**1. Setup Project & Dependencies**

Create a new Web API project:

dotnet new webapi -n LibraryManagement

cd LibraryManagement

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Tools

dotnet add package FluentValidation.AspNetCore

**2. Define Entities**

**Book.cs**

using System.ComponentModel.DataAnnotations;

namespace LibraryManagement.Models

{

public class Book

{

public int Id { get; set; }

[Required]

[MaxLength(100)]

public string Title { get; set; }

[Required]

[MaxLength(50)]

public string Author { get; set; }

[Range(1000, 2024)]

public int PublishedYear { get; set; }

public int? LibraryCardId { get; set; }

public LibraryCard LibraryCard { get; set; }

}

}

**LibraryCard.cs**

using System.ComponentModel.DataAnnotations;

namespace LibraryManagement.Models

{

public class LibraryCard

{

public int Id { get; set; }

[Required]

[RegularExpression(@"LC-\d{5}")]

public string CardNumber { get; set; }

[Required]

[MaxLength(100)]

public string MemberName { get; set; }

public DateTime ExpiryDate { get; set; }

public ICollection<Book> Books { get; set; }

}

}

**3. Fluent Validation**

**BookValidator.cs**

using FluentValidation;

using LibraryManagement.Models;

public class BookValidator : AbstractValidator<Book>

{

public BookValidator()

{

RuleFor(b => b.Title).NotEmpty().MaximumLength(100);

RuleFor(b => b.Author).NotEmpty().MaximumLength(50);

RuleFor(b => b.PublishedYear).InclusiveBetween(1000, 2024);

}

}

**LibraryCardValidator.cs**

using FluentValidation;

using LibraryManagement.Models;

public class LibraryCardValidator : AbstractValidator<LibraryCard>

{

public LibraryCardValidator()

{

RuleFor(lc => lc.CardNumber).NotEmpty().Matches(@"LC-\d{5}");

RuleFor(lc => lc.MemberName).NotEmpty().MaximumLength(100);

}

}

**4. AppDbContext**

using Microsoft.EntityFrameworkCore;

using LibraryManagement.Models;

namespace LibraryManagement.Data

{

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

public DbSet<Book> Books { get; set; }

public DbSet<LibraryCard> LibraryCards { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

// One-to-Many relationship

modelBuilder.Entity<Book>()

.HasOne(b => b.LibraryCard)

.WithMany(lc => lc.Books)

.HasForeignKey(b => b.LibraryCardId);

// Seed LibraryCards

modelBuilder.Entity<LibraryCard>().HasData(

new LibraryCard

{

Id = 1,

CardNumber = "LC-12345",

MemberName = "John Doe",

ExpiryDate = new DateTime(2025, 12, 31)

},

new LibraryCard

{

Id = 2,

CardNumber = "LC-54321",

MemberName = "Jane Smith",

ExpiryDate = new DateTime(2024, 10, 15)

}

);

}

}

}

**5. Configure Services (Program.cs)**

builder.Services.AddDbContext<AppDbContext>(options =>

options.UseSqlServer("User ID=sa;password=examlyMssql@123; server=localhost;Database=appdb;trusted\_connection=false;Persist Security Info=False;Encrypt=False"));

builder.Services.AddControllers()

.AddFluentValidation(fv => fv.RegisterValidatorsFromAssemblyContaining<Program>());

builder.Services.AddSwaggerGen();

**6. LibraryController**

using LibraryManagement.Data;

using LibraryManagement.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

[ApiController]

[Route("api/[controller]")]

public class LibraryController : ControllerBase

{

private readonly AppDbContext \_context;

public LibraryController(AppDbContext context)

{

\_context = context;

}

// Display Books for a LibraryCard

[HttpGet("{libraryCardId}/books")]

public IActionResult DisplayBooksForLibraryCard(int libraryCardId)

{

var books = \_context.Books

.Where(b => b.LibraryCardId == libraryCardId)

.Select(b => new { b.Id, b.Title, b.Author, b.PublishedYear })

.ToList();

if (!books.Any())

return NotFound("No books found for this library card.");

return Ok(books);

}

// Add Book

[HttpPost("books")]

public IActionResult AddBook([FromBody] Book book)

{

if (!ModelState.IsValid)

return BadRequest(ModelState);

\_context.Books.Add(book);

\_context.SaveChanges();

return CreatedAtAction(nameof(DisplayBooksForLibraryCard), new { libraryCardId = book.LibraryCardId }, book);

}

// Display all books

[HttpGet("books")]

public IActionResult DisplayAllBooks()

{

var books = \_context.Books

.Select(b => new { b.Id, b.Title, b.Author, b.PublishedYear })

.ToList();

return Ok(books);

}

// Search books by title

[HttpGet("books/search")]

public IActionResult SearchBooksByTitle(string query)

{

var books = \_context.Books

.Where(b => b.Title.Contains(query, StringComparison.OrdinalIgnoreCase))

.Select(b => new { b.Id, b.Title, b.Author, b.PublishedYear })

.ToList();

return Ok(books);

}

}

**7. Run Migrations**

dotnet dotnet-ef migrations add InitialSetup

dotnet dotnet-ef database update

**8. Test the API**

* Start the app:

dotnet run

* Open Swagger at http://localhost:5000/swagger to test endpoints:

1. **GET** /api/library/books – Display all books
2. **POST** /api/library/books – Add a new book
3. **GET** /api/library/{libraryCardId}/books – Display books for a card
4. **GET** /api/library/books/search?query=abc – Search books by title