ShoppingContext.cs

```
using Microsoft.EntityFrameworkCore;
using ShoppingApp.Models;
using System.Collections.Generic;
using System.Reflection.Emit;

namespace ShoppingApp.Contexts
{
    public class ShoppingContext : DbContext
    {
        public ShoppingContext(DbContextOptions options) : base(options)
        {

        }
        public DbSet<User> Users { get; set; }
        public DbSet<Product> Products { get; set; }
        public DbSet<Cart> Carts { get; set; }
        public DbSet<CartItems> CartItems { get; set; }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            //Fluent API
            modelBuilder.Entity<Cart>(cart =>
            {
                cart.HasKey(ck => ck.cartNumber);
            });
            modelBuilder.Entity<CartItems>(ci =>
            {
                ci.HasKey(cik => new { cik.CartNumber, cik.Product_Id });
            });
            modelBuilder.Entity<CartItems>()
                .HasOne<Product>(ci => ci.Product)
                .WithMany(p => p.CartItems);
        }
    }
}
```

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;
using ShoppingApp.Models;
using System.Diagnostics;

namespace ShoppingApp.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

UserController.cs

```csharp
using Microsoft.AspNetCore.Mvc;
using ShoppingApp.Interfaces;
using ShoppingApp.Models.DTOs;

namespace ShoppingApp.Controllers
{
    public class UserController : Controller
    {
        private readonly IUserService _userService;


        public UserController(IUserService userService)
        {
            _userService = userService;
        }
        public IActionResult Login()
        {
            return View();
        }
        [HttpPost]
        public IActionResult Login(UserViewModel viewModel)
        {
            var user = _userService.Login(viewModel);
            if (user != null)
            {
                return RedirectToAction("Index", "Home");
            }
            return View();
        }
        public IActionResult Register()
        {
            return View();
        }
        [HttpPost]
        public IActionResult Register(UserViewModel viewModel)
        {
            var user = _userService.Register(viewModel);
            if (user != null)
            {
                return RedirectToAction("Index", "Home");
            }
            return View();
        }
    }
}
```

IRepository.cs

```csharp
namespace ShoppingApp.Interfaces
{
    public interface IRepository<K,T>
    {
        T GetById(K key);
        IList<T> GetAll();
        T Add(T entity);
        T Update(T entity);
        T Delete(K key);
    }
}
```

IUserService.cs

```
using ShoppingApp.Models.DTOs;

namespace ShoppingApp.Interfaces
{
    public interface IUserService
    {
        UserDTO Login(UserDTO userDTO);
        UserDTO Register(UserDTO userDTO);
    }
}
```

UserDTO.cs

```
using System.ComponentModel.DataAnnotations;

namespace ShoppingApp.Models.DTOs
{
    public class UserDTO
    {
        [Required(ErrorMessage = "Username cannot be empty")]
        public string Username { get; set; }

        public string Role { get; set; }
        public string Token { get; set; }
        [Required(ErrorMessage = "Password cannot be empty")]
        public string Password { get; set; }
    }
}
```

UserViewModel.cs

```
using System.ComponentModel.DataAnnotations;
namespace ShoppingApp.Models.DTOs
{
    public class UserViewModel : UserDTO
    {
        [Required(ErrorMessage = "Re type password cannot be empty")]
        [Compare("Password", ErrorMessage = "Password and retype password do not match")]
        public string ReTypePassword { get; set; }
    }
}
```

Cart.cs

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

namespace ShoppingApp.Models
{
    public class Cart
    {

        public int cartNumber { get; set; }
        public string Username { get; set; }

        [ForeignKey("Username")]
        public User User { get; set; }
    }
}
```

CartItems.cs

```
namespace ShoppingApp.Models
{
    public class CartItems
    {
        public int CartNumber { get; set; }
        public int Product_Id { get; set; }
        public float Price { get; set; }
        public int Quantity { get; set; }
        public Product Product { get; set; }
    }
}
```

Product.cs

```
namespace ShoppingApp.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public float Price { get; set; }
        public int Quantity { get; set; }
        public string Picture { get; set; }
        public ICollection<CartItems> CartItems { get; set; }
    }
}
```

User.cs

```
using System.ComponentModel.DataAnnotations;

namespace ShoppingApp.Models
{
    public class User
    {
        [Key]
        public string Username { get; set; }
        public byte[] Password { get; set; }
        public string Role { get; set; }
        public byte[] Key { get; set; }

