بخش اول

توضیحات پروژه:

پروژه نهایی لیسانس رشته نرم افزار با عنوان انتشارات آنلاین دانشگاه

این انتشارات دو نوع خدمات ارائه میدهد، مورد اول انجام خدمات چاپ به صورت آنلاین است.

دانشجو میتواند چهار نوع درخواست چاپ برای ادمین ارسال کند:

1. در خواست چاپ سیاه سفید :

در مرحله اول کاربر اطلاعات لازم مثل جنس کاغذ و پشت و رو و بارگذاری فایل را در فرم وارد می کند. بعد از تایید کردن این مرحله ، در مرحله دوم با توجه به تعداد صفحاتی که کاربر خواسته بود و تعداد صفحات فایل ارسالی و جنس و کاغذ و موارد دیگر قیمت محاسبه میشود (صحافی قیمت ندارد و قیمت آن در مراجعه حضوری مشخص میشود) و به کاربر نمایش داده شده و دکمه افزودن به سبد خرید نمایش داده می شود.

در مرحله بعد وارد سبد خرید می شویم. درخواست های چاپ کاربر + محصولات لوازم تحریر اضافه شده به سبد خرید باید نمایش یابد و قیمت نیز محاسبه شود.

دو دکمه داریم: تایید نهایی و ادامه خرید

در صورتی که تایید نهایی زده شد سفارش ثبت می شود و از طریق سفارش های من قابل پیگیری است. در صفحه سفارش های من محصولات سفارش داده شده و خدمات نمایش پیدا میکند. با این تفاوت که در خدمات چاپ کاربر باید منتظر تایید ادمین و به اتمام رسیدن کار باشد و در سفارشات من وضعیت خدمات چاپ مشخص باشد.

کار که اتمام رسید دکمه ای ظاهر میشود و با کلیک روی آن صفحه SUCCES

نمایش می یابد که شامل کد و کیو ار کدی است که هنگام مراجعه حضوری باید به اوپراتور انتشارات نمایش داده شود. (پرداخت به صورت حضوری)

2. درخواست چاپ رنگی: به همان صوررت
3. درخواست چاپ پلان : قیمتی محاسبه نمیشود
4. لمینیت کاغذ : به همان صورت

یکی دیگر از ویژگی های سایت داشتن فروشگاه آنلاین لوازم التحریر است و مراحل سفارش کالا نیز به روش خدمات چاپ انجام میشود.

مدل ها و ارتباطات پایگاه داده:

کلاس PrintRequest

```
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models
{
    public class PrintRequest
    {
        [Key]
        public int Id { get; set; }

        public string UserId { get; set; } // اتصال به Identity
        public User User { get; set; }

        public string ServiceType { get; set; } // Color, BlackWhite, Plan,laminate

        public string Status { get; set; } = "Submitted"; // Submitted, Processing,
Completed, Rejected

        public DateTime CreatedAt { get; set; } = DateTime.Now;
        public decimal? TotalPrice { get; set; }
        public ColorPrintDetail ColorPrintDetail { get; set; }
        public BlackWhitePrintDetail BlackWhitePrintDetail { get; set; }
        public PlanPrintDetail PlanPrintDetail { get; set; }
        public LaminateDetail LaminateDetail { get; set; }

        public ICollection<PrintFile> Files { get; set; }
        public ICollection<PickupPrintItem> PickupPrintItems { get; set; } = new
List<PickupPrintItem>();
    }
}
```

## کلاس BlackWhitePrintDetail

```
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models
{
    public class BlackWhitePrintDetail
    {
        [Key]
        public int PrintRequestId { get; set; }
        public PrintRequest PrintRequest { get; set; }

        public string PaperType { get; set; }
        public string PaperSize { get; set; }
        public string PrintSide { get; set; }
        public int CopyCount { get; set; }
        public int TotalPages { get; set; }
        public string Description { get; set; }
        public string BindingType { get; set; } = “ هیچکدام ، طلق و شیرازه ، هیچکدام"; //منگنه
،
        public string? FilesJson { get; set; }  // رشته‌ای که لیست فایل‌ها را به صورت JSON
نگه می‌دارد
        public decimal TotalPrice { get; set; }
    }
}
```

## کلاس ColorPrintDetail

```
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models
{
    public class ColorPrintDetail
    {
        [Key]
        public int PrintRequestId { get; set; }
        public PrintRequest PrintRequest { get; set; }

        public string PaperType { get; set; } // Glossy, Normal
        public string PaperSize { get; set; } // A4, A5, A3
        public int CopyCount { get; set; }
        public string PrintSide { get; set; }
        public decimal TotalPrice { get; set; }
        public int TotalPages { get; set; }
        public string? AdditionalDescription { get; set; } // توضیحات اضافه مثل موقعیت نقشه،
حاشیه، یا تنظیمات خاص
        public string BindingType { get; set; } = " هیچکدام ، طلق و شیرازه ، هیچکدام"; //منگنه
،
        public string? FilesJson { get; set; }  // رشته‌ای که لیست فایل‌ها را به صورت JSON
نگه می‌دارد
    }
}
```

## کلاس PlanPrintDetail

```
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models
{
    public class PlanPrintDetail
    {
        [Key]
        public int PrintRequestId { get; set; }
        public PrintRequest PrintRequest { get; set; }
        public string SizeOrScaleDescription { get; set; }
        public string? AdditionalDescription { get; set; } // توضیحات اضافه مثل موقعیت نقشه، حاشیه، یا تنظیمات خاص
        public string PaperType { get; set; } // Glossy, Normal
        public string printType { get; set; } //  BW , color

        public int CopyCount { get; set; }

        public string BindingType { get; set; } = " هیچ‌کدام"; //منگنه ، طلق و شیرازه ، هیچ‌کدام ، سیمی
        public string? FilesJson { get; set; }  // رشته‌ای که لیست فایل‌ها را به صورت JSON نگه می‌دارد
    }
}
```

## و کلاس LaminateDetail

```
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models
{
    public class LaminateDetail
    {
        [Key]
        public int PrintRequestId { get; set; }
        public PrintRequest PrintRequest { get; set; }

        public string PaperType { get; set; }
        public string PaperSize { get; set; }
        public string PrintSide { get; set; }
        public string printType { get; set; } //  BW , color
        public int CopyCount { get; set; }
        public int TotalPages { get; set; }
        public string Description { get; set; }
        public string LaminateType { get; set; } = "مات"; //مات ، براق
        public string CornerType { get; set; } //گوشه گرد، گوشه تیز
        public string? FilesJson { get; set; }  // رشته‌ای که لیست فایل‌ها را به صورت JSON نگه می‌دارد
        public decimal TotalPrice { get; set; }
    }
}
```

## کلاس PrintFile

```
namespace MainChapar.Models
{
    public class PrintFile
    {
        public int Id { get; set; }

        public int PrintRequestId { get; set; }
        public PrintRequest PrintRequest { get; set; }

        public string FileName { get; set; }
        public string FilePath { get; set; }
        public int PageCount { get; set; }
    }
}
```

## کلاس PrintPricing

```
namespace MainChapar.Models
{
    public class PrintPricing
    {

        public int Id { get; set; }
        public string PaperSize { get; set; } // A4, A5, A3
        public string PaperType { get; set; } // ساده، گلاسه سبک، گلاسه سنگین
        public bool IsDoubleSided { get; set; }
        public decimal PricePerPage { get; set; }
        public bool IsAvailable { get; set; }
        public string PrintType { get; set; } // "BlackWhite" یا "Color"

    }
}
```

