**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=3306"

},

"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.Address).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);

}

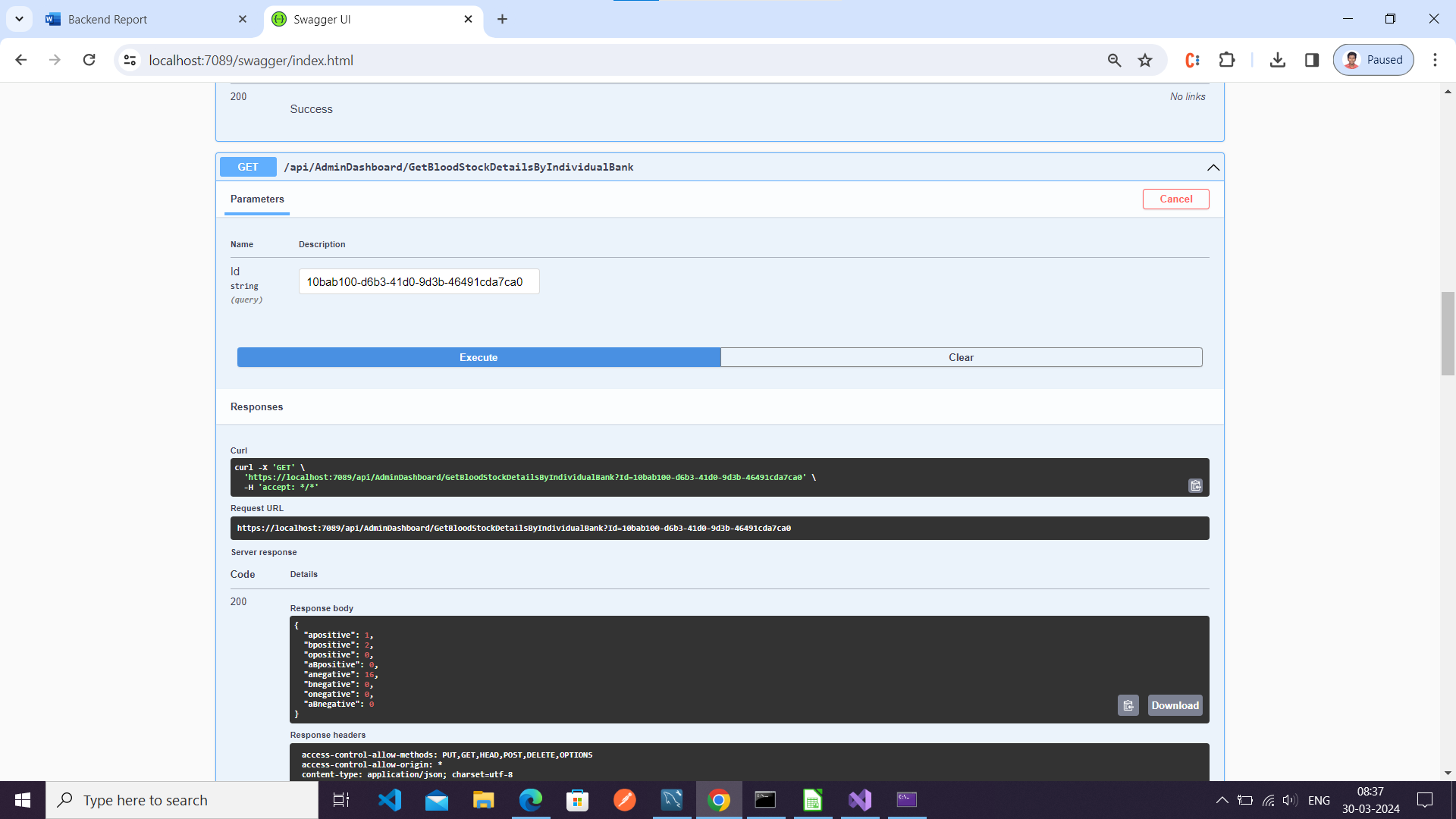
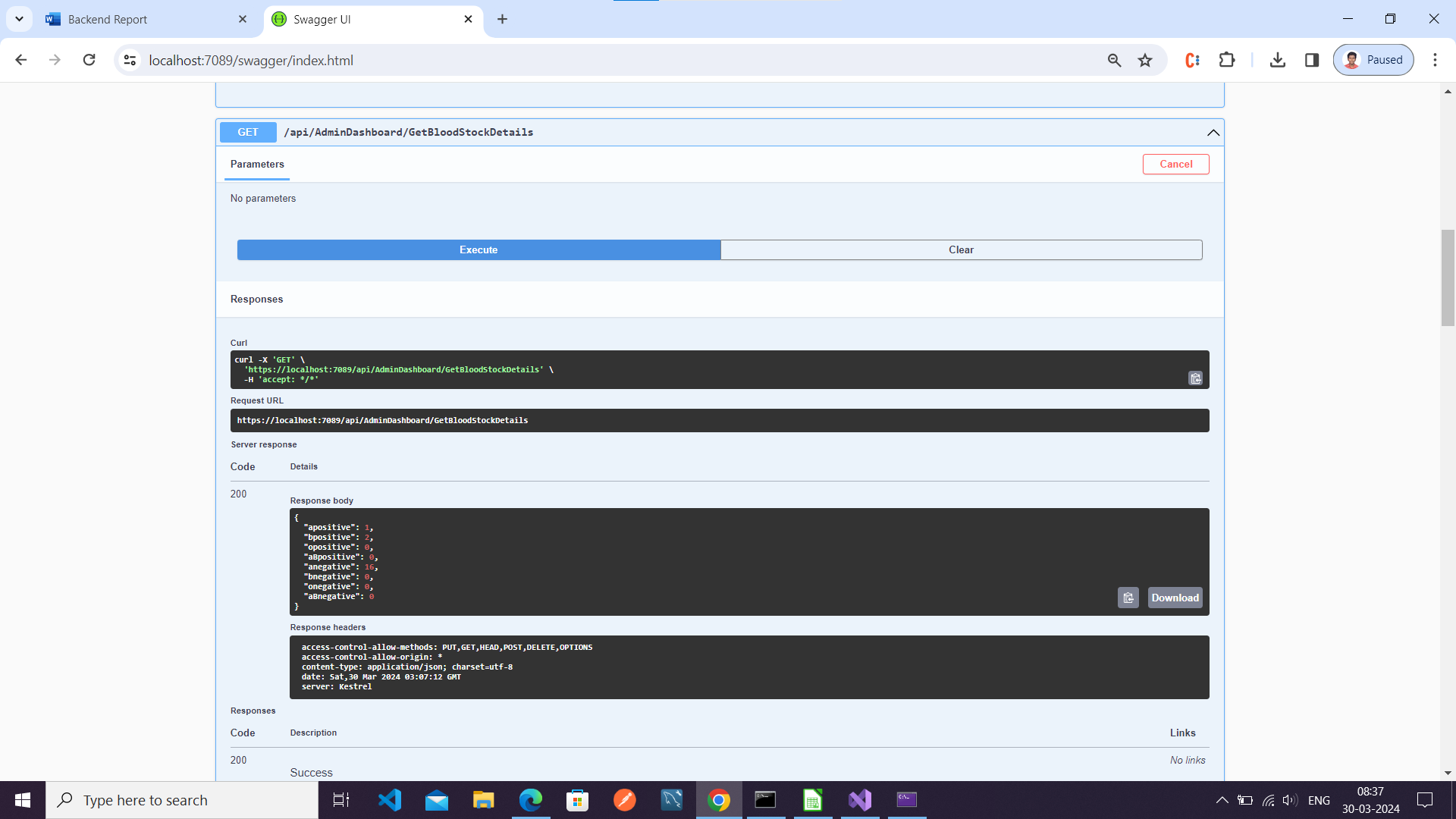
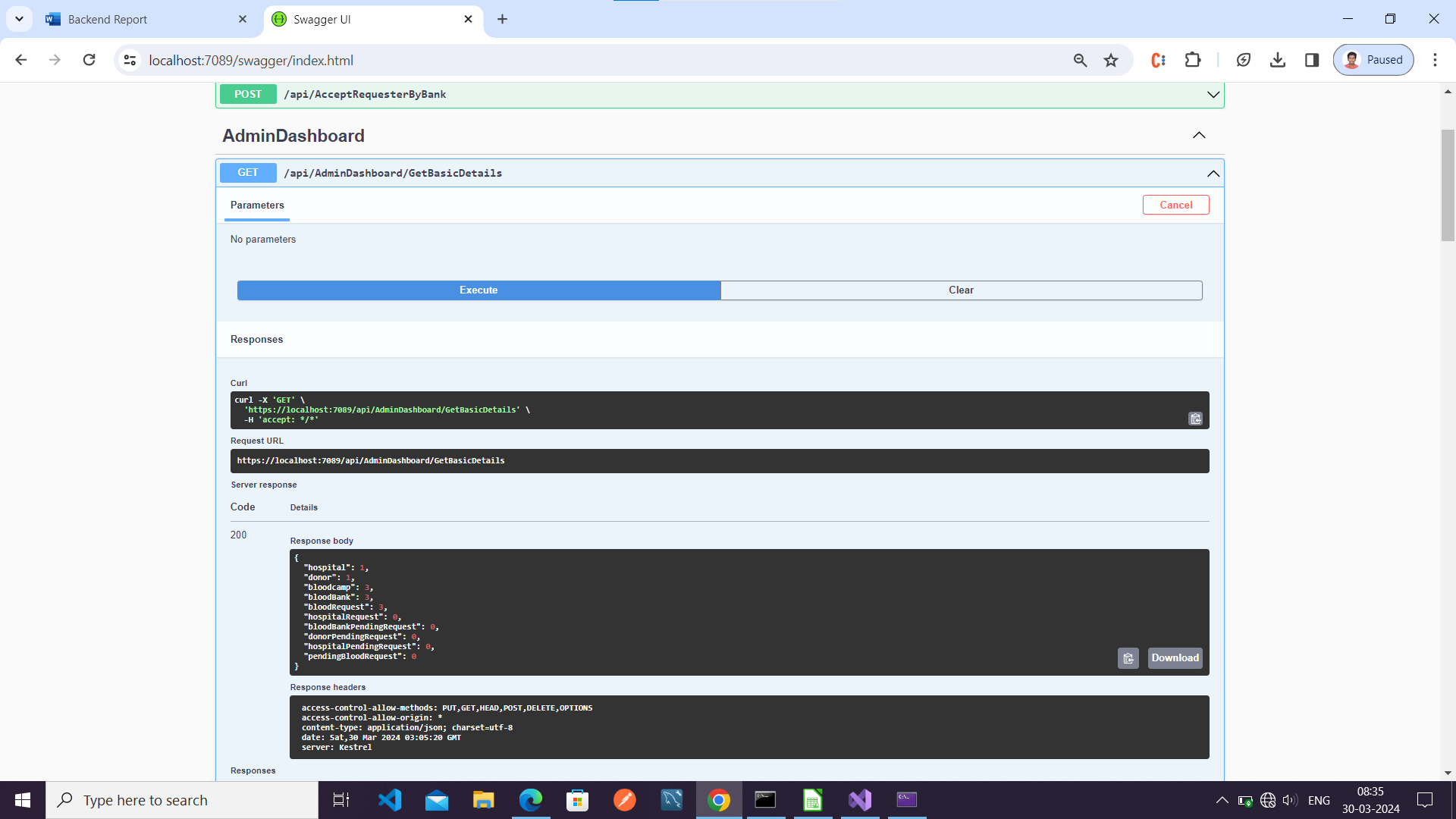
}