    }
}
```

CartItemsRepository.cs

```csharp
using Microsoft.EntityFrameworkCore;
using ShoppingApp.Contexts;
using ShoppingApp.Interfaces;
using ShoppingApp.Models;

namespace ShoppingApp.Repositories
{
    public class CartItemsRepository : IRepository<int, CartItems>
    {
        private readonly ShoppingContext _context;
        public CartItemsRepository(ShoppingContext context)
        {
            _context = context;
        }

        public CartItems Add(CartItems entity)
        {
            _context.CartItems.Add(entity);
            _context.SaveChanges();
            return entity;
        }

        public CartItems Delete(int key)
        {
            var item = _context.CartItems.FirstOrDefault(ci => ci.Product_Id == key);
            if (item != null)
            {
                _context.CartItems.Remove(item);
                _context.SaveChanges();
                return item;
            }
            return null;
        }

        public IList<CartItems> GetAll()
        {
            if (_context.CartItems.Count() == 0)
                return null;
            return _context.CartItems.ToList();
        }

        public CartItems GetById(int key)
        {
            var item = _context.CartItems.FirstOrDefault(ci => ci.Product_Id == key);
            return item;
        }

        public CartItems Update(CartItems entity)
        {
            var cart = GetById(entity.Product_Id);
            if (cart != null)
            {
                _context.Entry<CartItems>(cart).State = EntityState.Modified;
                _context.SaveChanges();
                return cart;
            }
            return null;
        }
    }
}
```

CartRepository.cs

```
using Microsoft.EntityFrameworkCore;
using ShoppingApp.Contexts;
using ShoppingApp.Interfaces;
using ShoppingApp.Models;

namespace ShoppingApp.Repositories
{
    public class CartRepository : IRepository<int, Cart>
    {
        private readonly ShoppingContext _context;
        public CartRepository(ShoppingContext context)
        {
            _context = context;
        }
        public Cart Add(Cart entity)
        {
            _context.Carts.Add(entity);
            _context.SaveChanges();
            return entity;
        }

        public Cart Delete(int key)
        {
            var cart = GetById(key);
            if (cart != null)
            {
                _context.Carts.Remove(cart);
                _context.SaveChanges();
                return cart;
            }
            return null;
        }

        public IList<Cart> GetAll()
        {

            if (_context.Carts.Count() == 0)
                return null;
            return _context.Carts.ToList();
        }

        public Cart GetById(int key)
        {
            var cart = _context.Carts.SingleOrDefault(u => u.cartNumber == key);
            return cart;
        }

        public Cart Update(Cart entity)
        {
            var cart = GetById(entity.cartNumber);
            if (cart != null)
            {
                _context.Entry<Cart>(cart).State = EntityState.Modified;
                _context.SaveChanges();
                return cart;
            }
            return null;
        }
    }
}
```

ProductRepository.cs

```
using Microsoft.EntityFrameworkCore;
using ShoppingApp.Contexts;
using ShoppingApp.Interfaces;
using ShoppingApp.Models;

namespace ShoppingApp.Repositories
{
    public class ProductRepository : IRepository<int, Product>
    {
        private readonly ShoppingContext _context;
        public ProductRepository(ShoppingContext context)
        {
            _context = context;
        }
        public Product Add(Product entity)
        {
            _context.Products.Add(entity);
            _context.SaveChanges();
            return entity;
        }

        public Product Delete(int key)
        {
            var product = GetById(key);
            if (product != null)
            {
                _context.Products.Remove(product);
                _context.SaveChanges();
                return product;
            }
            return null;
        }

        public IList<Product> GetAll()
        {
            if (_context.Products.Count() == 0)
                return null;
            return _context.Products.ToList();
        }

        public Product GetById(int key)
        {
            var product = _context.Products.SingleOrDefault(u => u.Id == key);
            return product;
        }

        public Product Update(Product entity)
        {
            var product = GetById(entity.Id);
            if (product != null)
            {
                _context.Entry<Product>(product).State = EntityState.Modified;
                _context.SaveChanges();
                return product;
            }
            return null;
        }
    }
}
```

UserRepository.cs