## کلاس Product

```
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models
{
    public class Product
    {
        public int Id { get; set; }

        [Required(ErrorMessage = "لطفاً عنوان را وارد کنید")]
        public string Title { get; set; }

        public string? Description { get; set; }

        public string? FullDesc { get; set; }
```

```
        [Required(ErrorMessage = "لطفاً قیمت را وارد کنید")]
        public decimal Price { get; set; }

        public decimal? Discount { get; set; }

        public string? ImageName { get; set; }

        [Required(ErrorMessage = "موجودی انبار را وارد کنید")]
        public int Qty { get; set; }

        public string? Tags { get; set; }

        public bool IsAvailable { get; set; }

        public virtual ICollection<ProductGallery> ProductGalleries { get; set; } =
new List<ProductGallery>();


        public int CategoryId { get; set; }
        public Category? Category { get; set; }
    }
}
```

کلاس Order

```
namespace MainChapar.Models
{
    public class Order
    {
        public int Id { get; set; }

        public DateTime CreatedAt { get; set; } = DateTime.Now;
        public bool IsConfirmed { get; set; } = false; // تأیید دستی یا اتومات
        public bool IsCollected { get; set; } = false; // آیا کاربر محصولو دریافت کرده
        public string PickupCode { get; set; } // اینو تبدیل به QR می‌کنیم

        // پراپرتی محاسبه‌ای برای قیمت کل
        public decimal TotalPrice
        {
            get
            {
                return OrderDetails?.Sum(od => od.Quantity * od.UnitPrice) ?? 0;
            }
        }

        //nav to user
        public string UserId { get; set; }
        public User User { get; set; }
        // nav to OrderDetails
        public virtual ICollection<OrderDetail> OrderDetails { get; set; }
    }
}
```

## کلاس OrderDetail

```
namespace MainChapar.Models
{
    public class OrderDetail
    {
        public int Id { get; set; }
        public int Quantity { get; set; }
        public decimal UnitPrice { get; set; }

        //nav to Order
        public int OrderId { get; set; }
        public Order Order { get; set; }

        //nav to Product
        public int ProductId { get; set; }
        public Product Product { get; set; }
    }
}
```

## کلاس Category

```
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models
{
    public class Category
    {
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        public string Slug { get; set; }         // انگلیسی، برای URL

        // Navigation property
        public List<Product>? Products { get; set; }
    }
}
```

## کلاس PickupPrintItem

```
namespace MainChapar.Models
{
    public class PickupPrintItem
    {
        public int Id { get; set; }

        public int PrintRequestId { get; set; }
        public PrintRequest PrintRequest { get; set; }

        public int PickupRequestId { get; set; }
        public PickupRequest PickupRequest { get; set; }

        // محاسبه قیمت نهایی بر اساس PrintRequest
        public decimal TotalPrice => PrintRequest?.TotalPrice ?? 0;
    }}
```

## کلاس PickupProductItem

```
namespace MainChapar.Models
{
    public class PickupProductItem
    {
        public int Id { get; set; }

        public int Quantity { get; set; }

        // Navigation to Product
        public int ProductId { get; set; }
        public Product Product { get; set; }

        // Navigation to PickupRequest
        public int PickupRequestId { get; set; }
        public PickupRequest PickupRequest { get; set; }

        // محاسبه قیمت نهایی این آیتم
        public decimal TotalPrice => (Product?.Price ?? 0 - Product?.Discount ?? 0)
* Quantity;
    }
}
```

## کلاس PickupRequest

```
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models
{
    public class PickupRequest
    {
        [Key]
        public int Id { get; set; }
        public string QrCodeToken { get; set; } // یکتا برای تولید QR
        public DateTime CreatedAt { get; set; } = DateTime.Now;
        public bool IsDelivered { get; set; } = false;

        public string Status { get; set; }
        // وضعیت‌ها:
        // "در انتظار تأیید"
        // "تأیید شده"
        // "در حال آماده‌سازی"
        // "آماده برای تحویل"
        // "تحویل داده شده"
        // "رد شده"

        //nav to user
        public string UserId { get; set; }
        public User User { get; set; }

        public ICollection<PickupProductItem> ProductItems { get; set; } = new
List<PickupProductItem>();
        public ICollection<PickupPrintItem> PickupPrintItems { get; set; } = new
List<PickupPrintItem>();


    }}
```

## کلاس ProductGallery

```
namespace MainChapar.Models
{
    public class ProductGallery
    {
        public int Id { get; set; }
        public string? ImageName { get; set; }

        public int ProductId { get; set; }
        public virtual Product Product { get; set; } = null!;
    }
}
```

## کلاس Comment

```
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models
{
    public class Comment
    {
        public int Id { get; set; }
        [Required]
        public string UserId { get; set; }       // آیدی کاربر
        public User User { get; set; }

        [Required]
        [MaxLength(1000)]
        public string Text { get; set; }
        public DateTime CreatedAt { get; set; } = DateTime.Now;
        public int ProductId { get; set; }
        public Product Product { get; set; }
    }
}
```

## کلاس User

```
using Microsoft.AspNetCore.Identity;

namespace MainChapar.Models
{
    public class User:IdentityUser
    {
        public string Name { get; set; }
        public string LName { get; set; }

        public ICollection<Comment> Comments { get; set; }
    }
}
```

## DTO ها:

```
using MainChapar.Helpers;
using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models.DTO
{
    public class BlackWhitePrintRequestDto
    {
        public string PaperType { get; set; }
        public string PaperSize { get; set; }
        public string PrintSide { get; set; }
        public int CopyCount { get; set; }
        public List<IFormFile> Files { get; set; }
    }
}


using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models.DTO
{
    public class ColorPrintRequestDto
    {
        [Required]
        public string PaperType { get; set; }

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

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

        [Required]
        [Range(1, 1000)]
        public int CopyCount { get; set; }
        public List<IFormFile> Files { get; set; }
    }
}

namespace MainChapar.Models.DTO
{
    public class LaminatePrintRequestDTO
    {
        public string PaperType { get; set; }
        public string PaperSize { get; set; }
        public string PrintSide { get; set; }
        public int CopyCount { get; set; }
        public string printType { get; set; } // BW , color
        public string LaminateType { get; set; } = "مات"; //مات ، براق
        public string CornerType { get; set; } //گوشه گرد، گوشه تیز
        public List<IFormFile> Files { get; set; }
    }
}
namespace MainChapar.Models.DTO
```

```
{
    public class LoginDTO
    {
        public string UserName { get; set; }
        public string Password { get; set; }
    }
}


namespace MainChapar.Models.DTO
{
    public class PlanPrintRequestDto
    {
        public int CopyCount { get; set; }

        public string SizeOrScaleDescription { get; set; }

        public string PaperType { get; set; }

        public string printType { get; set; } // BW , color

        public string? AdditionalDescription { get; set; }

        public string BindingType { get; set; } = "هیچکدام";

        public List<IFormFile> Files { get; set; }

    }
}


using System.ComponentModel.DataAnnotations;

namespace MainChapar.Models.DTO
{
    public class RegisterDTO
    {
        [Required]
        public string FirstName { get; set; }
        [Required]
        public string LastName { get; set; }
        [Required]
        [EmailAddress]
        public string Email { get; set; }
        [Required]
        public string Username { get; set; }
        [Required]
        [DataType(DataType.Password)]
        public string Password { get; set; }
        [Required]
        [Compare(nameof(Password))]
        public string ConfPass { get; set; }
    }
}


using System.ComponentModel.DataAnnotations;
```

```
namespace MainChapar.Models.DTOs
{
    public class UserDTO
    {
        [Required]
        public string FirstName { get; set; }
        [Required]
        public string LastName { get; set; }

        [Required]
        [EmailAddress]
        public string Email { get; set; }
        [Required] public string Username { get; set; }
        [Required]
        [DataType(DataType.Password)]
        public string Password { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Compare(nameof(Password))]
        public string ConfPassword { get; set; }
    }
}


namespace MainChapar.Models.DTO
{
    public class RoleDTO
    {
        public string Name { get; set; }
        public string Id { get; set; }

    }
}
```

