OrderBlockChaininfo.cs

```
namespace TheDesiAchaar.API.Models
{
    public class OrderBlockChainInfo
    {
        public int OrderId { get; set; }
        public string OrderStatus { get; set; }
        public DateTime OrderUpdateDate { get; set; }
    }
}
```

TheDesiAcharChainService.cs

```
using Nethereum.Contracts.ContractHandlers;
using Nethereum.RPC.Eth.DTOs;
using System.Threading;
using System.Threading.Tasks;
using TheDesiAchar.Contracts.Contracts.TheDesiAcharChain.ContractDefinition;

namespace TheDesiAchar.Contracts.Contracts.TheDesiAcharChain
{
    public partial class TheDesiAcharChainService
    {
        public Task<TransactionReceipt>
DeployContractAndWaitForReceiptAsync(Nethereum.Web3.Web3 web3,
TheDesiAcharChainDeployment theDesiAcharChainDeployment, CancellationTokenSource
cancellationTokenSource = null)
        {
            return
web3.Eth.GetContractDeploymentHandler<TheDesiAcharChainDeployment>().SendRequestA
ndWaitForReceiptAsync(theDesiAcharChainDeployment, cancellationTokenSource);
        }

        public Task<string> DeployContractAsync(Nethereum.Web3.Web3 web3,
TheDesiAcharChainDeployment theDesiAcharChainDeployment)
        {
            return
web3.Eth.GetContractDeploymentHandler<TheDesiAcharChainDeployment>().SendRequestA
sync(theDesiAcharChainDeployment);
        }

        public async Task<TheDesiAcharChainService>
DeployContractAndGetServiceAsync(Nethereum.Web3.Web3 web3,
TheDesiAcharChainDeployment theDesiAcharChainDeployment, CancellationTokenSource
cancellationTokenSource = null)
        {
            var receipt = await DeployContractAndWaitForReceiptAsync(web3,
theDesiAcharChainDeployment, cancellationTokenSource);
            return new TheDesiAcharChainService(web3, receipt.ContractAddress);
        }

        protected Nethereum.Web3.Web3 Web3{ get; }

        public ContractHandler ContractHandler { get; }
```

```
        public TheDesiAcharChainService(Nethereum.Web3.Web3 web3, string
contractAddress)
        {
            Web3 = web3;
            ContractHandler = web3.Eth.GetContractHandler(contractAddress);
        }

        public Task<string> RequestMessageQueryAsync(RequestMessageFunction
requestMessageFunction, BlockParameter blockParameter = null)
        {
            return ContractHandler.QueryAsync<RequestMessageFunction,
string>(requestMessageFunction, blockParameter);
        }


        public Task<string> RequestMessageQueryAsync(BlockParameter
blockParameter = null)
        {
            return ContractHandler.QueryAsync<RequestMessageFunction,
string>(null, blockParameter);
        }

        public Task<string> RequestorQueryAsync(RequestorFunction
requestorFunction, BlockParameter blockParameter = null)
        {
            return ContractHandler.QueryAsync<RequestorFunction,
string>(requestorFunction, blockParameter);
        }


        public Task<string> RequestorQueryAsync(BlockParameter blockParameter =
null)
        {
            return ContractHandler.QueryAsync<RequestorFunction, string>(null,
blockParameter);
        }

        public Task<string> ResponderQueryAsync(ResponderFunction
responderFunction, BlockParameter blockParameter = null)
        {
            return ContractHandler.QueryAsync<ResponderFunction,
string>(responderFunction, blockParameter);
        }


        public Task<string> ResponderQueryAsync(BlockParameter blockParameter =
null)
        {
            return ContractHandler.QueryAsync<ResponderFunction, string>(null,
blockParameter);
        }

        public Task<string> ResponseMessageQueryAsync(ResponseMessageFunction
responseMessageFunction, BlockParameter blockParameter = null)
        {
            return ContractHandler.QueryAsync<ResponseMessageFunction,
string>(responseMessageFunction, blockParameter);
        }


        public Task<string> ResponseMessageQueryAsync(BlockParameter
blockParameter = null)
        {
```

```
            return ContractHandler.QueryAsync<ResponseMessageFunction,
string>(null, blockParameter);
        }

        public Task<string> SendRequestRequestAsync(SendRequestFunction
sendRequestFunction)
        {
             return ContractHandler.SendRequestAsync(sendRequestFunction);
        }

        public Task<TransactionReceipt>
SendRequestRequestAndWaitForReceiptAsync(SendRequestFunction sendRequestFunction,
CancellationTokenSource cancellationToken = null)
        {
             return
ContractHandler.SendRequestAndWaitForReceiptAsync(sendRequestFunction,
cancellationToken);
        }

        public Task<string> SendRequestRequestAsync(string requestMessage)
        {
            var sendRequestFunction = new SendRequestFunction();
                sendRequestFunction.RequestMessage = requestMessage;

             return ContractHandler.SendRequestAsync(sendRequestFunction);
        }

        public Task<TransactionReceipt>
SendRequestRequestAndWaitForReceiptAsync(string requestMessage,
CancellationTokenSource cancellationToken = null)
        {
            var sendRequestFunction = new SendRequestFunction();
                sendRequestFunction.RequestMessage = requestMessage;

             return
ContractHandler.SendRequestAndWaitForReceiptAsync(sendRequestFunction,
cancellationToken);
        }

        public Task<string> SendResponseRequestAsync(SendResponseFunction
sendResponseFunction)
        {
             return ContractHandler.SendRequestAsync(sendResponseFunction);
        }

        public Task<TransactionReceipt>
SendResponseRequestAndWaitForReceiptAsync(SendResponseFunction
sendResponseFunction, CancellationTokenSource cancellationToken = null)
        {
             return
ContractHandler.SendRequestAndWaitForReceiptAsync(sendResponseFunction,
cancellationToken);
        }

        public Task<string> SendResponseRequestAsync(string responseMessage)
        {
            var sendResponseFunction = new SendResponseFunction();
                sendResponseFunction.ResponseMessage = responseMessage;

             return ContractHandler.SendRequestAsync(sendResponseFunction);
        }
```

```
        public Task<TransactionReceipt>
SendResponseRequestAndWaitForReceiptAsync(string responseMessage,
CancellationTokenSource cancellationToken = null)
        {
            var sendResponseFunction = new SendResponseFunction();
                sendResponseFunction.ResponseMessage = responseMessage;

             return
ContractHandler.SendRequestAndWaitForReceiptAsync(sendResponseFunction,
cancellationToken);
        }

        public Task<byte> StateQueryAsync(StateFunction stateFunction,
BlockParameter blockParameter = null)
        {
            return ContractHandler.QueryAsync<StateFunction, byte>(stateFunction,
blockParameter);
        }


        public Task<byte> StateQueryAsync(BlockParameter blockParameter = null)
        {
            return ContractHandler.QueryAsync<StateFunction, byte>(null,
blockParameter);
        }
    }
}
```