}

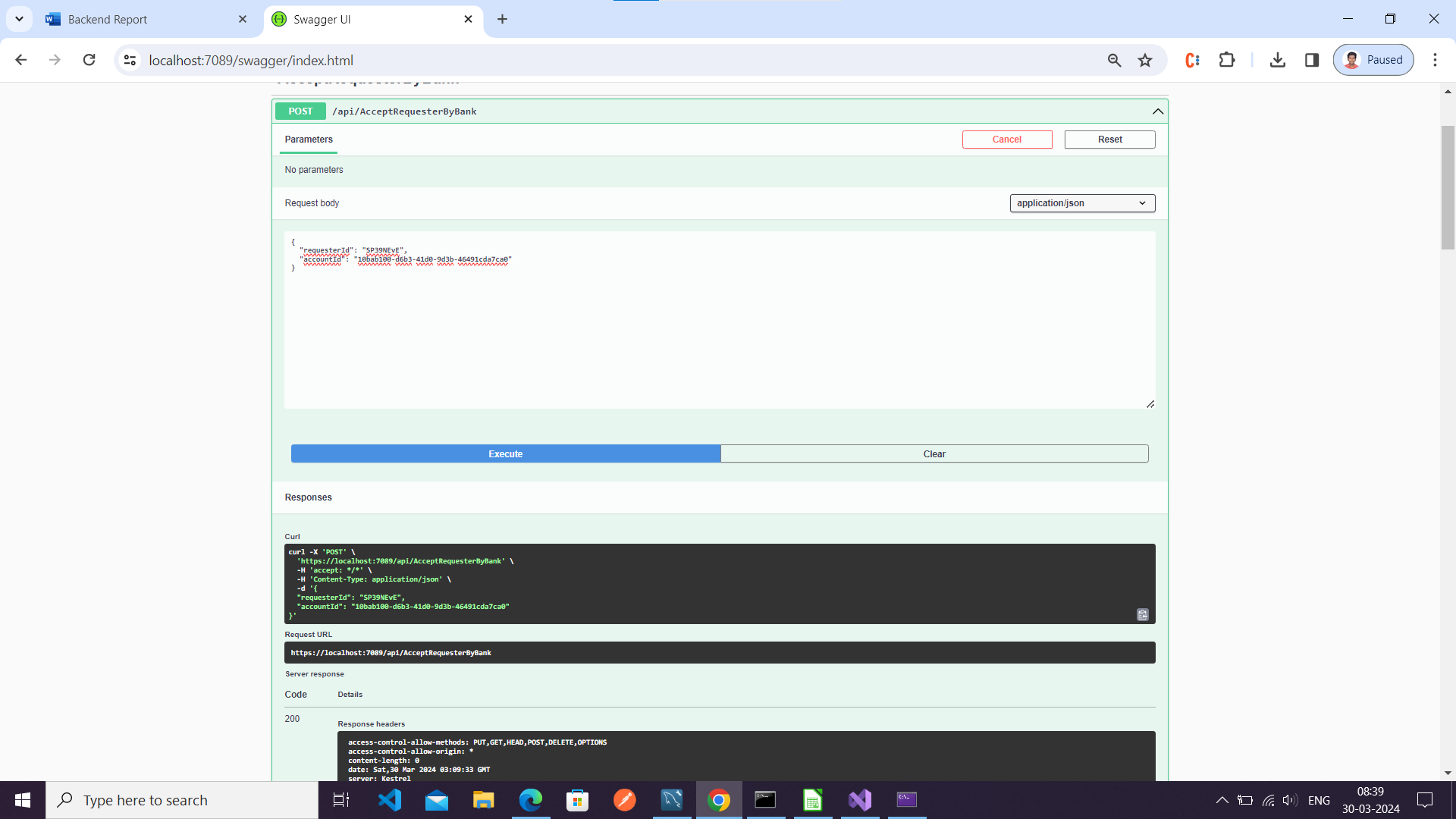
**Output**

**SWAGGER OUTPUT**

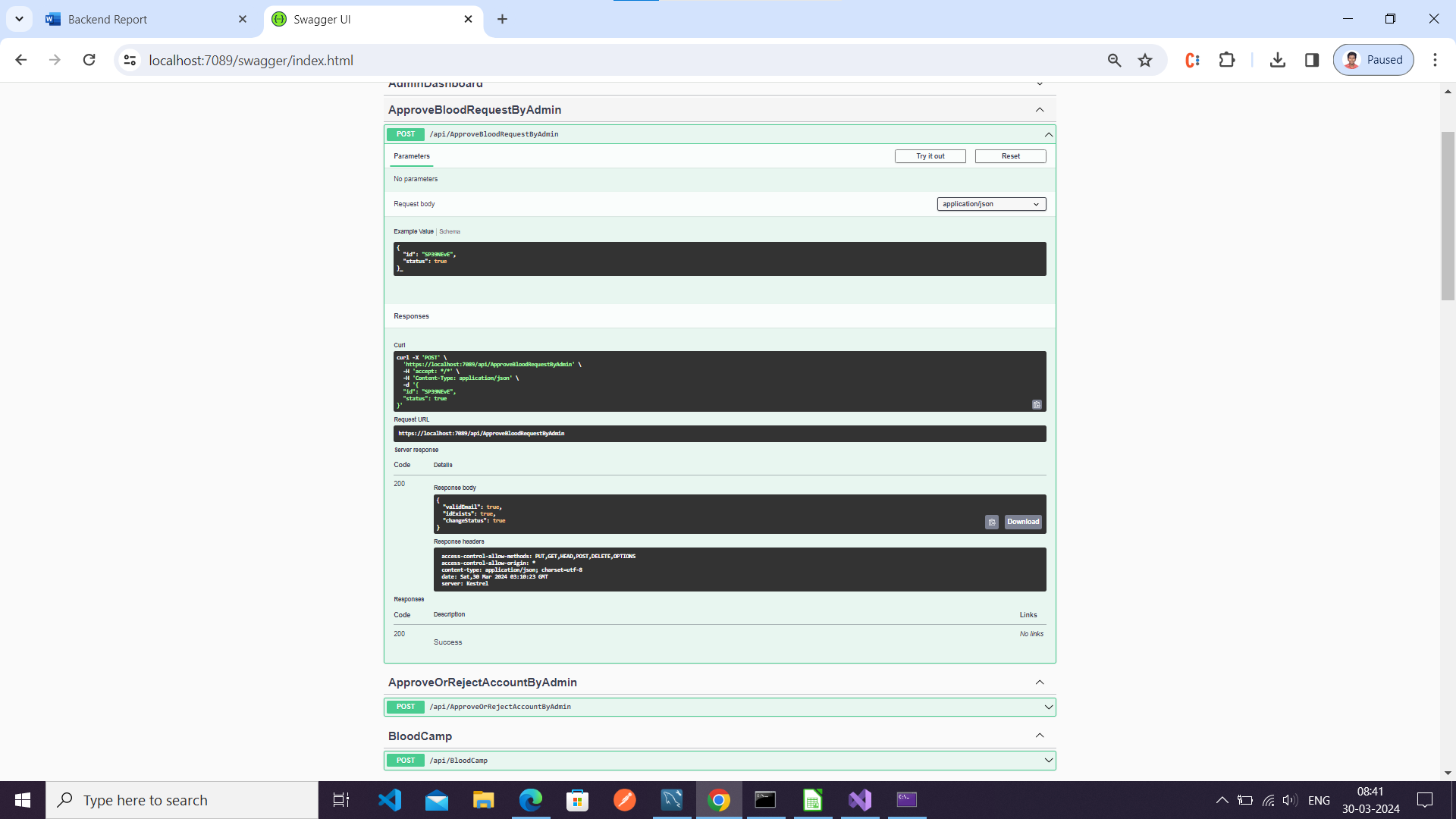
**ADMIN DASHBOARD**

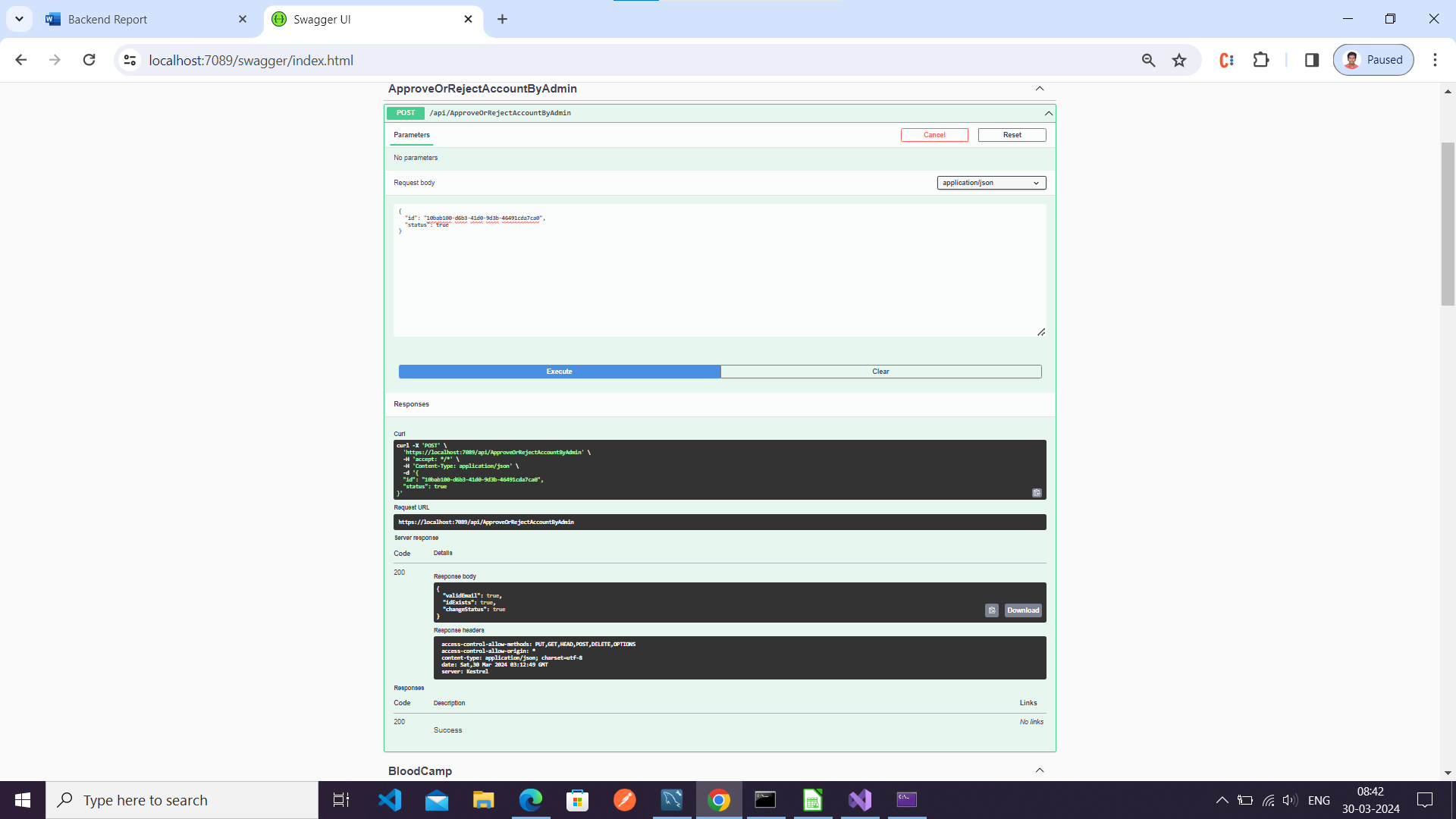


**Accept Requester By Bank**

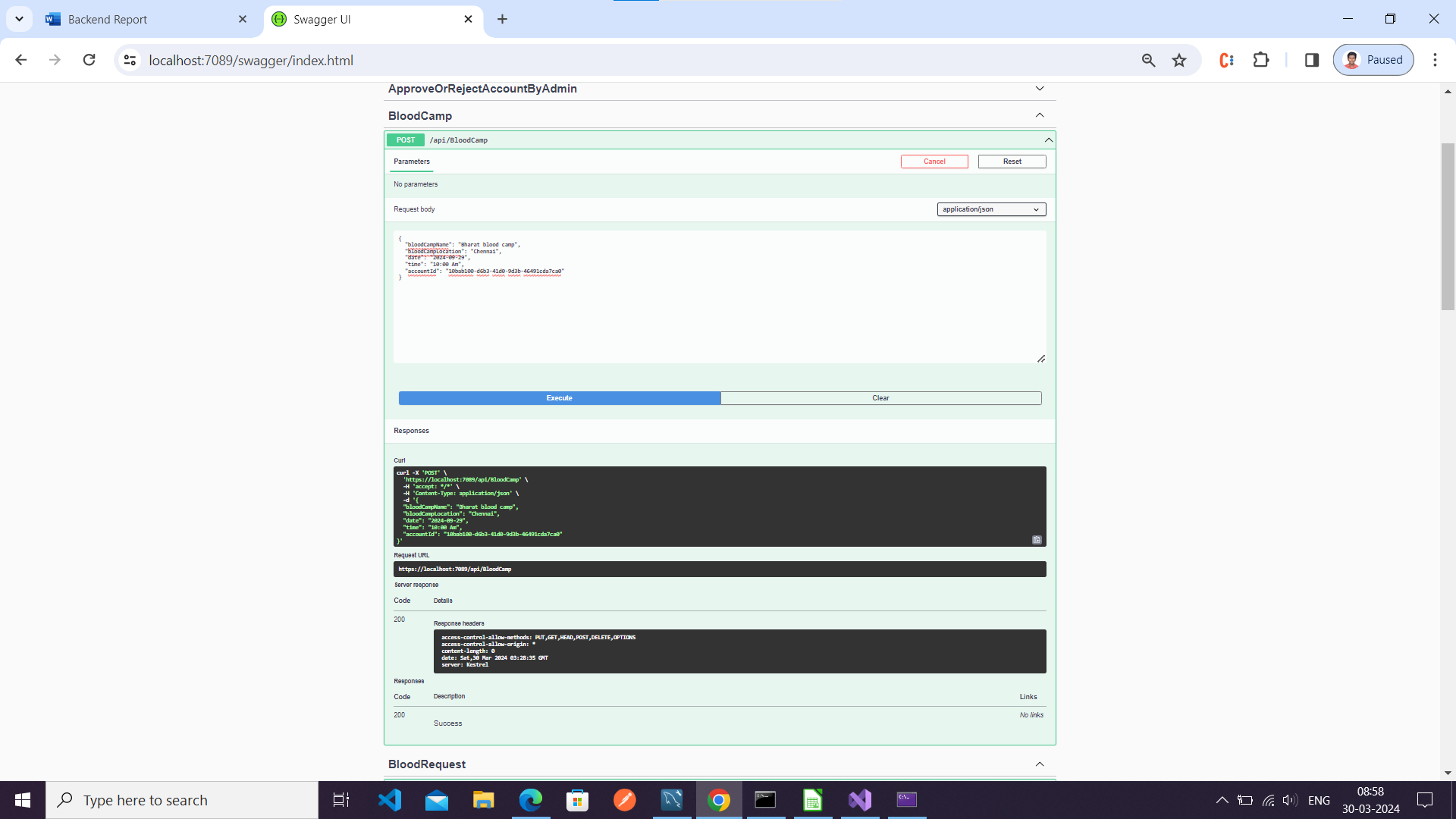