ویو مدل ها:

```
using MainChapar.Helpers;
using System.ComponentModel.DataAnnotations;

namespace MainChapar.ViewModel.Print
{
    public class BlackWhitePrintRequestViewModel
    {

        [Required(ErrorMessage = "نوع کاغذ الزامی است")]
        public string PaperType { get; set; }

        [Required(ErrorMessage = "اندازه کاغذ الزامی است")]
        public string PaperSize { get; set; }

        [Required(ErrorMessage = "نوع چاپ الزامی است")]
        public string PrintSide { get; set; }

        [Required(ErrorMessage = "تعداد کپی الزامی است")]
        [Range(1, 1000, ErrorMessage = "تعداد کپی باید بین 1 تا 1000 باشد")]
        public int CopyCount { get; set; }

        [Display(Name = "توضیحات برای اپراتور")]
        public string Description { get; set; }

        [Display(Name = "نوع صحافی")]
        public string BindingType { get; set; } = "هیچکدام";

        [AtLeastOneFileRequired(ErrorMessage = "حداقل یک فایل باید انتخاب شود")]
        public List<IFormFile> Files { get; set; }
    }
}


using MainChapar.Models;
using System.ComponentModel.DataAnnotations;

namespace MainChapar.ViewModel.Print
{
    public class BWPrintSummaryViewModel
    {
        public string PaperType { get; set; }
        public string PaperSize { get; set; }
        public string PrintSide { get; set; }
        public int CopyCount { get; set; }
        public string Description { get; set; }
        public string BindingType { get; set; }
        public List<PrintFileViewModel> Files { get; set; } = new();
        public int TotalPages { get; set; }
        public decimal TotalPrice { get; set; }
    }
}
namespace MainChapar.ViewModel.Print
{
    public class PrintFileViewModel
    {
```

```
        public string FileName { get; set; }        // اسم فایل اصلی (مثلاً project.pdf)
        public string FilePath { get; set; }        //  مسیر فایل ذخیره‌شده در سرور (برای دانلود
یا نمایش)
        public int PageCount { get; set; }          // تعداد صفحات فایل (برای محاسبه قیمت)
        public decimal? FilePrice { get; set; }      // هزینه چاپ همین فایل خاص
    }
}


namespace MainChapar.ViewModel
{
    public class CartPrintItemViewModel
    {
        public int Id { get; set; }                // شناسه PickupPrintItem
        public int PrintRequestId { get; set; }  // شناسه درخواست چاپ
        public string ServiceType { get; set; }  // نوع سرویس (Color, BlackWhite,
Plan)
        public string Status { get; set; }       // وضعیت درخواست
        public decimal TotalPrice { get; set; }  // قیمت نهایی
        public DateTime CreatedAt { get; set; }  // زمان ایجاد درخواست
    }
}


namespace MainChapar.ViewModel
{
    public class CartProductItemViewModel
    {
        public int Id { get; set; }           // شناسه PickupProductItem
        public int ProductId { get; set; }    // شناسه محصول
        public string Title { get; set; }     // عنوان محصول
        public string ImageName { get; set; } // تصویر محصول (برای نمایش)
        public int Quantity { get; set; }     // تعداد سفارش داده شده
        public decimal UnitPrice { get; set; } // قیمت واحد پس از تخفیف (Price -
Discount)
        public decimal TotalPrice { get; set; } // قیمت کل (UnitPrice * Quantity)
        public int AvailableStock { get; set; } // موجودی انبار (برای اطلاع کاربر)

    }
}


using MainChapar.Models;

namespace MainChapar.ViewModel
{
    public class CartViewModel
    {
        public List<CartProductItemViewModel> Products { get; set; }
        public List<CartPrintItemViewModel> PrintRequests { get; set; }

        public decimal TotalPrice =>
            PrintRequests.Sum(p => p.TotalPrice) +
            Products.Sum(p => p.TotalPrice);
    }
}
using MainChapar.Models;
```

```
namespace MainChapar.ViewModel
{
    public class UserOrdersViewModel
    {
        public List<Order> ProductOrders { get; set; } = new();
        public List<PrintRequest> PrintRequests { get; set; } = new();
    }
}


using System.ComponentModel.DataAnnotations;

namespace MainChapar.ViewModel.Admin
{
    public class PrintPricingCreateViewModel
    {
        [Required(ErrorMessage = "نوع چاپ را انتخاب کنید.")]
        public string PrintType { get; set; }

        [Required(ErrorMessage = "نوع کاغذ را انتخاب کنید.")]
        public string PaperType { get; set; }

        [Required(ErrorMessage = "سایز کاغذ را انتخاب کنید.")]
        public string PaperSize { get; set; }

        public bool IsDoubleSided { get; set; }

        [Required(ErrorMessage = "قیمت به ازای هر صفحه را وارد کنید.")]
        [Range(0.01, double.MaxValue, ErrorMessage = "قیمت باید عدد مثبت باشد.")]
        public decimal PricePerPage { get; set; }

        public bool IsAvailable { get; set; }

        // برای لیست‌ها:
        //public List<string> PrintTypes { get; set; }
        //public List<string> PaperTypes { get; set; }
        //public List<string> PaperSizes { get; set; }

    }
}


namespace MainChapar.ViewModel.Admin
{
    public class PrintRequestListViewModel
    {
        public int Id { get; set; }
        public string UserFullName { get; set; }
        public string ServiceType { get; set; }  // رنگی، سیاه‌وسفید، پلان
        public DateTime CreatedAt { get; set; }
        public string Status { get; set; }       // Submitted, Processing, Rejected,
Completed
        public decimal TotalPrice { get; set; }
    }
}
```

کنترلر ها:

