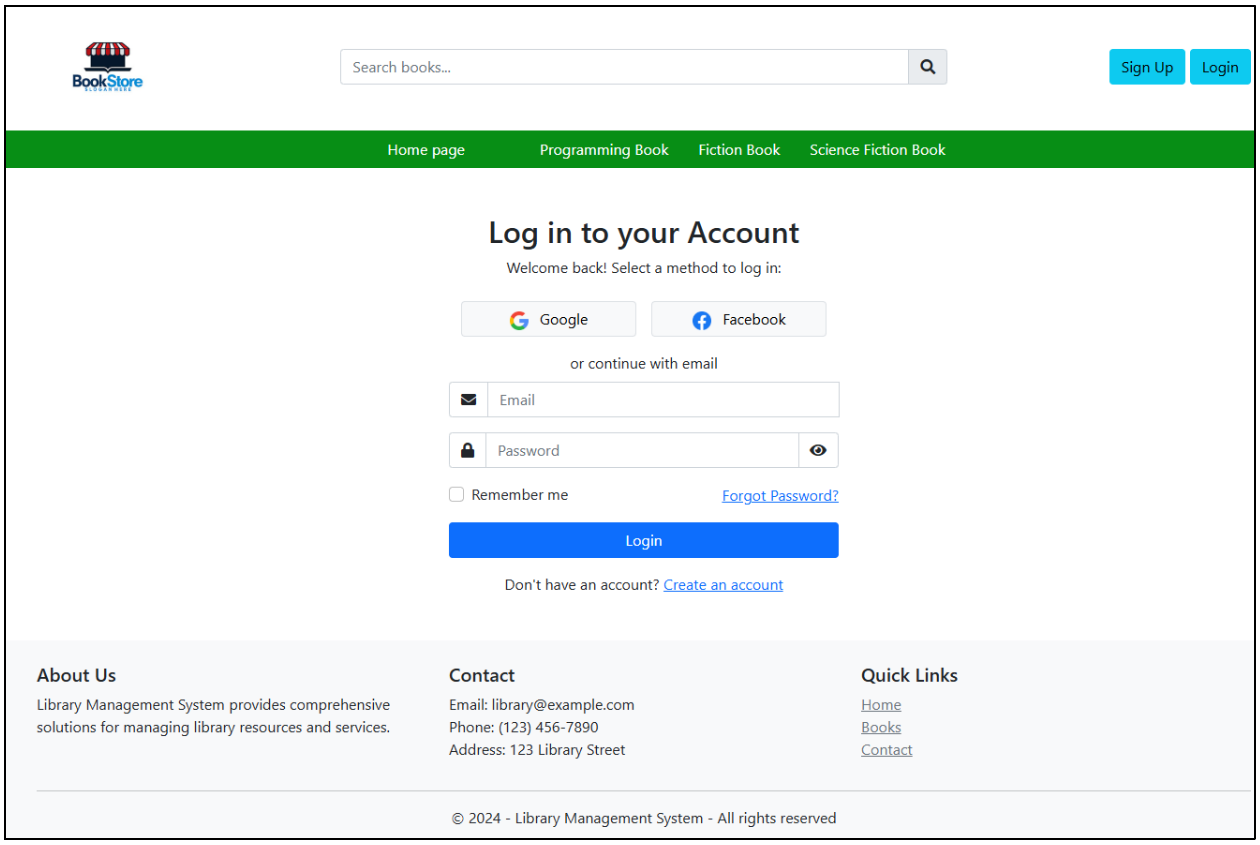
Exercise 1: Implement Authentication for User Login



Search books...

Sign Up | Login

Home page | Programming Book | Fiction Book | Science Fiction Book

Log in to your Account

Welcome back! Select a method to log in:

Google | Facebook

or continue with email

Email

Password

Remember me   Forgot Password?

Login

Don't have an account? Create an account

About Us

Library Management System provides comprehensive solutions for managing library resources and services.