**Approve Blood Request By Admin**

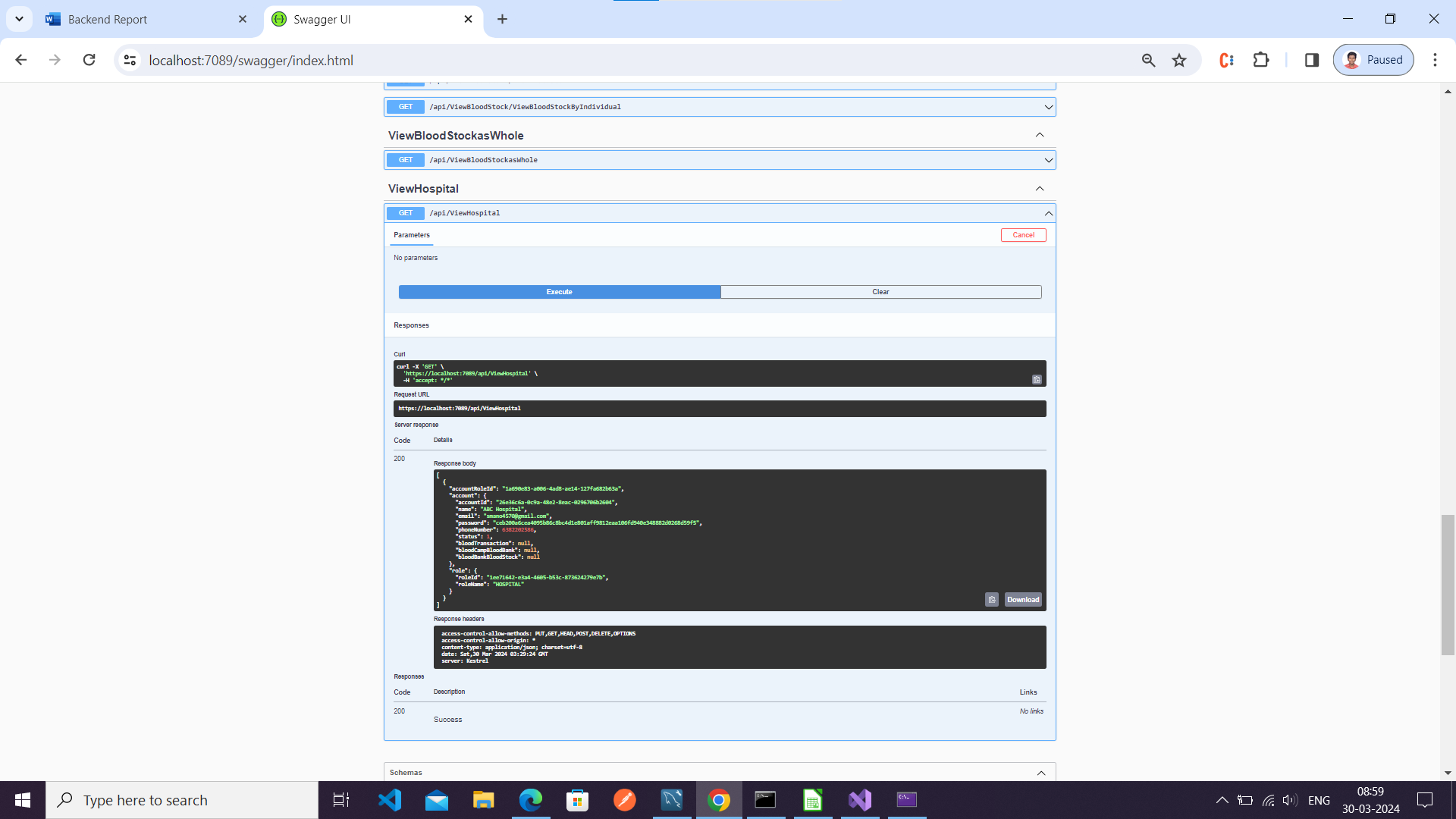
**Approve Or Reject Account By Admin**



**Blood camp**



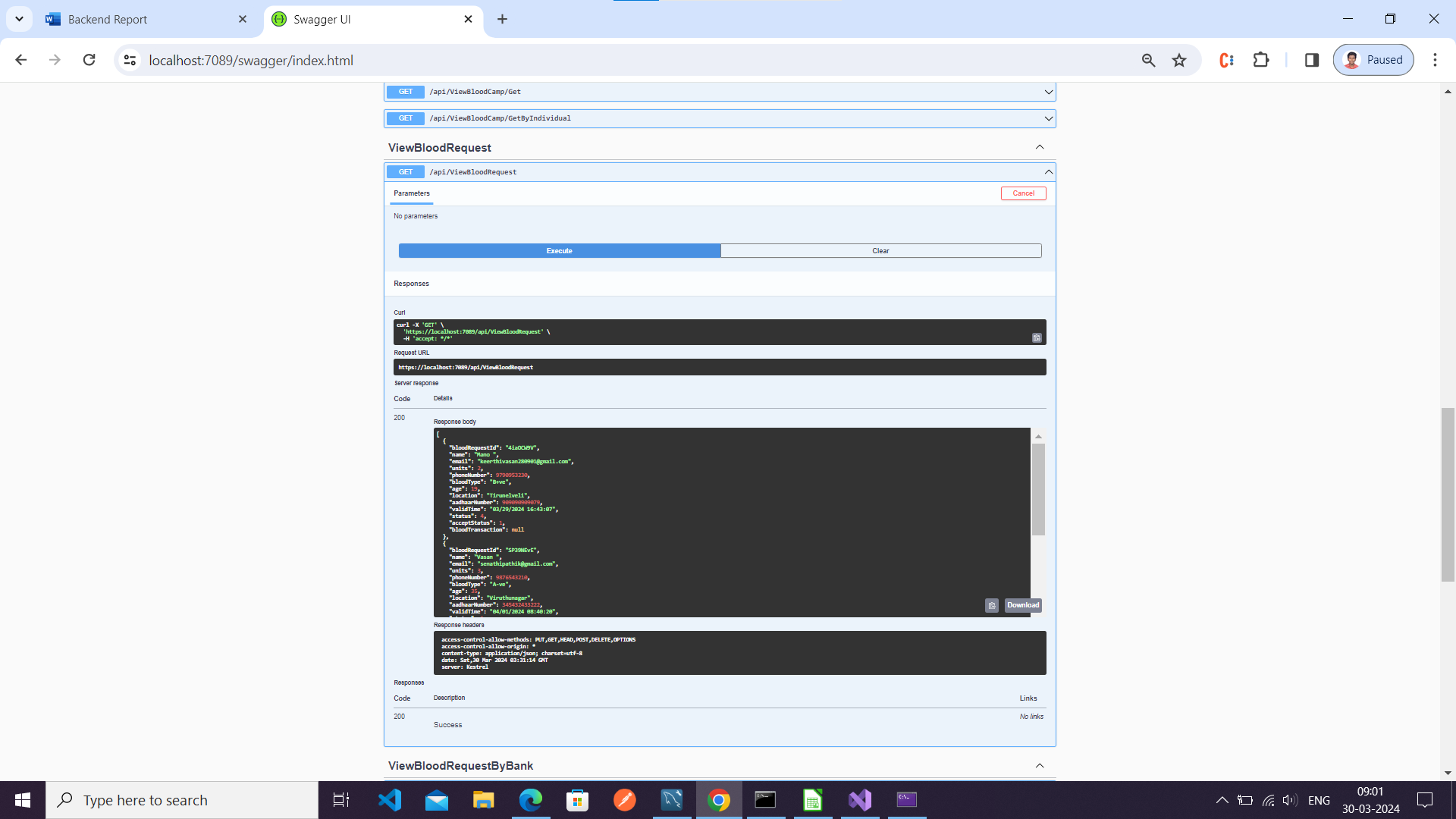
View Hospital



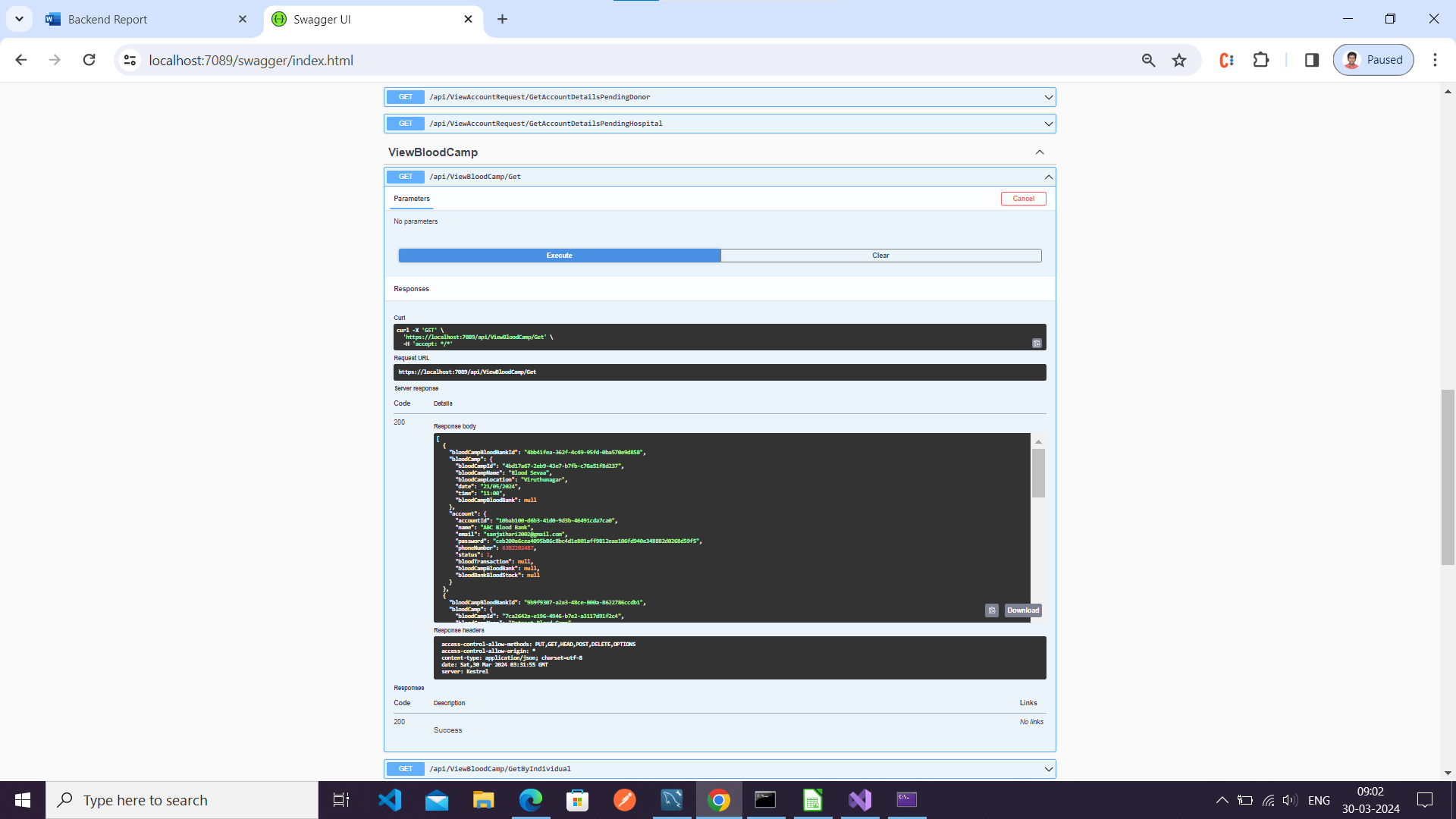
**View Blood Stock As Whole**



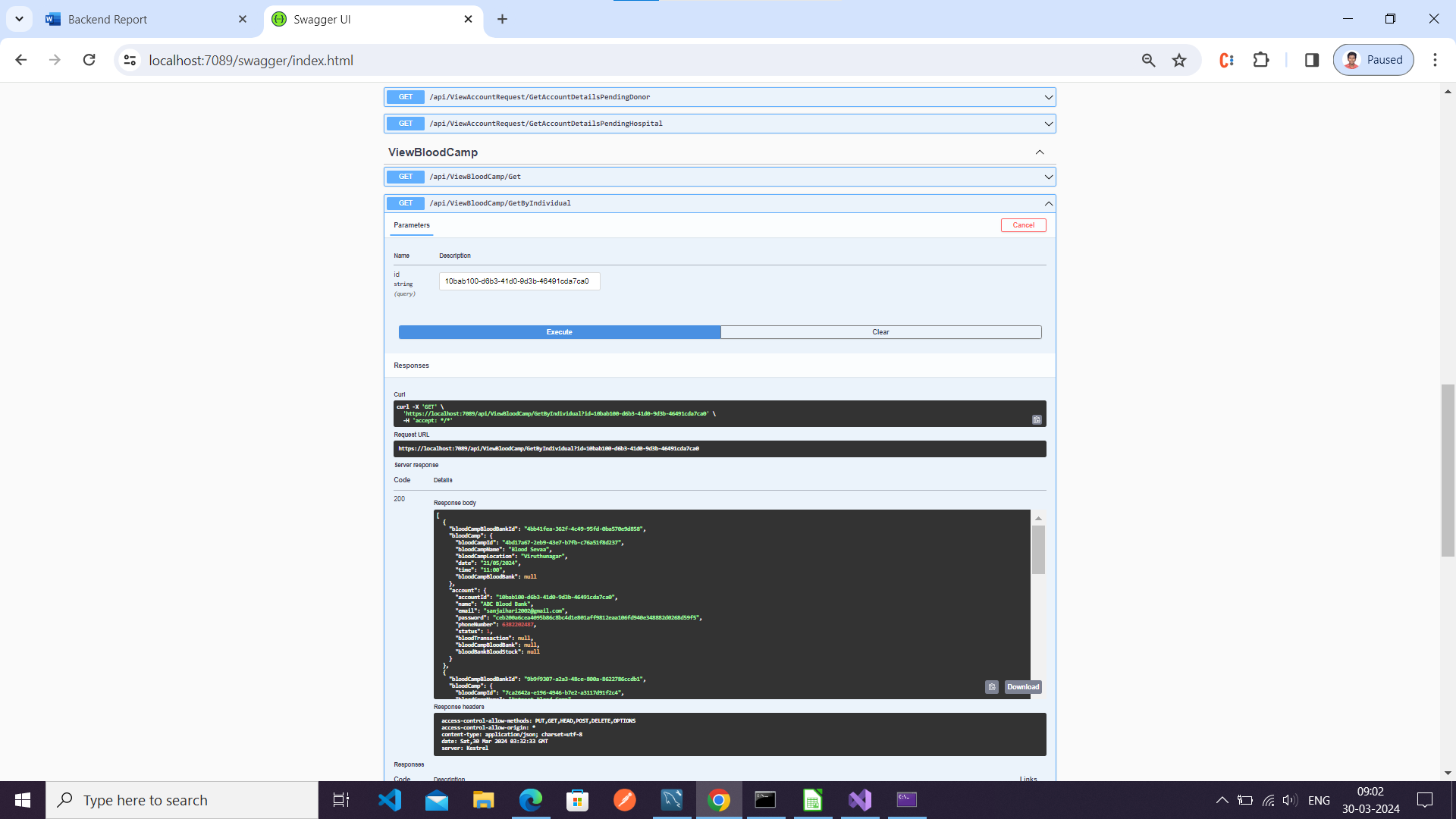
