   public class AdminDashboardController : ControllerBase

   {

      private readonly AppDbContext _context;

      public AdminDashboardController(AppDbContext context)

      {

         _context = context;

      }

      [HttpGet]

      public ActionResult GetBasicDetails()

      {

         var Hospital = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a =>
a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus == 2).Where(x =>
x.Account.Status != 0).Count();

         var Donor = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a =>
a.Account).Include(s => s.Address).ToList().Where(x => x.UserDetails.rolestatus == 3).Where(x =>
x.Account.Status != 0).Count();

         var Bloodcamp = _context.BloodCampBloodBank.ToList().Count();

         var BloodBank = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a =>
a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus == 1).Where(x =>
x.Account.Status != 0).Count();

         var BloodRequest = _context.BloodRequest.ToList().Count();

         var PendingBloodRequest = _context.BloodRequest.Where(x=>x.Status==0).ToList().Count();
```

```
        var HospitalRequest = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a
=> a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus == 2).Where(x =>
x.Account.Status == 0).Count();

        var BloodBankPendingRequest = _context.AccountUserDetailsAddress.Include(x =>
x.UserDetails).Include(a => a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus ==
1).Where(x => x.Account.Status == 0).Count();

        var DonorPendingRequest = _context.AccountUserDetailsAddress.Include(x =>
x.UserDetails).Include(a => a.Account).Include(s => s.Address).ToList().Where(x =>
x.UserDetails.rolestatus == 3).Where(x => x.Account.Status == 0).Count();

        var HospitalPendingRequest = _context.AccountUserDetailsAddress.Include(x =>
x.UserDetails).Include(a => a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus ==
2).Where(x => x.Account.Status == 0).Count();

         var Result=new Dashboard()

        {

           Hospital=Hospital,

           Donor=Donor,

           BloodBank=BloodBank,

           Bloodcamp=Bloodcamp,

           BloodRequest=BloodRequest,

           HospitalRequest=HospitalRequest,

           BloodBankPendingRequest=BloodBankPendingRequest,

           DonorPendingRequest=DonorPendingRequest,

           HospitalPendingRequest =HospitalPendingRequest,

           PendingBloodRequest= PendingBloodRequest

        };

        return Ok(Result);

    }

    [HttpGet]

    public IActionResult GetBloodStockDetails()

    {

        var Apositive = _context.BloodStock.Where(x=>x.BloodType=="A+ve").Sum(x=>x.Units);
```

```
        var Bpositive = _context.BloodStock.Where(x => x.BloodType == "B+ve").Sum(x => x.Units);
        var Opositive = _context.BloodStock.Where(x => x.BloodType == "O+ve").Sum(x => x.Units);
        var ABpositive = _context.BloodStock.Where(x => x.BloodType == "AB+ve").Sum(x => x.Units);
        var Anegative = _context.BloodStock.Where(x => x.BloodType == "A-ve").Sum(x => x.Units);
        var Bnegative = _context.BloodStock.Where(x => x.BloodType == "B-ve").Sum(x => x.Units);
        var Onegative = _context.BloodStock.Where(x => x.BloodType == "O-ve").Sum(x => x.Units);
        var ABnegative = _context.BloodStock.Where(x => x.BloodType == "AB-ve").Sum(x => x.Units);
        BloodDashboard bloodDashboard = new BloodDashboard
        {
            ABnegative = ABnegative,
            Bnegative = Bnegative,
            Anegative = Anegative,
            Onegative = Onegative,
            Opositive = Opositive,
            ABpositive = ABpositive,
            Apositive = Apositive,
            Bpositive   = Bpositive,
        };
        return Ok(bloodDashboard);
    }
    [HttpGet]
    public IActionResult GetBloodStockDetailsByIndividualBank(string Id)
    {
        var account= _context.Account.Find(Id);


         var Apositive = _context.BloodStock.Include(x => x.Account).Where(x => x.Account ==
account).Where(x => x.BloodType == "A+ve").Sum(x => x.Units);
        var Bpositive = _context.BloodStock.Include(x => x.Account).Where(x => x.Account ==
account).Where(x => x.BloodType == "B+ve").Sum(x => x.Units);
```

```
        var Opositive = _context.BloodStock.Include(x => x.Account).Where(x => x.Account ==
account).Where(x => x.BloodType == "O+ve").Sum(x => x.Units);

        var ABpositive = _context.BloodStock.Include(x => x.Account).Where(x => x.Account ==
account).Where(x => x.BloodType == "AB+ve").Sum(x => x.Units);

        var Anegative = _context.BloodStock.Include(x => x.Account).Where(x => x.Account ==
account).Where(x => x.BloodType == "A-ve").Sum(x => x.Units);

        var ABnegative = _context.BloodStock.Include(x => x.Account).Where(x => x.Account ==
account).Where(x => x.BloodType == "AB-ve").Sum(x => x.Units);

        var Onegative = _context.BloodStock.Include(x => x.Account).Where(x => x.Account ==
account).Where(x => x.BloodType == "O-ve").Sum(x => x.Units);

        var Bnegative = _context.BloodStock.Include(x => x.Account).Where(x => x.Account ==
account).Where(x => x.BloodType == "B-ve").Sum(x => x.Units);

         BloodDashboard bloodDashboard = new BloodDashboard

        {

           ABnegative = ABnegative,

           Bnegative = Bnegative,

           Anegative = Anegative,

           Onegative = Onegative,

           Opositive = Opositive,

           ABpositive = ABpositive,

           Apositive = Apositive,

           Bpositive = Bpositive,

        };

        return Ok(bloodDashboard);

      }

   }

}
```

ApproveBloodRequestByAdmin.cs

```
using BloodBankManagementWebapi.ApiModel;

using BloodBankManagementWebapi.Model;
```

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using BloodBankManagementWebapi.OtherOperation;
using Microsoft.VisualStudio.Web.CodeGenerators.Mvc.Templates.BlazorIdentity.Pages.Manage;
using Org.BouncyCastle.Asn1.Ocsp;

namespace BloodBankManagementWebapi.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class ApproveBloodRequestByAdminController : ControllerBase
   {
      private readonly AppDbContext _context;
      public ApproveBloodRequestByAdminController(AppDbContext context)
      {
         _context = context;
      }
      [HttpPost]
      public IActionResult ApproveOrRequestBloodRequest([FromBody] ApproveOrReject
approveOrReject)
      {
          ApproveOrRejectReturnMessage message = null;
         if(approveOrReject.Id==null
|| !_context.BloodRequest.Any(s=>s.BloodRequestId==approveOrReject.Id))
         {
            message = new ApproveOrRejectReturnMessage()
            {
               ValidEmail= false,
```

```
            IdExists= false,
            ChangeStatus =false
        };
         return Ok(message);
    }
    BloodRequest oldRequest=_context.BloodRequest.Where(s => s.BloodRequestId ==
approveOrReject.Id).FirstOrDefault()!;
    if(approveOrReject.Status == true)
    {
        oldRequest.Status = 1;
        oldRequest.ValidTime=DateTime.Now.AddDays(2).ToString();
        _context.BloodRequest.Update(oldRequest);
        if(EmailGenerator.SendEmail(oldRequest.Name, oldRequest.Email, "BloodRequestApprove",
oldRequest.BloodRequestId, oldRequest.ValidTime))
        {
            if (_context.SaveChanges()>0)
            {
                message = new ApproveOrRejectReturnMessage()
                {
                    ValidEmail = true,
                    IdExists = true,
                    ChangeStatus = true
                };
                return Ok(message);
            }
            else
            {
                message = new ApproveOrRejectReturnMessage()
                {
```

```
                ValidEmail = true,
                IdExists = true,
                ChangeStatus = false
            };
            return Ok(message);
        }
    }
    else
    {
        message = new ApproveOrRejectReturnMessage()
        {
            ValidEmail = false,
            IdExists = true,
            ChangeStatus = false
        };
        return Ok(message);
    }
}
else
{
    oldRequest.Status = 0;
    _context.BloodRequest.Update(oldRequest);


    if (EmailGenerator.SendEmail(oldRequest.Name, oldRequest.Email, "BloodRequestReject",
oldRequest.BloodRequestId, oldRequest.ValidTime))
    {
        if (_context.SaveChanges() > 0)
        {
            message = new ApproveOrRejectReturnMessage()
```

```
        {
            ValidEmail = true,
            IdExists = true,
            ChangeStatus = true
        };
        return Ok(message);


    }
    else
    {
        message = new ApproveOrRejectReturnMessage()
        {
            ValidEmail = true,
            IdExists = true,
            ChangeStatus = false
        };
        return Ok(message);
    }
}
else
{
    message = new ApproveOrRejectReturnMessage()
    {
        ValidEmail = false,
        IdExists = true,
        ChangeStatus = false
    };
    return Ok(message);
}
```

```
        }
      }
    }
}
```

ApproveOrRejectAccountByAdminController.cs

```
using BloodBankManagementWebapi.ApiModel;
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.Model;
using BloodBankManagementWebapi.OtherOperation;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
namespace BloodBankManagementWebapi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ApproveOrRejectAccountByAdminController : ControllerBase
    {
        private readonly AppDbContext _context;
        public ApproveOrRejectAccountByAdminController(AppDbContext context)
        {
            _context = context;
        }

        [HttpPost]
        public IActionResult ApproveOrRequestAccount([FromBody] ApproveOrReject approveOrReject)
        {
            ApproveOrRejectReturnMessage message = null;
            if (approveOrReject.Id == null || !_context.Account.Any(s => s.AccountId == approveOrReject.Id))
```

```
        {
            message = new ApproveOrRejectReturnMessage()
            {
                ValidEmail = false,
                IdExists = false,
                ChangeStatus = false
            };
            return Ok(message);
        }
        Account oldRequest = _context.Account.Where(s => s.AccountId ==
approveOrReject.Id).FirstOrDefault()!;
        if (approveOrReject.Status == true)
        {
            oldRequest.Status = 1;
            _context.Account.Update(oldRequest);
            if (EmailGenerator.SendEmailForAccountApproval(oldRequest.Name, oldRequest.Email,
"AccountApprove"))
            {
                if (_context.SaveChanges() > 0)
                {
                    message = new ApproveOrRejectReturnMessage()
                    {
                        ValidEmail = true,
                        IdExists = true,
                        ChangeStatus = true
                    };
                    return Ok(message);
                }
                else
```

```
            {
                message = new ApproveOrRejectReturnMessage()
                {
                    ValidEmail = true,
                    IdExists = true,
                    ChangeStatus = false
                };
                return Ok(message);
            }
        }
        else
        {
            message = new ApproveOrRejectReturnMessage()
            {
                ValidEmail = false,
                IdExists = true,
                ChangeStatus = false
            };
            return Ok(message);
        }
    }
    else
    {
        oldRequest.Status = 2;
        _context.Account.Update(oldRequest);
        if (EmailGenerator.SendEmailForAccountApproval(oldRequest.Name, oldRequest.Email,
"AccountReject"))
        {
            if (_context.SaveChanges() > 0)
```

```
    {
        message = new ApproveOrRejectReturnMessage()
        {
            ValidEmail = true,
            IdExists = true,
            ChangeStatus = true
        };
        return Ok(message);
    }
    else
    {
        message = new ApproveOrRejectReturnMessage()
        {
            ValidEmail = true,
            IdExists = true,
            ChangeStatus = false
        };
        return Ok(message);
    }
}
else
{
    message = new ApproveOrRejectReturnMessage()
    {
        ValidEmail = false,
        IdExists = true,
        ChangeStatus = false
    };
    return Ok(message);
```

```
            }
        }
      }
    }
}
```