```csharp
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata.Internal;
using ShoppingApp.Contexts;
using ShoppingApp.Interfaces;
using ShoppingApp.Models;

namespace ShoppingApp.Repositories
{
    public class UserRepository : IRepository<string, User>
    {
        private readonly ShoppingContext _context;

        public UserRepository(ShoppingContext context)
        {
            _context = context;
        }
        public User Add(User entity)
        {
            _context.Users.Add(entity);
            _context.SaveChanges();
            return entity;
        }

        public User Delete(string key)
        {
            var user = GetById(key);
            if (user != null)
            {
                _context.Users.Remove(user);
                _context.SaveChanges();
                return user;
            }
            return null;
        }

        public IList<User> GetAll()
        {
            if (_context.Users.Count() == 0)
                return null;
            return _context.Users.ToList();
        }

        public User GetById(string key)
        {
            var user = _context.Users.SingleOrDefault(u => u.Username == key);
            return user;
        }

        public User Update(User entity)
        {
            var user = GetById(entity.Username);
            if (user != null)
            {
                _context.Entry<User>(user).State = EntityState.Modified;
                _context.SaveChanges();
                return user;
            }
            return null;
        }
    }
}
```

UserService.cs

```
using ShoppingApp.Interfaces;
using ShoppingApp.Models;
using ShoppingApp.Models.DTOs;
using ShoppingApp.Repositories;
using System.Security.Cryptography;
using System.Text;

namespace ShoppingApp.Services
{
    public class UserService : IUserService
    {
        private readonly IRepository<string, User> _repository;

        public UserService(IRepository<string, User> repository)
        {
            _repository = repository;
        }
        public UserDTO Login(UserDTO userDTO)
        {
            var user = _repository.GetById(userDTO.Username);
            if (user != null)
            {
                HMACSHA512 hmac = new HMACSHA512(user.Key);
                var userpass = hmac.ComputeHash(Encoding.UTF8.GetBytes(userDTO.Password));
                for (int i = 0; i < userpass.Length; i++)
                {
                    if (user.Password[i] != userpass[i])
                        return null;
                }
                userDTO.Password = "";
                return userDTO;
            }
            return null;
        }

        public UserDTO Register(UserDTO userDTO)
        {
            HMACSHA512 hmac = new HMACSHA512();
            User user = new User()
            {
                Username = userDTO.Username,
                Password = hmac.ComputeHash(Encoding.UTF8.GetBytes(userDTO.Password)),
                Key = hmac.Key,
                Role = userDTO.Role
            };
            var result = _repository.Add(user);
            if (result != null)
            {
                userDTO.Password = "";
                return userDTO;
            }
            return null;

        }


    }
}
```

Login.cshtml

```
@model ShoppingApp.Models.DTOs.UserDTO

@{
    ViewData["Title"] = "Login";
}

<h1>Login</h1>

<h4>UserDTO</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Login">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Username" class="control-label"></label>
                <input asp-for="Username" class="form-control" />
                <span asp-validation-for="Username" class="text-danger"></span>
            </div>

            <div class="form-group">
                <label asp-for="Password" class="control-label"></label>
                <input asp-for="Password" type="password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Register.cshtml