**View Blood Request**



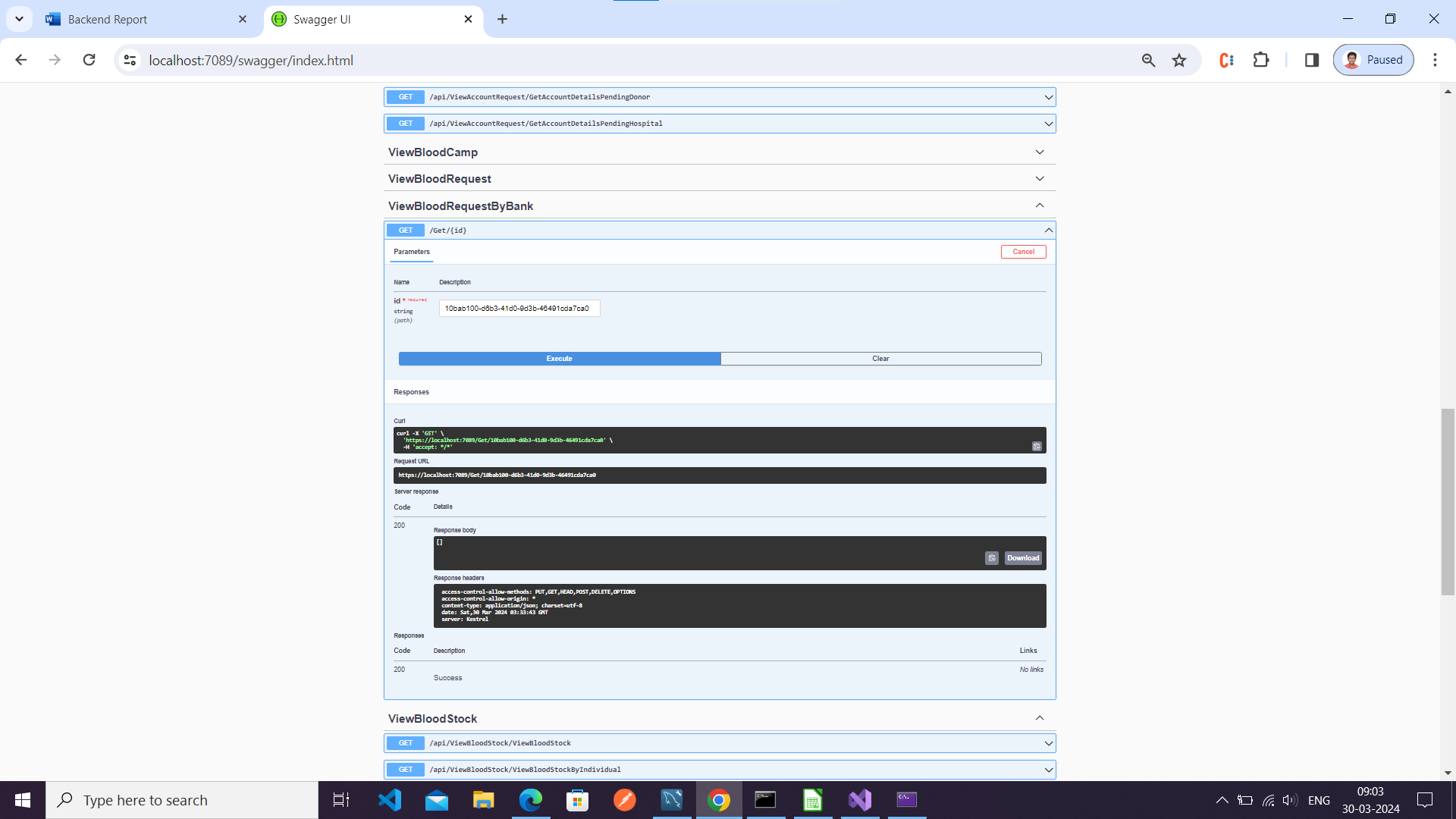
**View Blood Camp**



By individual blood bank



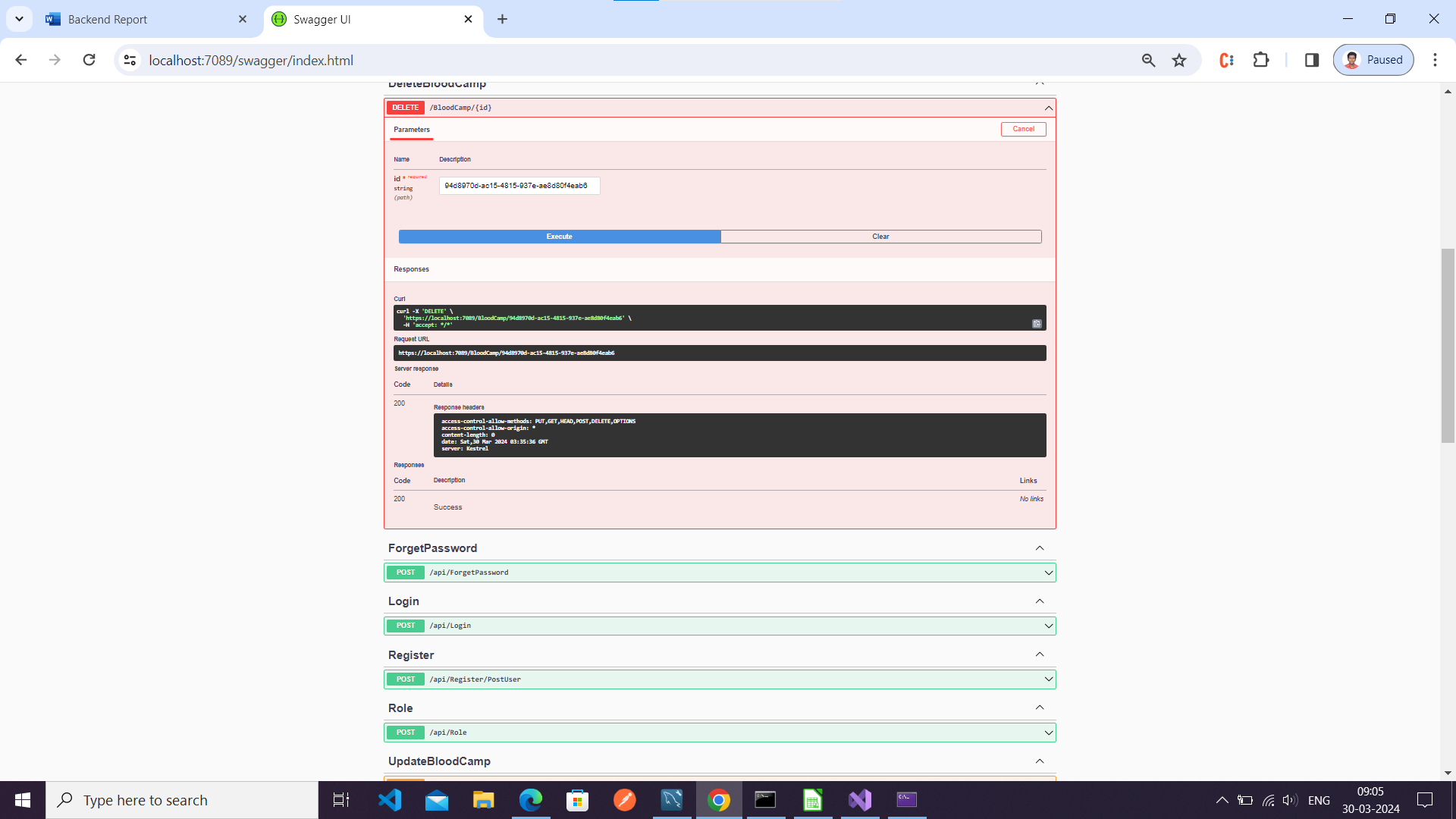
View Blood Request by Bank



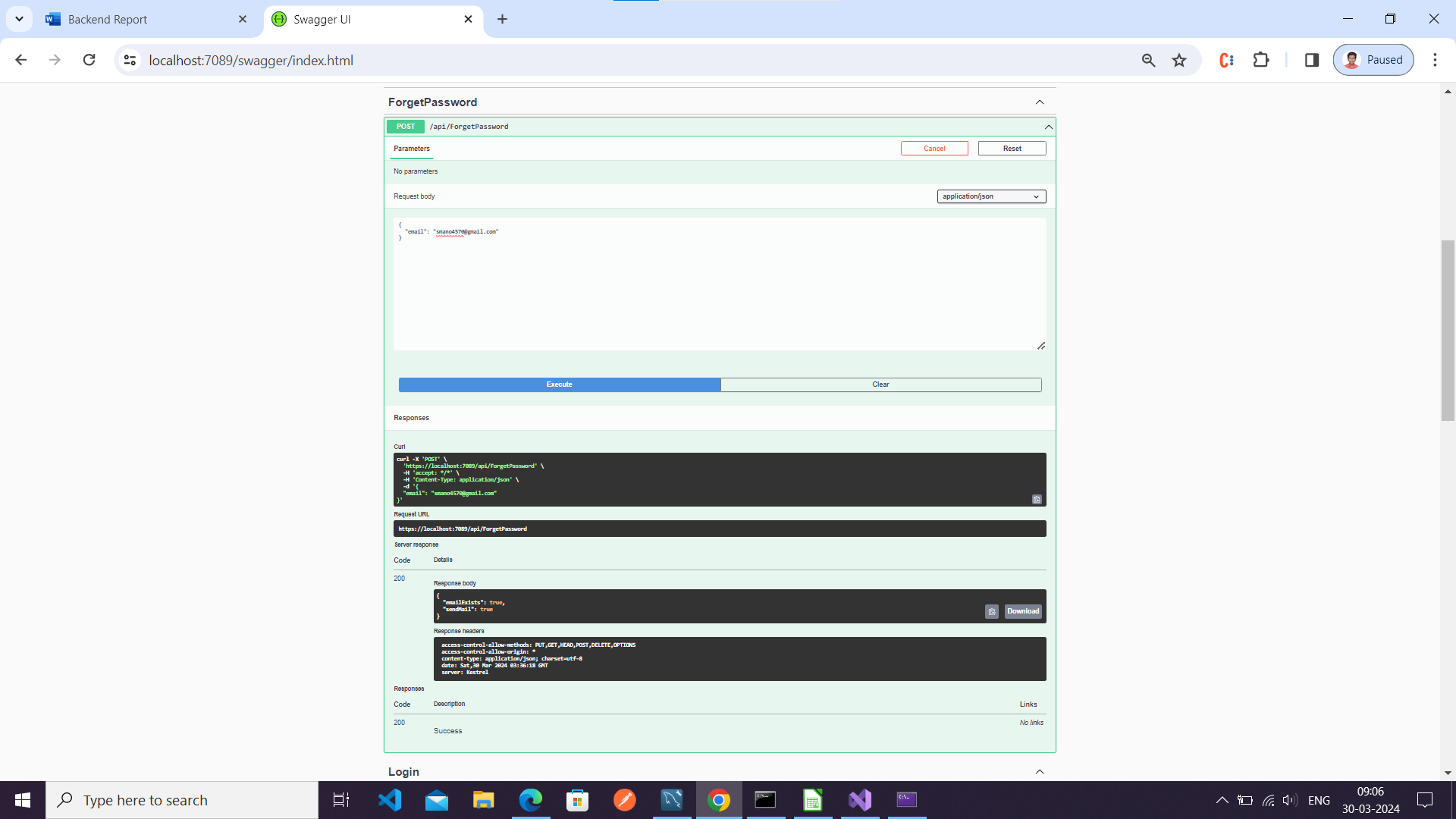
View Accepted bank



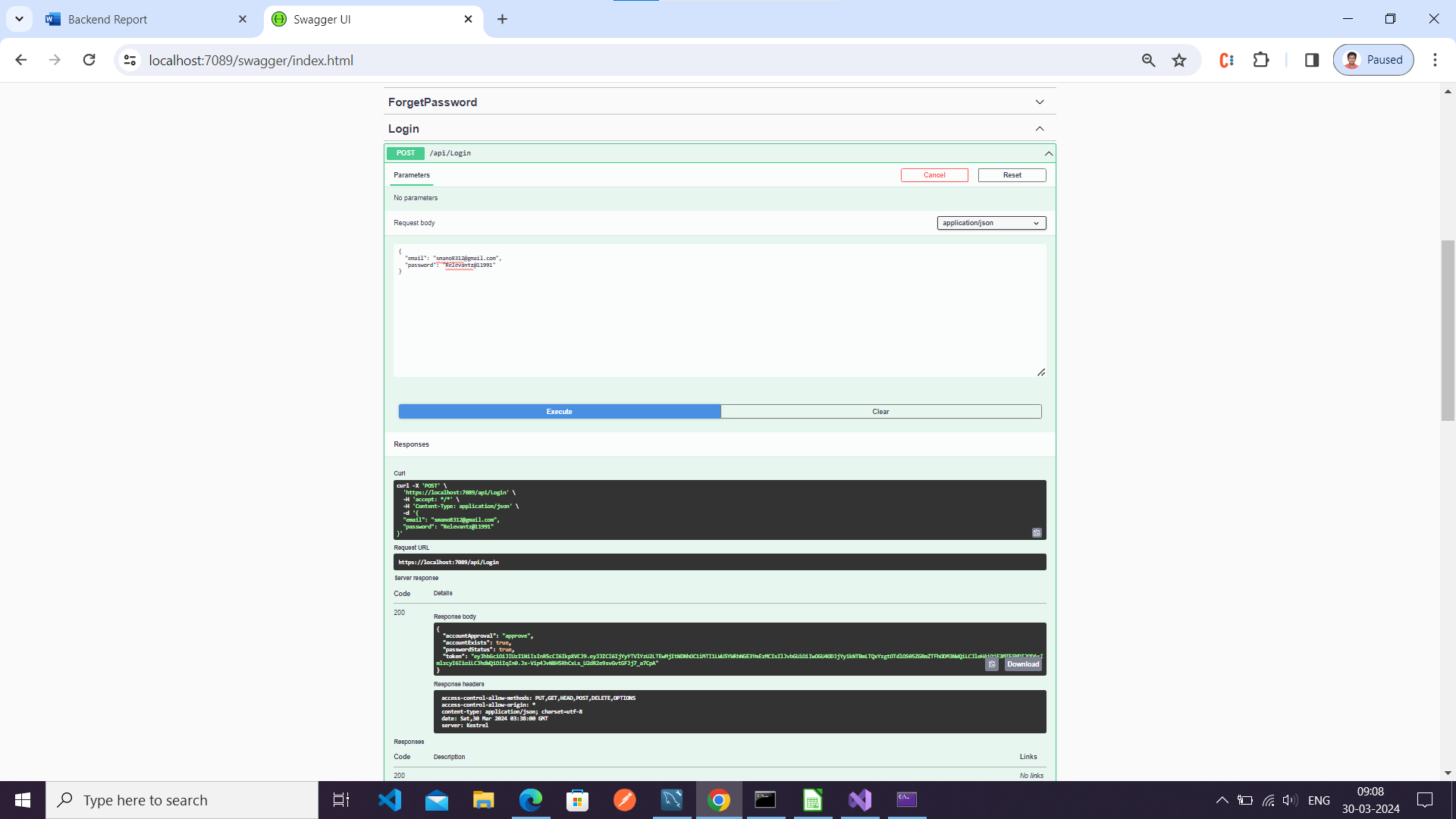