BloodCampController.cs

```
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.ApiModel;
using BloodBankManagementWebapi.Model;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
namespace BloodBankManagementWebapi.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class BloodCampController : ControllerBase
   {
      private readonly AppDbContext _context;
      public BloodCampController(AppDbContext context)
      {
         _context = context;
      }
      [HttpPost]
      public IActionResult AddBloodCamp([FromBody]BloodCampModel bloodCampModel)
      {
         BloodCamp bloodCamp = new BloodCamp()
         {
            BloodCampId= Guid.NewGuid().ToString(),
```

```
            BloodCampName= bloodCampModel.BloodCampName,
            BloodCampLocation= bloodCampModel.BloodCampLocation,
            Date= bloodCampModel.Date,
            Time = bloodCampModel.Time
        };
        Account account = _context.Account.Find(bloodCampModel.AccountId);
        BloodCampBloodBank bloodCampBloodBank = new BloodCampBloodBank()
        {
            BloodCampBloodBankId = Guid.NewGuid().ToString(),
            BloodCamp = bloodCamp,
            Account = account
        };
        _context.BloodCamp.Add(bloodCamp);
        _context.BloodCampBloodBank.Add(bloodCampBloodBank);
        _context.SaveChanges();
        return Ok();
      }
    }
}
```

BloodRequestController.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using BloodBankManagementWebapi.Model;
using BloodBankManagementWebapi.ApiModel;
using Microsoft.EntityFrameworkCore;
namespace BloodBankManagementWebapi.Controllers
{
```

```
[Route("api/[controller]/[action]")]
[ApiController]
public class BloodRequestController : ControllerBase
{
    private readonly AppDbContext _context;
    public BloodRequestController(AppDbContext context)
    {
        _context = context;
    }
    [HttpPost]
    public IActionResult RequestBlood([FromBody] BloodRequestDto bloodRequestDto)
    {
        Random rand = new Random();
        if (bloodRequestDto == null)
        {
            return BadRequest();
        }
        BloodRequest bloodRequest= new BloodRequest()
        {
            BloodRequestId = Convert.ToBase64String(Guid.NewGuid().ToByteArray()).Replace("/", "-
").Replace("+", "-").Substring(rand.Next(0, 10), 8),
            Name=bloodRequestDto.Name,
            Email=bloodRequestDto.Email,
            PhoneNumber=bloodRequestDto.PhoneNumber,
            BloodType=bloodRequestDto.BloodType,
            Age=bloodRequestDto.Age,
            Location=bloodRequestDto.Location,
            Units=bloodRequestDto.Units,
            AcceptStatus=0,
```

```
        AadhaarNumber=bloodRequestDto.AadhaarNumber,
        ValidTime=bloodRequestDto.ValidTime,
        Status = bloodRequestDto.Status
    };
    _context.BloodRequest.Add(bloodRequest);
    if (_context.SaveChanges()>0)
    {
        return Ok(bloodRequest.BloodRequestId);
    }
    return BadRequest();
}
[HttpPost]
public IActionResult CheckStatus([FromBody]CheckRequest Request)
{
    var bloodrequest = _context.BloodRequest.Find(Request.Id);
    CheckBloodRequestStatusMessage message = null;
    List<BloodBankStock> bloodstock = new List<BloodBankStock>();
    if (Request == null || !_context.BloodRequest.Any(s=>s.BloodRequestId==Request.Id))
    {
        message = new CheckBloodRequestStatusMessage()
        {
            IdExists = false,
            Status = null,
            bloodRequestBloodBank= bloodstock
        };
        return Ok(message);
    }
    BloodRequest bloodRequest=_context.BloodRequest.Find(Request.Id);
```

```
var BloodBank = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a => a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus == 1).Where(x => x.Account.Status != 0).Where(x => x.UserDetails.Location == bloodrequest!.Location).Select(l => new BloodRequestBloodBank
{
    Id = l.Account.AccountId,
    Name = l.Account.Name,
    Location = l.UserDetails.Location
}).ToList();
if (bloodRequest.Status == 1)
{
    foreach (var bank in BloodBank)
    {
        var o = _context.BloodStock.Include(c => c.Account).Where(c => c.BloodType == bloodrequest.BloodType).Where(d => d.Account.AccountId == bank.Id).ToList().Select(p => new BloodBankStock
        {
            BloodBank = p.Account.Name,
            Location = bank.Location,
            Units = p.Units
        }).FirstOrDefault();
        bloodstock.Add(o);
    };
    message = new CheckBloodRequestStatusMessage()
    {
        IdExists = true,
        Status="approved",
        bloodRequestBloodBank= bloodstock
    };
    return Ok(message);
```

```
        }
        else if(bloodRequest.Status == 2)
        {
            message = new CheckBloodRequestStatusMessage()
            {
                IdExists = true,
                Status = "rejected",
                bloodRequestBloodBank = bloodstock
            };
            return Ok(message);
        }
        else
        {
            message = new CheckBloodRequestStatusMessage()
            {
                IdExists = true,
                Status = "pending",
                bloodRequestBloodBank = bloodstock
            };
            return Ok(message);
        }
    }
  }
}
```

BloodStockController.cs

```
using BloodBankManagementWebapi.ApiModel;
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.Model;
```

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
namespace BloodBankManagementWebapi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class BloodStockController : ControllerBase
    {
        private readonly AppDbContext _context;
        public BloodStockController(AppDbContext context)
        {
            _context = context;
        }
        [HttpPost]
        public IActionResult AddBloodStock([FromBody] BloodStockDto bloodStockDto)
        {
            var bloodstockcount = _context.BloodStock.Include(z => z.Account).Where(x => x.BloodType ==
bloodStockDto.BloodType).Where(i => i.Account.AccountId == bloodStockDto.AccountId).Count();
            if (bloodstockcount == 0)
            {
                Account account = _context.Account.Find(bloodStockDto.AccountId)!;
                BloodStock bloodStock = new BloodStock()
                {
                    BloodStockId = Guid.NewGuid().ToString(),
                    BloodType = bloodStockDto.BloodType,
                    Units = bloodStockDto.Units,
                    Account= account
                };
```

```
            _context.BloodStock.Add(bloodStock);
            _context.SaveChanges();
            return Ok();


        }
        else
        {
            var bloodstock = _context.BloodStock.Include(z => z.Account).Where(x => x.BloodType ==
bloodStockDto.BloodType).Where(i => i.Account.AccountId ==
bloodStockDto.AccountId).FirstOrDefault();
            var blood = _context.BloodStock.Where(x => x.BloodStockId ==
bloodstock.BloodStockId).FirstOrDefault();
             blood.Units += bloodStockDto.Units;
            _context.BloodStock.Update(blood);
            _context.SaveChanges();
            return Ok();
        }
    }
    [Route("GetStockByBank/{id}/{type}")]
    [HttpGet]
    public IActionResult Get(string id,string type)
    {
        var account = _context.Account.Find(id);
        var bloodstock = 0;
        var blood = _context.BloodStock.Include(a => a.Account).Where(e => e.Account ==
account).Where(d => d.BloodType == type).FirstOrDefault();
        if (blood == null)
        {
            bloodstock = 0;
        }
```

```
        else
        {
            bloodstock=blood.Units;
        }
        return Ok(bloodstock);
    }
  }
}
```

BloodTransactionController.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using BloodBankManagementWebapi.ApiModel;
using BloodBankManagementWebapi.Model;
using Microsoft.EntityFrameworkCore;
namespace BloodBankManagementWebapi.Controllers
{
  [Route("api/[controller]/[action]")]
  [ApiController]
  public class BloodTransactionController : ControllerBase
  {
    private readonly AppDbContext _context;
    public BloodTransactionController(AppDbContext appDbContext)
    {
      _context = appDbContext;
    }
    [HttpPost]
```

```
    public IActionResult AddBloodTransaction([FromBody]BloodTransactionModel
bloodTransactionModel)
    {
      if(_context.BloodRequest.Any(x=>x.BloodRequestId==bloodTransactionModel.BloodRequestId)) {
        var account = _context.Account.Find(bloodTransactionModel.AccountId);
        var bloodRequest = _context.BloodRequest.Find(bloodTransactionModel.BloodRequestId);
        bloodRequest.Status = 4;
        _context.BloodRequest.Update(bloodRequest);
        _context.SaveChanges();
        BloodTransaction bloodTransaction = new BloodTransaction()
        {
          BloodTransactionId = Guid.NewGuid().ToString(),
          BloodType = bloodTransactionModel.BloodType,
          units = bloodTransactionModel.units,
          Date = bloodTransactionModel.Date,
          Time = bloodTransactionModel.Time,
          Account = account,
          BloodRequest = bloodRequest
        };
        var bloodStock = _context.BloodStock.Include(x => x.Account).Where(x => x.Account ==
account).Where(x => x.BloodType == bloodTransactionModel.BloodType).Select(x => new BloodStock
        {
          BloodStockId = x.BloodStockId,
          BloodType = x.BloodType,
          Units = x.Units
        }).FirstOrDefault();
        bloodStock.Units = bloodStock.Units - bloodTransactionModel.units;
        _context.BloodStock.Update(bloodStock);
        _context.SaveChanges();
```

```
            _context.BloodTransaction.Add(bloodTransaction);
            _context.SaveChanges();
            return Ok();
        }
            return BadRequest();
    }
    [HttpPost]
    public IActionResult GetBloodTransaction([FromBody]GetBloodTransaction getBloodTransaction)
    {
        var bloodTransaction = _context.BloodTransaction.Include(x => x.Account).Include(x =>
x.BloodRequest).Where(x => x.Account.AccountId == getBloodTransaction.Id).ToList();
        return Ok(bloodTransaction);
    }
  }
}
```

ChangePasswordController.cs

