```
====================         BOOKSTORE.ADONET            =================

======= Database Creation ========
-- 1. Create database
CREATE DATABASE BookStoreDb;
GO



USE BookStoreDb;
GO

-- 2. Books table
IF OBJECT_ID('dbo.Books', 'U') IS NOT NULL DROP TABLE dbo.Books;
GO
CREATE TABLE dbo.Books (
Id INT IDENTITY(1,1) PRIMARY KEY,
Title NVARCHAR(200) NOT NULL,
Author NVARCHAR(150) NOT NULL,
Genre NVARCHAR(100) NULL,
ISBN NVARCHAR(20) NULL UNIQUE,
Price DECIMAL(10,2) NOT NULL CHECK (Price >= 0),
Stock INT NOT NULL DEFAULT 0 CHECK (Stock >= 0),
PublishedDate DATE NULL,
CreatedAt DATETIME2 NOT NULL DEFAULT SYSUTCDATETIME(),
UpdatedAt DATETIME2 NOT NULL DEFAULT SYSUTCDATETIME()
);
GO

-- 3. Helper: auto-update UpdatedAt
CREATE OR ALTER TRIGGER trg_Books_SetUpdatedAt
ON dbo.Books
AFTER UPDATE
```

```
AS

BEGIN

SET NOCOUNT ON;

UPDATE b

SET UpdatedAt = SYSUTCDATETIME()

FROM dbo.Books b

JOIN inserted i ON b.Id = i.Id;

END

GO


-- 4. Stored Procedures



-- Add

CREATE OR ALTER PROCEDURE dbo.usp_AddBook

@Title NVARCHAR(200),

@Author NVARCHAR(150),

@Genre NVARCHAR(100) = NULL,

@ISBN NVARCHAR(20) = NULL,

@Price DECIMAL(10,2),

@Stock INT = 0,

@PublishedDate DATE = NULL

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO dbo.Books (Title, Author, Genre, ISBN, Price, Stock,
PublishedDate)

VALUES (@Title, @Author, @Genre, @ISBN, @Price, @Stock, @PublishedDate);



SELECT SCOPE_IDENTITY() AS NewId;

END
```

```sql
GO

-- Update
CREATE OR ALTER PROCEDURE dbo.usp_UpdateBook
@Id INT,
@Title NVARCHAR(200),
@Author NVARCHAR(150),
@Genre NVARCHAR(100) = NULL,
@ISBN NVARCHAR(20) = NULL,
@Price DECIMAL(10,2),
@Stock INT,
@PublishedDate DATE = NULL
AS
BEGIN
SET NOCOUNT ON;
UPDATE dbo.Books
SET Title=@Title, Author=@Author, Genre=@Genre, ISBN=@ISBN,
Price=@Price, Stock=@Stock, PublishedDate=@PublishedDate
WHERE Id=@Id;
END
GO

-- Delete
CREATE OR ALTER PROCEDURE dbo.usp_DeleteBook
@Id INT
AS
BEGIN
SET NOCOUNT ON;
DELETE FROM dbo.Books WHERE Id=@Id;
END
GO
```

```sql
-- Get by Id

CREATE OR ALTER PROCEDURE dbo.usp_GetBookById

@Id INT

AS

BEGIN

SET NOCOUNT ON;

SELECT * FROM dbo.Books WHERE Id=@Id;

END

GO


-- Get all

CREATE OR ALTER PROCEDURE dbo.usp_GetAllBooks

AS

BEGIN

SET NOCOUNT ON;

SELECT * FROM dbo.Books ORDER BY CreatedAt DESC;

END

GO


-- Seed a few rows (optional)

INSERT INTO dbo.Books (Title, Author, Genre, ISBN, Price, Stock,
PublishedDate)

VALUES

(N'The Pragmatic Programmer', N'Andrew Hunt', N'Tech', N'978-0201616224',
42.50, 10, '1999-10-20'),

(N'Clean Code', N'Robert C. Martin', N'Tech', N'978-0132350884', 39.99, 7,
'2008-08-01');

GO
```

===== Project Source Code =====

===== Controllers\BooksController.cs =====

```
using BookStore.Web.Data;

using BookStore.Web.Models;

using Microsoft.AspNetCore.Mvc;

namespace BookStore.Web.Controllers

{

    public class BooksController : Controller

    {

        private readonly IBookRepository _repo;

        private readonly DisconnectedBookService _dsService;

        private readonly ILogger<BooksController> _logger;

        public BooksController(IBookRepository repo,
DisconnectedBookService

        dsService, ILogger<BooksController> logger)

        {

            _repo = repo; _dsService = dsService; _logger = logger;

        }

        // READ â€“ list with SqlDataReader (connected)

        public async Task<IActionResult> Index()

        {

            var books = await _repo.GetAllAsync();

            return View(books);

        }

        // READ â€“ details

        public async Task<IActionResult> Details(int id)

        {

            var book = await _repo.GetByIdAsync(id);

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

            return View(book);

        }

        // CREATE â€“ GET

        public IActionResult Create() => View(new Book

        {
            Stock = 0,
            Price =
        0
        });
        // CREATE â€“ POST (parameterized via stored proc)
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(Book model)
        {
            if (!ModelState.IsValid) return View(model);
            try
            {
                var newId = await _repo.AddAsync(model);
                TempData["Message"] = $"Book created with Id {newId}";
                return RedirectToAction(nameof(Index));
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Error creating book");
                ModelState.AddModelError(string.Empty, "An error occurred
whilecreating the book.");
                return View(model);
            }
        }
        // EDIT â€“ GET
        public async Task<IActionResult> Edit(int id)
        {
            var book = await _repo.GetByIdAsync(id);
            if (book == null) return NotFound();