Delete blood camp



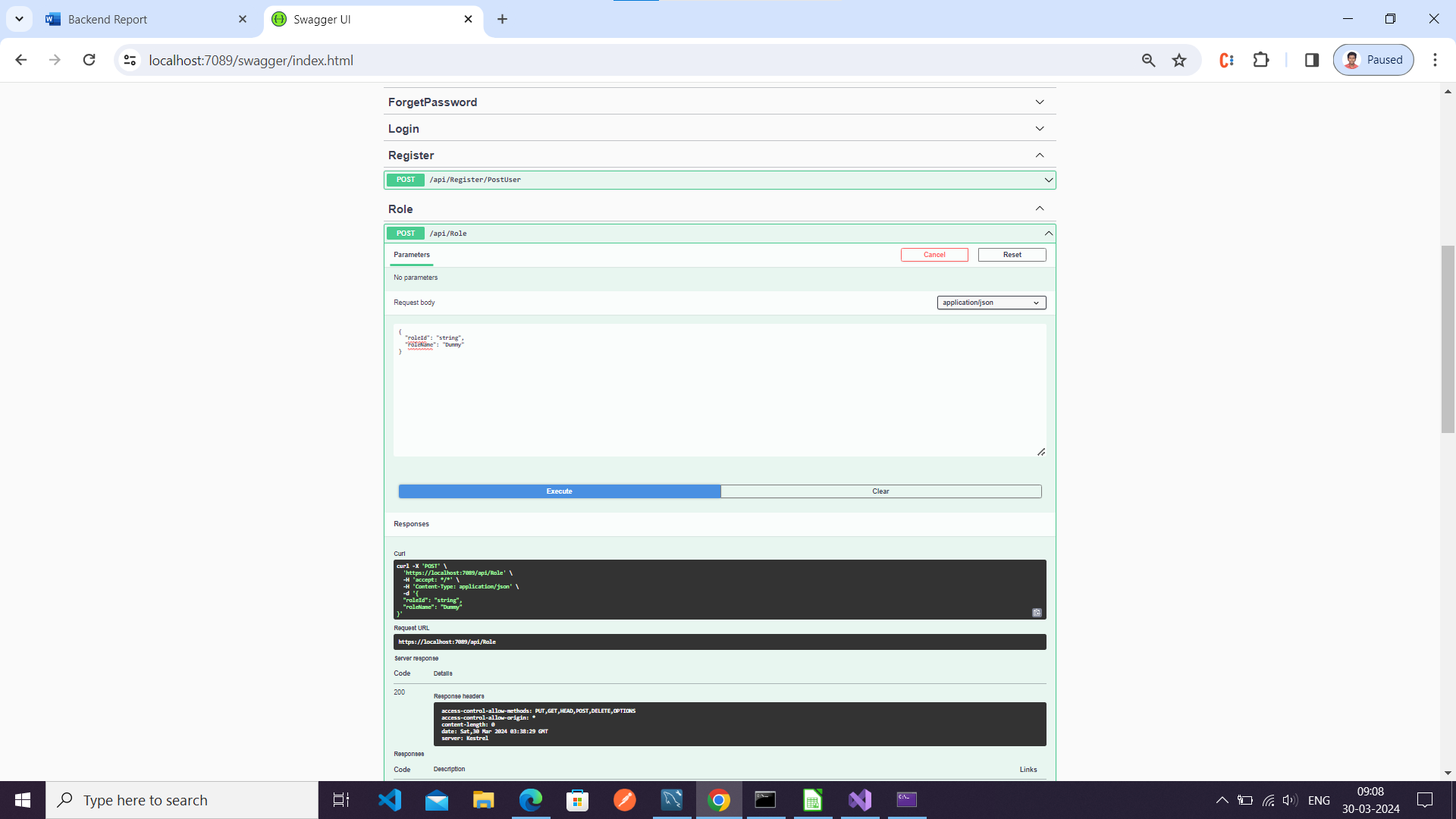
Forgot Password



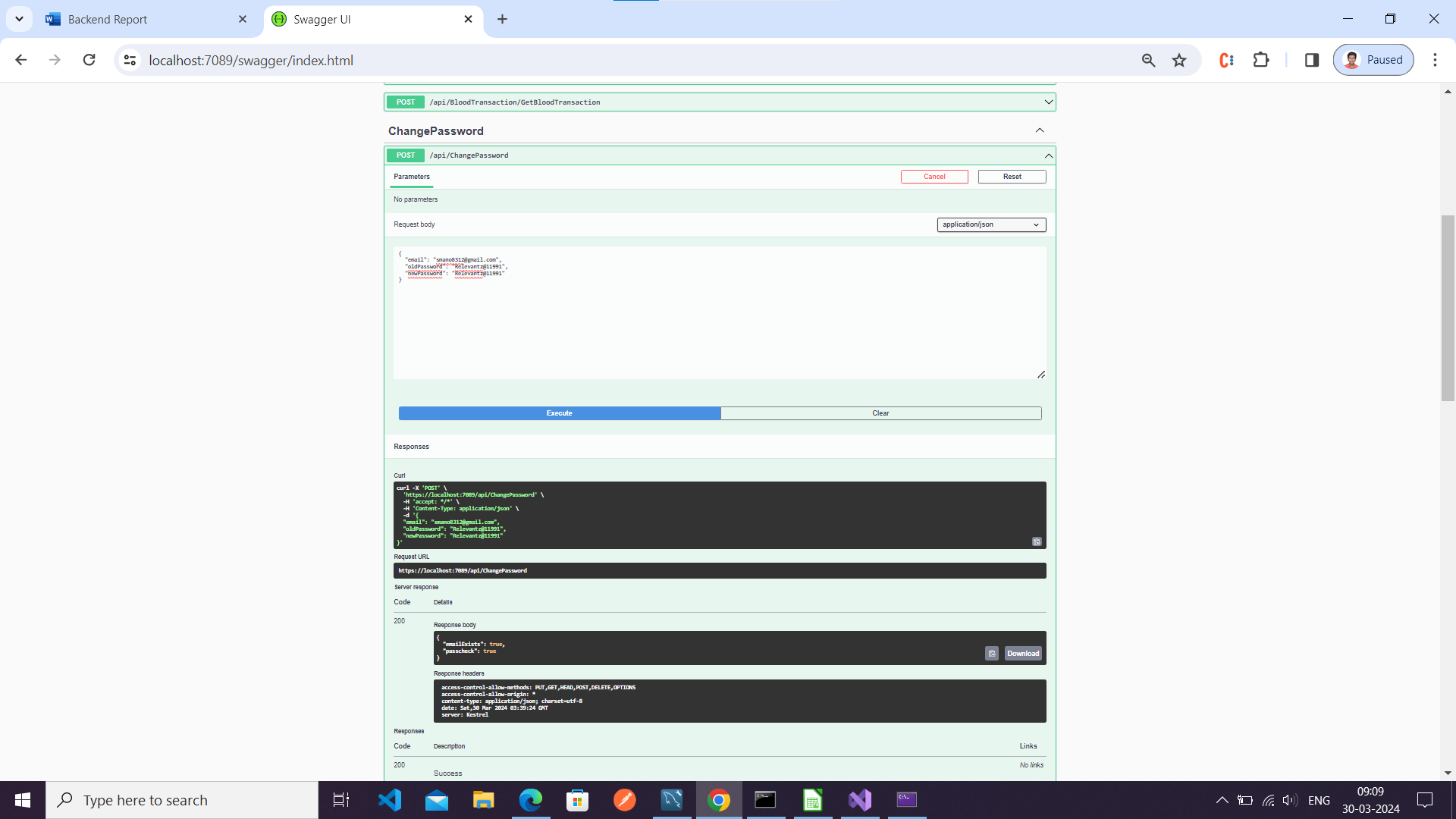
Login



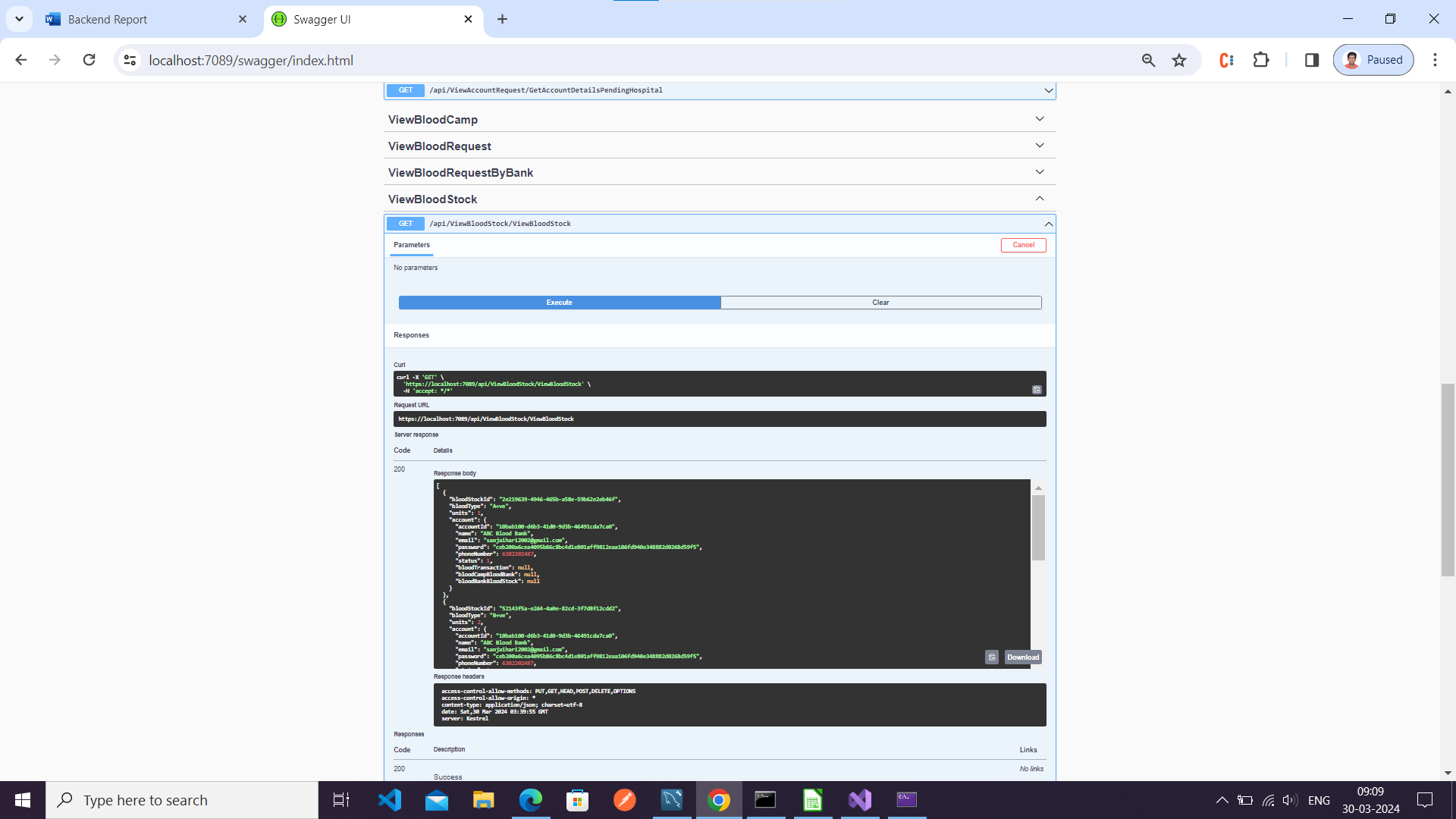
Role



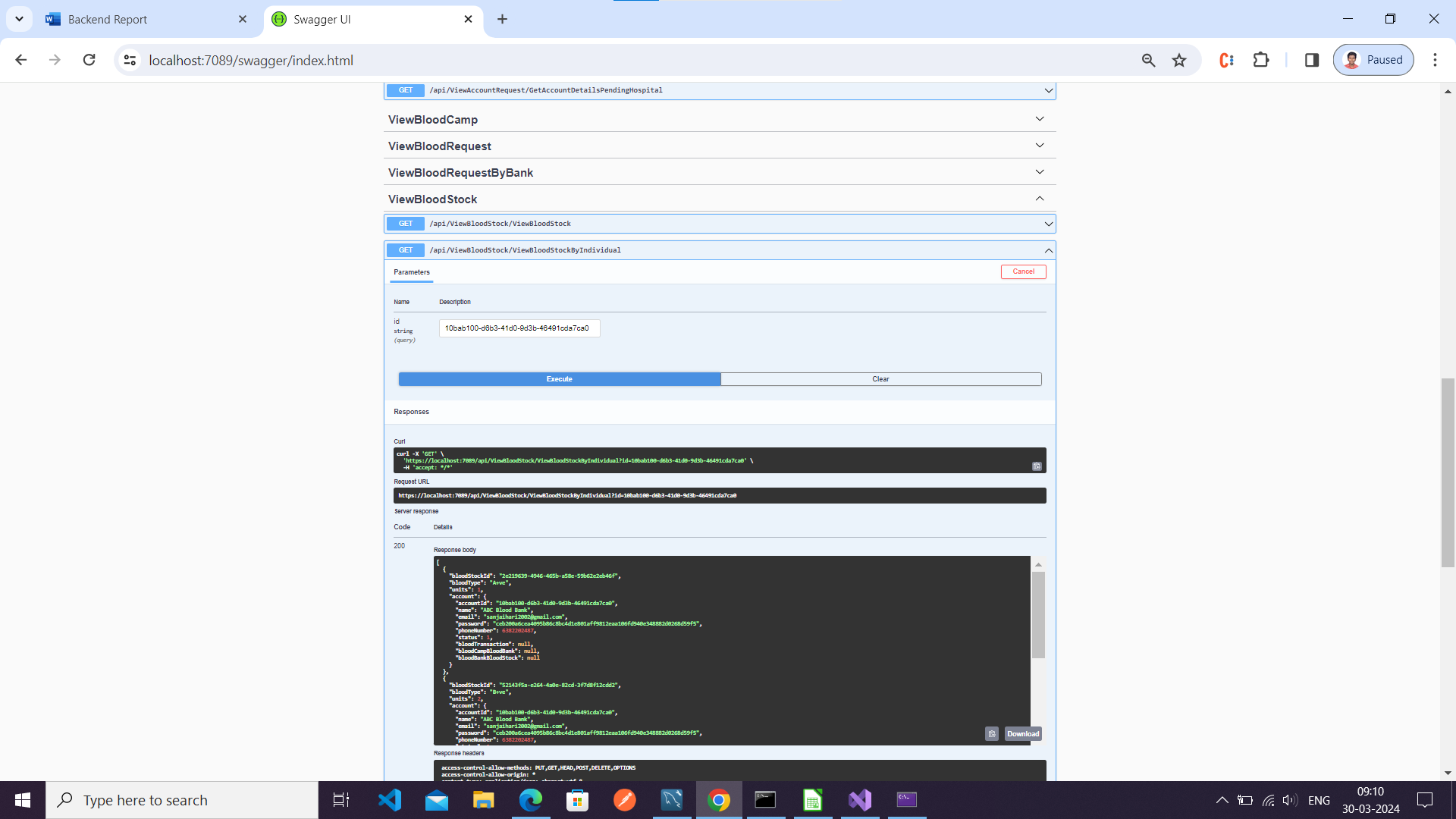
Change Password



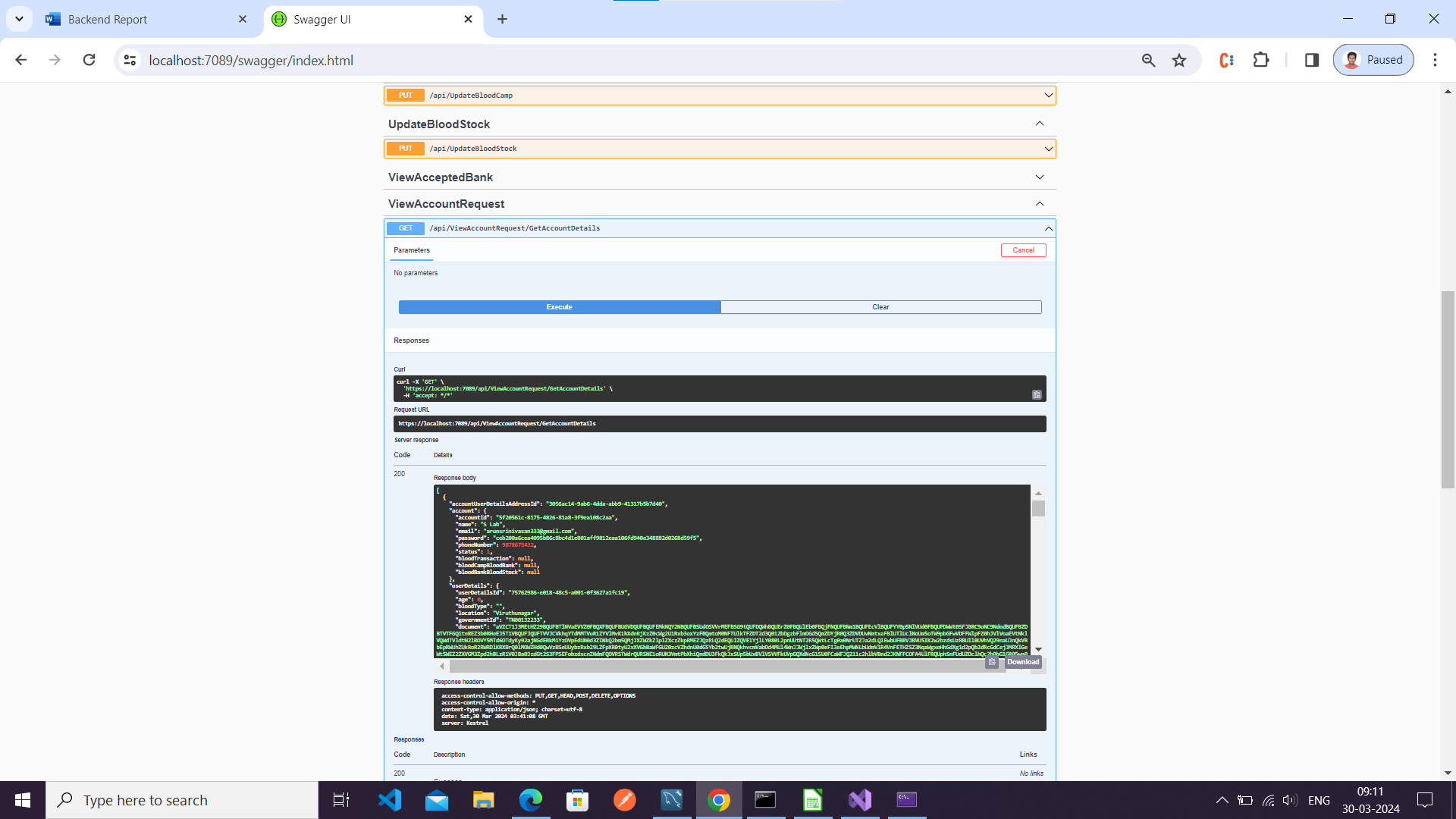
View Blood Stock