```
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.Model;
using BloodBankManagementWebapi.OtherOperation;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using BloodBankManagementWebapi.ApiModel;

namespace BloodBankManagementWebapi.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
```

```
public class ChangePasswordController : ControllerBase
{
    private readonly AppDbContext _context;
    public ChangePasswordController(AppDbContext context)
    {
        _context = context;
    }
    [HttpPost]
    public IActionResult ChangePassword([FromBody] ChangePassword changePassword)
    {
        ChangePasswordMessage changePasswordMessage;
        if (_context.Account.Any(user => user.Email == changePassword.Email))
        {
            Account olduser = _context.Account.Where(s => s.Email ==
changePassword.Email).FirstOrDefault()!;
            if (olduser.Password ==
SHA256Encrypt.ComputePasswordToSha256Hash(changePassword.OldPassword))
            {
                olduser.Password =
SHA256Encrypt.ComputePasswordToSha256Hash(changePassword.NewPassword);
                _context.Account.Update(olduser);
                _context.SaveChanges();
                changePasswordMessage = new ChangePasswordMessage()
                {
                    EmailExists = true,
                    Passcheck = true
                };
                return Ok(changePasswordMessage);
            }
            else
```

```
            {
                changePasswordMessage = new ChangePasswordMessage()
                {
                    EmailExists = true,
                    Passcheck = false
                };
                return Ok(changePasswordMessage);
            }
        }
        else
        {
            changePasswordMessage = new ChangePasswordMessage()
            {
                EmailExists = false,
                Passcheck = false
            };
            return Ok(changePasswordMessage);
        }
    }
  }
}
```

DeleteBloodCampController.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
namespace BloodBankManagementWebapi.Controllers
{
```

```
  [Route("api/[controller]")]
  [ApiController]
  public class DeleteBloodCampController : ControllerBase
  {
    private readonly AppDbContext _context;


    public DeleteBloodCampController(AppDbContext context)
    {
      _context = context;
    }
    [Route("/BloodCamp/{id}")]
    [HttpDelete]
    public IActionResult Delete(string id)
    {
      var bloodCampBank = _context.BloodCampBloodBank.Where(s => s.BloodCamp.BloodCampId ==
id).FirstOrDefault();
      var bloodCamp=_context.BloodCamp.Find(id);
      _context.BloodCampBloodBank.Remove(bloodCampBank);
      _context.BloodCamp.Remove(bloodCamp);
      _context.SaveChanges();
      return Ok();
    }
  }
}
```

ForgetPasswordController.cs

```
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.OtherOperation;
using Microsoft.AspNetCore.Http;
```

```
using BloodBankManagementWebapi.ApiModel;
using BloodBankManagementWebapi.Model;
using Microsoft.AspNetCore.Mvc;

namespace BloodBankManagementWebapi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ForgetPasswordController : ControllerBase
    {
        private readonly AppDbContext _context;
        public ForgetPasswordController(AppDbContext context)
        {
            _context= context;
        }
        [HttpPost]
        public IActionResult SendMailForPassword([FromBody] EmailClass Mail)
        {
            ForgetPasswordMessage message = null;
            if (!_context.Account.Any(s => s.Email == Mail.Email))
            {
                message = new ForgetPasswordMessage
                {
                    EmailExists = false,
                    SendMail = false
                };
                return Ok(message);
            }
            string Password = RandomPasswordGeneration.RandomPasswordGenerator();
```

```
        Account olduser = _context.Account.Where(s => s.Email == Mail.Email).FirstOrDefault()!;
        olduser.Password = SHA256Encrypt.ComputePasswordToSha256Hash(Password);
        //_context.Entry(olduser).State = EntityState.Detached;
        _context.Account.Update(olduser);
        _context.SaveChangesAsync();
        if (EmailGenerator.SendEmail(olduser.Name, olduser.Email, Password))
        {
            message = new ForgetPasswordMessage
            {
                EmailExists = true,
                SendMail = true
            };
            return Ok(message);
        }
        else
        {
            message = new ForgetPasswordMessage
            {
                EmailExists = true,
                SendMail = false
            };
            return Ok(message);
        }
    }
  }
}
```

LoginController.cs

```
using BloodBankManagementWebapi.ApiModel;
```

```
using BloodBankManagementWebapi.Model;
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.OtherOperation;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace BloodBankManagementWebapi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class LoginController : ControllerBase
    {
        private readonly AppDbContext _context;
        private readonly IConfiguration _configuration;
        public LoginController(AppDbContext context, IConfiguration configuration)
        {
            _context = context;
            _configuration = configuration;
        }
        [HttpPost]
        public IActionResult Login([FromBody] LoginModel loginModel)
        {
            AuthMessageModel authmessage;
            List<Claim> authClaim;
            if (loginModel != null)
            {
                if (_context.Account.Any(user => user.Email == loginModel.Email))
                {
```

```
        Account user = _context.Account.Where(user => user.Email ==
loginModel.Email).FirstOrDefault()!;
        if (user.Status==1)
        {
          if (user.Password ==
SHA256Encrypt.ComputePasswordToSha256Hash(loginModel.Password))
          {
            var accountroleid = _context.AccountRole.Where(userrole => userrole.Account == user)
              .Select(user => user.AccountRoleId)
              .FirstOrDefault()!;
            var roleid = _context.AccountRole.Where(ur => ur.AccountRoleId == accountroleid) // Filter
by userrole id
                          .Select(ur => ur.Role.RoleId) // Select the role id
                          .FirstOrDefault();
            authClaim = new List<Claim>()
          {
            new Claim("Id",user.AccountId),
            new Claim("Role",roleid!)
          };
            var token = Jwt.GenerateJWTToken(authClaim, _configuration);
            authmessage = new AuthMessageModel
            {
              AccountApproval = "approve",
              AccountExists = true,
              PasswordStatus = true,
              Token = token
            };
            return Ok(authmessage);
          }
          authmessage = new AuthMessageModel
```

```
            {
                AccountApproval = "approve",
                AccountExists = true,
                PasswordStatus = false,
                Token = null
            };
            return Ok(authmessage);


        }
        else if(user.Status==2)
        {
            if (user.Password ==
SHA256Encrypt.ComputePasswordToSha256Hash(loginModel.Password))
            {
                var accountroleid = _context.AccountRole.Where(userrole => userrole.Account == user)
                    .Select(user => user.AccountRoleId)
                    .FirstOrDefault()!;
                var roleid = _context.AccountRole.Where(ur => ur.AccountRoleId == accountroleid) // Filter
by userrole id
                                .Select(ur => ur.Role.RoleId) // Select the role id
                                .FirstOrDefault();
                authClaim = new List<Claim>()
            {
                new Claim("Id",user.AccountId),
                new Claim("Role",roleid!)
            };
                var token = Jwt.GenerateJWTToken(authClaim, _configuration);
                authmessage = new AuthMessageModel
                {
```

```
                AccountApproval = "reject",
                AccountExists = true,
                PasswordStatus = true,
                Token = token
            };
            return Ok(authmessage);
        }
        authmessage = new AuthMessageModel
        {
            AccountApproval = "reject",
            AccountExists = true,
            PasswordStatus = false,
            Token = null
        };
        return Ok(authmessage);
    }
    else
    {
        if (user.Password ==
SHA256Encrypt.ComputePasswordToSha256Hash(loginModel.Password))
        {
            var accountroleid = _context.AccountRole.Where(userrole => userrole.Account == user)
                .Select(user => user.AccountRoleId)
                .FirstOrDefault()!;
            var roleid = _context.AccountRole.Where(ur => ur.AccountRoleId == accountroleid) // Filter
by userrole id
                                .Select(ur => ur.Role.RoleId) // Select the role id
                                .FirstOrDefault();
            authClaim = new List<Claim>()
```

```
        {
            new Claim("Id",user.AccountId),
            new Claim("Role",roleid!)
        };
            var token = Jwt.GenerateJWTToken(authClaim, _configuration);
            authmessage = new AuthMessageModel
            {
                AccountApproval = "pending",
                AccountExists = true,
                PasswordStatus = true,
                Token = token
            };
            return Ok(authmessage);
        }
        authmessage = new AuthMessageModel
        {
            AccountApproval = "pending",
            AccountExists = true,
            PasswordStatus = false,
            Token = null
        };
        return Ok(authmessage);


    }
}
authmessage = new AuthMessageModel
{
    AccountApproval = "pending",
```

```
            AccountExists = false,
            PasswordStatus = false,
            Token = null
          };
          return Ok(authmessage);
        }
        return BadRequest();


      }
    }
}
```

RegisterController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.OtherOperation;
using System.Security.Claims;
using BloodBankManagementWebapi.ApiModel;
using BloodBankManagementWebapi.Model;
using static System.Net.Mime.MediaTypeNames;
using System.Text;
namespace BloodBankManagementWebapi.Controllers
{
   [Route("api/[controller]/[action]")]
   [ApiController]
   public class RegisterController : ControllerBase
   {
      private readonly AppDbContext _context;
```

```
//private readonly IConfiguration _configuration;
public RegisterController(AppDbContext context)
{
    _context = context;
//    _configuration = configuration;
}
[HttpPost]


public IActionResult PostUser(RegisterModel registerUser)
{
    SignUpMessageModel message = null;
    if (_context.Account.Any(s => s.Email == registerUser.Email) || _context.Account.Any(s =>
s.PhoneNumber == registerUser.PhoneNumber))
    {
        if (_context.Account.Any(s => s.Email == registerUser.Email) && _context.Account.Any(s =>
s.PhoneNumber == registerUser.PhoneNumber))
        {
            message = new SignUpMessageModel()
            {
                EmailExists = true,
                MobileNumberExists = true
            };
            return Ok(message);
        }
        if (_context.Account.Any(s => s.PhoneNumber == registerUser.PhoneNumber))
        {
            message = new SignUpMessageModel()
            {
                EmailExists = false,
```

```
            MobileNumberExists = true
        };
        return Ok(message);
    }
    message = new SignUpMessageModel()
    {
        EmailExists = true,
        MobileNumberExists = false
    };
    return Ok(message);
}
else
{
    Account account = new Account()
    {
        AccountId = Guid.NewGuid().ToString(),
        Name = registerUser.Name,
        Email = registerUser.Email,
        Password = SHA256Encrypt.ComputePasswordToSha256Hash(registerUser.Password),
        PhoneNumber = registerUser.PhoneNumber,
        Status = 0,

    };
    UserDetails userDetails = new UserDetails()
    {
        UserDetailsId = Guid.NewGuid().ToString(),
        Age = registerUser.Age,
        Location = registerUser.Location,
        GovernmentId = registerUser.GovernmentId,
```