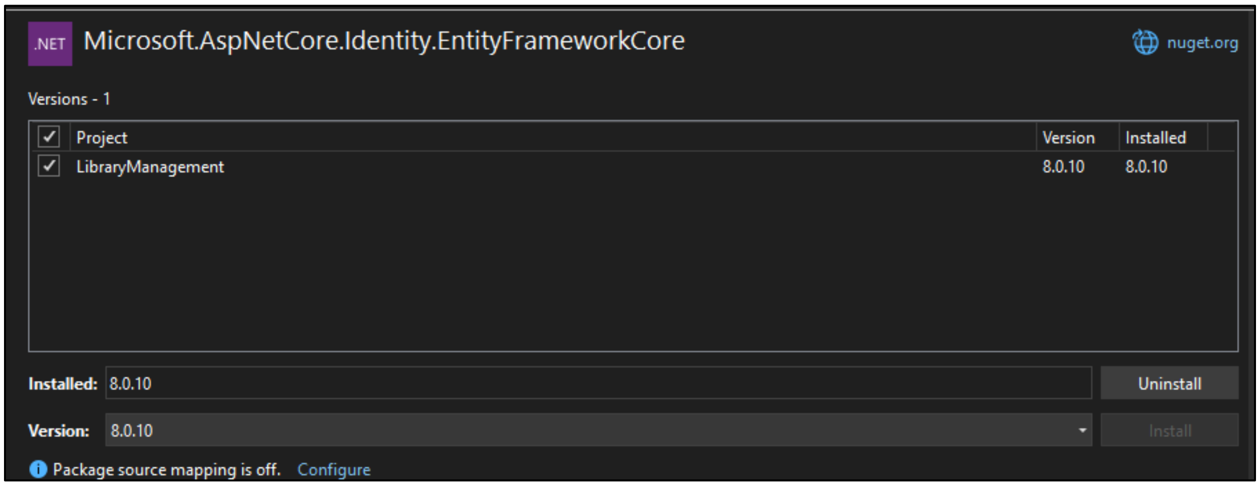
Contact

Email: library@example.com
Phone: (123) 456-7890
Address: 123 Library Street

Quick Links

Home
Books
Contact

© 2024 - Library Management System - All rights reserved



.NET Microsoft.AspNetCore.Identity.EntityFrameworkCore   nuget.org