IBlockChainService.cs

```
using Nethereum.RPC.Eth.DTOs;
using TheDesiAchaar.API.Models;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.API.Contracts
{
    public interface IBlockChainService
    {
        Task<Blockchain> CreateOrderTransactions(OrderBlockChainInfo
orderBlockChainInfo);
    }
```

BlockChainService.cs

```
using TheDesiAchar.Contracts.Contracts.TheDesiAcharChain.ContractDefinition;

namespace TheDesiAchaar.API.Services
{
    public class BlockChainService : IBlockChainService
    {
        private static readonly string _ethFrameWorkAddress =
"http://127.0.0.1:7545";
        private static readonly string _accountAddress =
"0xef416B8d6f13115D46f2f49aeB530E717d5ED52d";
```

```
        private static readonly string _privateKey =
"f488283351116c218f0553e33d3af5ebcf1efd417a615e849b3970ec8d558724";
        public async Task<Blockchain> CreateOrderTransactions(OrderBlockChainInfo
orderBlockChainInfo)
        {
            var contractData = new TheDesiAcharChainDeployment
            {
                Message = JsonSerializer.Serialize(orderBlockChainInfo),
                FromAddress = Web3.GetAddressFromPrivateKey(_privateKey)
            };
            var web3 = new Web3(_ethFrameWorkAddress);
            var chainService = new TheDesiAcharChainService(web3,
_accountAddress);
            var transactionReceipt = await
chainService.DeployContractAndWaitForReceiptAsync(web3, contractData, new
CancellationTokenSource());
            return new Blockchain
            {
                BlockHash = transactionReceipt.BlockHash,
                OrderId = orderBlockChainInfo.OrderId,
                From = transactionReceipt.From,
                TransactionHash = transactionReceipt.TransactionHash,
                BlockNumber = transactionReceipt.BlockNumber.ToString(),
                TransactionIndex =
transactionReceipt.TransactionIndex.ToString(),
                CreatedBy = "System",
                LastUpdatedBy = "System",
                CreatedOn = DateTime.UtcNow,
                LastUpdatedOn = DateTime.UtcNow
            };
        }
    }
}
```