```
        Document = Encoding.ASCII.GetBytes(registerUser.Document),
        AadhaarNumber = registerUser.AadhaarNumber,
        BloodType = registerUser.BloodType,




    };
    if (registerUser.Role=="ADMIN")
    {
        userDetails.rolestatus = 0;
    }
    else if(registerUser.Role == "HOSPITAL")
    {
        userDetails.rolestatus = 2;


    }
    else if (registerUser.Role == "DONOR")
    {
        userDetails.rolestatus = 3;


    }
    else
    {
        userDetails.rolestatus = 1;


    }
    Address address = new Address()
    {
        AddressId = Guid.NewGuid().ToString(),
```

```
    City = registerUser.City,
    DoorNo = registerUser.DoorNo,
    Street = registerUser.Street,
    State = registerUser.State,
    PostalCode = registerUser.PostalCode,
    Area = registerUser.Area,

};
AccountUserDetailsAddress accountUserDetailsAddress = new AccountUserDetailsAddress()
{
    AccountUserDetailsAddressId=Guid.NewGuid().ToString(),
    Address = address,
    Account=account,
    UserDetails=userDetails
};


Role role = _context.Role.Where(x => x.RoleName == registerUser.Role).FirstOrDefault()!;
AccountRole accountRole = new AccountRole()
{
    AccountRoleId = Guid.NewGuid().ToString(),
    Role = role,
    Account = account
};
_context.Account.Add(account);
_context.UserDetails.Add(userDetails);
_context.Address.Add(address);
_context.AccountUserDetailsAddress.Add(accountUserDetailsAddress);
_context.AccountRole.Add(accountRole);
```

```
            _context.SaveChanges();


            message = new SignUpMessageModel()
            {
                EmailExists = false,
                MobileNumberExists = false
            };
            return Ok(message);
        }


    }
  }
}
```

RoleController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.Model;
namespace BloodBankManagementWebapi.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class RoleController : ControllerBase
  {
    private readonly AppDbContext _context;
    public RoleController(AppDbContext context)
    {
      _context = context;
```

```
        }
        [HttpPost]
        public IActionResult PostRole([FromBody] Role roles)
        {
            Role roles1 = new Role()
            {
                RoleId = Guid.NewGuid().ToString(),
                RoleName = roles.RoleName.ToUpper()
            };
            _context.Role.Add(roles1);
            _context.SaveChanges();
            return Ok();
        }

    }
}
```

ViewAcceptedBankController.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using BloodBankManagementWebapi.Model;
using Microsoft.EntityFrameworkCore;

namespace BloodBankManagementWebapi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ViewAcceptedBankController : ControllerBase
```

```
  {
    private readonly AppDbContext _context;
    public ViewAcceptedBankController(AppDbContext context)
    {
      _context = context;
    }
    [Route("/GetBank/{id}")]
    [HttpGet]
    public IActionResult GetBankbyId(string id)
    {
      if (_context.BloodRequest.Any(x=>x.AcceptStatus==1))
      {
        var requester = _context.BloodRequest.Find(id);
        var bank = _context.bloodStockRequesters.Include(x => x.UserDetails).Include(x =>
x.Account).Include(x => x.BloodRequest).Include(x => x.Address).Where(x => x.BloodRequest ==
requester).ToList();
        return Ok(bank);
      }
      else
      {
        return Ok(null);
      }


    }
  }
}
```

ViewAccountRequestController.cs

```
using BloodBankManagementWebapi.DataContext;
```

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Text.Json.Serialization;
using System.Text.Json;
using BloodBankManagementWebapi.Model;
using Microsoft.IdentityModel.Tokens.Configuration;
using System.Text;
using BloodBankManagementWebapi.ApiModel;



namespace BloodBankManagementWebapi.Controllers
{
    [Route("api/[controller]/[action]")]
    [ApiController]
    public class ViewAccountRequestController : ControllerBase
    {
        private readonly AppDbContext _context;
        public ViewAccountRequestController(AppDbContext context)
        {
            _context = context;
        }
        [HttpGet]
        public IActionResult GetAccountDetails()
        {


            var accountDetails =
_context.AccountUserDetailsAddress.Include(x=>x.UserDetails).Include(a=>a.Account).Include(s=>s.A
ddress).ToList();
```

```
        return Ok(accountDetails);
    }
    [HttpGet]
    public IActionResult GetAccountDetailsById(string id)
    {
        var accountDetails = _context.UserDetails.Select(l => new UserDetailsDto
        {
            Document = Encoding.ASCII.GetString(l.Document),
        });


        return Ok(accountDetails);
    }
    [HttpGet]
    public IActionResult GetDonorById(string id)
    {
        var account = _context.Account.Find(id);
        var accountDetails = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a
=> a.Account).Include(s => s.Address).ToList().Where(x => x.UserDetails.rolestatus == 3).Where(x =>
x.Account.Status != 0).Where(x=>x.Account==account);


        return Ok(accountDetails);
    }
    //Approved Donor
    [HttpGet]
    public IActionResult GetAccountDetailsDonor()
    {
```

```
        var accountDetails = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a => a.Account).Include(s => s.Address).ToList().Where(x => x.UserDetails.rolestatus == 3).Where(x => x.Account.Status != 0);


        return Ok(accountDetails);
    }
    [HttpGet]
    public IActionResult GetAccountDetailsHospital()
    {


        var accountDetails = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a => a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus == 2).Where(x => x.Account.Status != 0).Select(l => new HospitalBloodBankDetails
        {
            Id=l.Account.AccountId,
            Name = l.Account.Name,
            Email = l.Account.Email,
            PhoneNumber = l.Account.PhoneNumber,
            Location = l.UserDetails.Location,
            GovernmentId = l.UserDetails.GovernmentId,


            Document = Encoding.ASCII.GetString(l.UserDetails.Document),
            DoorNo = l.Address.DoorNo,
            Street =l.Address.Street,
            Area =l.Address.Area,
            City =l.Address.City,
            State =l.Address.State,
            PostalCode =l.Address.PostalCode,
            Status=l.Account.Status
        }) ;
```

```
        return Ok(accountDetails);
    }
    [HttpGet]
    public IActionResult GetAccountDetailsBloodBank()
    {

        var accountDetails = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a
=> a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus == 1).Where(x =>
x.Account.Status != 0).Select(l => new HospitalBloodBankDetails
        {
            Id=l.Account.AccountId,
            Name = l.Account.Name,
            Email = l.Account.Email,
            PhoneNumber = l.Account.PhoneNumber,
            Location = l.UserDetails.Location,
            GovernmentId = l.UserDetails.GovernmentId,

            Document = Encoding.ASCII.GetString(l.UserDetails.Document),
            DoorNo = l.Address.DoorNo,
            Street = l.Address.Street,
            Area = l.Address.Area,
            City = l.Address.City,
            State = l.Address.State,
            PostalCode = l.Address.PostalCode,
            Status = l.Account.Status
        });
        return Ok(accountDetails);
```

```
    }
    [HttpGet]
    public IActionResult GetAccountDetailsPendingBloodBank()
    {

        var accountDetails = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a
=> a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus == 1).Where(x =>
x.Account.Status == 0).Select(l => new HospitalBloodBankDetails
        {
            Id = l.Account.AccountId,

            Name = l.Account.Name,
            Email = l.Account.Email,
            PhoneNumber = l.Account.PhoneNumber,
            Location = l.UserDetails.Location,
            GovernmentId = l.UserDetails.GovernmentId,

            Document = Encoding.ASCII.GetString(l.UserDetails.Document),
            DoorNo = l.Address.DoorNo,
            Street = l.Address.Street,
            Area = l.Address.Area,
            City = l.Address.City,
            State = l.Address.State,
            PostalCode = l.Address.PostalCode,
            Status = l.Account.Status
        });
        return Ok(accountDetails);
    }
    [HttpGet]
```

```
public IActionResult GetAccountDetailsPendingDonor()
{

    var accountDetails = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a => a.Account).Include(s => s.Address).ToList().Where(x => x.UserDetails.rolestatus == 3).Where(x => x.Account.Status == 0);

    return Ok(accountDetails);
}
[HttpGet]
public IActionResult GetAccountDetailsPendingHospital()
{

    var accountDetails = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a => a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus == 2).Where(x => x.Account.Status == 0).Select(l => new HospitalBloodBankDetails
    {
        Id= l.Account.AccountId,
        Name = l.Account.Name,
        Email = l.Account.Email,
        PhoneNumber = l.Account.PhoneNumber,
        Location = l.UserDetails.Location,
        GovernmentId = l.UserDetails.GovernmentId,

        Document = Encoding.ASCII.GetString(l.UserDetails.Document),
        DoorNo = l.Address.DoorNo,
        Street = l.Address.Street,
        Area = l.Address.Area,
        City = l.Address.City,
        State = l.Address.State,
```

```
            PostalCode = l.Address.PostalCode,
            Status = l.Account.Status
        });



        return Ok(accountDetails);
    }
  }



}
```

ViewBloodCampController.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace BloodBankManagementWebapi.Controllers
{
  [Route("api/[controller]/[action]")]
  [ApiController]
  public class ViewBloodCampController : ControllerBase
  {
    private readonly AppDbContext _context;
    public ViewBloodCampController(AppDbContext context)
    {
      _context = context;
    }
```

```
    [HttpGet]
    public IActionResult Get()
    {
        var BloodCamp= _context.BloodCampBloodBank.Include(x =>
x.BloodCamp).Include(x=>x.Account).ToList();
        return Ok(BloodCamp);


    }
    [HttpGet]
    public IActionResult GetByIndividual(string id)
    {
        var account = _context.Account.Find(id);
        var BloodCamp = _context.BloodCampBloodBank.Include(x => x.BloodCamp).Include(x =>
x.Account).Where(x=>x.Account==account);
        return Ok(BloodCamp);


    }


  }
}
```

ViewBloodRequestByBankController.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
namespace BloodBankManagementWebapi.Controllers
{
  [Route("api/[controller]")]
```

```
    [ApiController]
    public class ViewBloodRequestByBankController : ControllerBase
    {
        private readonly AppDbContext _context;
        public ViewBloodRequestByBankController(AppDbContext context)
        {
            _context = context;
        }
        [Route("/Get/{id}")]
        [HttpGet]
        public IActionResult Get(string id) {
            var account= _context.Account.Find(id);
            var bank = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(x =>
x.Account).Where(x => x.Account == account).FirstOrDefault();
            var bloodrequest = _context.BloodRequest.Where(x => x.Location ==
bank.UserDetails.Location).Where(x => x.Status == 1).ToList();
            return Ok(bloodrequest);
        }


    }
}
```

