```
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 PhaseEnd2.Data;
using PhaseEnd2.Models;

namespace PhaseEnd2.Controllers
{
    public class ProductsController : Controller
    {
        private readonly PhaseEndDbContext _context;

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

        // GET: Products
        public async Task<IActionResult> Index()
        {
              return _context.Product != null ?
                          View(await _context.Product.ToListAsync()) :
                          Problem("Entity set 'PhaseEndDbContext.Product'  is
null.");
        }

        // GET: Products/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Product == null)
            {
                return NotFound();
            }

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

            return View(product);
        }

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

        // POST: Products/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([Bind("ProductID,Name,UnitPrice,UnitInStock,ProductAvailable,ShortDescription
,PicturePath,Note")] Product product)
        {
            if (ModelState.IsValid)
            {
                _context.Add(product);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(product);
        }

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

            var product = await _context.Product.FindAsync(id);
            if (product == null)
            {
                return NotFound();
            }
            return View(product);
        }

        // POST: 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,
[Bind("ProductID,Name,UnitPrice,UnitInStock,ProductAvailable,ShortDescription,Pictur
ePath,Note")] Product product)
        {
            if (id != product.ProductID)
            {
                return NotFound();
            }

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

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

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

            return View(product);
        }

        // POST: Products/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.Product == null)
            {
                return Problem("Entity set 'PhaseEndDbContext.Product'  is null.");
            }
            var product = await _context.Product.FindAsync(id);
            if (product != null)
            {
                _context.Product.Remove(product);
            }

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

        private bool ProductExists(int id)
        {
          return (_context.Product?.Any(e => e.ProductID ==
id)).GetValueOrDefault();
        }
    }
}


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 PhaseEnd2.Data;
using PhaseEnd2.Models;

namespace PhaseEnd2.Controllers
{
    public class OrdersController : Controller
    {
        private readonly PhaseEndDbContext _context;

        public OrdersController(PhaseEndDbContext context)
        {
            _context = context;
        }

        // GET: Orders
        public async Task<IActionResult> Index()
        {
              return _context.Order != null ?
                          View(await _context.Order.ToListAsync()) :
                          Problem("Entity set 'PhaseEndDbContext.Order'  is null.");
        }

        // GET: Orders/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Order == null)
            {
                return NotFound();
            }

            var order = await _context.Order
                .FirstOrDefaultAsync(m => m.OrderID == id);
            if (order == null)
            {
                return NotFound();
            }

            return View(order);
        }

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

        // POST: Orders/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([Bind("OrderID,CustomerID,PaymentID,ShippingID,Discount,Taxes,TotalAmount,isC
ompleted,OrderDate,DIspatched,DispatchedDate,Shipped,ShippingDate,Deliver,DeliveryDa
te,Notes,CancelOrder")] Order order)
        {
            if (ModelState.IsValid)
            {
                _context.Add(order);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(order);
        }

        // GET: Orders/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.Order == null)
            {
                return NotFound();
            }

            var order = await _context.Order.FindAsync(id);
            if (order == null)
            {
                return NotFound();
            }
            return View(order);
        }

        // POST: Orders/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,
[Bind("OrderID,CustomerID,PaymentID,ShippingID,Discount,Taxes,TotalAmount,isComplete
d,OrderDate,DIspatched,DispatchedDate,Shipped,ShippingDate,Deliver,DeliveryDate,Note
s,CancelOrder")] Order order)
        {
            if (id != order.OrderID)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(order);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!OrderExists(order.OrderID))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(order);
        }

        // GET: Orders/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.Order == null)
            {
                return NotFound();
            }

            var order = await _context.Order
                .FirstOrDefaultAsync(m => m.OrderID == id);
            if (order == null)
            {
                return NotFound();
            }

            return View(order);
        }

        // POST: Orders/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.Order == null)
            {
                return Problem("Entity set 'PhaseEndDbContext.Order'  is null.");
            }
            var order = await _context.Order.FindAsync(id);
            if (order != null)
            {
                _context.Order.Remove(order);
            }

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

        private bool OrderExists(int id)
        {
          return (_context.Order?.Any(e => e.OrderID == id)).GetValueOrDefault();
        }
    }
}
```

```
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 PhaseEnd2.Data;
using PhaseEnd2.Models;

namespace PhaseEnd2.Controllers
{
    public class OrderDetailsController : Controller
    {
        private readonly PhaseEndDbContext _context;

        public OrderDetailsController(PhaseEndDbContext context)
        {
            _context = context;
        }

        // GET: OrderDetails
        public async Task<IActionResult> Index()
        {
            var phaseEndDbContext = _context.OrderDetail.Include(o =>
o.Order).Include(o => o.Product);
            return View(await phaseEndDbContext.ToListAsync());
        }

        // GET: OrderDetails/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.OrderDetail == null)
            {
                return NotFound();