OrdersController.cs

```
using Microsoft.AspNetCore.Mvc;
using TheDesiAchaar.API.Contracts;
using TheDesiAchaar.API.Models;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrdersController : ControllerBase
    {
        private readonly IOrderService _orderService;

        public OrdersController(IOrderService orderService)
        {
            _orderService = orderService;
        }

        [HttpGet("{userId}")]
        public ActionResult<IEnumerable<OrderDetails>> GetOrders(int userId, int
offset) =>
            Ok(_orderService.GetOrders(userId, offset));
```

```
        [HttpPost]
        public ActionResult<IEnumerable<Orders>> SaveOrders(IEnumerable<Orders>
orders) =>
            Ok(_orderService.SaveOrders(orders));
        [HttpGet]
        public ActionResult<IEnumerable<OrderDetails>> GetSellerOrders(int
userId, int offset) =>
            Ok(_orderService.GetOrdersForSeller(userId, offset));

        [HttpPut]
        public async Task<ActionResult<bool>> UpdateOrderStatusAsync(int orderId,
string orderStatus) =>
            Ok(await _orderService.UpdateOrderStatusAsync(orderId, orderStatus));
    }
}
```

OrderService.cs

```
using Microsoft.EntityFrameworkCore;
using TheDesiAchaar.API.Contracts;
using TheDesiAchaar.API.Models;
using TheDesiAchaar.Data;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.API.Services
{
    public class OrderService : IOrderService
    {
        private readonly DataContext _dataContext;
        private readonly IBlockChainService _blockChainService;
        private static readonly int _pageSize = 5;
        public OrderService(DataContext dataContext, IBlockChainService
blockChainService)
        {
            _dataContext = dataContext;
            _blockChainService = blockChainService;
        }

        public IEnumerable<OrderDetails> GetOrders(int userId, int offset)
        {
            return _dataContext.Orders.Where(_ => _.UserId ==
userId).OrderByDescending(_ => _.CreatedOn)
                .Join(_dataContext.Product, o => o.ProductId, p => p.Id, (o, p)
=> new { o, p }).Select(_ => new OrderDetails
                {
                    OrderDate = _.o.CreatedOn,
                    OrderStatus = _.o.Status,
                    Price = _.p.Price,
                    Quantity = _.o.Quantity,
                    ProductName = _.p.Name,
                    Id = _.o.Id,
                    Image = Convert.ToBase64String(_.p.Image)
                }).Skip(_pageSize * offset).Take(_pageSize).ToList();
        }

        public IEnumerable<Orders> SaveOrders(IEnumerable<Orders> orders)
        {
            var productIds = orders.Select(o => o.ProductId);
```

```
            var products = _dataContext.Product.Where(_ =>
productIds.Contains(_.Id));
            foreach (var order in orders)
            {
                order.CreatedOn = DateTime.UtcNow;
                order.LastUpdatedOn = DateTime.UtcNow;
                order.CreatedBy = "Test";
                order.LastUpdatedBy = "Test";
                var currentQty = products.First(_ => _.Id ==
order.ProductId).Quantity - order.Quantity;
                products.First(_ => _.Id == order.ProductId).Quantity =
currentQty;
            }
            _dataContext.AttachRange(products);
            _dataContext.Orders.AddRange(orders);
            _dataContext.SaveChanges();
            return orders;
        }

        public IEnumerable<OrderDetails> GetOrdersForSeller(int userId, int
offset)
        {
            return _dataContext.Orders.OrderByDescending(_ => _.CreatedOn)
                .Join(_dataContext.Product.Where(_ => _.UserId == userId), o =>
o.ProductId, p => p.Id, (o, p) => new { o, p })
                .Join(_dataContext.Users, ou => ou.o.UserId, u => u.Id, (ou, u)
=> new {ou.o, ou.p, u}).Select(_ => new OrderDetails
                {
                    OrderDate = _.o.CreatedOn,
                    OrderStatus = _.o.Status,
                    Price = _.p.Price,
                    Quantity = _.o.Quantity,
                    ProductName = _.p.Name,
                    Id = _.o.Id,
                    Image = Convert.ToBase64String(_.p.Image),
                    ClientAddress = _.u.Address,
                    Email = _.u.EmailAddress,
                    PhoneNumber = _.u.PhoneNumber

                }).Skip(_pageSize * offset).Take(_pageSize).ToList();
        }

        public async Task<bool> UpdateOrderStatusAsync(int orderId, string
orderStatus)
        {
            var order = await _dataContext.Orders.FirstAsync(_ => _.Id ==
orderId);
            order.Status = orderStatus;
            order.LastUpdatedOn = DateTime.UtcNow;
            _dataContext.Attach(order);
            await _dataContext.SaveChangesAsync();
            var transactionReceipt = await
_blockChainService.CreateOrderTransactions(
                new OrderBlockChainInfo {
                    OrderId = order.Id,
                    OrderStatus = orderStatus,
                    OrderUpdateDate = order.LastUpdatedOn });
            _dataContext.Blockchain.Add(transactionReceipt);
            await _dataContext.SaveChangesAsync();
            return true;
        }
    }
}
```