            return View(book);
        }
        // EDIT â€“ POST
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, Book model)
        {
            if (id != model.Id) return BadRequest();
            if (!ModelState.IsValid) return View(model);
            try
            {
                await _repo.UpdateAsync(model);
                TempData["Message"] = "Book updated";
                return RedirectToAction(nameof(Index));
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Error updating book");
                ModelState.AddModelError(string.Empty, "An error occurred
while updating the book.");
            return View(model);
            }
        }
        // DELETE â€“ GET
        public async Task<IActionResult> Delete(int id)
        {
            var book = await _repo.GetByIdAsync(id);
            if (book == null) return NotFound();
            return View(book);
        }
        // DELETE â€“ POST
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            try
            {
                await _repo.DeleteAsync(id);
                TempData["Message"] = "Book deleted";
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, "Error deleting book");
                TempData["Error"] = "An error occurred while deleting
thebook.";
            }
            return RedirectToAction(nameof(Index));
        }
        // DISCONNECTED demo â€“ increase all prices by 5% using
DataSet/DataTable
        [HttpPost]
        public async Task<IActionResult> IncreasePrices()
        {
            var rows = await _dsService.IncreaseAllPricesAsync(0.05m);
            TempData["Message"] = $"Prices updated for {rows} row(s).";
            return RedirectToAction(nameof(Index));
        }
    }
}

===== Controllers\HomeController.cs =====

using System.Diagnostics;
using Microsoft.AspNetCore.Mvc;
using BookStore.Web.Models;

namespace BookStore.Web.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 });
    }
}

===== Data\BookRepository.cs =====

using BookStore.Web.Models;
using Microsoft.Data.SqlClient;
using System.Data;
```

```
namespace BookStore.Web.Data
{
    public class BookRepository : IBookRepository
    {
        private readonly IDbConnectionFactory _factory;
        private readonly ILogger<BookRepository> _logger;
        public BookRepository(IDbConnectionFactory factory,
        ILogger<BookRepository> logger)
        {
            _factory = factory;
            _logger = logger;
        }
        // Connected â€“ SqlDataReader (forward-only, fast)
        public async Task<IEnumerable<Book>> GetAllAsync(CancellationToken
ct =
        default)
        {
            var results = new List<Book>();
            await using var conn = _factory.Create();
            await conn.OpenAsync(ct);
            const string sql = "SELECT * FROM dbo.Books ORDER BY CreatedAt
DESC";
            await using var cmd = new SqlCommand(sql, conn)
            {
                CommandType =
            CommandType.Text
            };
            await using var reader = await cmd.ExecuteReaderAsync(ct);
            while (await reader.ReadAsync(ct))
            {
                results.Add(MapBook(reader));
            }
            return results;
        }
        // Stored Procedure â€“ Get by Id
        public async Task<Book?> GetByIdAsync(int id, CancellationToken ct
=
        default)
        {
            await using var conn = _factory.Create();
            await conn.OpenAsync(ct);
            await using var cmd = new SqlCommand("dbo.usp_GetBookById",
conn)
            { CommandType = CommandType.StoredProcedure };
            cmd.Parameters.Add(new SqlParameter("@Id", SqlDbType.Int)
            {
                Value =
            id
            });
            await using var reader = await cmd.ExecuteReaderAsync(ct);
            return await reader.ReadAsync(ct) ? MapBook(reader) : null;
        }

        // Stored Procedure â€“ Add (returns new Id)
        public async Task<int> AddAsync(Book book, CancellationToken ct =
        default)
        {
            await using var conn = _factory.Create();
            await conn.OpenAsync(ct);
            await using var cmd = new SqlCommand("dbo.usp_AddBook", conn)
            { CommandType = CommandType.StoredProcedure };
            cmd.Parameters.AddRange(new[]
            {
new SqlParameter("@Title", SqlDbType.NVarChar, 200) { Value =
book.Title },
new SqlParameter("@Author", SqlDbType.NVarChar, 150) { Value =
book.Author },
new SqlParameter("@Genre", SqlDbType.NVarChar, 100) { Value =
(object?)book.Genre ?? DBNull.Value },
new SqlParameter("@ISBN", SqlDbType.NVarChar, 20) { Value =
(object?)book.ISBN ?? DBNull.Value },