توجه: به علت حجم زیاد فقط بخشی از برخی کنترلر هارا اورده ایم

# PrintController:

```
private readonly ApplicationDbContext _context;
 private readonly IWebHostEnvironment _env;
 private readonly UserManager<User> _userManager;
 public PrintController(ApplicationDbContext context, IWebHostEnvironment env,
UserManager<User> userManager)
 {
     _context = context;
     _env = env;
     _userManager = userManager;
 }
 public IActionResult Index()
 {
     return View();
 }
 // GET: نمایش فرم چاپ سیاه‌وسفید
 [HttpGet]
 public IActionResult BlackWhitePrintForm()
 {
     var vm = new BlackWhitePrintRequestViewModel();
     return View(vm);
 }
 [HttpPost]
 [ValidateAntiForgeryToken]
 public async Task<IActionResult>
BlackWhitePrintForm(BlackWhitePrintRequestViewModel vm)
 {
     if (!ModelState.IsValid)
         return View(vm);

     var pricing = await _context.printPricings.FirstOrDefaultAsync(p =>
         p.PaperType == vm.PaperType &&
         p.PaperSize == vm.PaperSize &&
         p.IsDoubleSided == string.Equals(vm.PrintSide.Trim(), "پشت و رو",
StringComparison.OrdinalIgnoreCase) &&
         p.IsAvailable);

     if (pricing == null)
     {
         ModelState.AddModelError("", "ترکیب انتخاب‌شده در حال حاضر فعال نیست.");
         return View(vm);
     }

     var uploadPath = Path.Combine(_env.WebRootPath, "uploads");
     if (!Directory.Exists(uploadPath))
         Directory.CreateDirectory(uploadPath);

     var files = new List<PrintFileViewModel>();
```

```
    decimal totalPrice = 0;

    foreach (var file in vm.Files)
    {
        if (file == null || file.Length == 0)
            continue;

        var ext = Path.GetExtension(file.FileName).ToLowerInvariant();
        if (ext != ".pdf" && ext != ".docx" && ext != ".jpg" && ext != ".jpeg" &&
ext != ".png")
        {
            ModelState.AddModelError("", $"فرمت فایل '{file.FileName}' پشتیبانی نمی‌شود.");
            return View(vm);
        }

        var fileName = Guid.NewGuid() + ext;
        var filePath = Path.Combine(uploadPath, fileName);

        using (var stream = new FileStream(filePath, FileMode.Create))
        {
            await file.CopyToAsync(stream);
        }

        int pageCount = 1;
        if (ext == ".pdf")
        {
            pageCount = await GetPdfPageCount(file);
        }

        // بررسی اینکه چاپ پشت و رو است یا خیر
        bool isDoubleSided = string.Equals(vm.PrintSide?.Trim(), "پشت و رو",
StringComparison.OrdinalIgnoreCase);
        int effectivePages;

        if (string.Equals(vm.PrintSide?.Trim(), "پشت و رو",
StringComparison.OrdinalIgnoreCase))
        {
            //محاسبه قیمت صفحات فرد
            effectivePages = (int)Math.Ceiling(pageCount / 2.0);
        }
        else
        {
            effectivePages = pageCount;
        }

        decimal filePrice = effectivePages * vm.CopyCount * pricing.PricePerPage;
        totalPrice += filePrice;

        files.Add(new PrintFileViewModel
        {
            FileName = file.FileName,
            FilePath = "~/uploads/" + fileName,
            PageCount = pageCount,
            FilePrice = filePrice
        });

    }
```

```
    var summary = new BWPrintSummaryViewModel
    {
        PaperType = vm.PaperType,
        PaperSize = vm.PaperSize,
        PrintSide = vm.PrintSide,
        CopyCount = vm.CopyCount,
        TotalPages = files.Sum(f => f.PageCount),
        TotalPrice = totalPrice,
        Files = files,
        Description = vm.Description,
        BindingType = vm.BindingType,
    };

    TempData["PrintSummary"] = JsonConvert.SerializeObject(summary);
    return RedirectToAction("BlackWhitePrintStep2");
}
public IActionResult BlackWhitePrintStep2()
{
    if (TempData["PrintSummary"] == null)
        return RedirectToAction("BlackWhitePrintForm");

    var json = TempData["PrintSummary"] as string;
    var vm = JsonConvert.DeserializeObject<BWPrintSummaryViewModel>(json);

    TempData["PrintSummary"] = json; // برای مراحل بعدی دوباره ست می‌کنیم
    return View(vm);
}
[Authorize]
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> AddToCartFromPrint()
{
    var json = TempData["PrintSummary"] as string;
    if (json == null)
        return RedirectToAction("BlackWhitePrintForm");

    var vm = JsonConvert.DeserializeObject<BWPrintSummaryViewModel>(json);

    // ساختن شی PrintRequest
    var printRequest = new PrintRequest
    {
        CreatedAt = DateTime.Now,
        Status = "در انتظار تأیید",
        ServiceType = "BlackWhite",
        BlackWhitePrintDetail = new BlackWhitePrintDetail
        {
            PaperType = vm.PaperType,
            PaperSize = vm.PaperSize,
            PrintSide = vm.PrintSide,
            CopyCount = vm.CopyCount,
            TotalPages = vm.TotalPages,
            TotalPrice = vm.TotalPrice,
            FilesJson = JsonConvert.SerializeObject(vm.Files),
            Description = vm.Description,
            BindingType = vm.BindingType,
        }
    };
    var userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
```

```
    if (string.IsNullOrEmpty(userId))
    {
        // کاربر لاگین نکرده → می‌تونی برگردونی به صفحه لاگین
        return RedirectToAction("Login", "Account");
    }

    printRequest.UserId = userId;

    _context.PrintRequests.Add(printRequest);
    await _context.SaveChangesAsync();

    // افزودن آیدی به سبد خرید داخل Session
    var existing = HttpContext.Session.GetString("CartPrints");
    var list = string.IsNullOrEmpty(existing)
        ? new List<int>()
        : JsonConvert.DeserializeObject<List<int>>(existing);

    list.Add(printRequest.Id);
    HttpContext.Session.SetString("CartPrints", JsonConvert.SerializeObject(list));

    return RedirectToAction("Index", "Cart");
 }
private async Task<int> GetPdfPageCount(IFormFile file)
 {
    using var stream = file.OpenReadStream();
    using var pdf = PdfReader.Open(stream, PdfDocumentOpenMode.ReadOnly);
    return pdf.PageCount;
 }
```

# ProductController:

```
using MainChapar.Data;
using MainChapar.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Newtonsoft.Json;

namespace MainChapar.Controllers
{
    public class ProductController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly UserManager<User> _userManager;

        public ProductController(ApplicationDbContext context, UserManager<User>
userManager)
        {
            _context = context;
            _userManager = userManager;
        }

        public async Task<IActionResult> Index()
```

```
        {
            var products = await _context.Products
             .Where(p => p.IsAvailable == true && p.Qty > 0)
             .ToListAsync();
            return View(products);
        }

        //public async Task<IActionResult> Index()
        //{
        //    var products = await _context.Products.OrderByDescending(x =>
x.Id).ToListAsync();
        //    return View(products);
        //}

        public async Task<IActionResult> Details(int id)
        {
            var product = await _context.Products
                .Include(p => p.ProductGalleries)
                .FirstOrDefaultAsync(p => p.Id == id);

            if (product == null) return NotFound();

            // دریافت نظرات این محصول
            var comments = await _context.comments
                .Where(c => c.ProductId == id)
                .Include(c => c.User)
                .ToListAsync();

            ViewBag.Comments = comments;

            return View(product);
        }

        public IActionResult Category(string categorySlug)
        {
            if (string.IsNullOrEmpty(categorySlug))
                return NotFound();

            // دسته‌بندی رو از دیتابیس می‌گیریم
            var category = _context.categories
                                   .FirstOrDefault(c => c.Slug == categorySlug);

            if (category == null)
                return NotFound();

            // محصولات آن دسته‌بندی
            var products = _context.Products
                                   .Where(p => p.CategoryId == category.Id)
                                   .ToList();

            // به ویو دسته‌بندی و محصولات میفرستیم
            ViewBag.CategoryName = category.Name;

            return View(products);
        }

        [HttpGet]
        public IActionResult Search(string query)
```

```
        {
            if (string.IsNullOrEmpty(query))
            {
                // اگر چیزی وارد نشده، همه یا هیچ محصولی نشان بده
                var allProducts = _context.Products.ToList();
                return View("SearchResults", allProducts);
            }

            // (جستجو در نام محصولات (یا هر فیلدی که میخوای
            var results = _context.Products
                .Where(p => p.Title.Contains(query))
                .ToList();

            return View("Search", results);
        }

        //ارسال به سبد خرید

        [HttpPost]
        [Authorize]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> AddToCart(int productId, int quantity = 1)
        {
            if (quantity < 1)
                quantity = 1;

            var product = await _context.Products.FindAsync(productId);
            if (product == null || !product.IsAvailable || product.Qty < quantity)
            {
                // محصول موجود نیست یا تعداد درخواستی بیشتر از موجودی است
                ModelState.AddModelError("", " محصول انتخاب شده موجود نیست یا تعداد درخواستی معتبر
نیست.");
                // میتونی برگردونی به صفحه محصول یا صفحه سبد خرید
                return RedirectToAction("Details", new { id = productId });
            }

            var userId = _userManager.GetUserId(User);
            if (string.IsNullOrEmpty(userId))
            {
                // کاربر لاگین نکرده
                return RedirectToAction("Login", "Account");
            }

            // دریافت سبد خرید محصولات از سشن
            var existingCartJson = HttpContext.Session.GetString("CartProducts");
            Dictionary<int, int> cartProducts;

            if (string.IsNullOrEmpty(existingCartJson))
                cartProducts = new Dictionary<int, int>();
            else
                cartProducts = JsonConvert.DeserializeObject<Dictionary<int,
int>>(existingCartJson);

            // اگر محصول قبلاً در سبد بود، تعداد رو افزایش بده
            if (cartProducts.ContainsKey(productId))
                cartProducts[productId] += quantity;
            else
                cartProducts[productId] = quantity;
```

```
            // ذخیره دوباره در سشن
            HttpContext.Session.SetString("CartProducts",
JsonConvert.SerializeObject(cartProducts));

            // هدایت به صفحه سبد خرید
            return RedirectToAction("Index", "Cart");
        }


        [HttpPost]
        //[Authorize]
        public async Task<IActionResult> AddComment(int productId, string text)
        {
            var userId = _userManager.GetUserId(User);

            var comment = new Comment
            {
                ProductId = productId,
                Text = text,
                UserId = userId
            };

            _context.comments.Add(comment);
            await _context.SaveChangesAsync();

            return RedirectToAction("Details", new { id = productId });
        }


    }
}
```