ViewBloodRequestController.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace BloodBankManagementWebapi.Controllers
{
```

```
    [Route("api/[controller]")]
    [ApiController]
    public class ViewBloodRequestController : ControllerBase
    {
        private readonly AppDbContext _context;
        public ViewBloodRequestController(AppDbContext context)
        {
            _context = context;
        }
        [HttpGet]
        public IActionResult Get()
        {
            var BloodRequest = _context.BloodRequest.ToList();
            return Ok(BloodRequest);
        }
    }
}
```

ViewBloodStockasWholeController.cs

```
using BloodBankManagementWebapi.ApiModel;
using BloodBankManagementWebapi.DataContext;
using BloodBankManagementWebapi.Model;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace BloodBankManagementWebapi.Controllers
{
    [Route("api/[controller]")]
```

```
[ApiController]
public class ViewBloodStockasWholeController : ControllerBase
{
    private readonly AppDbContext _context;
    public ViewBloodStockasWholeController(AppDbContext context)
    {
        _context = context;
    }

    [HttpGet]
    public IActionResult GetAllStock()
    {
        List<StockAllBank> bloodstock = new List<StockAllBank>();
        var BloodBank = _context.AccountUserDetailsAddress.Include(x => x.UserDetails).Include(a =>
a.Account).Include(s => s.Address).Where(x => x.UserDetails.rolestatus == 1).Where(x =>
x.Account.Status != 0).Select(l => new BloodRequestBloodBank
        {
            Id = l.Account.AccountId,
            Name = l.Account.Name,
            Location = l.UserDetails.Location



        }).ToList();

        foreach (var bank in BloodBank)
        {
            var o = _context.BloodStock.Include(c => c.Account).ToList().Select(p => new StockAllBank
            {
                BloodBankName=bank.Name,
```

```
                Location=bank.Location,
                BloodType=p.BloodType,
                Units=p.Units,
                Mobile=p.Account.PhoneNumber

            }).ToList();
            foreach(var i in o)
            {
                bloodstock.Add(i);
            }
        };

        return Ok(bloodstock);
    }
  }
}
```

ViewBloodStockController.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace BloodBankManagementWebapi.Controllers
{
  [Route("api/[controller]/[action]")]
  [ApiController]
  public class ViewBloodStockController : ControllerBase
  {
```

```
    private readonly AppDbContext _context;
    public ViewBloodStockController(AppDbContext context)
    {
      _context = context;
    }
    [HttpGet]
    public ActionResult ViewBloodStock()
    {
      var BloodStock= _context.BloodStock.Include(y=>y.Account).ToList();
      return Ok(BloodStock);
    }
    [HttpGet]
    public ActionResult ViewBloodStockByIndividual(string id)


    {
      var account = _context.Account.Find(id);
      var BloodStock = _context.BloodStock.Include(y =>
y.Account).ToList().Where(x=>x.Account==account);
      return Ok(BloodStock);
    }
  }
}
```

ViewHospitalController.cs

```
using BloodBankManagementWebapi.DataContext;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
namespace BloodBankManagementWebapi.Controllers
```

```
{
    [Route("api/[controller]")]
    [ApiController]
    public class ViewHospitalController : ControllerBase
    {
        private readonly AppDbContext _context;
        public ViewHospitalController(AppDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        public IActionResult GetAccountDetails()
        {
            var role = _context.Role.Where(s => s.RoleName == "HOSPITAL").FirstOrDefault();
            var hospitalaccount =
_context.AccountRole.Include(s=>s.Account).Include(a=>a.Role).Where(e=>e.Role==role).ToList();

            return Ok(hospitalaccount);
        }
    }
}
```

Output

SWAGGER OUTPUT

ADMIN DASHBOARD

Accept Requester By Bank

Approve Blood Request By Admin

Approve Or Reject Account By Admin

Blood camp

View Hospital

View Blood Stock As Whole

View Blood Request

View Blood Camp

By individual blood bank

View Blood Request by Bank

View Accepted bank

Delete blood camp

Forgot Password

Login

Role

Change Password

View Blood Stock

View Blood Stock By Bank

View Account Details

View Donor By Id

Get Donor Details

View Hospital Details

View Blood Bank Details

View Pending Bank Request

View Pending Donor Request

View Pending Hospital Request

Register

Add Blood Request

Check Request Status

Add Blood Stock

View Stock by bank and type

Add blood transaction

SONARQUBE REPORT

SonarQube – Checking quality of code

Step –1 :Start the sonarqube. Open the sonarqube.

Step – 2 :Create the project manually

Step – 3 : Name the project and key and click the setup button.

Step – 4: Now go with locally to analysis the project locally

Step – 5 : Generate the token for 30 days.

Step – 6 : Token is provided , then click on continue to continue the project.

Step – 7 : Then Choose the project .net => ,net core to analysis the dotnet webapi project.

Step – 8 : Copy the command .

Step – 9 : Copy the command:

Dotnet tool install –global dotnet-sonarscanner.

Ignore this step , if it is already installed.

Step – 10 : dotnet sonarscanner begin /k:”MicroProjectSonarQube” /d:sonar.host.url=”http://localhost:9000” /d:sonar.login=”sqp_5ea205519cbd1b188cbefa91b6c97144e9b123c”

This begin the scanner in your project by creating the link between scanner and project.

Step – 11 : Then run dotnet build command .

Step –12 : Finally run the command

Dotnet sonarscanner end /d:sonar.login=”sqp_5ea2a05519cbd1b188cbefa91b6c97144e9b123c”

This help to end the analysis and send report to sonarqube

Step – 13:Check the report in sonarqube

Intially it contain bug, vulnerability and code smell.

Step – 14: After clearing all issues, rerun the command to send the report to sonarqube

Step –15 : Check the sonarqube final report with 0 bug,0 vulnerability and 0 codesmells.

Step – 16: Finally end the analysis.

Graph:

Security hotspot

Measure

TESTING REPORT

Manual Testing

Project Name :	Blood Bank Management System
Module Name :	Admin Module, Donor Module, Blood Bank Module, Hospital Module, Blood Request Module, Login and register page, welcome page
Created By :	Mano Selvaraj - 11991

Test Case ID	Test Scenario	Test Case	Pre-Condition	Test Step	Test Data	Expected Result	Post Condition	Actual Result	Status (Pass/ Fail)
BBMS_0001	Verify Login Page	Check login with non exists email	Non exists email and correct password	Enter the non exists email and correct password	Non exists email	Error message shown like"Email is not exists"	Login page with error	Error message shown like"Email is not exists"	Pass
BBMS_0002	Verify Login Page	Check login with non invalid password	exists email and incorrect password	Enter the exists email and incorrect password	Incorrect password	Error message shown like"Invalid Password"	Login page with error	Error message shown like"Invalid Password"	Pass
BBMS_0003	Verify Login Page	Check login wth empty data	_	Click Login button with no data entry	_	Error message shown like"*Required! enter a email" and "*Required!enter a password"	Login page with error	Error message shown like"*Required!enter a email" and "*Required!enter a password"	Pass
BBMS_0004	Verify Login Page	Check login with correct data , redirect to home page	Correct email and password	Enter the exists email and correct password	Correct email and password	Redirect to home page	Home page	Redirect to home page	Pass
BBMS_	Verify Forget Password Page	Check forget password with non	Non exists email	Enter non exist email	Non exists email	Error message shown like"Email is not exists"	Error message is	Error message shown like"Email is not exists"	Pass

0005		existe email					displayed		
BBMS_0006	Verify Forget Password Page	Check with correct email	exists email	Enter the exists email	Exists email	Sent Mail with random password and redirects to change password page	Change Password Page	Sent Mail with random password and redirects to change password page	Pass
BBMS_0007	Verify Change Password Page	Check with incorrect password sent in email	Incorrect Password	Enter incorrect password	Incorrect password	Error message shown like "Old Password is wrong"	Error message is displayed	Error message shown like "Old Password is wrong"	Pass
BBMS_0008	Verify Change Password Page	Check with correct old password	correct old password	Enter the correct old password	Correct password	Redirect to login page	Login page	Redirect to login page	Pass
BBMS_0009	Login with admin credential goes to admin dashboard	Check with admin email and password	admin email and password	Enter the admin email and password	admin credential	Redirect to admin dashboard	Admin Dashboard	Redirect to admin dashboard	Pass
BBMS_0010	Login with blood bank credential goes to blood bank dashboard	Check with blood bank email and password	blood bank email and password	Enter the blood bank email and password	blood bank credential	Redirect to blood bank dashboard	Blood bank Dashboard	Redirect to blood bank dashboard	Pass
BBMS_0011	Login with hospital credential goes to hospital dashboard	Check with hospital email and password	hospital email and password	Enter the hospital email and password	hospital credential	Redirect to hospital dashboard	Hospital Dashboard	Redirect to hospital dashboard	Pass

BBMS_0012	Login with donor credential goes to donor home	Check with donor email and password	donor email and password	Enter the donor email and password	donor credential	Redirect to donor home page	Donor home page	Redirect to donor home page	Pass
BBMS_0013	Verify Blood Request Page	Check with entering all data	correct data	Enter the correct data and click the submit button	Correct data	Redirect to pending page with request id	Pending page	Redirect to pending page with request id	Pass
BBMS_0014	Verify Blood Request Page	Check with entering no data	no data	Click button without entering data	_	Error message shown here like "**Required!" in all field	Error message is displayed	Error message shown here like "**Required!" in all field	Pass
BBMS_0015	Verify the check blood request page	Check with incorrect id	incorrect id	Enter the incorrect id	incorrect id	Error message shown here like "Invalid Id"	Error message is displayed	Error message shown here like "Invalid Id"	Pass
BBMS_0016	Verify the check blood request page	Check with correct pending id	correct pending request id	Enter the correct pending request id	correct pending request id	Error message shown here like "Pending"	Error message is displayed	Error message shown here like "Pending"	Pass
BBMS_0017	Verify the check blood request page	Check with correct rejected id	correct rejected request id	Enter the correct rejected request id	correct rejected request id	Error message shown here like "Rejected"	Error message is displayed	Error message shown here like "Rejected"	Pass
BBMS_0018	Verify the check blood request page	Check with correct approved id	correct approved request id	Enter the correct approved request id	correct approved request id	List of available blood bank is shown with accepted bank is also shown	Bank details is displayed	List of available blood bank is shown with accepted bank is also shown	Pass
BBMS_0019	Verify the donor registration page	Check with entering all data	correct data	Enter the correct data and click the submit button	Correct data	Redirect to pending page	Pending page	Redirect to pending page	Pass