RegisterRequest.cs

```
namespace TheDesiAchaar.Models.Users;

using System.ComponentModel.DataAnnotations;

public class RegisterRequest
{
    [Required]
    public string FirstName { get; set; }

    [Required]
    public string LastName { get; set; }

    [Required]
    public string Username { get; set; }

    [Required]
    public string Password { get; set; }
    [Required]
    public string Role { get; set; }
}
```

ProductAddModel.cs

```
namespace TheDesiAchaar.API.Models
{
    public class ProductAddModel
    {
        public string Name { get; set; }
        public int Quantity { get; set; }
        public decimal Price { get; set; }
        public string Image { get; set; }
        public int UserId { get; set; }
        public DateTime DateOfMfg { get; set; }
        public DateTime DateOfExp { get; set; }
    }
}
```

IOrderService.cs

```
using TheDesiAchaar.API.Models;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.API.Contracts
{
    public interface IOrderService
    {
```

```
        IEnumerable<OrderDetails> GetOrders(int userId, int offset);
        IEnumerable<Orders> SaveOrders(IEnumerable<Orders> orders);
        IEnumerable<OrderDetails> GetOrdersForSeller(int userId, int offset);
        Task<bool> UpdateOrderStatusAsync(int orderId, string orderStatus);
    }
}
```

Order.cs

```
namespace TheDesiAchaar.Models
{
    public class Orders : DbEntity<int>
    {
        public int UserId { get; set; }
        public int ProductId { get; set; }
        public int Quantity { get; set; }
        public string Status { get; set; }
    }
}
```

OrderDetails.cs

```
namespace TheDesiAchaar.API.Models
{
    public class OrderDetails
    {
        public string ProductName { get; set; }
        public int Id { get; set; }
        public int Quantity { get; set; }
        public decimal Price { get; set; }
        public DateTime OrderDate { get; set; }
        public string OrderStatus { get; set; }
        public string Image { get; set; }
        public string? ClientAddress { get; set; }
        public string? Email { get; set; }
        public string? PhoneNumber { get; set; }
    }
}
```

JwtMiddleware.cs

```
namespace TheDesiAchaar.Authorization;

using TheDesiAchaar.API.Contracts;
using TheDesiAchaar.Services;

public class JwtMiddleware
{
    private readonly RequestDelegate _next;
```

```
    public JwtMiddleware(RequestDelegate next)
    {
        _next = next;
    }

    public async Task Invoke(HttpContext context, IUserService userService,
IJwtUtils jwtUtils)
    {
        var token =
context.Request.Headers["Authorization"].FirstOrDefault()?.Split(" ").Last();
        var userId = jwtUtils.ValidateToken(token);
        if (userId != null)
        {
            // attach user to context on successful jwt validation
            context.Items["User"] = userService.GetById(userId.Value);
        }

        await _next(context);
    }
}
```