# CartController:

```
using MainChapar.Data;

using MainChapar.Models;

using MainChapar.ViewModel;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Newtonsoft.Json;

using QRCoder;

using System.Security.Claims;

namespace MainChapar.Controllers

{
```

```
    public class CartController : Controller
    {
        private readonly ApplicationDbContext _context;
        private readonly UserManager<User> _userManager;

        public CartController(ApplicationDbContext context, UserManager<User>
userManager)
        {
            _context = context;
            _userManager = userManager;
        }

        // نمایش سبد خرید - با استفاده از ViewModel جدید
        public async Task<IActionResult> Index()
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            if (string.IsNullOrEmpty(userId))
            {
                TempData["Error"] = "ابتدا وارد حساب کاربری شوید.";
                return RedirectToAction("Index", "Home");
            }

            // جستجوی آخرین سبد خرید باز (در انتظار تحویل) کاربر
            var pickupRequest = await _context.pickupRequests
                .Include(p => p.ProductItems)
                    .ThenInclude(pi => pi.Product)
                .Include(p => p.PickupPrintItems)
                    .ThenInclude(ppi => ppi.PrintRequest)
                .Where(p => p.UserId == userId && !p.IsDelivered)
                .OrderByDescending(p => p.CreatedAt)
                .FirstOrDefaultAsync();
```

```
            if (pickupRequest == null)
            {
                // اگر سبد خالی است ویومدل خالی برمیگردانیم
                return View(new CartViewModel());
            }

            var cartVM = new CartViewModel();

            // تبدیل PickupProductItem به CartProductItemViewModel
            cartVM.Products = pickupRequest.ProductItems.Select(pi => new
CartProductItemViewModel
            {
                Id = pi.Id,
                ProductId = pi.ProductId,
                Title = pi.Product.Title,
                ImageName = string.IsNullOrEmpty(pi.Product.ImageName) ?
"default.png" : pi.Product.ImageName,
                Quantity = pi.Quantity,
                UnitPrice = (pi.Product.Price - (pi.Product.Discount ?? 0)),
                TotalPrice = (pi.Product.Price - (pi.Product.Discount ?? 0)) *
pi.Quantity,
                AvailableStock = pi.Product.Qty
            }).ToList();

            // تبدیل PickupPrintItem به CartPrintItemViewModel
            cartVM.PrintRequests = pickupRequest.PickupPrintItems.Select(pi => new
CartPrintItemViewModel
            {
                Id = pi.Id,
                PrintRequestId = pi.PrintRequestId,
                ServiceType = pi.PrintRequest.ServiceType,
                Status = pi.PrintRequest.Status,
```

```
                TotalPrice = pi.PrintRequest.TotalPrice ?? 0,
                CreatedAt = pi.PrintRequest.CreatedAt
            }).ToList();

            return View(cartVM);
        }

        // به‌روزرسانی تعداد محصول در سبد
        [HttpPost]
        public IActionResult UpdateQuantity(int productId, int quantity)
        {
            if (quantity < 1)
                quantity = 1;

            var productJson = HttpContext.Session.GetString("CartProducts");
            Dictionary<int, int> cartDict = new Dictionary<int, int>();

            if (!string.IsNullOrEmpty(productJson))
                cartDict = JsonConvert.DeserializeObject<Dictionary<int,
int>>(productJson);

            if (cartDict.ContainsKey(productId))
                cartDict[productId] = quantity;

            HttpContext.Session.SetString("CartProducts",
JsonConvert.SerializeObject(cartDict));

            return RedirectToAction("Index");
        }

        // ثبت نهایی سبد خرید با ذخیره در دیتابیس و کاهش موجودی محصولات
        [HttpPost]
```

```
        public async Task<IActionResult> Submit()
        {
            var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
            if (string.IsNullOrEmpty(userId))
            {
                TempData["Error"] = "کاربر شناسایی نشد. لطفاً دوباره وارد شوید.";
                return RedirectToAction("Index");
            }

            var printJson = HttpContext.Session.GetString("CartPrints");
            var productJson = HttpContext.Session.GetString("CartProducts");

            if (string.IsNullOrEmpty(printJson) &&
string.IsNullOrEmpty(productJson))
            {
                TempData["Error"] = "سبد خرید خالی است!";
                return RedirectToAction("Index");
            }

            try
            {
                var pickupRequest = new PickupRequest
                {
                    CreatedAt = DateTime.Now,
                    Status = "در انتظار تأیید",
                    QrCodeToken = Guid.NewGuid().ToString(),
                    UserId = userId
                };

                _context.pickupRequests.Add(pickupRequest);
                await _context.SaveChangesAsync();
```

```
                // ذخیره چاپ‌ها
                if (!string.IsNullOrEmpty(printJson))
                {
                    var printIds =
JsonConvert.DeserializeObject<List<int>>(printJson);

                    var existingPrints = await _context.PrintRequests
                        .Where(p => printIds.Contains(p.Id))
                        .Select(p => p.Id)
                        .ToListAsync();

                    foreach (var id in existingPrints)
                    {
                        _context.pickupPrintItems.Add(new PickupPrintItem
                        {
                            PrintRequestId = id,
                            PickupRequestId = pickupRequest.Id
                        });
                    }
                }

                // ذخیره محصولات و کاهش موجودی
                if (!string.IsNullOrEmpty(productJson))
                {
                    var dict = JsonConvert.DeserializeObject<Dictionary<int,
int>>(productJson);

                    var validProductIds = await _context.Products
                        .Where(p => dict.Keys.Contains(p.Id))
                        .Select(p => p.Id)
                        .ToListAsync();
```

```
                foreach (var item in dict)
                {
                    if (!validProductIds.Contains(item.Key)) continue;
                    if (item.Value <= 0) continue;

                    var product = await _context.Products.FindAsync(item.Key);
                    if (product == null) continue;

                    if (product.Qty < item.Value)
                    {
                        TempData["Error"] = $"موجودی محصول {product.Title} کافی
نیست.";
                        return RedirectToAction("Index");
                    }

                    product.Qty -= item.Value;
                    if (product.Qty <= 0)
                        product.IsAvailable = false;

                    _context.pickupProducts.Add(new PickupProductItem
                    {
                        ProductId = item.Key,
                        Quantity = item.Value,
                        PickupRequestId = pickupRequest.Id
                    });
                }
            }

            await _context.SaveChangesAsync();
```

```
            // پاک کردن سشن
            HttpContext.Session.Remove("CartPrints");
            HttpContext.Session.Remove("CartProducts");

            return RedirectToAction("Success", new { id = pickupRequest.Id });
        }
        catch (Exception)
        {
            TempData["Error"] = "خطایی در ثبت سبد خرید رخ داد.";
            return RedirectToAction("Index");
        }
    }

    // نمایش سفارش‌های کاربر
    public async Task<IActionResult> Orders()
    {
        var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
        if (string.IsNullOrEmpty(userId))
        {
            TempData["Error"] = "کاربر شناسایی نشد. لطفاً وارد شوید.";
            return RedirectToAction("Index", "Home");
        }

        var orders = await _context.pickupRequests
            .Where(o => o.UserId == userId)
            .Include(o => o.ProductItems)
                .ThenInclude(pi => pi.Product)
            .Include(o => o.PickupPrintItems)
                .ThenInclude(ppi => ppi.PrintRequest)
            .OrderByDescending(o => o.CreatedAt)
            .ToListAsync();
```

```
    return View(orders);
}

// نمایش جزئیات سفارش
public async Task<IActionResult> Details(int id)
{
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

    var order = await _context.pickupRequests
        .Where(p => p.Id == id && p.UserId == userId)
        .Include(p => p.ProductItems)
            .ThenInclude(pi => pi.Product)
        .Include(p => p.PickupPrintItems)
            .ThenInclude(pp => pp.PrintRequest)
        .FirstOrDefaultAsync();

    if (order == null)
        return NotFound();

    return View(order);
}
//حذف محصول
[HttpPost]
public IActionResult RemoveFromCart(int productId)
{
    var productJson = HttpContext.Session.GetString("CartProducts");
    if (string.IsNullOrEmpty(productJson))
        return RedirectToAction("Index");
```

```
            var cartDict = JsonConvert.DeserializeObject<Dictionary<int,
int>>(productJson);


            if (cartDict.ContainsKey(productId))

            {

                cartDict.Remove(productId);

                HttpContext.Session.SetString("CartProducts",
JsonConvert.SerializeObject(cartDict));

            }


            return RedirectToAction("Index");

        }


        // صفحه موفقیت و تولید QR کد (در صورت آماده بودن سفارش)

        public async Task<IActionResult> Success(int id)

        {

            var request = await _context.pickupRequests

                .Include(r => r.PickupPrintItems)

                    .ThenInclude(pi => pi.PrintRequest)

                .Include(r => r.ProductItems)

                    .ThenInclude(pi => pi.Product)

                .FirstOrDefaultAsync(r => r.Id == id);


            if (request == null)

                return NotFound();


            bool hasPrints = request.PickupPrintItems?.Any() == true;

            bool allPrintsReady = request.PickupPrintItems?.All(p =>
p.PrintRequest.Status == "آماده تحویل") ?? false;

            bool hasOnlyProducts = !hasPrints && (request.ProductItems?.Any() ==
true);


            if ((hasPrints && allPrintsReady) || hasOnlyProducts)
```

```
            {
                string qrText = $"PickupRequest:{id}";

                using (var qrGenerator = new QRCodeGenerator())
                using (var qrData = qrGenerator.CreateQrCode(qrText,
QRCodeGenerator.ECCLevel.Q))
                {
                    var qrCode = new SvgQRCode(qrData);
                    string svgImage = qrCode.GetGraphic(5);
                    ViewBag.QrCode = $"data:image/svg+xml;utf8,{svgImage}";
                }
            }
            else
            {
                ViewBag.QrCode = null;
                ViewBag.Message = "سفارش شما در حال پردازش است و هنوز آماده تحویل نمی‌باشد.";
            }

            ViewBag.RequestId = id;
            return View(request);
        }
    }
}
```