BBMS_0020	Verify the donor registration page	Check with entering no data	no data	Click button without entering data	_	Error message shown here like "**Required!" in all field	Error message is displayed	Error message shown here like "**Required!" in all field	Pass
BBMS_0021	Verify the blood bank registration page	Check with entering all data	correct data	Enter the correct data and click the submit button	Correct data	Redirect to pending page	Pending page	Redirect to pending page	Pass
BBMS_0022	Verify the blood bank registration page	Check with entering no data	no data	Click button without entering data	_	Error message shown here like "**Required!" in all field	Error message is displayed	Error message shown here like "**Required!" in all field	Pass
BBMS_0023	Verify the hospital registration page	Check with entering all data	correct data	Enter the correct data and click the submit button	Correct data	Redirect to pending page	Pending page	Redirect to pending page	Pass
BBMS_0024	Verify the hospital registration page	Check with entering no data	no data	Click button without entering data	_	Error message shown here like "**Required!" in all field	Error message is displayed	Error message shown here like "**Required!" in all field	Pass
BBMS_0025	Verify the Login Page	Check the login with rejected or pending account details	correct email and password where account status is pending / rejected	Enter the correct email and password where account status is pending / rejected and click the login button	correct email and password where account status is pending / rejected	Redirect to pending or rejected page	Pending/Rejected Page	Redirect to pending or rejected page	Pass

BBMS_0026	Verify the add blood stock by blood bank user	Check the blood stock after adding the blood stock	blood type and units	Enter blood type and units	blood type and units	Blood Stock is increased	Check the stock in view page	Blood Stock is increased	Pass
BBMS_0027	Verify the add blood stock by blood bank user	Check the blood stock with no data	_	No data entered and click the add button	_	Error message shown here like "**Required!" in all field	Error message is displayed	Error message shown here like "**Required!" in all field	Pass
BBMS_0028	Verify the add blood camp by blood bank user	Check the blood camp after adding the camp	Blood Camp name,location,date and time	Enter the Blood Camp name,location,date and time data and click add blood camp button	Blood Camp name, location,date and time	Blood camp is added	Check the camp data in view page	Blood camp is added	Pass
BBMS_0029	Verify the add blood camp by blood bank user	Check with no data	_	No data entered and click the add button	_	Error message shown here like "**Required!" in all field	Error message is displayed	Error message shown here like "**Required!" in all field	Pass
BBMS_0030	Verify the add blood transaction by blood bank user	Check the blood transaction after adding the blood transaction details	request id, blood type and units	Enter the request id, blood type and units	request id, blood type and units	Blood Stock is decreased and data is added in view page	Check data in view page	Blood Stock is decreased and data is added in view page	Pass
BBMS_0031	Verify the add blood transaction by blood bank user	Check with no data	_	No data entered and click the add button	_	Error message shown here like "**Required!" in all field	Error message is displayed	Error message shown here like "**Required!" in all field	Pass

Selenium

Output

Check Blood Bank Page Visibility

```
using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Threading;

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using OpenQA.Selenium.Firefox;

using OpenQA.Selenium.Remote;

using OpenQA.Selenium.Support.UI;

using OpenQA.Selenium.Interactions;

using NUnit.Framework;

namespace Selenium

{

   [TestFixture]

   public class ViewBankTest

   {
```

```
    private IWebDriver driver;
    public IDictionary<string, object> vars { get; private set; }
    private IJavaScriptExecutor js;
    [SetUp]
    public void SetUp()
    {
      driver = new ChromeDriver();
      js = (IJavaScriptExecutor)driver;
      vars = new Dictionary<string, object>();
    }
    [TearDown]
    protected void TearDown()
    {
      driver.Quit();
    }
    [Test]
    public void viewBank()
    {
      driver.Navigate().GoToUrl("http://localhost:3000/admindashboard");
      driver.Manage().Window.Size = new System.Drawing.Size(1552, 840);
      driver.FindElement(By.CssSelector(".nav-item:nth-child(4) .d-none")).Click();
      driver.Close();
    }
  }

}
```

Check Dashboard Page Visibility

```
using System;
using System.Collections;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
namespace Selenium
{
  [TestFixture]
  public class ViewDashboardTest
  {
    private IWebDriver driver;
    public IDictionary<string, object> vars { get; private set; }
    private IJavaScriptExecutor js;
    [SetUp]
    public void SetUp()
    {
      driver = new ChromeDriver();
      js = (IJavaScriptExecutor)driver;
      vars = new Dictionary<string, object>();
    }
    [TearDown]
    protected void TearDown()
    {
      driver.Quit();
```

```
    }
    [Test]
    public void viewDashboard()
    {
      driver.Navigate().GoToUrl("http://localhost:3000/admindashboard");
      driver.Manage().Window.Size = new System.Drawing.Size(1552, 840);
      driver.FindElement(By.CssSelector(".nav-item:nth-child(1) .d-none")).Click();
      driver.FindElement(By.CssSelector(".nav-item:nth-child(1) .d-none")).Click();
      driver.Close();
    }
  }

}
```

Check Hospital Page Visibility

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
namespace Selenium
{
```

```
  [TestFixture]
  public class AdminDashboardViewHospitalTest
  {
    private IWebDriver driver;
    public IDictionary<string, object> vars { get; private set; }
    private IJavaScriptExecutor js;
    [SetUp]
    public void SetUp()
    {
      driver = new ChromeDriver();
      js = (IJavaScriptExecutor)driver;
      vars = new Dictionary<string, object>();
    }
    [TearDown]
    protected void TearDown()
    {
      driver.Quit();
    }
    [Test]
    public void adminDashboardViewHospital()
    {
      driver.Navigate().GoToUrl("http://localhost:3000/viewhospital");
      driver.Manage().Window.Size = new System.Drawing.Size(1552, 840);
      driver.FindElement(By.CssSelector(".nav-item:nth-child(2) .d-none")).Click();
    }
  }

}
```

Check Logout Button

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
namespace Selenium
{
  [TestFixture]
  public class LogoutTest
  {
    private IWebDriver driver;
    public IDictionary<string, object> vars { get; private set; }
    private IJavaScriptExecutor js;
    [SetUp]
    public void SetUp()
    {
      driver = new ChromeDriver();
      js = (IJavaScriptExecutor)driver;
      vars = new Dictionary<string, object>();
    }
    [TearDown]
```

```
        protected void TearDown()
        {
            driver.Quit();
        }
        [Test]
        public void logout()
        {
            driver.Navigate().GoToUrl("http://localhost:3000/admindashboard");
            driver.Manage().Window.Size = new System.Drawing.Size(1552, 840);
            driver.FindElement(By.CssSelector(".navbg > .d-none")).Click();
            driver.Close();
        }
    }

}
```

Check Bank Request Page Visibility

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
```

```
namespace Selenium
{
    [TestFixture]
    public class ViewBankRequestTest
    {
        private IWebDriver driver;
        public IDictionary<string, object> vars { get; private set; }
        private IJavaScriptExecutor js;
        [SetUp]
        public void SetUp()
        {
            driver = new ChromeDriver();
            js = (IJavaScriptExecutor)driver;
            vars = new Dictionary<string, object>();
        }
        [TearDown]
        protected void TearDown()
        {
            driver.Quit();
        }
        [Test]
        public void viewBankRequest()
        {
            driver.Navigate().GoToUrl("http://localhost:3000/admindashboard");
            driver.Manage().Window.Size = new System.Drawing.Size(1552, 840);
            driver.FindElement(By.CssSelector(".nav-item:nth-child(8) .d-none")).Click();
            driver.Close();
        }
    }
```

```
}
```

Check Blood Request Page Visibility

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
namespace Selenium
{
  [TestFixture]
  public class ViewBloodRequestTest
  {
    private IWebDriver driver;
    public IDictionary<string, object> vars { get; private set; }
    private IJavaScriptExecutor js;
    [SetUp]
    public void SetUp()
    {
      driver = new ChromeDriver();
      js = (IJavaScriptExecutor)driver;
      vars = new Dictionary<string, object>();
```

```
    }
    [TearDown]
    protected void TearDown()
    {
      driver.Quit();
    }
    [Test]
    public void viewBloodRequest()
    {
      driver.Navigate().GoToUrl("http://localhost:3000/admindashboard");
      driver.Manage().Window.Size = new System.Drawing.Size(1552, 840);
      driver.FindElement(By.CssSelector(".nav-item:nth-child(8) .d-none")).Click();
      driver.FindElement(By.CssSelector(".nav-item:nth-child(9) .d-none")).Click();
      driver.Close();
    }
  }

}
```

Check Blood Stock Page Visibility

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
```

```
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
namespace Selenium
{
  [TestFixture]
  public class ViewBloodStockTest
  {
    private IWebDriver driver;
    public IDictionary<string, object> vars { get; private set; }
    private IJavaScriptExecutor js;
    [SetUp]
    public void SetUp()
    {
      driver = new ChromeDriver();
      js = (IJavaScriptExecutor)driver;
      vars = new Dictionary<string, object>();
    }
    [TearDown]
    protected void TearDown()
    {
      driver.Quit();
    }
    [Test]
    public void viewBloodStock()
    {
      driver.Navigate().GoToUrl("http://localhost:3000/admindashboard");
      driver.Manage().Window.Size = new System.Drawing.Size(1552, 840);
      driver.FindElement(By.CssSelector(".nav-item:nth-child(10) .d-none")).Click();
      driver.Close();
```

```
      }
    }


}
```

Jest

Coding:

View Hospital Component Test

```
import React from 'react';
import { render, screen, waitFor } from '@testing-library/react';
import axios from 'axios';
import ViewHospitalComponent from '../Component/ViewHospitalComponent';


jest.mock('axios');


describe('ViewHospitalComponent', () => {
  it('renders hospital data fetched from API', async () => {
    const mockData = {
      data: [
        {
          "id": "26e36c6a-0c9a-48e2-8eac-0296706b2604",
          "name": "ABC Hospital",
          "email": "smano4570@gmail.com",
          "phoneNumber": 6382202586,
          "location": "Chennai",
          "governmentId": "1245433",
          "document": "document",
          "doorNo": "21/2",
          "street": "Salai Street,",
```

```
            "area": "Ktc Nagar",
            "city": "Tirunelveli",
            "state": "Tamil Nadu",
            "postalCode": "627011",
            "status": 1



        }
    ]
}



    axios.get.mockResolvedValue(mockData);

    const { getByTestId,getAllByTestId } = render(<ViewHospitalComponent />);



    await waitFor(() => {
      expect(getByTestId('row-26e36c6a-0c9a-48e2-8eac-0296706b2604')).toBeInTheDocument();