new SqlParameter("@Price", SqlDbType.Decimal) { Precision=10,
Scale=2, Value = book.Price },
new SqlParameter("@Stock", SqlDbType.Int) { Value =
book.Stock },
new SqlParameter("@PublishedDate", SqlDbType.Date) { Value =
(object?)book.PublishedDate ?? DBNull.Value }
});
            var newId = 0;
            await using var reader = await cmd.ExecuteReaderAsync(ct);
            if (await reader.ReadAsync(ct))
            {
                newId = Convert.ToInt32(reader["NewId"]);
            }
            return newId;
        }
        // Stored Procedure â€“ Update
        public async Task UpdateAsync(Book book, CancellationToken ct =
default)
        {
            await using var conn = _factory.Create();
            await conn.OpenAsync(ct);
            await using var cmd = new SqlCommand("dbo.usp_UpdateBook",
conn)
            { CommandType = CommandType.StoredProcedure };
            cmd.Parameters.AddRange(new[]
            {
            new SqlParameter("@Id", SqlDbType.Int) { Value = book.Id },
            new SqlParameter("@Title", SqlDbType.NVarChar, 200) { Value
=book.Title },

            new SqlParameter("@Author", SqlDbType.NVarChar, 150) { Value
=book.Author },

            new SqlParameter("@Genre", SqlDbType.NVarChar, 100) { Value =

            (object?)book.Genre ?? DBNull.Value },new
SqlParameter("@ISBN", SqlDbType.NVarChar, 20) { Value =(object?)book.ISBN
?? DBNull.Value },new SqlParameter("@Price", SqlDbType.Decimal) {
Precision=10,Scale=2, Value = book.Price },

            new SqlParameter("@Stock", SqlDbType.Int) { Value =book.Stock
},

            new SqlParameter("@PublishedDate", SqlDbType.Date) { Value
=(object?)book.PublishedDate ?? DBNull.Value }

            });

            await cmd.ExecuteNonQueryAsync(ct);

        }

        // Stored Procedure â€" Delete

        public async Task DeleteAsync(int id, CancellationToken ct =
default)

        {

            await using var conn = _factory.Create();

            await conn.OpenAsync(ct);

            await using var cmd = new SqlCommand("dbo.usp_DeleteBook",
conn)

            { CommandType = CommandType.StoredProcedure };

            cmd.Parameters.Add(new SqlParameter("@Id", SqlDbType.Int)

            {

                Value =

            id

            });

            await cmd.ExecuteNonQueryAsync(ct);

        }

        // Disconnected â€" DataSet via SqlDataAdapter

        public async Task<DataSet> GetAllDataSetAsync(CancellationToken ct
=
        default)
        {
            await using var conn = _factory.Create();
            await conn.OpenAsync(ct);
            const string sql = "SELECT * FROM dbo.Books ORDER BY CreatedAt
DESC";
            using var adapter = new SqlDataAdapter(sql, conn);
            var ds = new DataSet();
            adapter.Fill(ds, "Books"); // Fill DataSet with DataTable
named "Books"
            return await Task.FromResult(ds);
        }
        // Disconnected update: apply DataTable changes back to DB
        public async Task<int> UpdateViaDataSetAsync(DataSet ds,
        CancellationToken ct = default)
        {
            if (!ds.Tables.Contains("Books")) return 0;
            await using var conn = _factory.Create();
            await conn.OpenAsync(ct);
            const string sql = "SELECT Id, Title, Author, Genre, ISBN,
Price, Stock, PublishedDate FROM dbo.Books";
            using var adapter = new SqlDataAdapter(sql, conn);
            using var builder = new
            SqlCommandBuilder(adapter); // auto-generate
INSERT/UPDATE/DELETE .Ensure commands are parameterized (SqlCommandBuilder
handles this)
            var rows = adapter.Update(ds, "Books");
            return rows;
        }
        private static Book MapBook(SqlDataReader reader)
        {
            return new Book
            {
                Id = reader.GetInt32(reader.GetOrdinal("Id")),
                Title = reader.GetString(reader.GetOrdinal("Title")),
                Author = reader.GetString(reader.GetOrdinal("Author")),
                Genre = reader.IsDBNull(reader.GetOrdinal("Genre")) ? null
: reader.GetString(reader.GetOrdinal("Genre")),
                ISBN = reader.IsDBNull(reader.GetOrdinal("ISBN")) ? null :
reader.GetString(reader.GetOrdinal("ISBN")),
                Price = reader.GetDecimal(reader.GetOrdinal("Price")),
                Stock = reader.GetInt32(reader.GetOrdinal("Stock")),
                PublishedDate =
reader.IsDBNull(reader.GetOrdinal("PublishedDate")) ? (DateTime?)null
:reader.GetDateTime(reader.GetOrdinal("PublishedDate")),
                CreatedAt =
reader.GetDateTime(reader.GetOrdinal("CreatedAt")),
                UpdatedAt =
reader.GetDateTime(reader.GetOrdinal("UpdatedAt"))
            };