# userOrderController:

```
using MainChapar.Data;
using MainChapar.Models;
using MainChapar.ViewModel;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using QRCoder;

namespace MainChapar.Controllers
{
    public class UserOrderController : Controller
    {
        private readonly ApplicationDbContext _context;
```

```
        private readonly UserManager<User> _userManager;

        public UserOrderController(ApplicationDbContext context, UserManager<User>
userManager)
        {
            _context = context;
            _userManager = userManager;
        }

        // (محصولات + خدمات چاپی) نمایش همه سفارش‌های کاربر جاری
        public async Task<IActionResult> Index()
        {
            var userId = _userManager.GetUserId(User);

            // (محصولات) سفارش‌های فروشگاهی
            var productOrders = await _context.orders
                .Where(o => o.UserId == userId)
                .Include(o => o.OrderDetails)
                    .ThenInclude(od => od.Product)
                .OrderByDescending(o => o.CreatedAt)
                .ToListAsync();

            // سفارش‌های خدمات چاپی
            var printRequests = await _context.PrintRequests
                .Where(r => r.UserId == userId)
                .Include(r => r.ColorPrintDetail)
                .Include(r => r.BlackWhitePrintDetail)
                .Include(r => r.PlanPrintDetail)
                .Include(r => r.Files)
                .OrderByDescending(r => r.CreatedAt)
                .ToListAsync();


            var vm = new UserOrdersViewModel
            {
                ProductOrders = productOrders,
                PrintRequests = printRequests
            };


            return View(vm);

        }

        // (اختیاری) جزئیات یک سفارش فروشگاهی
        public async Task<IActionResult> ProductOrderDetails(int id)
        {
            var userId = _userManager.GetUserId(User);

            var order = await _context.orders
                .Where(o => o.Id == id && o.UserId == userId)
                .Include(o => o.OrderDetails)
                    .ThenInclude(od => od.Product)
                .FirstOrDefaultAsync();

            if (order == null) return NotFound();

            return View(order);
```

```
        }


        // تولید QR کد برای تحویل حضوری
        // لینک QR شامل لینک کامل به صفحه ادمین با پارامتر کد تحویل است
        public IActionResult GenerateQr(string pickupCode)
        {
            if (string.IsNullOrEmpty(pickupCode)) return NotFound();

            // ساخت URL لینک به متد FindByCode در کنترلر Order در ناحیه Admin
            string url = Url.Action("FindByCode", "Order", new { area = "Admin",
code = pickupCode }, Request.Scheme);

            using var qrGenerator = new QRCodeGenerator();
            using var qrData = qrGenerator.CreateQrCode(url,
QRCodeGenerator.ECCLevel.Q);
            using var qrCode = new PngByteQRCode(qrData);
            byte[] qrCodeBytes = qrCode.GetGraphic(20);

            return File(qrCodeBytes, "image/png");
        }
    }
}
```

# UserController:

```
using MainChapar.Models.DTOs;
using MainChapar.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using MainChapar.Models.DTO;

namespace MainChapar.Controllers
{
    public class UserController : Controller
    {
        private readonly SignInManager<User> _signInManager;
        private readonly UserManager<User> _userManager;

        public UserController(SignInManager<User> signInManager, UserManager<User>
userManager)
        {
            _signInManager = signInManager;
            _userManager = userManager;
        }
        public IActionResult Index()
        {
            return View();
        }
        public IActionResult Register()
        {
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> Register(UserDTO register)
```

```
{
    if (!ModelState.IsValid)
    {
        return View(register);
    }

    // ایجاد کاربر جدید
    var newUser = new User
    {
        UserName = register.Username,
        Email = register.Email,
        Name = register.FirstName,
        LName = register.LastName,
    };

    // ذخیره کاربر با استفاده از UserManager
    var result = await _userManager.CreateAsync(newUser, register.Password);

    if (!result.Succeeded)
    {
        // اگر ایجاد کاربر با شکست مواجه شد
        foreach (var error in result.Errors)
        {
            ModelState.AddModelError("", error.Description);
        }
        return View(register);
    }

    // پیدا کردن کاربر ایجاد شده
    var usr = await _userManager.FindByNameAsync(register.Username);
    if (usr == null)
    {
        ModelState.AddModelError("", "کاربر ایجاد شده پیدا نشد.");
        return View(register);
    }

    // اضافه کردن نقش به کاربر
    var result2 = await _userManager.AddToRoleAsync(usr, "user");

    if (result2.Succeeded)
    {
        return RedirectToAction("Index", "Home");
    }

    // در صورتی که اضافه کردن نقش به کاربر با شکست مواجه شد
    foreach (var error in result2.Errors)
    {
        ModelState.AddModelError("", error.Description);
    }

    return View(register);
}


[HttpGet]
public IActionResult Login()
{
    return View();
```

```
        }

        [HttpPost]
        public IActionResult Login(LoginDTO loginUser)
        {
            if (!ModelState.IsValid)
            {
                return View(loginUser); // ارسال مدل LoginDTO به ویو
            }

            var user = _userManager.FindByNameAsync(loginUser.UserName).Result;
            if (user == null)
            {
                ModelState.AddModelError("", "نام کاربری وجود ندارد.");
                return View(loginUser);
            }

            var result = _signInManager.PasswordSignInAsync(user,
loginUser.Password, true, true).Result;

            if (result.Succeeded)
            {
                return RedirectToAction("Index", "Home");
            }

            ModelState.AddModelError("", "نام کاربری یا رمز عبور اشتباه است.");
            return View(loginUser);
        }

        public IActionResult Logout()
        {
            _signInManager.SignOutAsync();
            return RedirectToAction("Login", "User");
        }

        public IActionResult UserList()
        {
            var users = _userManager.Users.Select(p => new
            {
                p.Id,
                p.Name,
                p.Email,
                p.LName,
                p.UserName,
            });
            return View(users);
        }

    }
}
```