    });
  });
});
```

View Donor Component Test

```
import React from 'react';
import { render, waitFor } from '@testing-library/react';
import axios from 'axios';
import ViewDonorComponent from '../Component/ViewDonorComponent';
```

```
jest.mock('axios');


describe('ViewDonorComponent', () => {
  it('renders donor information', async () => {
    const data = {
      data: [
        {
          "accountUserDetailsAddressId": "50fa9fae-bfb2-4e5c-a219-2ebdc1fb4f47",
          "account": {
            "accountId": "477b7d70-299c-4428-a130-7b5bf5176586",
            "name": "Keerthi Vasan",
            "email": "keerthivasan280901@gmail.com",
            "password": "ceb200a6cea4095b86c8bc4d1e801aff9812eaa106fd940e348882d0268d59f5",
            "phoneNumber": 9089678776,
            "status": 1,
            "bloodTransaction": null,
            "bloodCampBloodBank": null,
            "bloodBankBloodStock": null
          },
          "userDetails": {
            "userDetailsId": "1b60eba4-66f9-496f-a764-7afb10d7ee87",
            "age": 21,
            "bloodType": "B+ve",
            "location": "Chennai",
            "governmentId": "",
            "document": "",
            "aadhaarNumber": 345623458900,
            "rolestatus": 3
```

```
            },
            "address": {
              "addressId": "b6872d86-1eb1-4fca-b2ad-3f2c798440b7",
              "doorNo": "21",
              "street": "Vk Street",
              "area": "Narayana nagar",
              "city": "Perambalur",
              "state": "Tamil Nadu",
              "postalCode": "627001"
            }
          }
        ]
      };

      axios.get.mockResolvedValue(data);

        const { getByTestId } = render(<ViewDonorComponent />);
      await waitFor(() => {
        expect(getByTestId('row-477b7d70-299c-4428-a130-7b5bf5176586')).toBeInTheDocument();



      });
    });
});
```

View Blood Transaction Component Test

```
import React from 'react';
import { render, waitFor } from '@testing-library/react';
import axios from 'axios';
import ViewBloodTransaction from '../Component/ViewBloodTransaction';
```

```
import Cookies from 'js-cookie';


jest.mock('axios');
jest.mock('js-cookie');


describe('ViewBloodTransaction', () => {
  it('renders blood transaction information', async () => {
    const data = {
      data: [
        {
          "bloodTransactionId": "cbb7f7ec-2ebf-45b3-af88-12729cb4fa41",
          "accountId": "10bab100-d6b3-41d0-9d3b-46491cda7ca0",
          "account": {
            "accountId": "10bab100-d6b3-41d0-9d3b-46491cda7ca0",
            "name": "ABC Blood Bank",
            "email": "sanjaihari2002@gmail.com",
            "password": "ceb200a6cea4095b86c8bc4d1e801aff9812eaa106fd940e348882d0268d59f5",
            "phoneNumber": 6382202487,
            "status": 1,
            "bloodTransaction": null,
            "bloodCampBloodBank": null,
            "bloodBankBloodStock": null
          },
          "bloodRequestId": "T2937kCU",
          "bloodRequest": {
            "bloodRequestId": "T2937kCU",
            "name": "Sanjai",
            "email": "sanjairock85@gmail.com",
            "units": 2,
```

```
        "phoneNumber": 9867541243,
        "bloodType": "B+ve",
        "age": 20,
        "location": "Chennai",
        "aadhaarNumber": 235489098762,
        "validTime": "03/29/2024 17:11:20",
        "status": 4,
        "acceptStatus": 1,
        "bloodTransaction": null
      },
      "bloodType": "B+ve",
      "units": 0,
      "date": "string",
      "time": "string"
    }
  ]
};

axios.post.mockResolvedValue(data);
Cookies.get.mockReturnValue('sample-id');

const { getByTestId } = render(<ViewBloodTransaction />);
await waitFor(() => {
    expect(getByTestId('row-cbb7f7ec-2ebf-45b3-af88-12729cb4fa41')).toBeInTheDocument();



  });
 });
});
```

Unauthorized Component Test

```
import React from 'react';
import { render } from '@testing-library/react';
import UnAuthorizedComponent from '../Component/UnAuthorizedComponent';

describe('UnAuthorizedComponent', () => {
  it('renders without crashing', () => {
    const { getByTestId } = render(<UnAuthorizedComponent />);
    expect(getByTestId('unauthorized-section')).toBeInTheDocument();
  });

  it('displays the unauthorized image', () => {
    const { getByTestId } = render(<UnAuthorizedComponent />);
    const img = getByTestId('unauthorized-image');
    expect(img).toBeInTheDocument();
    expect(img).toHaveAttribute('src', expect.stringContaining('accessdenied.png'));
  });
});
```

Reject Component Test

```
import React from 'react';
import { render } from '@testing-library/react';
import RejectComponent from '../Component/RejectComponent';

describe('RejectComponent', () => {
  it('renders the reject image', () => {
    const { getByTestId } = render(<RejectComponent />);
    const imgElement = getByTestId('reject-image');
```

```
    expect(imgElement).toBeInTheDocument();
    expect(imgElement).toHaveAttribute('src', expect.stringContaining('reject.png'));
  });
});
```

Pending Component Test

```
import React from 'react';
import { render, screen } from '@testing-library/react';
import { BrowserRouter } from 'react-router-dom';
import PendingComponent from '../Component/PendingComponent';

describe('PendingComponent', () => {
  it('renders the pending message', () => {
    render(
      <BrowserRouter>
        <PendingComponent />
      </BrowserRouter>
    );

    const pendingMessage = screen.getByText(/Your Request has been waiting for management approval/i);
    expect(pendingMessage).toBeInTheDocument();
  });



});
```

Not Found Component Test

```
import React from 'react';
import { render } from '@testing-library/react';
```

```
import NotFoundComponent from '../Component/NotFoundComponent';

describe('NotFoundComponent', () => {
  it('renders without crashing', () => {
    const { getByTestId } = render(<NotFoundComponent />);
    expect(getByTestId('notfound-section')).toBeInTheDocument();
  });

  it('displays the not found image with correct dimensions', () => {
    const { getByTestId } = render(<NotFoundComponent />);
    const img = getByTestId('notfound-image');
    expect(img).toBeInTheDocument();
    expect(img).toHaveAttribute('src', expect.stringContaining('notfound.png'));
    expect(img).toHaveAttribute('width', '350px');
    expect(img).toHaveAttribute('height', '150px');
  });
});
```

Login Component Test

```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react';
import LoginComponent from '../Component/LoginComponent';
import axios from 'axios'
import { BrowserRouter } from 'react-router-dom';

const mockedNavigate = jest.fn();
jest.mock('react-router-dom', () => ({
  ...jest.requireActual('react-router-dom'),
  useNavigate: () => mockedNavigate,
```

```
}));

describe('LoginComponent', () => {
  it('renders email and password inputs', () => {
    const { getByTestId } = render(
    <BrowserRouter><LoginComponent /></BrowserRouter>);
    const emailInput = getByTestId('email-input');
    const passwordInput = getByTestId('password-input');

    expect(emailInput).toBeInTheDocument();
    expect(passwordInput).toBeInTheDocument();
  });




  it('submits form with valid email and password', async () => {
    const mockAxios = jest.spyOn(axios, 'post').mockResolvedValueOnce({
      data: {
        accountExists: true,
        passwordStatus: true,
        accountApproval: 'approve',
        token: 'mockToken',
      },
    });

    const { getByTestId } = render(<BrowserRouter><LoginComponent /></BrowserRouter>);
    const emailInput = getByTestId('email-input');
    const passwordInput = getByTestId('password-input');
```

```
    const loginButton = getByTestId('login-button');

    fireEvent.change(emailInput, { target: { value: 'validemail@example.com' } });
    fireEvent.change(passwordInput, { target: { value: 'validpassword' } });
    fireEvent.click(loginButton);
    await waitFor(() => {
      expect(mockAxios).toHaveBeenCalledWith('https://localhost:7089/api/Login', {
        Email: 'validemail@example.com',
        Password: 'validpassword',
      });
    });
  });
});
```

Forget Password Test

```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react';
import ForgetPasswordComponent from '../Component/ForgetPasswordComponent';
import axios from 'axios';
import { BrowserRouter } from 'react-router-dom';

jest.mock('axios');

describe('ForgetPasswordComponent', () => {
  it('renders correctly', () => {
    const { getByTestId } = render(
      <BrowserRouter>
```

```
    <ForgetPasswordComponent />
   </BrowserRouter>
  );
  expect(getByTestId('email-input')).toBeInTheDocument();
  expect(getByTestId('submit-button')).toBeInTheDocument();
});



it('submits the form with valid email', async () => {
  axios.post.mockResolvedValue({ data: { emailExists: true, sendMail: true } });
  const { getByTestId } = render(
   <BrowserRouter>
    <ForgetPasswordComponent />
   </BrowserRouter>
  );
  fireEvent.change(getByTestId('email-input'), { target: { value: 'test@example.com' } });
  fireEvent.click(getByTestId('submit-button'));

  await waitFor(() => {
   expect(axios.post).toHaveBeenCalledWith('https://localhost:7089/api/ForgetPassword', {
    email: 'test@example.com',
   });
  });
});
```

```
});
```

Change Password Test

```
import React from 'react';

import { render, fireEvent, waitFor } from '@testing-library/react';

import ChangePasswordComponent from '../Component/ChangePasswordComponent';

import axios from 'axios';

import { BrowserRouter as Router } from 'react-router-dom';

jest.mock('axios');

const mockedNavigate = jest.fn();

jest.mock('react-router-dom', () => ({

  ...jest.requireActual('react-router-dom'),

  useNavigate: () => mockedNavigate,

}));


describe('ChangePasswordComponent', () => {

  it('renders correctly', () => {

    const { getByTestId } = render(<Router>

      <ChangePasswordComponent />

    </Router>);

    expect(getByTestId('email-input')).toBeInTheDocument();

    expect(getByTestId('old-password-input')).toBeInTheDocument();

    expect(getByTestId('new-password-input')).toBeInTheDocument();
```

```
  expect(getByTestId('submit-button')).toBeInTheDocument();
 });


 it('submits the form with valid data', async () => {
  axios.post.mockResolvedValue({ data: { emailExists: true, passcheck: true } });
  const { getByTestId } = render(<ChangePasswordComponent />);
  fireEvent.change(getByTestId('email-input'), { target: { value: 'test@example.com' } });
  fireEvent.change(getByTestId('old-password-input'), { target: { value: 'oldPassword123' } });
  fireEvent.change(getByTestId('new-password-input'), { target: { value: 'newPassword123' } });
  fireEvent.click(getByTestId('submit-button'));


  await waitFor(() => {
   expect(axios.post).toHaveBeenCalledWith('https://localhost:7089/api/ChangePassword', {
    email: 'test@example.com',
    oldPassword: 'oldPassword123',
    newPassword: 'newPassword123',
   });
  });
 });