        }
    }
}
```

```
===== Data\DisConnectedBookService.cs =====
```

```
using System.Data;
namespace BookStore.Web.Data
{
    public class DisconnectedBookService
    {
        private readonly IBookRepository _repo;
        private readonly ILogger<DisconnectedBookService> _logger;
        public DisconnectedBookService(IBookRepository repo,
        ILogger<DisconnectedBookService> logger)
        {
            _repo = repo; _logger = logger;
        }
        // Example: increase all prices by a percent in-memory (DataTable)
then push changes

        public async Task<int> IncreaseAllPricesAsync(decimal pct,

        CancellationToken ct = default)

        {

            var ds = await _repo.GetAllDataSetAsync(ct);

            var table = ds.Tables["Books"]!;

            foreach (DataRow row in table.Rows)

            {

                var old = Convert.ToDecimal(row["Price"]);

                row["Price"] = Math.Round(old * (1 + pct), 2);

            }

            return await _repo.UpdateViaDataSetAsync(ds, ct);

        }

    }

}


===== Data\IBookRepository.cs =====


using BookStore.Web.Models;

using System.Data;




namespace BookStore.Web.Data

{

public interface IBookRepository

{

        Task<IEnumerable<Book>> GetAllAsync(CancellationToken ct =
default); // SqlDataReader (connected)

        Task<Book?> GetByIdAsync(int id, CancellationToken ct = default);
// Stored proc

        Task<int> AddAsync(Book book, CancellationToken ct = default); //
Stored proc
```

```
        Task UpdateAsync(Book book, CancellationToken ct = default); //
Stored proc

        Task DeleteAsync(int id, CancellationToken ct = default); //
Stored proc



        Task<DataSet> GetAllDataSetAsync(CancellationToken ct = default);
// SqlDataAdapter (disconnected)

        Task<int> UpdateViaDataSetAsync(DataSet ds, CancellationToken ct =
default); // Disconnected update

}

}


===== Data\IDbConnectionFactory.cs =====


using System.Data;

using Microsoft.Data.SqlClient;

using Microsoft.Extensions.Configuration;



namespace BookStore.Web.Data

{

    public interface IDbConnectionFactory

    {

        SqlConnection Create();

    }


 public class DbConnectionFactory : IDbConnectionFactory

    {

        private readonly string _connectionString;


        public DbConnectionFactory(IConfiguration configuration)

        {
            // "DefaultConnection" should match your appsettings.json
            _connectionString =
configuration.GetConnectionString("DefaultConnection")
                                ?? throw new
InvalidOperationException("Connection string not found.");
        }

        public SqlConnection Create()
        {
            return new SqlConnection(_connectionString);
        }
    }
}

===== Data\SqlConnectionFactory.cs =====

using Microsoft.Data.SqlClient;
using Microsoft.Extensions.Configuration;



namespace BookStore.Web.Data
{
    public class SqlConnectionFactory : IDbConnectionFactory
    {
        private readonly string _connString;
        public SqlConnectionFactory(IConfiguration config)
        {
        _connString = config.GetConnectionString("Default")!;
        }
        public SqlConnection Create() => new SqlConnection(_connString);
    }
}
```

```
===== Models\Book.cs =====

using System;
using System.ComponentModel.DataAnnotations;


namespace BookStore.Web.Models
{
    public class Book
    {
        public int Id { get; set; }


        [Required, StringLength(200)]
        public string Title { get; set; } = string.Empty;


        [Required, StringLength(150)]
        public string Author { get; set; } = string.Empty;


        [StringLength(100)]
        public string? Genre { get; set; }


        [StringLength(20)]
        public string? ISBN { get; set; }


        [Range(0, 999999)]
        public decimal Price { get; set; }
        [Range(0, int.MaxValue)]
        public int Stock { get; set; }


        [DataType(DataType.Date)]
        public DateTime? PublishedDate { get; set; }


        public DateTime CreatedAt { get; set; }
        public DateTime UpdatedAt { get; set; }
    }
}

===== Models\ErrorViewModel.cs =====

namespace BookStore.Web.Models;

public class ErrorViewModel
{
    public string? RequestId { get; set; }

    public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
}

===== Views\Books\Create.cshtml =====

@* @model BookStore.Web.Models.Book
@{
    ViewData["Title"] = "Create";
}
<h2>Create</h2>
```

```
<form method="post">

    <div asp-validation-summary="ModelOnly" class="text-danger"></div>

    <div class="mb-3"><label>Title</label><input asp-for="Title"
class="formcontrol" /></div>

    <div class="mb-3"><label>Author</label><input asp-for="Author"