UserControll.cs

```
namespace TheDesiAchaar.Controllers;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;
using TheDesiAchaar.API.Contracts;
using TheDesiAchaar.Authorization;
using TheDesiAchaar.Helpers;
using TheDesiAchaar.Models.Users;

[Authorize]
[ApiController]
[Route("api/[controller]")]
public class UserController : ControllerBase
{
    private readonly IUserService _userService;

    public UserController(
        IUserService userService,
        IOptions<AppSettings> appSettings)
    {
        _userService = userService;
    }

    [AllowAnonymous]
    [HttpPost("authenticate")]
    public IActionResult Authenticate(AuthenticateRequest model)
    {
        var response = _userService.Authenticate(model);
        return Ok(response);
    }

    [AllowAnonymous]
    [HttpPost("register")]
    public IActionResult Register(RegisterRequest model)
    {
        _userService.Register(model);
        return Ok(new { message = "Registration successful" });
    }
```

```
    [HttpGet]
    public IActionResult GetAll()
    {
        var users = _userService.GetAll();
        return Ok(users);
    }

    [HttpGet("{id}")]
    public IActionResult GetById(int id)
    {
        var user = _userService.GetById(id);
        return Ok(user);
    }

    [HttpPut("{id}")]
    public IActionResult Update(int id, UpdateRequest model)
    {
        _userService.Update(id, model);
        return Ok(new { message = "User updated successfully" });
    }

    [HttpDelete("{id}")]
    public IActionResult Delete(int id)
    {
        _userService.Delete(id);
        return Ok(new { message = "User deleted successfully" });
    }
}
```

User.cs

```
namespace TheDesiAchaar.Entities;

using System.ComponentModel.DataAnnotations;
using System.Text.Json.Serialization;

public class User
{
    [Key]
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    [MaxLength(800)]
    public string? Address { get; set; }
    [MaxLength(100)]
    public string? EmailAddress { get; set; }
    [MaxLength(15)]
    public string? PhoneNumber { get; set; }
    public string Username { get; set; }
    public string Role { get; set; }
    [JsonIgnore]
    public string PasswordHash { get; set; }
}
```

UpdateRequest.cs

```
using System.ComponentModel.DataAnnotations;
```

```
namespace TheDesiAchaar.Models.Users;

public class UpdateRequest
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Username { get; set; }
    public string Password { get; set; }
    [MaxLength(800)]
    public string Address { get; set; }
    [MaxLength(100)]
    public string EmailAddress { get; set; }
    [MaxLength(15)]
    public string PhoneNumber { get; set; }
}
```

Program.cs

```
using Microsoft.EntityFrameworkCore;
using TheDesiAchaar.API.Contracts;
using TheDesiAchaar.API.Services;
using TheDesiAchaar.Authorization;
using TheDesiAchaar.Data;
using TheDesiAchaar.Helpers;
using TheDesiAchaar.Services;

var myAllowSpecificOrigins = "_myAllowSpecificOrigins";
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddDbContext<DataContext>(options =>
{

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection
"));
});
builder.Services.AddScoped<IJwtUtils, JwtUtils>();
builder.Services.AddTransient<IUserService, UserService>();
builder.Services.AddTransient<IProductService, ProductService>();
builder.Services.AddTransient<IOrderService, OrderService>();
builder.Services.AddTransient<IBlockChainService, BlockChainService>();
builder.Services.AddAutoMapper(typeof(Program));
builder.Services.Configure<AppSettings>(builder.Configuration.GetSection("AppSett
ings"));
builder.Services.AddCors(options =>
{
    options.AddPolicy(name: myAllowSpecificOrigins,
        builder =>
        {
            var allowedOrigins = new[] { "http://localhost:4200",
"http://localhost:4300" };
            builder.WithOrigins(allowedOrigins)
            .AllowAnyMethod()
            .AllowAnyHeader();
        });
});

var app = builder.Build();
if (app.Environment.IsDevelopment())
```

```
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
app.UseHttpsRedirection();
app.UseCors(myAllowSpecificOrigins);
app.UseAuthorization();
app.MapControllers();
app.UseMiddleware<ErrorHandlerMiddleware>();
app.UseMiddleware<JwtMiddleware>();
app.Run();
```