});
```

Blood Bank Request Component Test

```
import React from 'react';

import { render, fireEvent, waitFor } from '@testing-library/react';

import axios from 'axios';

import BloodBankRequestComponent from '../Component/BloodBankRequestComponent'; // Adjust the import path as necessary



jest.mock('axios');



describe('BloodBankRequestComponent', () => {

  it('renders hospital data and can approve a request', async () => {

    const hospitalData = [

      {

        "id": "10bab100-d6b3-41d0-9d3b-46491cda7ca0",

        "name": "ABC Blood Bank",

        "email": "sanjaihari2002@gmail.com",

        "phoneNumber": 6382202487,

        "location": "Chennai",

        "governmentId": "1245433",

        "document": "document",

        "doorNo": "24",

        "street": "VK Street,",

        "area": "Ktc Nagar",

        "city": "Viruthunagar",

        "state": "Tamil Nadu",

        "postalCode": "627011",
```

```
      "status":0
    }

    ];

    axios.get.mockResolvedValue({ data: hospitalData });

    const { getByTestId } = render(<BloodBankRequestComponent />);
    await waitFor(() => expect(getByTestId('row-10bab100-d6b3-41d0-9d3b-46491cda7ca0')).toBeInTheDocument());

  });

});
```

Add Blood Stock Component Test

```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react';
import AddBloodStockComponent from '../Component/AddBloodStockComponent';
import axios from 'axios';
import Cookies from 'js-cookie';
```

```
jest.mock('axios');

jest.mock('js-cookie', () => ({

  get: jest.fn(),

}));


describe('AddBloodStockComponent', () => {

  beforeEach(() => {

    jest.clearAllMocks();

  });


  it('renders correctly', () => {

    const { getByTestId } = render(<AddBloodStockComponent />);

    expect(getByTestId('headname')).toHaveTextContent('Add Blood Stock');

  });


  it('submits the form with correct blood type and units', async () => {

    Cookies.get.mockReturnValue('scbc238wej873cewd72');

    const { getByTestId } = render(<AddBloodStockComponent />);

    const bloodTypeSelect = getByTestId('BloodType');

    const unitInput = getByTestId('Unit');

    const addButton = getByTestId('addbutton');


    fireEvent.change(bloodTypeSelect, { target: { value: 'A+ve' } });

    fireEvent.change(unitInput, { target: { value: '5' } });
```

```
    axios.post.mockResolvedValue({ status: 200 });


    fireEvent.click(addButton);


    await waitFor(() => {
      expect(axios.post).toHaveBeenCalledWith('https://localhost:7089/api/BloodStock', {
        bloodStockId: '',
        bloodType: 'A+ve',
        units: 5,
        accountId: 'scbc238wej873cewd72',
      });
    });
  });
});
```

Add Blood Camp Test

```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react';
import '@testing-library/jest-dom';
import AddBloodCampComponent from '../Component/AddBloodCampComponent';
import axios from 'axios';
import Cookies from 'js-cookie';
```

```
jest.mock('axios');

jest.mock('js-cookie');


describe('AddBloodCampComponent', () => {

  beforeEach(() => {

    Cookies.get.mockReturnValue('test-id');

  });


  it('renders without crashing', () => {

    const { getByTestId } = render(<AddBloodCampComponent />);

    expect(getByTestId('headname')).toHaveTextContent('Blood Camp');

  });


  it('submits the form with correct data', async () => {

    axios.post.mockResolvedValue({ status: 200 });

    const { getByTestId } = render(<AddBloodCampComponent />);


    fireEvent.change(getByTestId('bloodCampName'), { target: { value: 'Test Camp' } });

    fireEvent.change(getByTestId('bloodCampLocation'), { target: { value: 'Test Location' } });

    fireEvent.change(getByTestId('date'), { target: { value: '2024-01-30' } });

    fireEvent.change(getByTestId('time'), { target: { value: '10:00' } });

    fireEvent.click(getByTestId('bloodcampbtn'));


    await waitFor(() => {

      expect(axios.post).toHaveBeenCalledWith('https://localhost:7089/api/BloodCamp', {
```

```
      bloodCampName: 'Test Camp',

      bloodCampLocation: 'Test Location',

      date: '2024-01-30',

      time: '10:00',

      accountId: 'test-id'

    });

  });

 });

});
```

Output

```
    √ renders correctly (229 ms)
    √ submits the form with valid data (45 ms)

Test Suites: 13 passed, 13 total
Tests:       20 passed, 20 total
Snapshots:   0 total
Time:        25.688 s
Ran all test suites related to changed files.
```

```
(c) Microsoft Corporation. All rights reserved.

D:\MicroProject\bloodbankmanagementapp>npm test

> bloodbankmanagementapp@0.1.0 test
> react-scripts test --verbose
```

```
C:\Windows\system32\cmd.exe
      at validatePropertiesInDevelopment (node_modules/react-dom/cjs/react-dom.development.js:9541:5)
      at setInitialProperties (node_modules/react-dom/cjs/react-dom.development.js:9830:5)
      at finalizeInitialChildren (node_modules/react-dom/cjs/react-dom.development.js:10950:3)
      at completeWork (node_modules/react-dom/cjs/react-dom.development.js:22193:17)
      at completeUnitOfWork (node_modules/react-dom/cjs/react-dom.development.js:26593:16)
      at performUnitOfWork (node_modules/react-dom/cjs/react-dom.development.js:26568:5)
      at workLoopSync (node_modules/react-dom/cjs/react-dom.development.js:26466:5)
      at renderRootSync (node_modules/react-dom/cjs/react-dom.development.js:26434:7)
      at performConcurrentWorkOnRoot (node_modules/react-dom/cjs/react-dom.development.js:25738:74)
      at flushActQueue (node_modules/react/cjs/react.development.js:2667:24)
      at act (node_modules/react/cjs/react.development.js:2582:11)
      at node_modules/@testing-library/react/dist/act-compat.js:63:25
      at renderRoot (node_modules/@testing-library/react/dist/pure.js:159:26)
      at render (node_modules/@testing-library/react/dist/pure.js:246:10)
      at Object.<anonymous> (src/Test/ChangePasswordTest.test.js:15:35)

  console.log
    { data: { emailExists: true, sendMail: true } }

      at src/Component/ForgetPasswordComponent.js:37:17

 PASS  src/Test/ForgetPasswordTest.test.js (16.095 s)
  ForgetPasswordComponent
    √ renders correctly (223 ms)
    √ submits the form with valid email (41 ms)

  console.log
    { data: { emailExists: true, passcheck: true } }

      at src/Component/ChangePasswordComponent.js:54:17

 PASS  src/Test/ChangePasswordTest.test.js (16.103 s)
  ChangePasswordComponent
    √ renders correctly (229 ms)
    √ submits the form with valid data (45 ms)

Test Suites: 13 passed, 13 total
Tests:       20 passed, 20 total
Snapshots:   0 total
Time:        25.688 s
Ran all test suites related to changed files.

Watch Usage
 › Press a to run all tests.
 › Press f to run only failed tests.
 › Press q to quit watch mode.
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.
Type here to search
ENG
09:42
30-03-2024
```

CI / CD

- **Docker**
- **Jenkins**

Docker

React

```
FROM node:latest

WORKDIR /app/reactapp

COPY package*.json /app/reactapp

RUN npm install

COPY . /app/reactapp

CMD [ "npm" , "start" ]
```

Dotnet Webapi

```
FROM --platform=$BUILDPLATFORM mcr.microsoft.com/dotnet/sdk:8.0-alpine AS build

COPY . /source

WORKDIR /source

ARG TARGETARCH

RUN --mount=type=cache,id=nuget,target=/root/.nuget/packages \

   dotnet publish -a ${TARGETARCH/amd64/x64} --use-current-runtime --self-contained false -o /app

FROM mcr.microsoft.com/dotnet/aspnet:8.0-alpine AS final

WORKDIR /app

COPY --from=build /app .

USER $APP_UID

ENTRYPOINT ["dotnet", "BloodBankManagementWebapi.dll"]
```

Compose File

```
version: '3.4'

networks:
```

```
  dev:
    driver: bridge
services:
  myapp:
    depends_on:
      - "simple"
    build:
      context: ./bloodbankmanagementapp
      dockerfile: Dockerfile
    ports:
      - "8090:3000"
    networks:
      - dev
  simple:
    depends_on:
      - "app_db"
    ports:
      - "8081:80"
    build:
      context: ./BloodBankManagementWebapi/BloodBankManagementWebapi
      dockerfile: Dockerfile
    environment:
      - ConnectionStrings__DefaultConnection=Server=app_db;Port=3306;Database=sampledb;Uid=root;Pwd=root
      - ASPNETCORE_URLS=http://+80
```

```
    networks:
      - dev
  app_db:
    image: mysql:latest
    container_name: app_db
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=sampledb
      - MYSQL_USER=mano
      - MYSQL_PASSWORD=admin123
    ports:
      - "6703:3306"
    restart: always
    volumes:
      - app_data:/var/lib/mysql
    networks:
      - dev
volumes:
app_data:
```

Output

In command line

In Docker Desktop

Images

Container

In Browser

React

Dotnet webapi

Jenkins

Step – 1 : Open jenkins

Step – 2 : Create a pipeline project

Step – 3 : Do script for automation

Script ?

```
node
{
    stage('1. GIT Checkout') {
        git 'https://github.com/smano-20052002/MicroProjectFinal.git'
    }
    dir('BloodBankManagementWebapi'){
        stage('2. Project Build') {
            bat 'dotnet build'
        }
        stage('2. Project Test') {
            bat 'dotnet test'
        }
        stage('3. CodeQuality Test') {
            bat 'dotnet sonarscanner begin /k:"MicroProjectCICD" /d:sonar.host.url="http://localhost:9000"  /d:sonar.login="sqp_669dbc79281094d7a0a490d73b9
        }
        stage('4. CodeQuality Build'){
            bat 'dotnet build'
        }
        stage('5. CodeQualty Report'){
            bat 'dotnet sonarscanner end /d:sonar.login="sqp_669dbc79281094d7a0a490d73b94c5889d47d6a2"'
        }
    }
    stage('6. Dockerization'){
        bat 'docker compose up --build'
    }

}
```

Step – 4: Run the automation

In Docker

In SonarQube