class="formcontrol" /></div>

    <div class="mb-3"><label>Genre</label><input asp-for="Genre"
class="formcontrol" /></div>

    <div class="mb-3"><label>ISBN</label><input asp-for="ISBN"
class="formcontrol" /></div>

    <div class="mb-3"><label>Price</label><input asp-for="Price"
class="formcontrol" /></div>

    <div class="mb-3"><label>Stock</label><input asp-for="Stock"
class="formcontrol" /></div>

    <div class="mb-3"><label>Published Date</label><input
aspfor="PublishedDate" type="date" class="form-control" /></div>

    <button type="submit" class="btn btn-primary">Save</button>

    <a asp-action="Index" class="btn btn-secondary">Cancel</a>

</form> *@


@model BookStore.Web.Models.Book

@{

    ViewData["Title"] = "Create";

}

<h2>Create New Book</h2>

<form method="post">

    <div asp-validation-summary="ModelOnly" class="text-danger"></div>


    <div class="mb-3">

        <label>Title</label>

        <input asp-for="Title" class="form-control" />

    </div>


    <div class="mb-3">
```

```
        <label>Author</label>
        <input asp-for="Author" class="form-control" />
    </div>

    <div class="mb-3">
        <label>Genre</label>
        <input asp-for="Genre" class="form-control" />
    </div>

    <div class="mb-3">
        <label>ISBN</label>
        <input asp-for="ISBN" class="form-control" />
    </div>

    <div class="mb-3">
        <label>Price</label>
        <input asp-for="Price" class="form-control" />
    </div>

    <div class="mb-3">
        <label>Stock</label>
        <input asp-for="Stock" class="form-control" />
    </div>

    <div class="mb-3">
        <label>Published Date</label>
        <input asp-for="PublishedDate" type="date" class="form-control" />
    </div>

    <button type="submit" class="btn btn-primary">Save</button>
    <a asp-action="Index" class="btn btn-secondary">Cancel</a>
</form>
```

```
===== Views\Books\Delete.cshtml =====

@model BookStore.Web.Models.Book
@{
    ViewData["Title"] = "Delete";
}

<h2 class="text-danger mb-4">Delete Book</h2>

<div class="delete-container shadow-sm p-4 rounded">
    <h4 class="mb-3">Are you sure you want to delete this book?</h4>

    <dl class="row mb-4">
        <dt class="col-sm-2 fw-bold">Title</dt>
        <dd class="col-sm-10">@Model.Title</dd>

        <dt class="col-sm-2 fw-bold">Author</dt>
        <dd class="col-sm-10">@Model.Author</dd>
    </dl>

    <form method="post" class="d-flex gap-2">
        <input type="hidden" asp-for="Id" />
        <button type="submit" class="btn btn-danger px-4">Delete</button>
        <a class="btn btn-secondary px-4" asp-action="Index">Cancel</a>
    </form>
</div>

<style>
    .delete-container {
        background-color: #fff8f8;
        border: 1px solid #f5c2c7;
    }

    h2 {
        font-weight: 600;
    }

    dl dt {
        color: #333;
    }

    dl dd {
        color: #555;
    }

    .btn-danger {
        transition: 0.3s ease;
    }

    .btn-danger:hover {
        background-color: #bb2d3b;
    }

    .btn-secondary:hover {
        background-color: #495057;
    }
</style>

===== Views\Books\Details.cshtml =====

@model BookStore.Web.Models.Book
@{
    ViewData["Title"] = "Details";
```

```
}


<div class="details-container">
    <h2 class="page-title">ðŸ“– Book Details</h2>

    <div class="details-card shadow">
        <dl class="row">
            <dt class="col-sm-3">Title</dt>
            <dd class="col-sm-9">@Model.Title</dd>

            <dt class="col-sm-3">Author</dt>
            <dd class="col-sm-9">@Model.Author</dd>

            <dt class="col-sm-3">Genre</dt>
            <dd class="col-sm-9">@Model.Genre</dd>

            <dt class="col-sm-3">ISBN</dt>
            <dd class="col-sm-9">@Model.ISBN</dd>

            <dt class="col-sm-3">Price</dt>
            <dd class="col-sm-9">â‚¹ @Model.Price</dd>

            <dt class="col-sm-3">Stock</dt>
            <dd class="col-sm-9">@Model.Stock</dd>

            <dt class="col-sm-3">Published Date</dt>
            <dd class="col-sm-9">@Model.PublishedDate?.ToString("yyyy-MM-
dd")</dd>
        </dl>

        <div class="action-buttons">
            <a class="btn btn-warning" asp-action="Edit" asp-route-
id="@Model.Id">âœ�ï¸� Edit</a>
```

```
            <a class="btn btn-secondary" asp-action="Index">â¬…ï¸
Back</a>
        </div>
    </div>
</div>

<style>
.details-container {
    max-width: 700px;
    margin: 40px auto;
    padding: 20px;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.page-title {
    text-align: center;
    margin-bottom: 25px;
    font-weight: bold;
    color: #2c3e50;
}

.details-card {
    background-color: #ffffff;
    padding: 25px;
    border-radius: 12px;