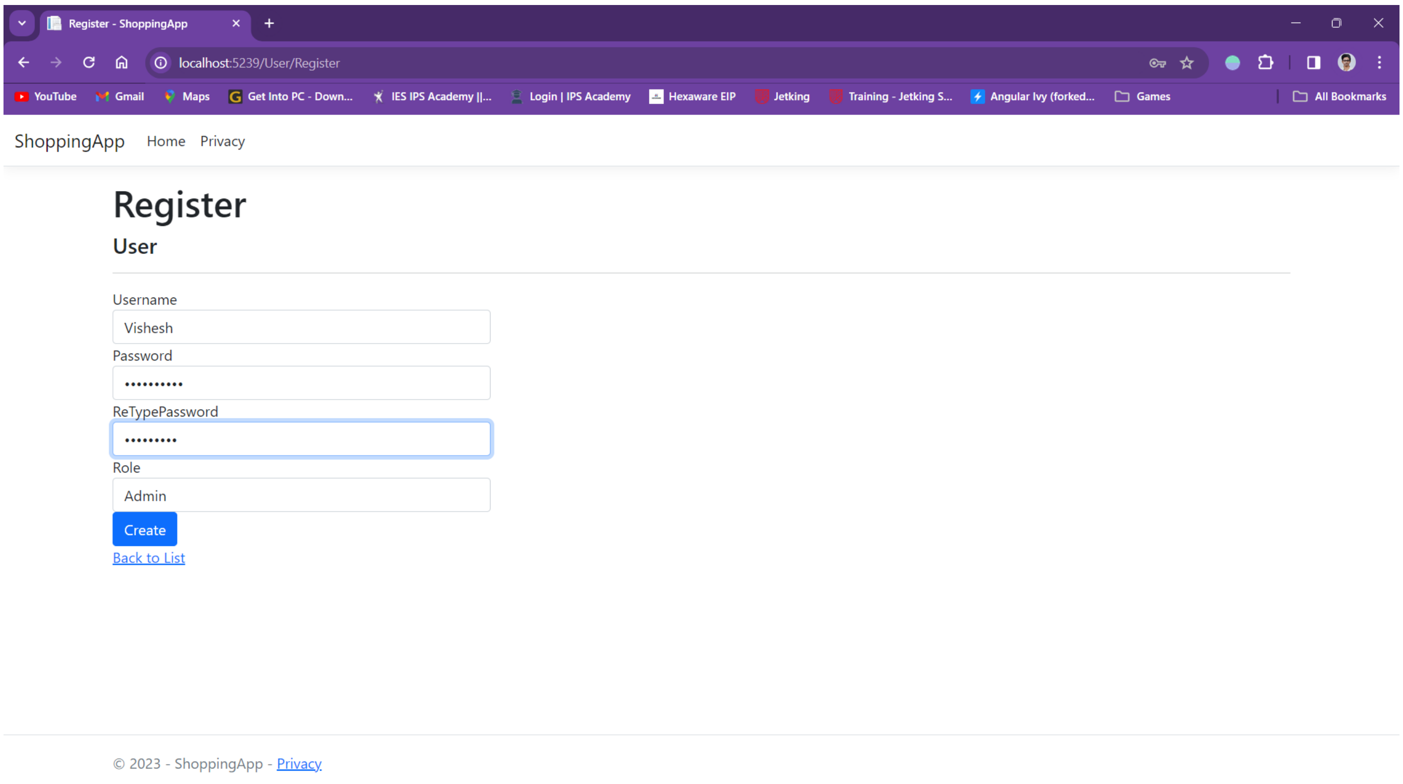
```
@model ShoppingApp.Models.DTOs.UserDTO

@{
    ViewData["Title"] = "Login";
}

<h1>Login</h1>

<h4>UserDTO</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Login">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Username" class="control-label"></label>
                <input asp-for="Username" class="form-control" />
                <span asp-validation-for="Username" class="text-danger"></span>
            </div>

            <div class="form-group">
                <label asp-for="Password" class="control-label"></label>
                <input asp-for="Password" type="password" class="form-control" />
                <span asp-validation-for="Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "ConnectionStrings": {
    "shoppingCon": "Data source=NAZ\\DEMOINSTANCE; user id=sa;password=Trainingpass;Initial
catalog=dbShopping06Nov2023"
  },
  "AllowedHosts": "*"
}
```

Program.cs

```
using Microsoft.EntityFrameworkCore;
using ShoppingApp.Contexts;
using ShoppingApp.Interfaces;
using ShoppingApp.Models;
using ShoppingApp.Repositories;
using ShoppingApp.Services;

namespace ShoppingApp
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            // Add services to the container.
            builder.Services.AddControllersWithViews();
            builder.Services.AddDbContext<ShoppingContext>(opts =>
            {
                opts.UseSqlServer(builder.Configuration.GetConnectionString("shoppingCon"));
            });

            builder.Services.AddScoped<IRepository<string,User>, UserRepository>();
            builder.Services.AddScoped<IRepository<int, Product>, ProductRepository>();
            builder.Services.AddScoped<IRepository<int, Cart>, CartRepository>();
            builder.Services.AddScoped<IRepository<int, CartItems>, CartItemsRepository>();
            builder.Services.AddScoped<IUserService, UserService>();

            var app = builder.Build();

            // Configure the HTTP request pipeline.
            if (!app.Environment.IsDevelopment())
            {
                app.UseExceptionHandler("/Home/Error");
            }
            app.UseStaticFiles();

            app.UseRouting();

            app.UseAuthorization();

            app.MapControllerRoute(
                name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");

            app.Run();
        }
    }
}
```