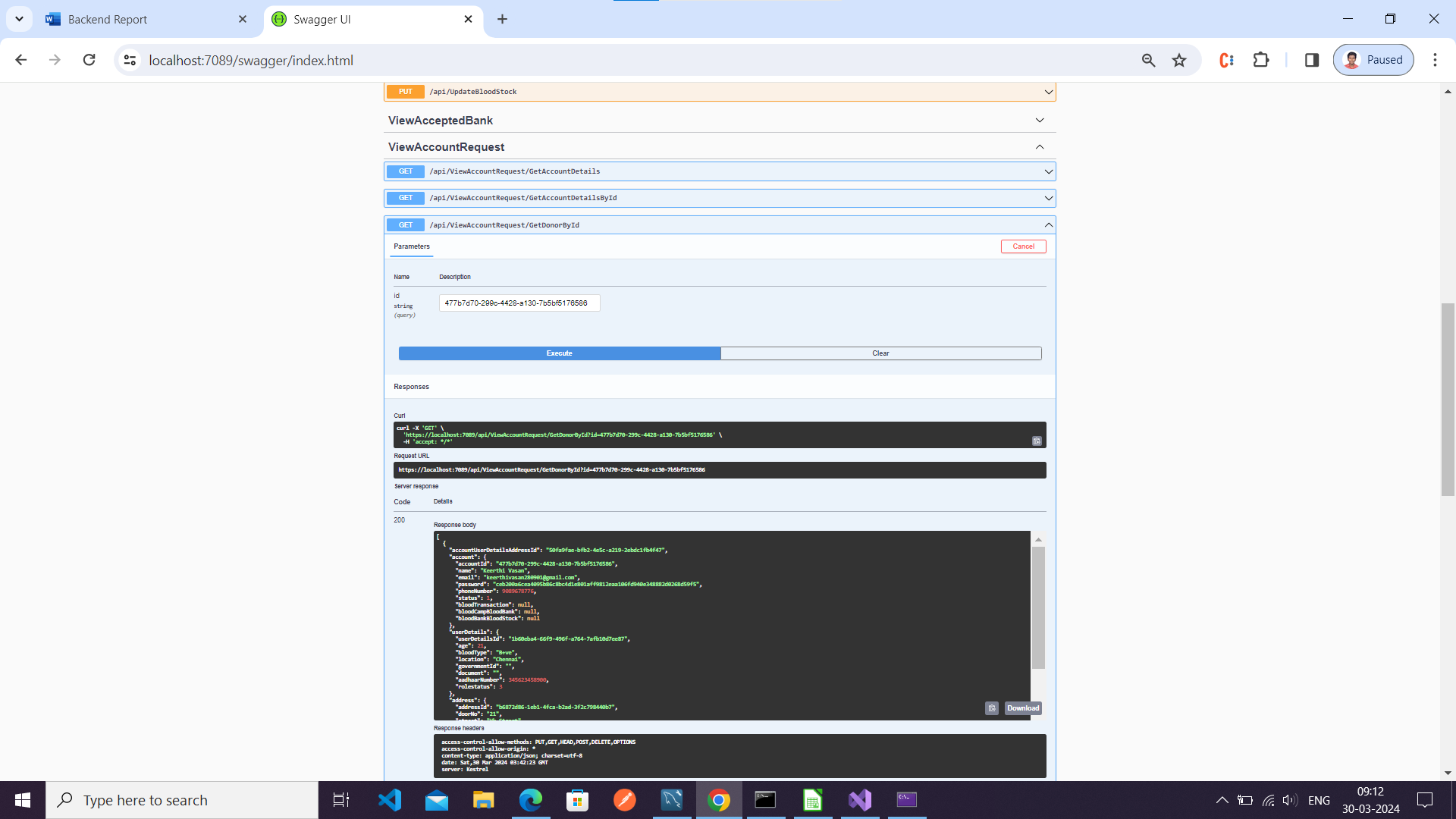
View Blood Stock By Bank



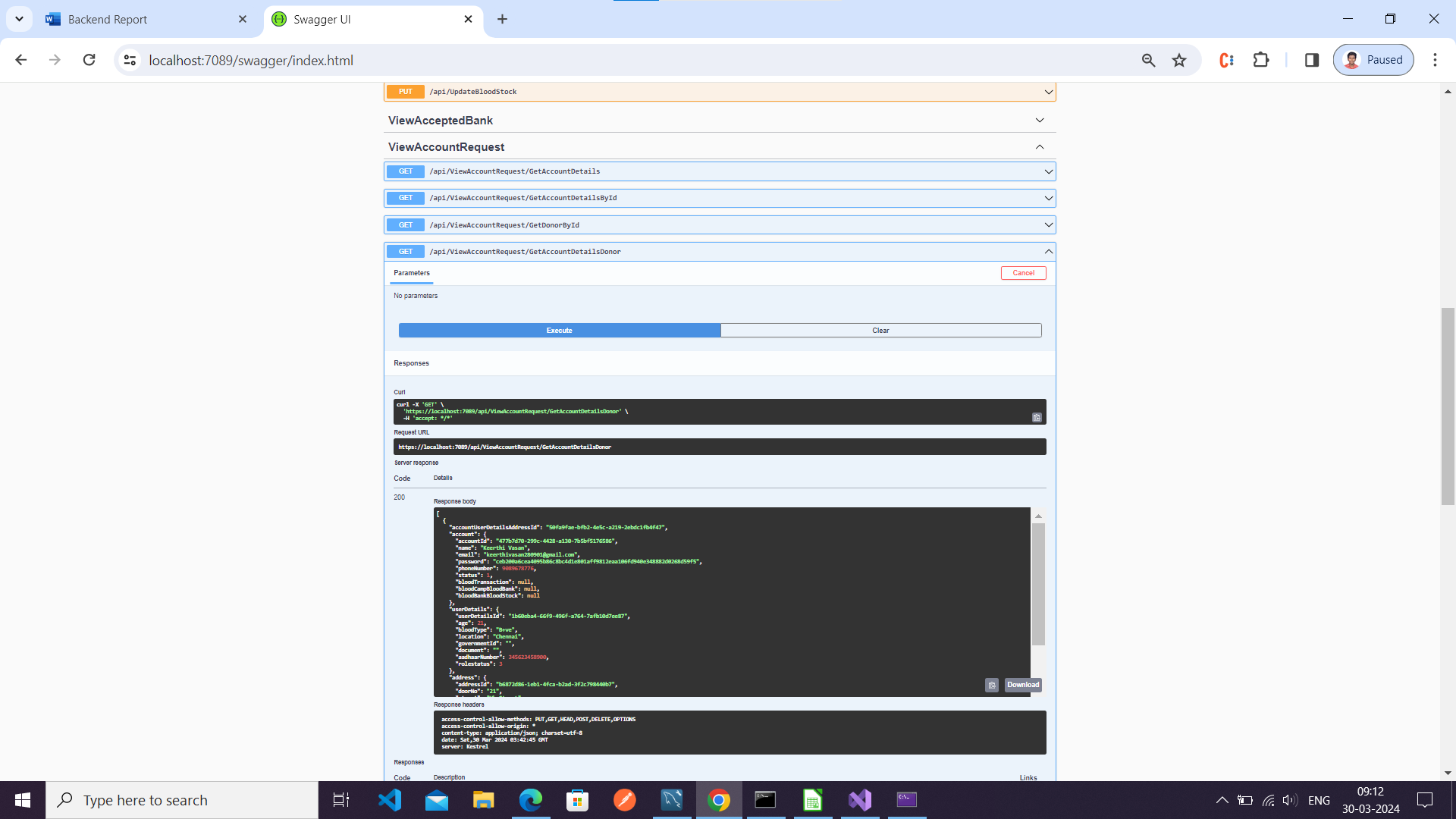
View Account Details



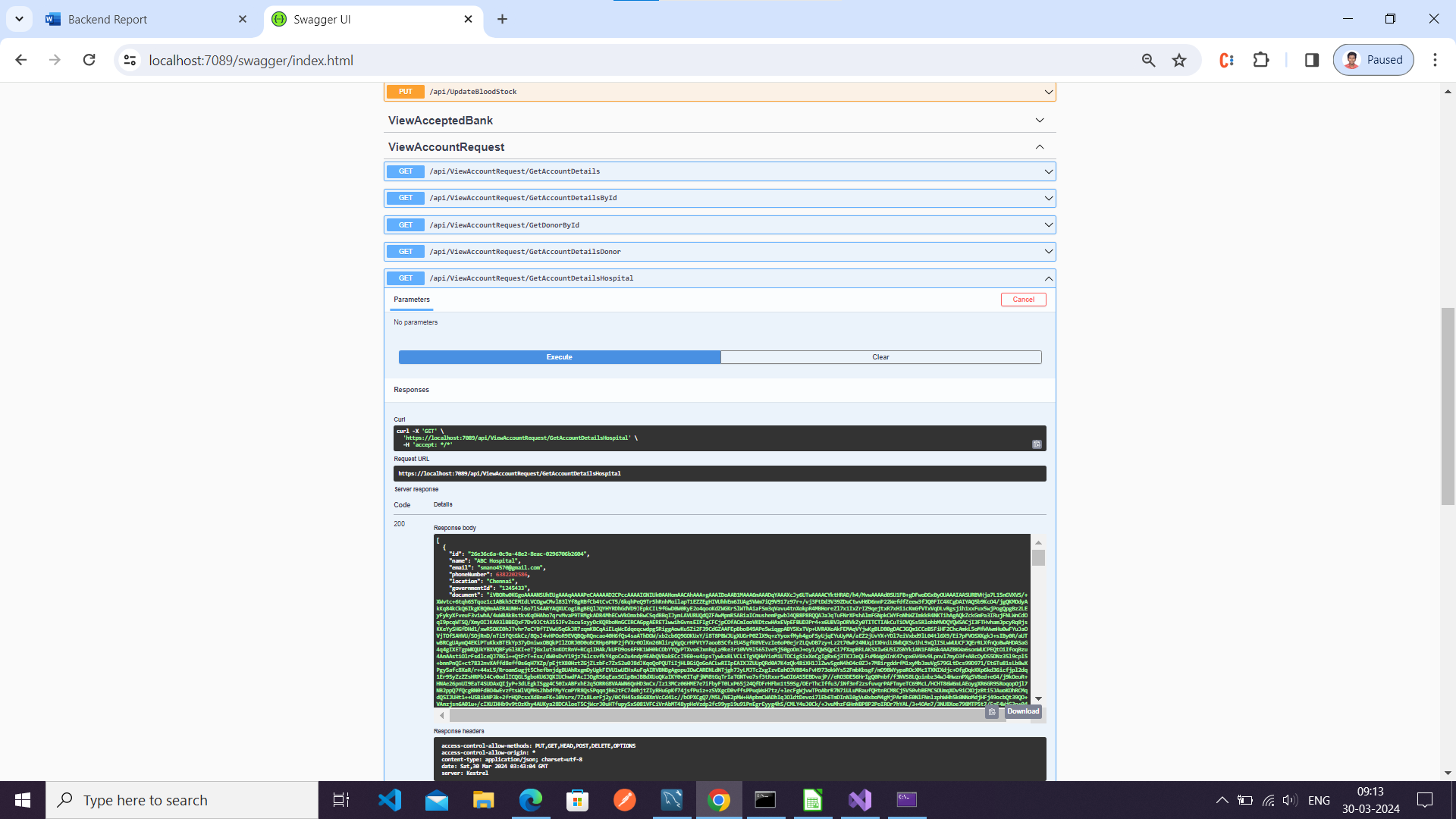
View Donor By Id



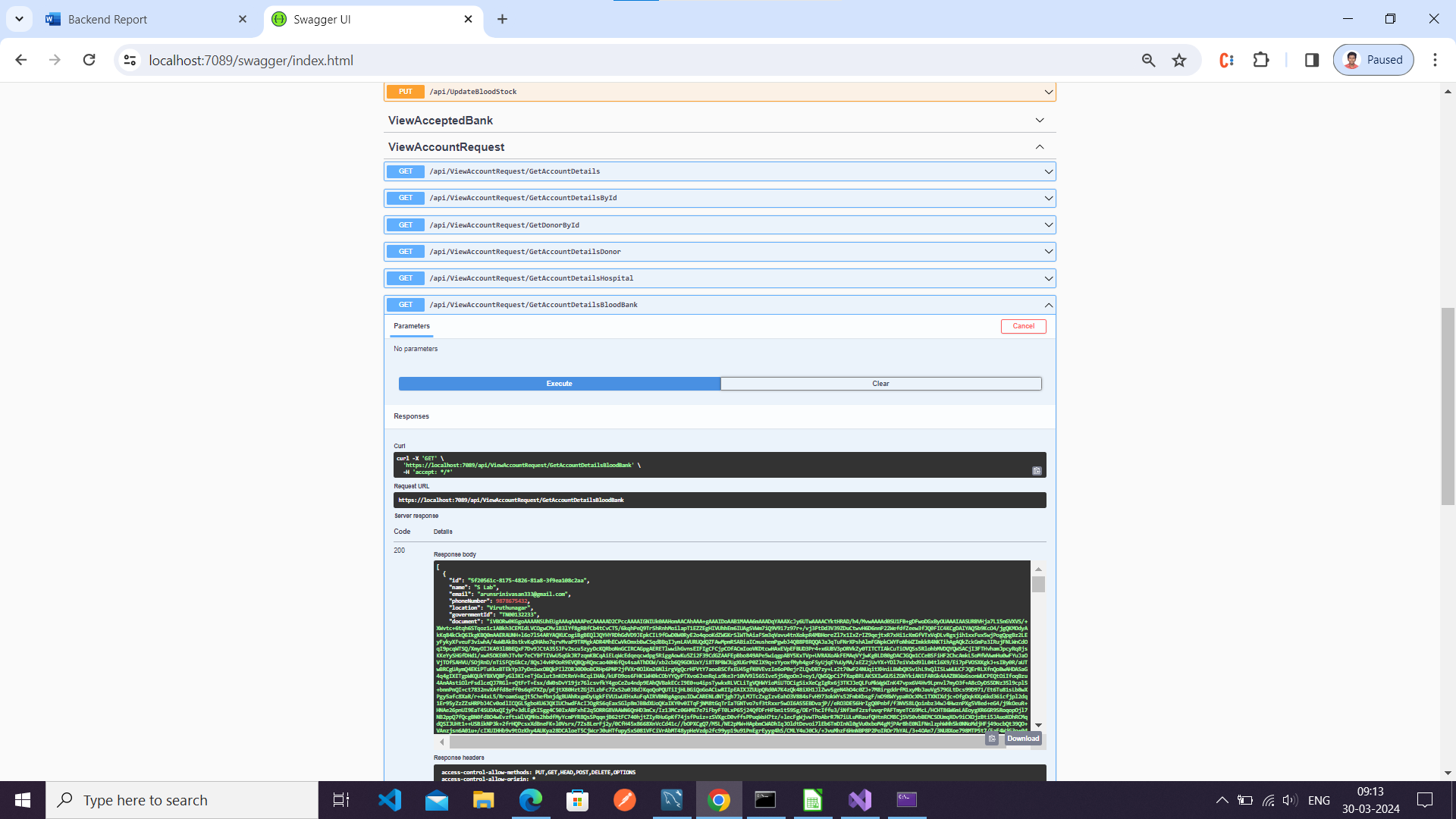
Get Donor Details



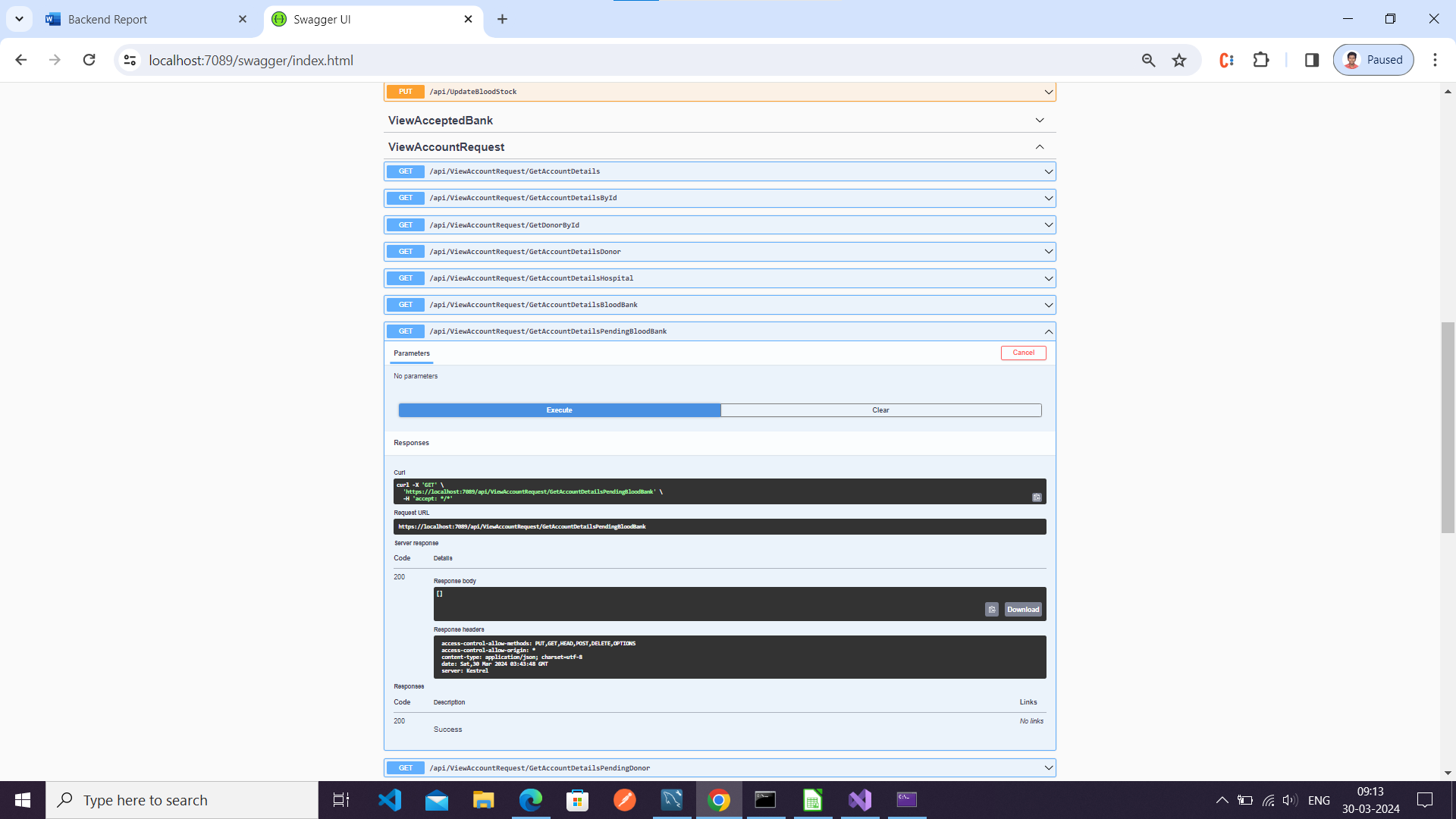