    border: 1px solid #e3e3e3;
}

.details-card dt {
    font-weight: bold;
    color: #34495e;
    padding: 8px 0;
```

```css
}

.details-card dd {
    margin-bottom: 12px;
    color: #555;
    padding: 8px 0;
}

.action-buttons {
    margin-top: 20px;
    text-align: center;
}

.btn {
    padding: 10px 20px;
    border-radius: 8px;
    font-size: 14px;
    margin: 0 5px;
    transition: all 0.3s ease;
    cursor: pointer;
    text-decoration: none;
    display: inline-block;
}

.btn-warning {
    background-color: #f39c12;
    border: none;
    color: white;
}

.btn-warning:hover {
    background-color: #e67e22;
    transform: scale(1.05);
}

.btn-secondary {
    background-color: #95a5a6;
    border: none;
    color: white;
}

.btn-secondary:hover {
    background-color: #7f8c8d;
    transform: scale(1.05);
}

.shadow {
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}
</style>

===== Views\Books\Edit.cshtml =====

@model BookStore.Web.Models.Book
@{
    ViewData["Title"] = "Edit";
}

<div class="edit-container">
    <h2 class="page-title">âœ�ï¸� Edit Book</h2>
    <form method="post">
        <input type="hidden" asp-for="Id" />
        <div asp-validation-summary="ModelOnly" class="text-danger mb-
3"></div>
```

```
<div class="mb-3">
    <label>Title</label>
    <input asp-for="Title" class="form-control" />
</div>

<div class="mb-3">
    <label>Author</label>
    <input asp-for="Author" class="form-control" />
</div>

<div class="mb-3">
    <label>Genre</label>
    <input asp-for="Genre" class="form-control" />
</div>

<div class="mb-3">
    <label>ISBN</label>
    <input asp-for="ISBN" class="form-control" />
</div>

<div class="mb-3">
    <label>Price</label>
    <input asp-for="Price" class="form-control" />
</div>

<div class="mb-3">
    <label>Stock</label>
    <input asp-for="Stock" class="form-control" />
</div>

<div class="mb-3">
```

```
            <label>Published Date</label>

            <input asp-for="PublishedDate" type="date" class="form-
control" />

        </div>


        <button type="submit" class="btn btn-primary">Update</button>

        <a asp-action="Index" class="btn btn-secondary">Cancel</a>

    </form>

</div>


<style>

.edit-container {

    max-width: 600px;

    margin: 40px auto;

    padding: 25px 30px;

    background: #fff;

    border-radius: 12px;

    box-shadow: 0 6px 18px rgba(0, 0, 0, 0.1);

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, san


===== Views\Books\Index.cshtml =====


@model IEnumerable<BookStore.Web.Models.Book>

@{

    ViewData["Title"] = "Books";

}


<div class="index-container">

    <h2 class="page-title">ðŸ"š Book List</h2>


    <div class="action-buttons mb-3">

        <a class="btn btn-primary" asp-action="Create">âž• Add New</a>
        <form method="post" asp-action="IncreasePrices" class="d-inline">
            <button type="submit" class="btn btn-warning">Increase All
Prices by 5%</button>
        </form>
    </div>

    @if (TempData["Message"] != null)
    {
        <div class="alert alert-success">@TempData["Message"]</div>
    }
    @if (TempData["Error"] != null)
    {
        <div class="alert alert-danger">@TempData["Error"]</div>
    }

    <div class="table-responsive shadow-sm p-3 bg-white rounded">
        <table class="table table-striped table-hover mb-0">
            <thead class="table-dark">
                <tr>
                    <th>Title</th>
                    <th>Author</th>
                    <th>Genre</th>
                    <th>ISBN</th>
                    <th>Price</th>
                    <th>Stock</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var b in Model)
                {
                    <tr>
```

```
                        <td>@b.Title</td>
                        <td>@b.Author</td>
                        <td>@b.Genre</td>
                        <td>@b.ISBN</td>
                        <td>â‚¹ @b.Price</td>
                        <td>@b.Stock</td>
                        <td>
                            <a class="btn btn-info btn-sm" asp-
action="Details" asp-route-id="@b.Id">Details</a>
                            <a class="btn btn-warning btn-sm" asp-
action="Edit" asp-route-id="@b.Id">Edit</a>
                            <a class="btn btn-danger btn-sm" asp-
action="Delete" asp-route-id="@b.Id">Delete</a>
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
</div>

<style>
.index-container {
    max-width: 900px;
    margin: 30px auto;
    padding: 20px;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.page-title {
    text-align: center;
    margin-bottom: 25px;
    font-weight: 600;
    color: #2c3e50;
}

.action-buttons {
    display: flex;
    gap: 10px;
    flex-wrap: wrap;