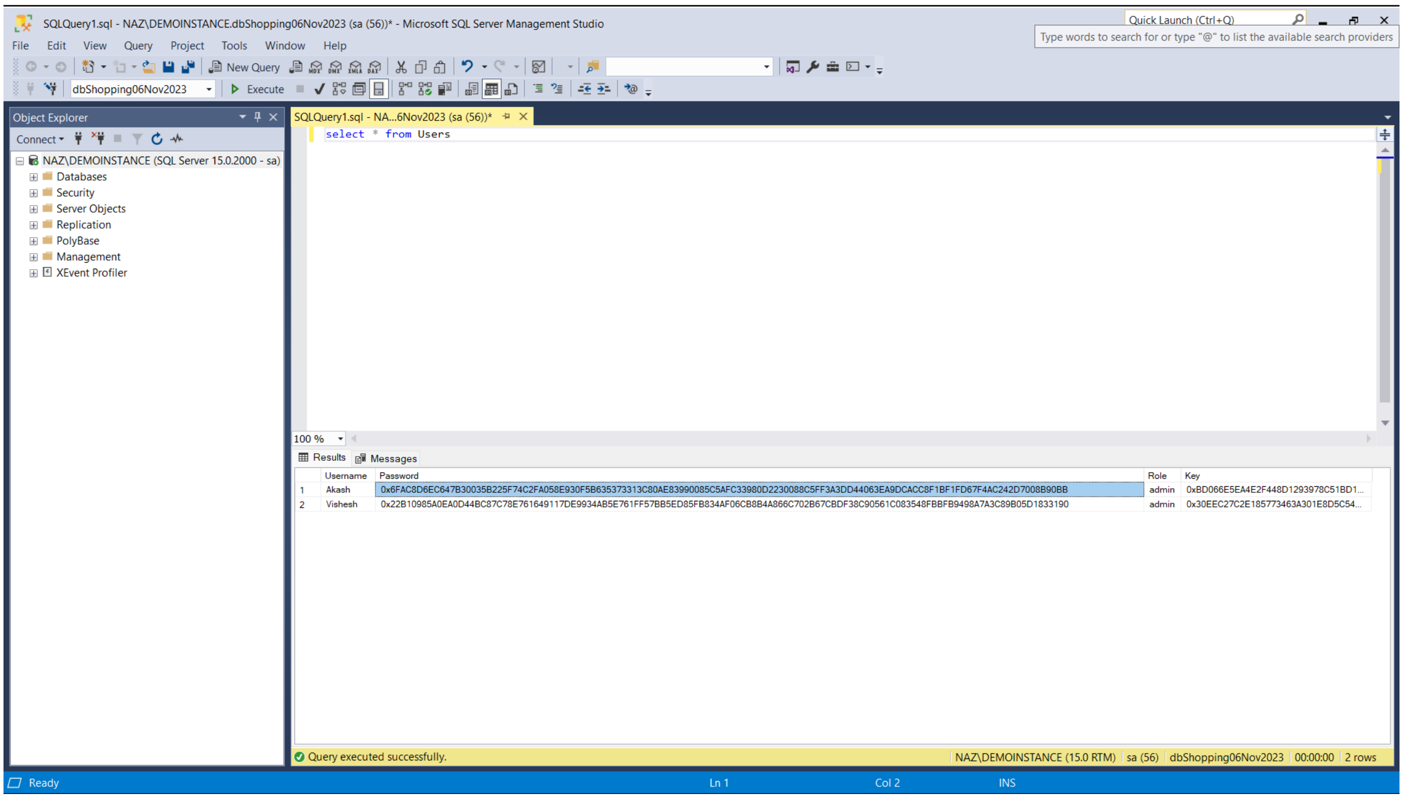


Register

User

Username	Vishesh
Password	••••••••••
ReTypePassword	•••••••••
Role	Admin

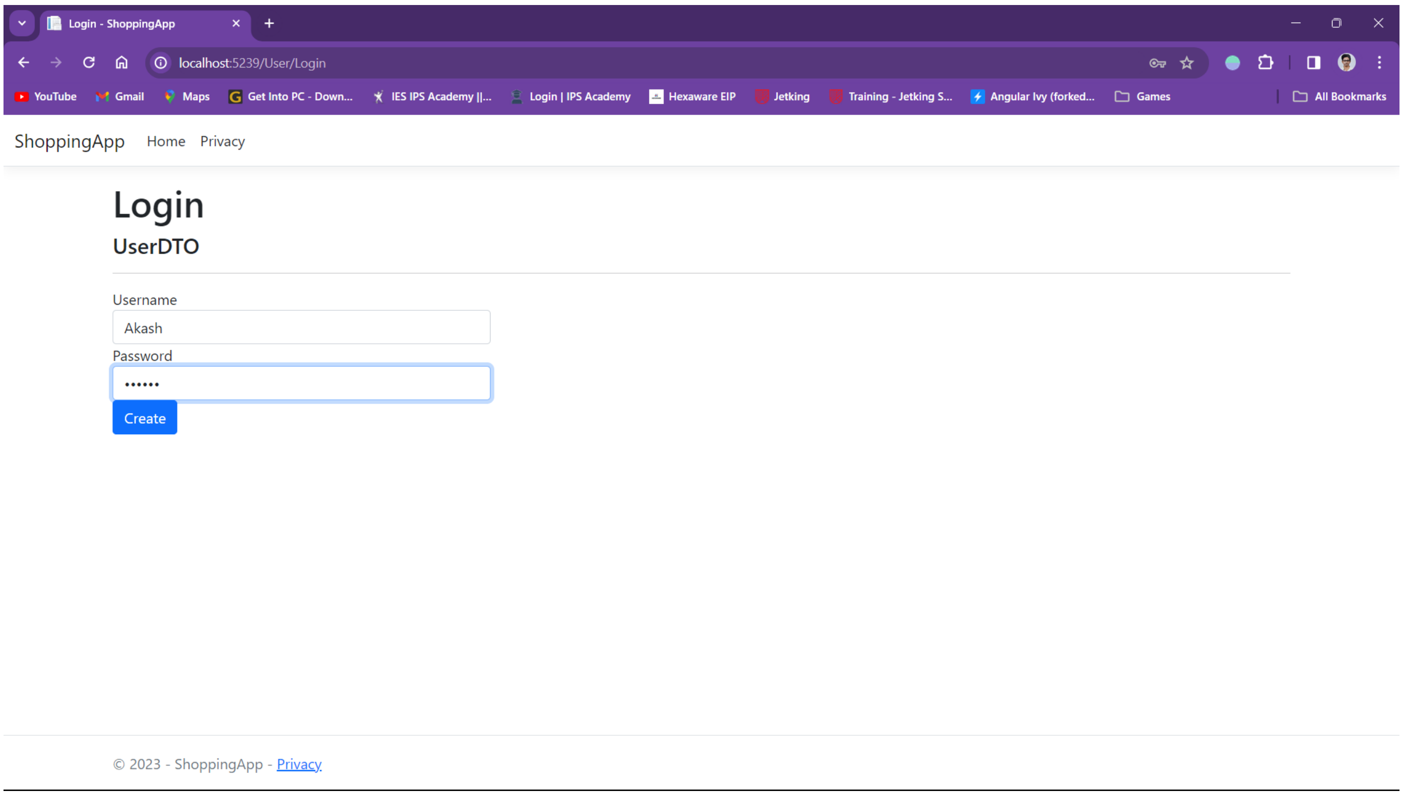
© 2023 - ShoppingApp - Privacy


SQLQuery1.sql - NAZ\DEMOINSTANCE.dbShopping06Nov2023 (sa (56))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
Type words to search for or type "@" to list the available search providers
File
Edit
View
Query
Project
Tools
Window
Help
New Query
dbShopping06Nov2023
Execute
Object Explorer
Connect
NAZ\DEMOINSTANCE (SQL Server 15.0.2000 - sa)
Databases
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
SQLQuery1.sql - NA...6Nov2023 (sa (56))*
100 %
Results
Messages

```
select * from Users
```

	Username	Password	Role	Key
1	Akash	0x6FAC8D6EC647B30035B225F74C2FA058E930F5B635373313C80AE83990085C5AFC33980D2230088C5FF3A3DD44063EA9DCACC8F1BF1FD67F4AC242D7008B90BB	admin	0xBD066E5EA4E2F448D1293978C51BD1...
2	Vishesh	0x22B10985A0EA0D44BC87C78E761649117DE9934AB5E761FF57BB5ED85FB834AF06CB8B4A866C702B67CBDF38C90561C083548FBBFB9498A7A3C89B05D1833190	admin	0x30EEC27C2E185773463A301E8D5C54...

Query executed successfully. NAZ\DEMOINSTANCE (15.0 RTM) | sa (56) | dbShopping06Nov2023 | 00:00:00 | 2 rows


Login - ShoppingApp

localhost:5239/User/Login

ShoppingApp Home Privacy

Login

UserDTO

Username

Akash

Password

••••••

Create

© 2023 - ShoppingApp - Privacy