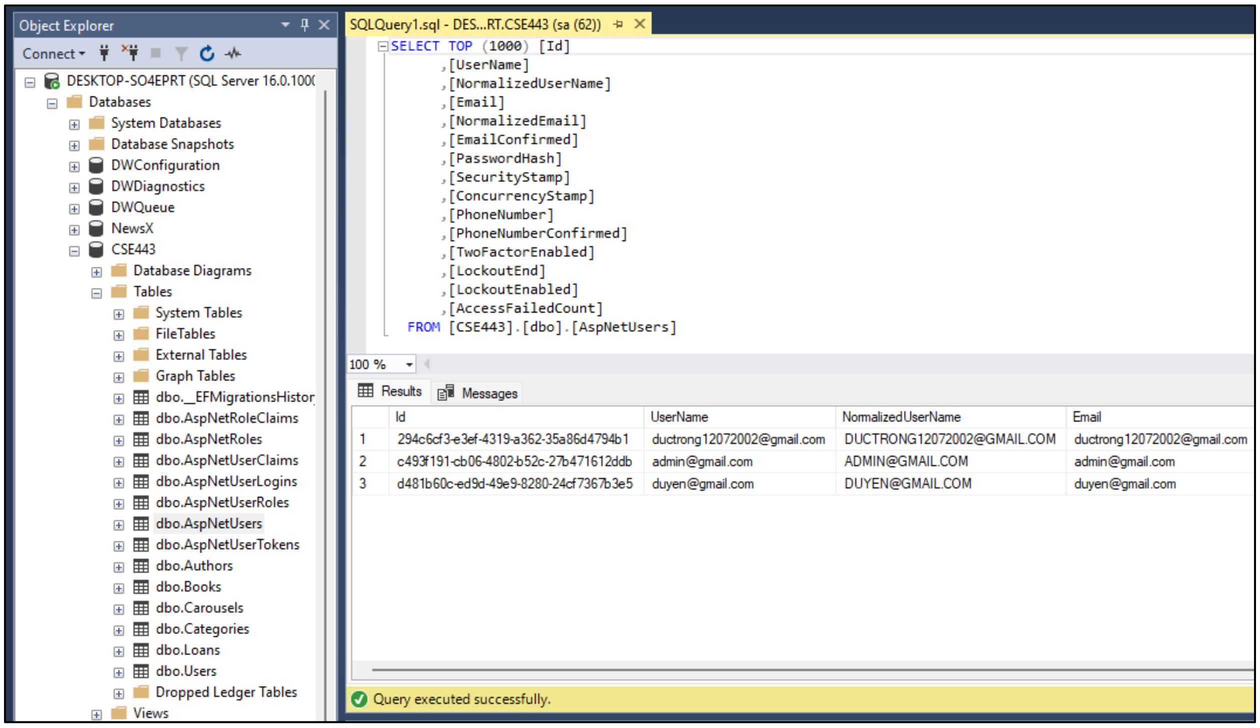
Versions - 1

✓ Project	Version	Installed
✓ LibraryManagement	8.0.10	8.0.10

Installed: 8.0.10   Uninstall

Version: 8.0.10   Install

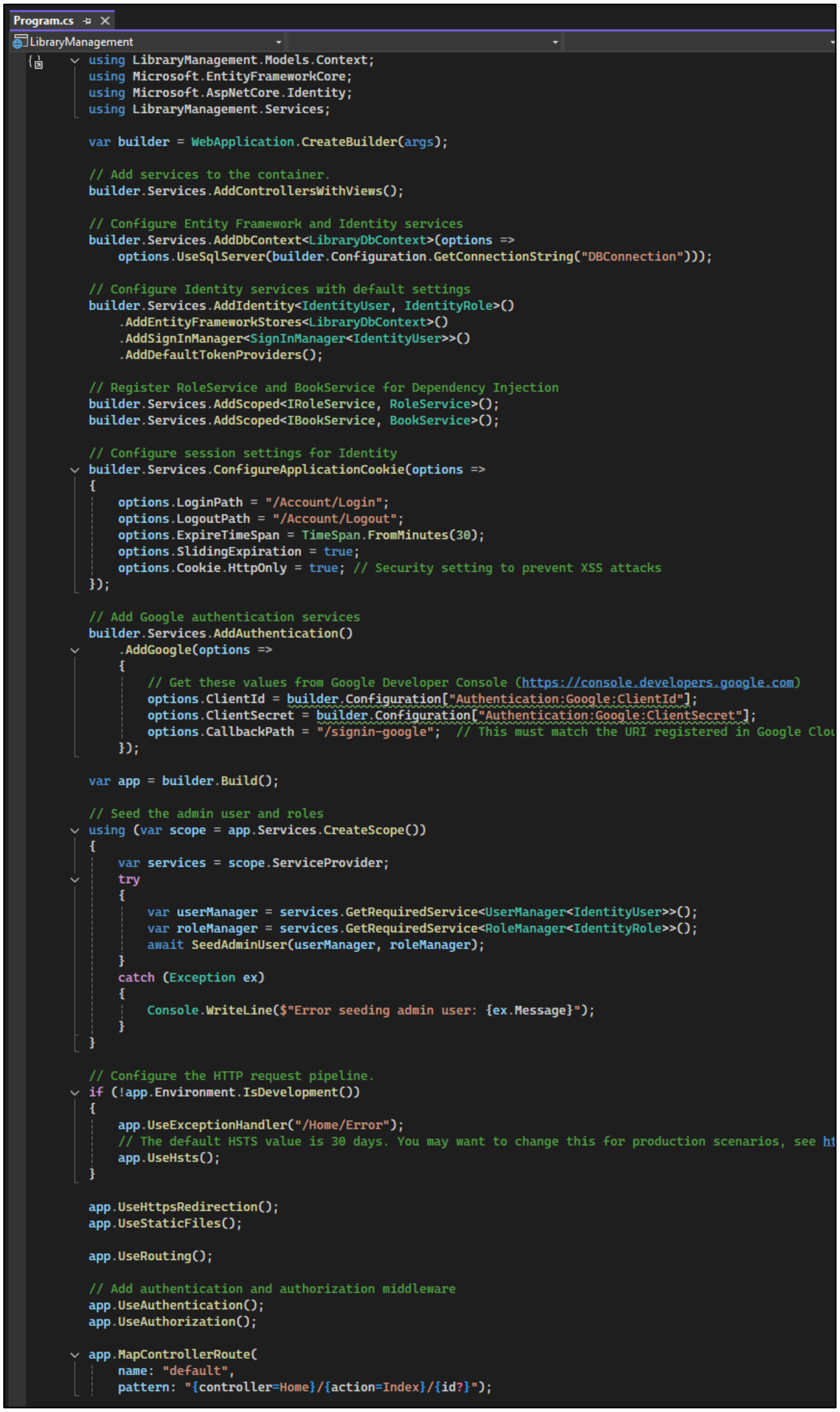
Package source mapping is off. Configure