ProductService.cs

```
using Microsoft.AspNetCore.Mvc;
using TheDesiAchaar.API.Contracts;
using TheDesiAchaar.API.Models;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ProductsController : ControllerBase
    {
        private readonly IProductService _productService;

        public ProductsController(IProductService productService)
        {
            _productService = productService;
        }

        [HttpGet("{id}")]
        public ActionResult<Product> GetProductById(int id) =>
            Ok(_productService.GetProductById(id));

        [HttpGet()]
        public ActionResult<IEnumerable<ProductModel>> GetProducts(int
pageNumber, int offset) =>
            Ok(_productService.GetProducts(pageNumber, offset));

        [HttpGet("by-user/{userId}")]
        public ActionResult<IEnumerable<ProductModel>> GetProductsByUser(int
userId, int pageNumber, int offset) =>
            Ok(_productService.GetProductsByUserId(userId, pageNumber, offset));

        [HttpPost]
        public ActionResult<Product> SaveProduct(ProductAddModel product) =>
            Ok(_productService.SaveProduct(product));

        [HttpPut("{id}")]
        public ActionResult<Product> UpdateProduct(int id, ProductAddModel
product) =>
            Ok(_productService.UpdateProduct(id, product));

        [HttpDelete("{id}")]
        public ActionResult DeleteProduct(int id)
        {
            _productService.DeleteProduct(id);
            return Ok();
```

```
        }

        [HttpPost("checkout")]
        public IActionResult GetProductsByIds(IEnumerable<int> ids) =>
            Ok(_productService.GetProductsByIds(ids));
    }
}
```

ProductsController.cs

```
using Microsoft.AspNetCore.Mvc;
using TheDesiAchaar.API.Contracts;
using TheDesiAchaar.API.Models;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ProductsController : ControllerBase
    {
        private readonly IProductService _productService;

        public ProductsController(IProductService productService)
        {
            _productService = productService;
        }

        [HttpGet("{id}")]
        public ActionResult<Product> GetProductById(int id) =>
            Ok(_productService.GetProductById(id));

        [HttpGet()]
        public ActionResult<IEnumerable<ProductModel>> GetProducts(int
pageNumber, int offset) =>
            Ok(_productService.GetProducts(pageNumber, offset));

        [HttpGet("by-user/{userId}")]
        public ActionResult<IEnumerable<ProductModel>> GetProductsByUser(int
userId, int pageNumber, int offset) =>
            Ok(_productService.GetProductsByUserId(userId, pageNumber, offset));

        [HttpPost]
        public ActionResult<Product> SaveProduct(ProductAddModel product) =>
            Ok(_productService.SaveProduct(product));

        [HttpPut("{id}")]
        public ActionResult<Product> UpdateProduct(int id, ProductAddModel
product) =>
            Ok(_productService.UpdateProduct(id, product));

        [HttpDelete("{id}")]
        public ActionResult DeleteProduct(int id)
        {
            _productService.DeleteProduct(id);
            return Ok();
        }

        [HttpPost("checkout")]
        public IActionResult GetProductsByIds(IEnumerable<int> ids) =>
```

```
            Ok(_productService.GetProductsByIds(ids));
    }
}
```

Product.cs

```
namespace TheDesiAchaar.Models
{
    public class Product: DbEntity<int>
    {
        public string Name { get; set; }
        public int Quantity { get; set; }
        public decimal Price { get; set; }
        public byte[] Image { get; set; }
        public int UserId { get; set; }
        public DateTime DateOfMfg { get; set; }
        public DateTime DateOfExp { get; set; }

    }
}
```

ProductModel.cs

```
namespace TheDesiAchaar.API.Models
{
    public class ProductModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Quantity { get; set; }
        public decimal Price { get; set; }
        public string Image { get; set; }
        public DateTime DateOfMfg { get; set; }
        public DateTime DateOfExp { get; set; }

    }
}
```