View Hospital Details



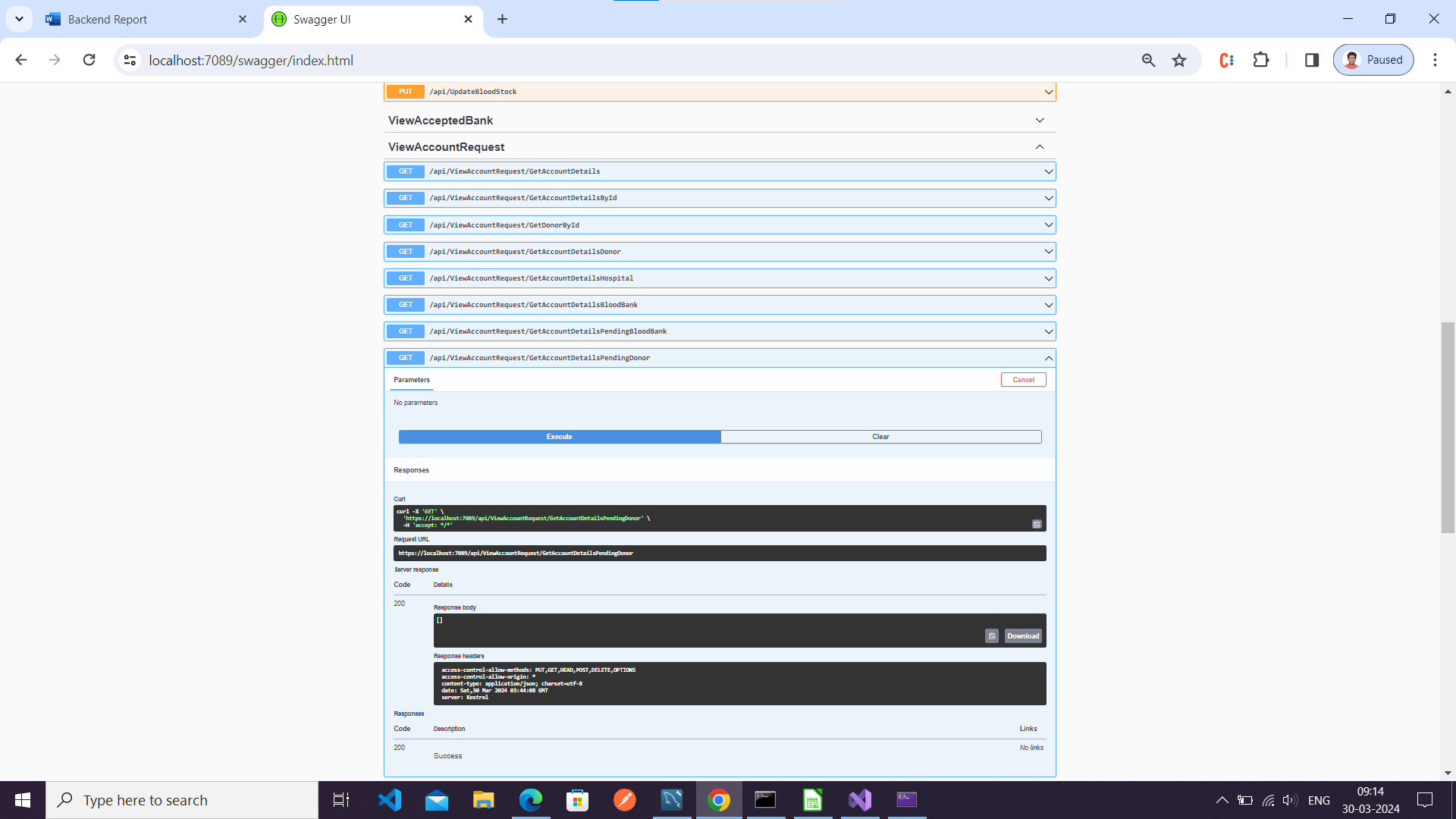
View Blood Bank Details



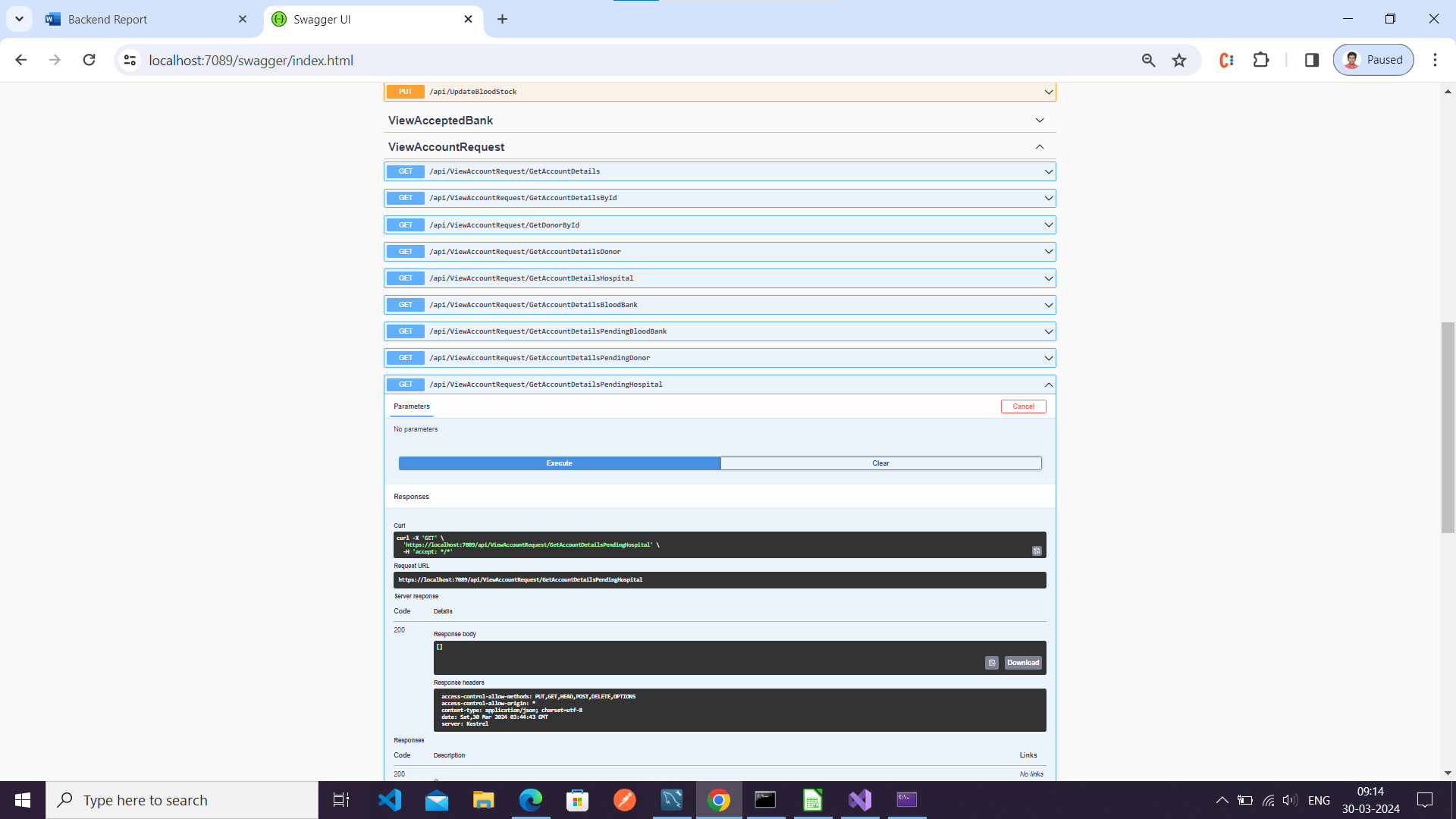
View Pending Bank Request



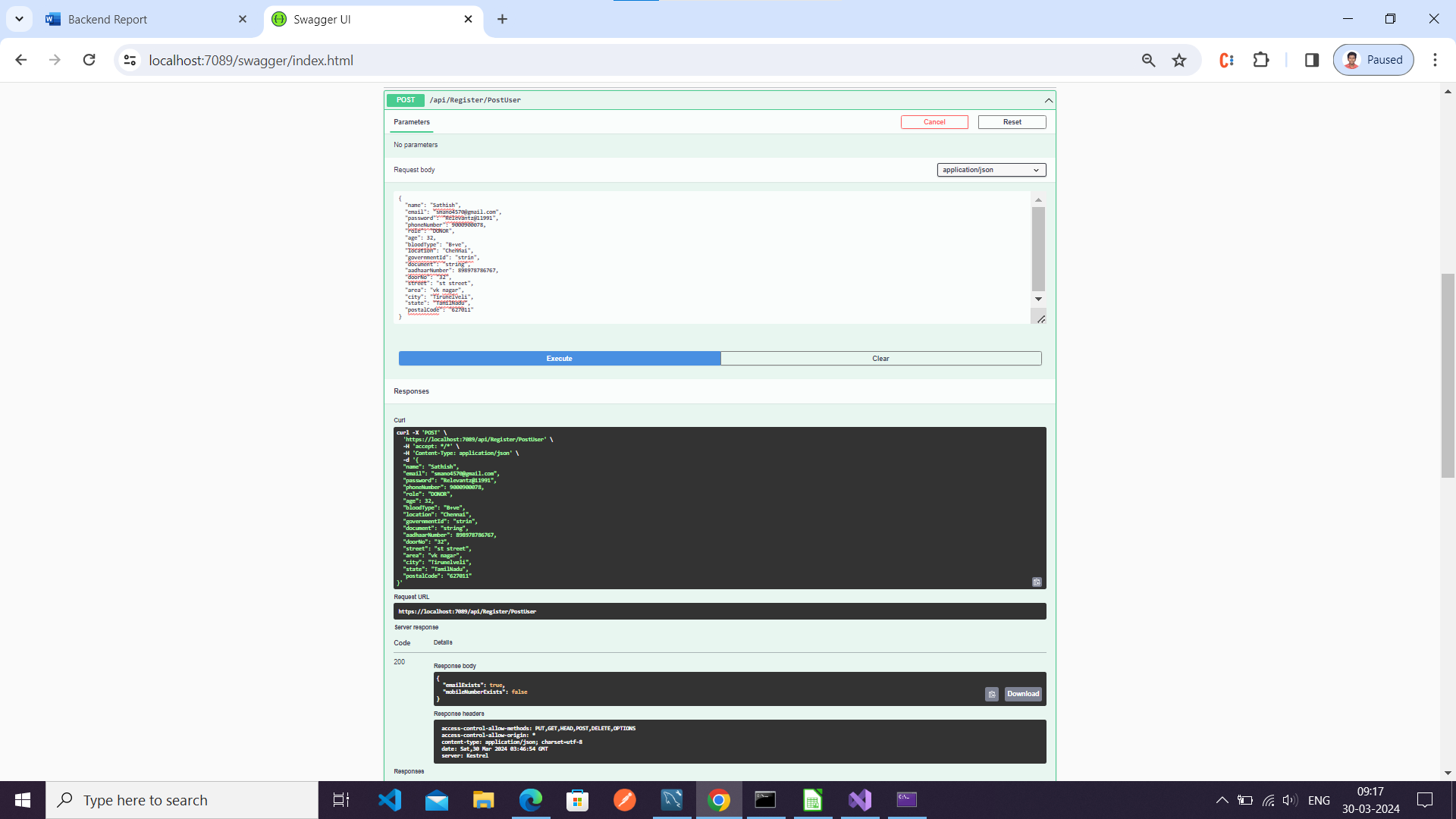
View Pending Donor Request



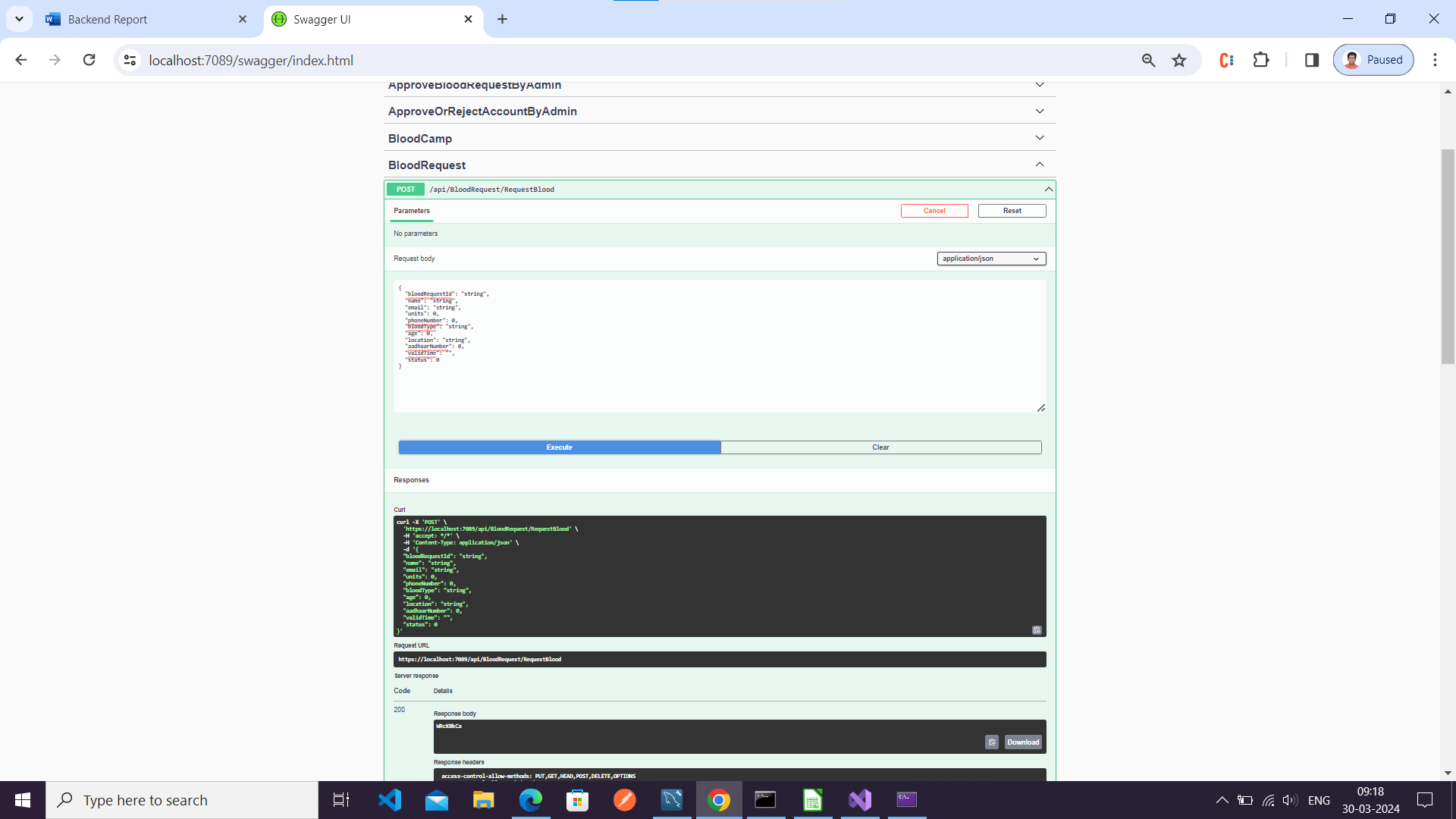