کنترلر های ناحیه ادمین:

## CategoryController:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using MainChapar.Data;
using MainChapar.Models;

namespace MainChapar.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class CategoryController : Controller
    {
        private readonly ApplicationDbContext _context;

        public CategoryController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Admin/Category
        public async Task<IActionResult> Index()
        {
            return View(await _context.categories.ToListAsync());
        }

        // GET: Admin/Category/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var category = await _context.categories
                .FirstOrDefaultAsync(m => m.Id == id);
            if (category == null)
            {
                return NotFound();
            }

            return View(category);
        }

        // GET: Admin/Category/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Admin/Category/Create
```

```
        // To protect from overposting attacks, enable the specific properties you
want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(Category category)
        {

            category.Slug = category.Slug.Replace(" ", "-").ToLower();

            if (ModelState.IsValid)
            {
                _context.categories.Add(category);
                _context.SaveChanges();
                 await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(category);

        }

        // GET: Admin/Category/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            var category = _context.categories.Find(id);
            if (category == null) return NotFound();
            return View(category);
        }

        // POST: Admin/Category/Edit/5
        // To protect from overposting attacks, enable the specific properties you
want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, Category category)
        {
            if (ModelState.IsValid)
            {
                _context.categories.Update(category);
                _context.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(category);
        }

        // GET: Admin/Category/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            var category = _context.categories.Find(id);
            if (category == null) return NotFound();
            return View(category);
        }

        // POST: Admin/Category/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
```

```
        {
            var category = _context.categories.Find(id);
            if (category == null) return NotFound();

            _context.categories.Remove(category);
            _context.SaveChanges();
            return RedirectToAction("Index");
        }

        private bool CategoryExists(int id)
        {
            return _context.categories.Any(e => e.Id == id);
        }
    }
}
```

# orderController:

```
using MainChapar.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using QRCoder;
using System.Drawing;
using System.IO;


namespace MainChapar.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class OrderController : Controller
    {
        private readonly ApplicationDbContext _context;

        public OrderController(ApplicationDbContext context)
        {
            _context = context;
        }

        // ------------------ لیست سفارشات ------------------
        public async Task<IActionResult> Index()
        {
            var orders = await _context.orders
                .Include(o => o.User)
                .Include(o => o.OrderDetails).ThenInclude(od => od.Product)
                .OrderByDescending(o => o.CreatedAt)
                .ToListAsync();

            return View(orders);
        }

        // ------------------ جزئیات سفارش ------------------
        public async Task<IActionResult> Details(int id)
        {
            var order = await _context.orders
                .Include(o => o.User)
                .Include(o => o.OrderDetails).ThenInclude(d => d.Product)
```

```
        .FirstOrDefaultAsync(o => o.Id == id);

    if (order == null) return NotFound();

    return View(order);
}

// ------------------ تأیید سفارش (در صورت نیاز) ------------------
[HttpPost]
public async Task<IActionResult> Confirm(int id)
{
    var order = await _context.orders.FindAsync(id);
    if (order == null) return NotFound();

    order.IsConfirmed = true;
    _context.Update(order);
    await _context.SaveChangesAsync();

    return RedirectToAction("Index");
}

// ------------------ علامت‌گذاری به عنوان تحویل شده ------------------
[HttpPost]
public async Task<IActionResult> MarkAsCollected(int id)
{
    var order = await _context.orders.FindAsync(id);
    if (order == null) return NotFound();

    order.IsCollected = true;
    _context.Update(order);
    await _context.SaveChangesAsync();

    TempData["Success"] = "وضعیت سفارش به 'تحویل شده' تغییر کرد.";
    return RedirectToAction("Index");
}

// ------------------ جستجو بر اساس PickupCode ------------------
[HttpGet]
public async Task<IActionResult> FindByCode(string code)
{
    if (string.IsNullOrEmpty(code))
    {
        TempData["Error"] = "کد وارد نشده است.";
        return RedirectToAction("Index");
    }

    var order = await _context.orders
        .Include(o => o.User)
        .Include(o => o.OrderDetails).ThenInclude(od => od.Product)
        .FirstOrDefaultAsync(o => o.PickupCode == code);

    if (order == null)
    {
        TempData["Error"] = "سفارشی با این کد پیدا نشد.";
        return RedirectToAction("Index");
    }

    return View("VerifyPickup", order); // یک ویوی ساده برای نمایش وضعیت
```

```
        }

        // ------------------ تأیید تحویل از طریق QR ------------------

        [HttpPost]
        public async Task<IActionResult> ConfirmByCode(int id)
        {
            var order = await _context.orders.FindAsync(id);
            if (order == null) return NotFound();

            if (order.IsCollected)
            {
                TempData["Error"] = "این سفارش قبلاً تحویل شده است.";
                return RedirectToAction("FindByCode", new { code = order.PickupCode
});
            }

            order.IsCollected = true;
            _context.Update(order);
            await _context.SaveChangesAsync();

            TempData["Success"] = "تحویل با موفقیت ثبت شد.";
            return RedirectToAction("Index");
        }


    }
}
```

# PricingController:

```
using MainChapar.Data;
using MainChapar.Models;
using MainChapar.ViewModel.Admin;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace MainChapar.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class PricingController : Controller
    {
        private readonly ApplicationDbContext _context;

        public PricingController(ApplicationDbContext context)
        {
            _context = context;
        }

        // نمایش همه قیمت‌ها
        public async Task<IActionResult> Index()
        {
            var prices = await _context.printPricings.ToListAsync();
            return View(prices);
        }
```

```
        public IActionResult CreatePricing()
        {
            // چون لیست‌ها را حذف کردیم، اینجا فقط ویو مدل خالی می‌فرستیم
            return View(new PrintPricingCreateViewModel());
        }

        [HttpPost]
        public async Task<IActionResult> CreatePricing(PrintPricingCreateViewModel
vm)
        {
            if (!ModelState.IsValid)
            {
                // چون لیست نداریم نیازی به مقداردهی مجدد نیست
                return View(vm);
            }

            var pricing = new PrintPricing
            {
                PrintType = vm.PrintType,
                PaperType = vm.PaperType,
                PaperSize = vm.PaperSize,
                IsDoubleSided = vm.IsDoubleSided,
                PricePerPage = vm.PricePerPage,
                IsAvailable = vm.IsAvailable
            };

            _context.Add(pricing);
            await _context.SaveChangesAsync();

            return RedirectToAction("Index", "Pricing", new { area = "Admin" });
        }
    }
}
```

# PrintRequest:

```
using MainChapar.Data;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MainChapar.Models;

namespace MainChapar.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class PrintRequestsController : Controller
    {
        private readonly ApplicationDbContext _context;

        public PrintRequestsController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Admin/PrintRequests
        public async Task<IActionResult> Index()
        {
```