    margin-bottom: 15px;
}

.btn {
    border-radius: 6px;
    padding: 6px 12px;
    transition: all 0.3s ease;
}

.btn-primary {
    background-color: #007bff;
    color: white;
    border: none;
}

.btn-primary:hover {
    background-color: #0056b3;
}

.btn-warning {
    background-color: #f39c12;
    color: white;
    border: none;
}
```

```
.btn-warning:hover {
    background-color: #e67e22;
}

.btn-danger {
    background-color: #c0392b;
    color: white;
    border: none;
}

.btn-danger:hover {
    background-color: #922b21;
}

.btn-info {
    background-color: #3498db;
    color: white;
    border: none;
}

.btn-info:hover {
    background-color: #2c81ba;
}

.table-responsive {
    overflow-x: auto;
}

.table-hover tbody tr:hover {
    background-color: #f1f1f1;
}
```

```
.alert-success {
    color: #155724;
    background-color: #d4edda;
    border: 1px solid #c3e6cb;
    border-radius: 6px;
    padding: 10px;
    margin-bottom: 15px;
}

.alert-danger {
    color: #721c24;
    background-color: #f8d7da;
    border: 1px solid #f5c6cb;
    border-radius: 6px;
    padding: 10px;
    margin-bottom: 15px;
}
</style>

===== Views\Home\Index.cshtml =====

@* @{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a
href="https://learn.microsoft.com/aspnet/core">building Web apps with
ASP.NET Core</a>.</p>
</div> *@

===== Views\Home\Privacy.cshtml =====
@* @{
    ViewData["Title"] = "Privacy Policy";
}
<h1>@ViewData["Title"]</h1>

<p>Use this page to detail your site's privacy policy.</p> *@

===== Views\Shared\Error.cshtml =====

@model ErrorViewModel
@{
    ViewData["Title"] = "Error";
}

<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your
request.</h2>

@if (Model.ShowRequestId)
{
    <p>
        <strong>Request ID:</strong> <code>@Model.RequestId</code>
    </p>
}

<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more
detailed information about the error that occurred.
</p>
<p>
```

```
    <strong>The Development environment shouldn't be enabled for deployed
applications.</strong>

    It can result in displaying sensitive information from exceptions to
end users.

    For local debugging, enable the <strong>Development</strong>
environment by setting the <strong>ASPNETCORE_ENVIRONMENT</strong>
environment variable to <strong>Development</strong>

    and restarting the app.

</p>


===== Views\Shared\_Layout.cshtml =====


<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>

    <title>@ViewData["Title"] - BookStore.Web</title>

    <link rel="stylesheet"
href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true"
/>

    <link rel="stylesheet" href="~/BookStore.Web.styles.css" asp-append-
version="true" />

</head>

<body>

    <header>

        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-
light bg-white border-bottom box-shadow mb-3">

            <div class="container-fluid">

                <a class="navbar-brand" asp-area="" asp-controller="Home"
asp-action="Index">BookStore.Web</a>

                <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target=".navbar-collapse" aria-
controls="navbarSupportedContent"
```

```
                        aria-expanded="false" aria-label="Toggle
navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex
justify-content-between">
                    <ul class="navbar-nav flex-grow-1">
                        @* <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Privacy">Privacy</a>
                        </li> *@
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2025 - BookStore.Web - <a asp-area="" asp-
controller="Home" asp-action="Privacy">Privacy</a>
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script
src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

    <script src="~/js/site.js" asp-append-version="true"></script>

    @await RenderSectionAsync("Scripts", required: false)

</body>

</html>


===== Views\Shared\_Layout.cshtml.css =====


/* Please see documentation at
https://learn.microsoft.com/aspnet/core/client-side/bundling-and-
minification

for details on configuring this project to bundle and minify static web
assets. */


a.navbar-brand {

  white-space: normal;

  text-align: center;

  word-break: break-all;

}


a {

  color: #0077cc;

}


.btn-primary {

  color: #fff;

  background-color: #1b6ec2;

  border-color: #1861ac;

}


.nav-pills .nav-link.active, .nav-pills .show > .nav-link {

  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

.border-top {
  border-top: 1px solid #e5e5e5;
}
.border-bottom {
  border-bottom: 1px solid #e5e5e5;
}

.box-shadow {
  box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}

button.accept-policy {
  font-size: 1rem;
  line-height: inherit;
}

.footer {