View Pending Hospital Request



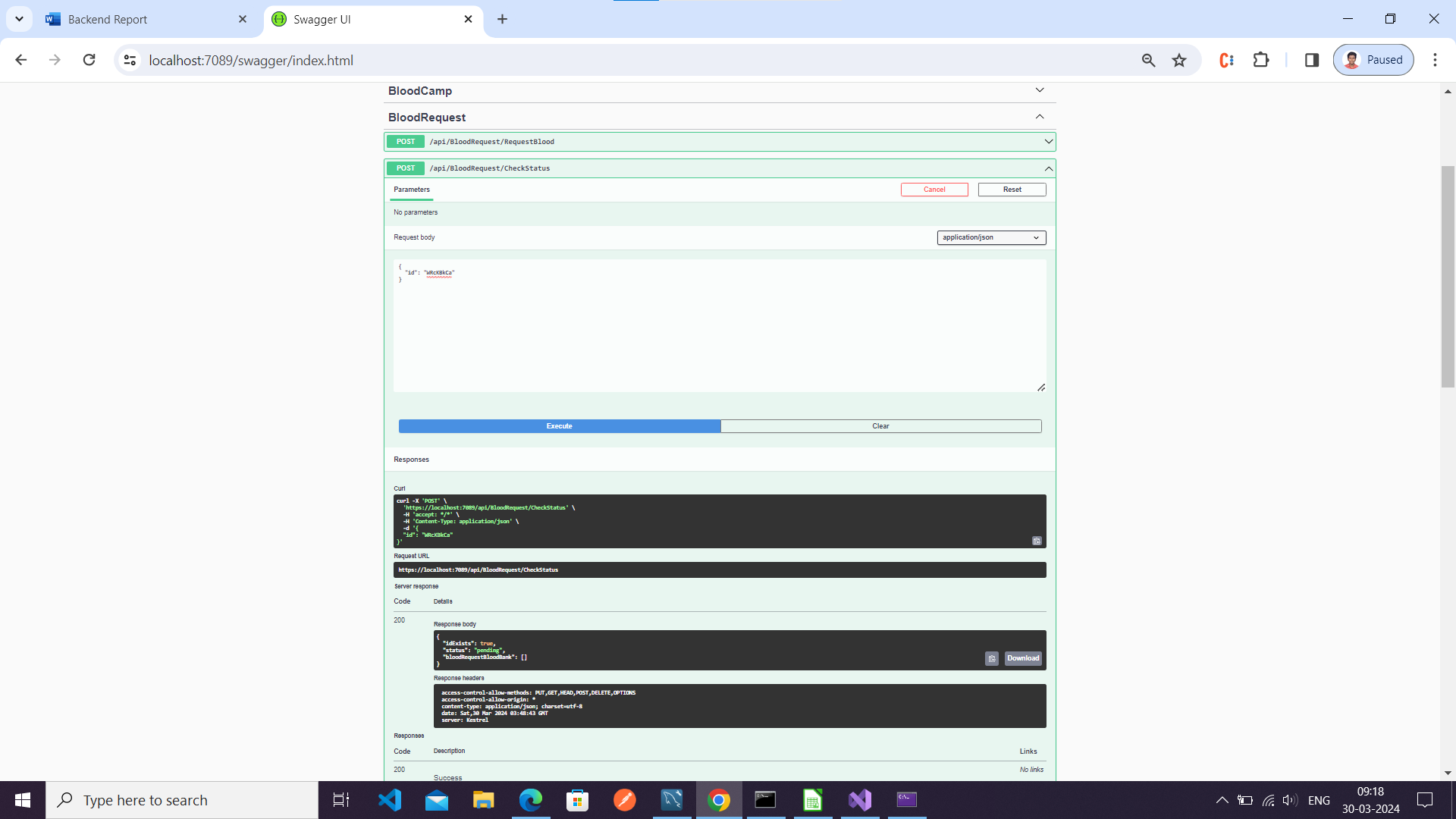
Register



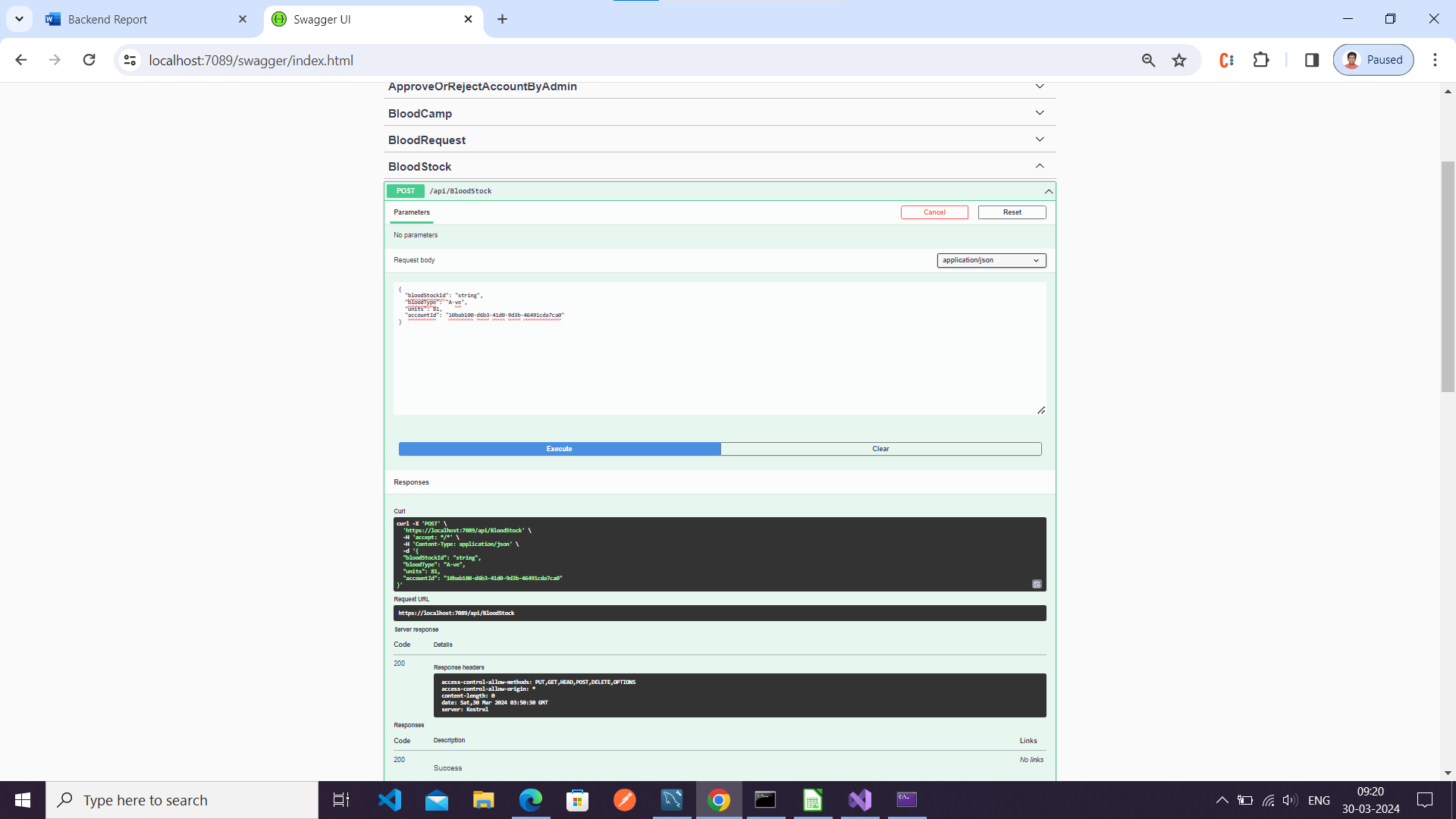
Add Blood Request



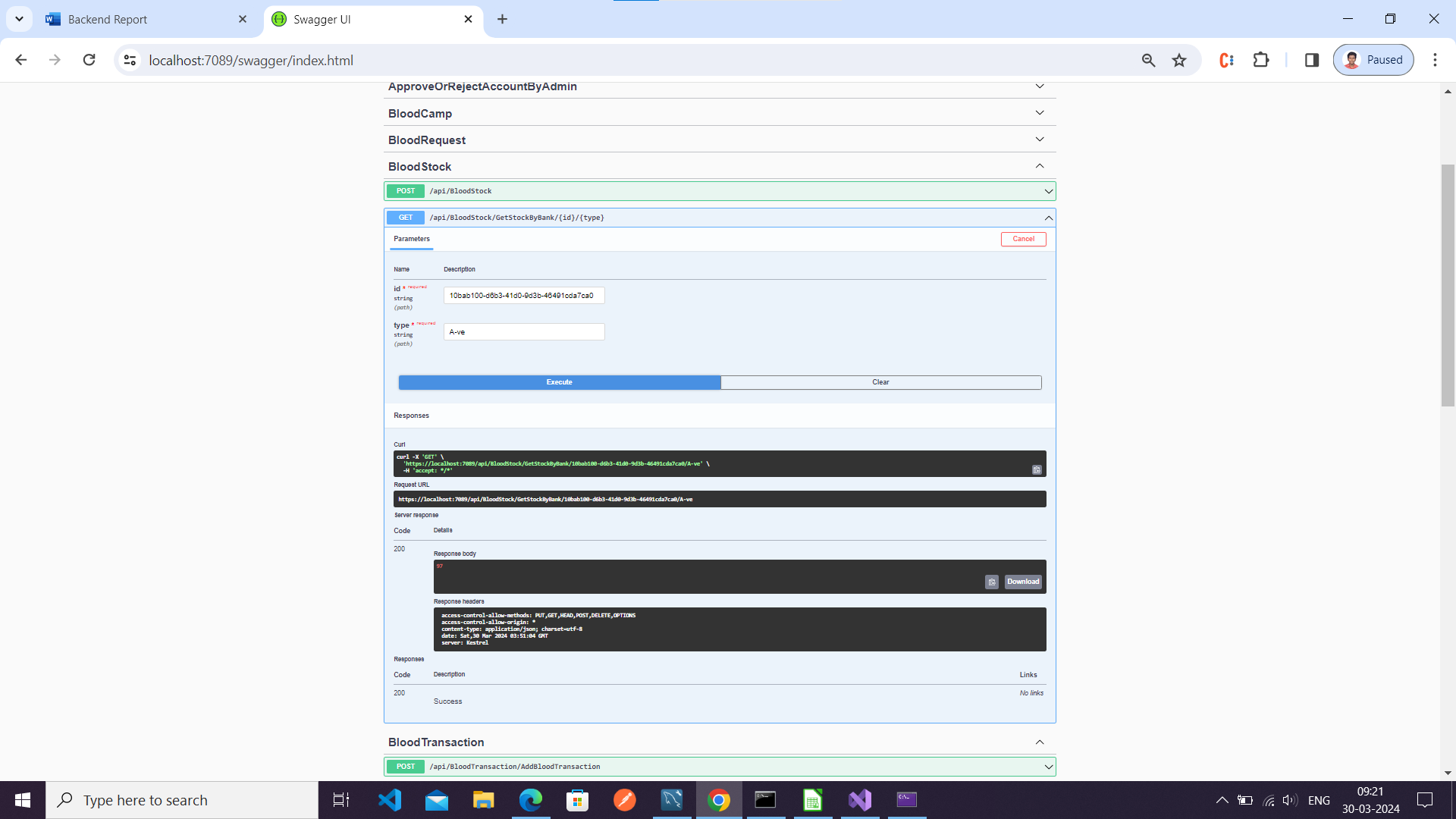
Check Request Status



Add Blood Stock



View Stock by bank and type



Add blood transaction