```
            var requests = await _context.PrintRequests.Include(p =>
p.User).ToListAsync();
            return View(requests);
        }

        // GET: Admin/PrintRequests/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            var request = await _context.PrintRequests
              .Include(r => r.User)
              .Include(r => r.ColorPrintDetail)
              .Include(r => r.BlackWhitePrintDetail)
              .Include(r => r.PlanPrintDetail)
              .Include(r => r.Files)
              .FirstOrDefaultAsync(r => r.Id == id);

            if (request == null)
                return NotFound();

            return View(request);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> UpdateStatus(int id, string status)
        {
            var request = await _context.PrintRequests.FindAsync(id);
            if (request == null)
                return NotFound();

            request.Status = status;
            await _context.SaveChangesAsync();

            return RedirectToAction(nameof(Details), new { id });
        }

        // POST: Admin/PrintRequests/SetStatus
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> SetStatus(int id, string status)
        {
            var request = await _context.PrintRequests.FindAsync(id);
            if (request == null) return NotFound();

            request.Status = status;
            await _context.SaveChangesAsync();

            return RedirectToAction(nameof(Index));
        }
    }
}
```

# ProductsController:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using MainChapar.Data;
using MainChapar.Models;

namespace MainChapar.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class ProductsController : Controller
    {
        private readonly ApplicationDbContext _context;

        public ProductsController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Admin/Products
        public async Task<IActionResult> Index()
        {
            return View(await _context.Products.Include(p =>
p.Category).ToListAsync());
        }

        // GET: Admin/Products/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
```

```
            {
                return NotFound();
            }

            var product = await _context.Products
                .FirstOrDefaultAsync(m => m.Id == id);
            if (product == null)
            {
                return NotFound();
            }

            return View(product);
        }



        // GET: Admin/Products/Create
        public IActionResult Create()
        {
            //ViewBag.Categories = new SelectList(_context.categories.ToList(),
"Id", "Name");
            ViewBag.categories = _context.categories.ToList();
            return View();
        }


        // POST: Admin/Products/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(Product product, IFormFile
MainImage, List<IFormFile> GalleryImage)
        {
            Console.WriteLine($"CategoryId from form: {product.CategoryId}");
            foreach (var key in ModelState.Keys)
```

```
            {
                var state = ModelState[key];
                foreach (var error in state.Errors)
                {
                    Console.WriteLine($"خطا در '{key}': {error.ErrorMessage}");
                }
            }
            //if (!ModelState.IsValid)
            //{
            //    ViewBag.Categories = new SelectList(_context.categories.ToList(),
"Id", "Name", product.CategoryId);
            //    return View(product);
            //}
            if (ModelState.IsValid)
            {
                // ذخیره عکس اصلی محصول
                if (MainImage != null && MainImage.Length > 0)
                {
                    var fileName = Guid.NewGuid().ToString() +
Path.GetExtension(MainImage.FileName);
                    var savePath = Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/assets/img/product", fileName);

                    using (var stream = new FileStream(savePath, FileMode.Create))
                    {
                        await MainImage.CopyToAsync(stream);
                    }

                    product.ImageName = fileName;
                }

                _context.Products.Add(product);
```

```
                await _context.SaveChangesAsync();


                // ذخیره تصاویر گالری محصول
                if (GalleryImage != null && GalleryImage.Count > 0)
                {
                    foreach (var item in GalleryImage)
                    {
                        if (item.Length > 0)
                        {
                            var galleryFileName = Guid.NewGuid().ToString() +
Path.GetExtension(item.FileName);
                            var galleryPath =
Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/assets/img/product",
galleryFileName);


                            using (var stream = new FileStream(galleryPath,
FileMode.Create))
                            {
                                await item.CopyToAsync(stream);
                            }


                            var newGallery = new ProductGallery()
                            {
                                ProductId = product.Id,
                                ImageName = galleryFileName
                            };


                            _context.ProductGallerys.Add(newGallery);
                        }
                    }


                    await _context.SaveChangesAsync();
                }
```

```
            return RedirectToAction(nameof(Index));
        }
        // در صورتی که ModelState نامعتبر باشد، دوباره فرم را با داده‌های قبلی برگردان
        ViewBag.Categories = new SelectList(_context.categories.ToList(), "Id",
"Name", product.CategoryId, "Slug");
        return View(product);



    }


    // GET: Admin/Products/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null)
        {
            return NotFound();
        }

        var product = await _context.Products.FindAsync(id);
        if (product == null)
        {
            return NotFound();
        }
        //--------------------
        ViewData["gallery"] = _context.ProductGallerys.Where(x => x.ProductId ==
product.Id).ToList();
        //----------------------
        return View(product);
    }


    // POST: Admin/Products/Edit/5
```

```
        // To protect from overposting attacks, enable the specific properties you
want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, Product product, IFormFile?
MainImage, IFormFile[]? GalleryImage)
        {
            if (id != product.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    //====================saveimage


                    if (MainImage != null)
                    {
                        string d = Directory.GetCurrentDirectory();
                        string fn = d + "\\wwwroot\\assets\\img\\product\\" +
product.ImageName;
                        if (System.IO.File.Exists(fn))
                        {
                            System.IO.File.Delete(fn);
                        }
                        using (var stream = new FileStream(fn, FileMode.Create))
                        {
                            MainImage.CopyTo(stream);
```

```
            }
        }

        //=======================================

        if (GalleryImage != null)
        {
            foreach (var item in GalleryImage)
            {
                var imagename = Guid.NewGuid() +
Path.GetExtension(item.FileName);

                //--------------
                string d = Directory.GetCurrentDirectory();
                string fn = d + "\\wwwroot\\assets\\img\\product\\" +
imagename;

                using (var stream = new FileStream(fn, FileMode.Create))
                {
                    item.CopyTo(stream);
                }

                //----------------
                var galleryitem = new ProductGallery()
                {
                    ImageName = imagename,
                    ProductId = product.Id,
                };
                _context.ProductGallerys.Add(galleryitem);

            }
        }
```

```
            _context.Update(product);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ProductExists(product.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(product);
}

// GET: Admin/Products/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var product = await _context.Products
        .FirstOrDefaultAsync(m => m.Id == id);
    if (product == null)
    {
```

```
                return NotFound();
            }

            return View(product);
        }

        // POST: Admin/Products/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            var product = await _context.Products.FindAsync(id);
            if (product != null)
            {

                //======================delete image

                string d = Directory.GetCurrentDirectory();
                string fn = d + "\\wwwroot\\assets\\img\\product\\";
                string mainimagepath = fn + product.ImageName;
                if (System.IO.File.Exists(mainimagepath))
                {
                    System.IO.File.Delete(mainimagepath);
                }

                //--------------------------
                var galleries = _context.ProductGallerys.Where(x => x.ProductId ==
id).ToList();
                if (galleries != null)
                {
                    foreach (var item in galleries)
                    {
```

```
                string galleryimagepath = fn + item.ImageName;



                if (System.IO.File.Exists(galleryimagepath))
                {
                    System.IO.File.Delete(galleryimagepath);
                }
            }
            _context.ProductGallerys.RemoveRange(galleries);
        }
        //==========================
        _context.Products.Remove(product);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool ProductExists(int id)
{
    return _context.Products.Any(e => e.Id == id);
}
public IActionResult DeleteGallery(int id)
{
    var gallery = _context.ProductGallerys.FirstOrDefault(x => x.Id == id);
    if (gallery == null)
    {
        return NotFound();
    }
    string d = Directory.GetCurrentDirectory();
    string fn = d + "\\wwwroot\\assets\\img\\product\\" + gallery.ImageName;
```

```
            if (System.IO.File.Exists(fn))
            {
                System.IO.File.Delete(fn);
            }
            _context.ProductGallerys.Remove(gallery);
            _context.SaveChanges();
            return Redirect("edit/" + gallery.ProductId);

        }
    }
}
```