  position: absolute;
  bottom: 0;
  width: 100%;
  white-space: nowrap;
  line-height: 60px;
}

===== Views\Shared\_ValidationScriptsPartial.cshtml =====

<script src="~/lib/jquery-
validation/dist/jquery.validate.min.js"></script>
```

```
<script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"></script>


===== Views\_ViewImports.cshtml =====


@using BookStore.Web

@using BookStore.Web.Models

@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers


===== Views\_ViewStart.cshtml =====


@{

    Layout = "_Layout";

}


===== wwwroot\css\site.css =====



body {

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

    background-color: #f4f7fa;

    color: #333;

    margin: 0;

    padding: 0;

}



.container {

    max-width: 900px;

    margin: 30px auto;

    padding: 20px;

}
```

```css
.page-title {
    text-align: center;
    margin-bottom: 30px;
    font-size: 28px;
    font-weight: 600;
    color: #2c3e50;
}


.form-card {
    background-color: #ffffff;
    padding: 25px;
    border-radius: 12px;
    border: 1px solid #e3e3e3;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
}

.form-control, input.formcontrol {
    width: 100%;
    padding: 10px 12px;
    margin-top: 5px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 6px;
    font-size: 14px;
    transition: all 0.3s ease;
}

.form-control:focus, input.formcontrol:focus {
    border-color: #3498db;
    outline: none;
```

```css
}


.btn {
    padding: 8px 16px;
    border-radius: 8px;
    font-size: 14px;
    margin: 3px 2px;
    transition: all 0.3s ease;
    cursor: pointer;
    text-decoration: none;
    display: inline-block;
}

/* Button colors */
.btn-primary {
    background-color: #007bff;
    color: white;
    border: none;
}
.btn-primary:hover { background-color: #0056b3; }

.btn-warning {
    background-color: #f39c12;
    color: white;
    border: none;
}
.btn-warning:hover { background-color: #e67e22; }

.btn-danger {
    background-color: #c0392b;
    color: white;
    border: none;
}
.btn-danger:hover { background-color: #922b21; }

.btn-info {
    background-color: #3498db;
    color: white;
    border: none;
}
.btn-info:hover { background-color: #2c81ba; }

.btn-secondary {
    background-color: #95a5a6;
    color: white;
    border: none;
}
.btn-secondary:hover { background-color: #7f8c8d; }



.table-responsive {
    overflow-x: auto;
    background-color: #fff;
    padding: 15px;
    border-radius: 12px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
}

table {
    width: 100%;
    border-collapse: collapse;
}
```

```
table th, table td {
    padding: 12px 10px;
    text-align: left;
}

table thead {
    background-color: #34495e;
    color: white;
    font-weight: 600;
}

.table-striped tbody tr:nth-of-type(odd) {
    background-color: #f2f2f2;
}

.table-hover tbody tr:hover {
    background-color: #f1f1f1;
}



.details-card {
    background-color: #ffffff;
    padding: 25px;
    border-radius: 12px;
    border: 1px solid #e3e3e3;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
}

.details-card dt {
    font-weight: bold;
    color: #34495e;
    padding: 8px 0;
```

```
}

.details-card dd {
    margin-bottom: 12px;
    color: #555;
    padding: 8px 0;
}


.alert {
    padding: 10px 15px;
    border-radius: 6px;
    margin-bottom: 15px;
    font-weight: 500;
}

.alert-success {
    color: #155724;
    background-color: #d4edda;
    border: 1px solid #c3e6cb;
}

.alert-danger {
    color: #721c24;
    background-color: #f8d7da;
    border: 1px solid #f5c6cb;
}


.action-buttons {
    display: flex;
    gap: 10px;
    flex-wrap: wrap;
    margin-top: 15px;
}



.text-center {
    text-align: center;
}



@media (max-width: 768px) {
    .table th, .table td {
        padding: 8px 5px;
        font-size: 13px;
    }

    .btn {
        font-size: 12px;
        padding: 6px 10px;
    }
}

===== wwwroot\js\site.js =====

// Please see documentation at
https://learn.microsoft.com/aspnet/core/client-side/bundling-and-
minification
// for details on configuring this project to bundle and minify static web
assets.

// Write your JavaScript code.

===== Program.cs =====
using BookStore.Web.Data;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

// Register data access services

builder.Services.AddSingleton<IDbConnectionFactory,
DbConnectionFactory>();

builder.Services.AddScoped<IBookRepository, BookRepository>();


builder.Services.AddScoped<DisconnectedBookService>();

var app = builder.Build();

if (!app.Environment.IsDevelopment())

{

    app.UseExceptionHandler("/Home/Error");

    app.UseHsts();

}

else

{

    app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Books}/{action=Index}/{id?}");

app.Run();
```