ProductAddModel.cs

```
namespace TheDesiAchaar.API.Models
{
    public class ProductAddModel
    {
        public string Name { get; set; }
        public int Quantity { get; set; }
        public decimal Price { get; set; }
        public string Image { get; set; }
```

```
        public int UserId { get; set; }
        public DateTime DateOfMfg { get; set; }
        public DateTime DateOfExp { get; set; }
    }
}
```

OrdersController.cs

```
using Microsoft.AspNetCore.Mvc;
using TheDesiAchaar.API.Contracts;
using TheDesiAchaar.API.Models;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.API.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrdersController : ControllerBase
    {
        private readonly IOrderService _orderService;

        public OrdersController(IOrderService orderService)
        {
            _orderService = orderService;
        }

        [HttpGet("{userId}")]
        public ActionResult<IEnumerable<OrderDetails>> GetOrders(int userId, int
offset) =>
            Ok(_orderService.GetOrders(userId, offset));

        [HttpPost]
        public ActionResult<IEnumerable<Orders>> SaveOrders(IEnumerable<Orders>
orders) =>
            Ok(_orderService.SaveOrders(orders));
        [HttpGet]
        public ActionResult<IEnumerable<OrderDetails>> GetSellerOrders(int
userId, int offset) =>
            Ok(_orderService.GetOrdersForSeller(userId, offset));

        [HttpPut]
        public async Task<ActionResult<bool>> UpdateOrderStatusAsync(int orderId,
string orderStatus) =>
            Ok(await _orderService.UpdateOrderStatusAsync(orderId, orderStatus));
    }
}
```

OrderService.cs

```
using Microsoft.EntityFrameworkCore;
using TheDesiAchaar.API.Contracts;
using TheDesiAchaar.API.Models;
using TheDesiAchaar.Data;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.API.Services
{
```

```
    public class OrderService : IOrderService
    {
        private readonly DataContext _dataContext;
        private readonly IBlockChainService _blockChainService;
        private static readonly int _pageSize = 5;
        public OrderService(DataContext dataContext, IBlockChainService
blockChainService)
        {
            _dataContext = dataContext;
            _blockChainService = blockChainService;
        }

        public IEnumerable<OrderDetails> GetOrders(int userId, int offset)
        {
            return _dataContext.Orders.Where(_ => _.UserId ==
userId).OrderByDescending(_ => _.CreatedOn)
                .Join(_dataContext.Product, o => o.ProductId, p => p.Id, (o, p)
=> new { o, p }).Select(_ => new OrderDetails
                {
                    OrderDate = _.o.CreatedOn,
                    OrderStatus = _.o.Status,
                    Price = _.p.Price,
                    Quantity = _.o.Quantity,
                    ProductName = _.p.Name,
                    Id = _.o.Id,
                    Image = Convert.ToBase64String(_.p.Image)
                }).Skip(_pageSize * offset).Take(_pageSize).ToList();
        }

        public IEnumerable<Orders> SaveOrders(IEnumerable<Orders> orders)
        {
            var productIds = orders.Select(o => o.ProductId);
            var products = _dataContext.Product.Where(_ =>
productIds.Contains(_.Id));
            foreach (var order in orders)
            {
                order.CreatedOn = DateTime.UtcNow;
                order.LastUpdatedOn = DateTime.UtcNow;
                order.CreatedBy = "Test";
                order.LastUpdatedBy = "Test";
                var currentQty = products.First(_ => _.Id ==
order.ProductId).Quantity - order.Quantity;
                products.First(_ => _.Id == order.ProductId).Quantity =
currentQty;
            }
            _dataContext.AttachRange(products);
            _dataContext.Orders.AddRange(orders);
            _dataContext.SaveChanges();
            return orders;
        }

        public IEnumerable<OrderDetails> GetOrdersForSeller(int userId, int
offset)
        {
            return _dataContext.Orders.OrderByDescending(_ => _.CreatedOn)
                .Join(_dataContext.Product.Where(_ => _.UserId == userId), o =>
o.ProductId, p => p.Id, (o, p) => new { o, p })
                .Join(_dataContext.Users, ou => ou.o.UserId, u => u.Id, (ou, u)
=> new {ou.o, ou.p, u}).Select(_ => new OrderDetails
                {
                    OrderDate = _.o.CreatedOn,
                    OrderStatus = _.o.Status,
                    Price = _.p.Price,
```

```
                    Quantity = _.o.Quantity,
                    ProductName = _.p.Name,
                    Id = _.o.Id,
                    Image = Convert.ToBase64String(_.p.Image),
                    ClientAddress = _.u.Address,
                    Email = _.u.EmailAddress,
                    PhoneNumber = _.u.PhoneNumber

                }).Skip(_pageSize * offset).Take(_pageSize).ToList();
        }

        public async Task<bool> UpdateOrderStatusAsync(int orderId, string
orderStatus)
        {
            var order = await _dataContext.Orders.FirstAsync(_ => _.Id ==
orderId);
            order.Status = orderStatus;
            order.LastUpdatedOn = DateTime.UtcNow;
            _dataContext.Attach(order);
            await _dataContext.SaveChangesAsync();
            var transactionReceipt = await
_blockChainService.CreateOrderTransactions(
                new OrderBlockChainInfo {
                    OrderId = order.Id,
                    OrderStatus = orderStatus,
                    OrderUpdateDate = order.LastUpdatedOn });
            _dataContext.Blockchain.Add(transactionReceipt);
            await _dataContext.SaveChangesAsync();
            return true;
        }
    }
}
```