            }

            var orderDetail = await _context.OrderDetail
                .Include(o => o.Order)
                .Include(o => o.Product)
                .FirstOrDefaultAsync(m => m.OrderDetailsID == id);
            if (orderDetail == null)
            {
                return NotFound();
            }

            return View(orderDetail);
        }

        // GET: OrderDetails/Create
        public IActionResult Create()
        {
            ViewData["OrderID"] = new SelectList(_context.Order, "OrderID",
"OrderID");
            ViewData["ProductID"] = new SelectList(_context.Set<Product>(),
"ProductID", "ProductID");
            return View();
        }
        // POST: OrderDetails/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([Bind("OrderDetailsID,OrderID,ProductID,UnitPrice,Quantity,OrderDate")] 
OrderDetail orderDetail)
        {
            if (ModelState.IsValid)
            {
                _context.Add(orderDetail);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["OrderID"] = new SelectList(_context.Order, "OrderID", 
"OrderID", orderDetail.OrderID);
            ViewData["ProductID"] = new SelectList(_context.Set<Product>(), 
"ProductID", "ProductID", orderDetail.ProductID);
            return View(orderDetail);
        }

        // GET: OrderDetails/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.OrderDetail == null)
            {
                return NotFound();
            }

            var orderDetail = await _context.OrderDetail.FindAsync(id);
            if (orderDetail == null)
            {
                return NotFound();
            }
            ViewData["OrderID"] = new SelectList(_context.Order, "OrderID", 
"OrderID", orderDetail.OrderID);
            ViewData["ProductID"] = new SelectList(_context.Set<Product>(), 
"ProductID", "ProductID", orderDetail.ProductID);
            return View(orderDetail);
        }

        // POST: OrderDetails/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, 
[Bind("OrderDetailsID,OrderID,ProductID,UnitPrice,Quantity,OrderDate")] OrderDetail 
orderDetail)
        {
            if (id != orderDetail.OrderDetailsID)
            {
                return NotFound();
            }
            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(orderDetail);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!OrderDetailExists(orderDetail.OrderDetailsID))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            ViewData["OrderID"] = new SelectList(_context.Order, "OrderID",
"OrderID", orderDetail.OrderID);
            ViewData["ProductID"] = new SelectList(_context.Set<Product>(),
"ProductID", "ProductID", orderDetail.ProductID);
            return View(orderDetail);
        }

        // GET: OrderDetails/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.OrderDetail == null)
            {
                return NotFound();
            }

            var orderDetail = await _context.OrderDetail
                .Include(o => o.Order)
                .Include(o => o.Product)
                .FirstOrDefaultAsync(m => m.OrderDetailsID == id);
            if (orderDetail == null)
            {
                return NotFound();
            }

            return View(orderDetail);
        }

        // POST: OrderDetails/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.OrderDetail == null)
            {
                return Problem("Entity set 'PhaseEndDbContext.OrderDetail'  is
null.");
            }
            var orderDetail = await _context.OrderDetail.FindAsync(id);
            if (orderDetail != null)
            {
                _context.OrderDetail.Remove(orderDetail);
            }

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

        private bool OrderDetailExists(int id)
        {
          return (_context.OrderDetail?.Any(e => e.OrderDetailsID ==
id)).GetValueOrDefault();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;


namespace PhaseEnd2.Models
{
    public class Product
    {
        public Product()
        {
            this.OrderDetails = new HashSet<OrderDetail>();

        }
        [Key]
        public int ProductID { get; set; }
        [Display(Name = "Product Name")]
        public string Name { get; set; }
        [Display(Name = "Supplier")]



        public decimal UnitPrice { get; set; }
        [Display(Name = "Previous Price")]

        public Nullable<int> UnitInStock { get; set; }
        [Display(Name = "Available?")]
        public Nullable<bool> ProductAvailable { get; set; }
        [Display(Name = "Description")]
        public string ShortDescription { get; set; }
        [Display(Name = "Picture")]
        public string PicturePath { get; set; }
        public string Note { get; set; }
        public virtual ICollection<OrderDetail> OrderDetails { get; set; }

    }
}


using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace PhaseEnd2.Models
{
    public class OrderDetail
    {
        [Key]
        public int OrderDetailsID { get; set; }
        public int OrderID { get; set; }
        public int ProductID { get; set; }
        public Nullable<decimal> UnitPrice { get; set; }
        public Nullable<int> Quantity { get; set; }
        public Nullable<System.DateTime> OrderDate { get; set; }

        public virtual Order Order { get; set; }
        public virtual Product Product { get; set; }
    }
}


using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace PhaseEnd2.Models
{
    public class Order
    {
        public Order()
        {
            this.OrderDetails = new HashSet<OrderDetail>();
        }

        [Key]
        public int OrderID { get; set; }
        public int CustomerID { get; set; }
        public Nullable<int> PaymentID { get; set; }
        public Nullable<int> ShippingID { get; set; }
        public Nullable<int> Discount { get; set; }
        public Nullable<int> Taxes { get; set; }
        public Nullable<int> TotalAmount { get; set; }
        public Nullable<bool> isCompleted { get; set; }
        public Nullable<System.DateTime> OrderDate { get; set; }
        public Nullable<bool> DIspatched { get; set; }
        public Nullable<System.DateTime> DispatchedDate { get; set; }
        public Nullable<bool> Shipped { get; set; }
        public Nullable<System.DateTime> ShippingDate { get; set; }
        public Nullable<bool> Deliver { get; set; }
        public Nullable<System.DateTime> DeliveryDate { get; set; }
        public string Notes { get; set; }
        public Nullable<bool> CancelOrder { get; set; }


        public virtual ICollection<OrderDetail> OrderDetails { get; set; }
    }
}
```

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "PhaseEndDbContext": "Server=DESKTOP-
MFQ8M0P;Database=PhaseEnd2.Data;Trusted_Connection=True;MultipleActiveResultSets=tru
e;TrustServerCertificate=True;"
  }
}
```