IProductService.cs

```
using TheDesiAchaar.API.Models;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.API.Contracts
{
    public interface IProductService
    {
        IEnumerable<ProductModel> GetProducts(int pageNumber, int offset);
        Product GetProductById(int id);
        IEnumerable<ProductModel> GetProductsByUserId(int userId, int pageNumber,
int offset);
        IEnumerable<ProductModel> GetProductsByIds(IEnumerable<int> ids);
        Product SaveProduct(ProductAddModel product);
        Product UpdateProduct(int id, ProductAddModel product);
        void DeleteProduct(int id);
    }
}
```

IUserService.cs

```
using TheDesiAchaar.Entities;
```

```
using TheDesiAchaar.Models.Users;

namespace TheDesiAchaar.API.Contracts
{
    public interface IUserService
    {
        AuthenticateResponse Authenticate(AuthenticateRequest model);
        IEnumerable<User> GetAll();
        User GetById(int id);
        void Register(RegisterRequest model);
        void Update(int id, UpdateRequest model);
        void Delete(int id);
    }
}
```

DataContext.cs

```
using Microsoft.EntityFrameworkCore;
using TheDesiAchaar.Entities;
using TheDesiAchaar.Models;

namespace TheDesiAchaar.Data;

public class DataContext: DbContext
{
    public DataContext(DbContextOptions options) : base(options) { }

    public DbSet<User> Users { get; set; }
    public DbSet<Blockchain> Blockchain { get; set; }
    public DbSet<Orders> Orders { get; set; }
    public DbSet<Product> Product { get; set; }
    public DbSet<ProductSupplier> ProductSupplier { get; set; }
    public DbSet<Supplier> Supplier { get; set; }


}
```

AuthenticateResponse.cs

```
namespace TheDesiAchaar.Models.Users;

public class AuthenticateResponse
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Username { get; set; }
    public string Address { get; set; }
    public string PhoneNumber { get; set; }
    public string Token { get; set; }
}
```

AutoMapperProfile.cs

```
namespace TheDesiAchaar.Helpers;

using AutoMapper;
using TheDesiAchaar.Entities;
using TheDesiAchaar.Models.Users;

public class AutoMapperProfile : Profile
{
    public AutoMapperProfile()
    {
        // User -> AuthenticateResponse
        CreateMap<User, AuthenticateResponse>();

        // RegisterRequest -> User
        CreateMap<RegisterRequest, User>();

        // UpdateRequest -> User
        CreateMap<UpdateRequest, User>()
            .ForAllMembers(x => x.Condition(
                (src, dest, prop) =>
                {
                    // ignore null & empty string properties
                    if (prop == null) return false;
                    if (prop.GetType() == typeof(string) &&
string.IsNullOrEmpty((string)prop)) return false;

                    return true;
                }
            ));
    }
}
```