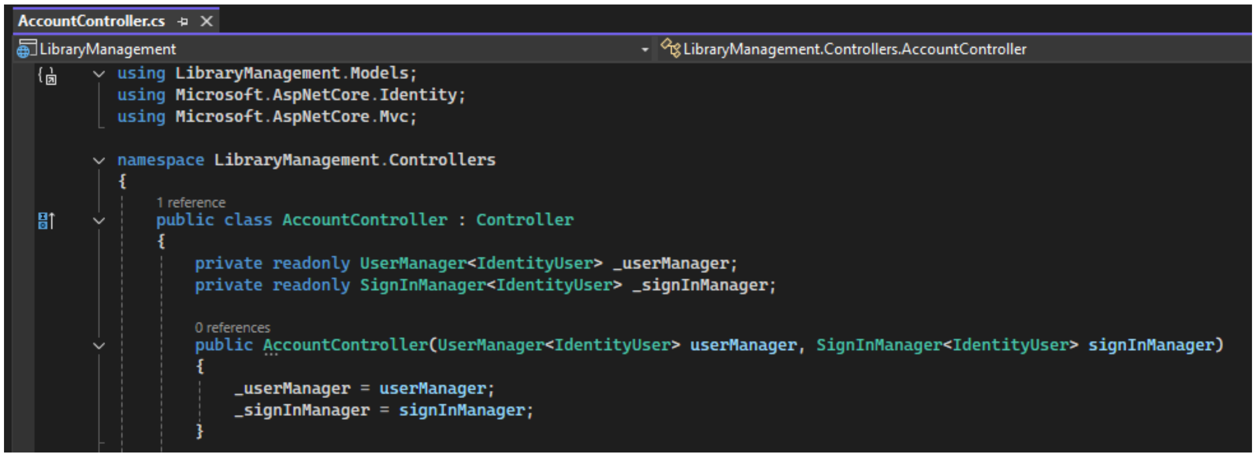
```
SELECT TOP (1000) [Id]
      ,[UserName]
      ,[NormalizedUserName]
      ,[Email]
      ,[NormalizedEmail]
      ,[EmailConfirmed]
      ,[PasswordHash]
      ,[SecurityStamp]
      ,[ConcurrencyStamp]
      ,[PhoneNumber]
      ,[PhoneNumberConfirmed]
      ,[TwoFactorEnabled]
      ,[LockoutEnd]
      ,[LockoutEnabled]
      ,[AccessFailedCount]
  FROM [CSE443].[dbo].[AspNetUsers]
```

	Id	UserName	NormalizedUserName	Email
1	294c6cf3-e3ef-4319-a362-35a86d4794b1	ductrong12072002@gmail.com	DUCTRONG12072002@GMAIL.COM	ductrong12072002@gmail.com
2	c493f191-cb06-4802-b52c-27b471612ddb	admin@gmail.com	ADMIN@GMAIL.COM	admin@gmail.com
3	d481b60c-ed9d-49e9-8280-24cf7367b3e5	duyen@gmail.com	DUYEN@GMAIL.COM	duyen@gmail.com

Query executed successfully.


Program.cs
LibraryManagement

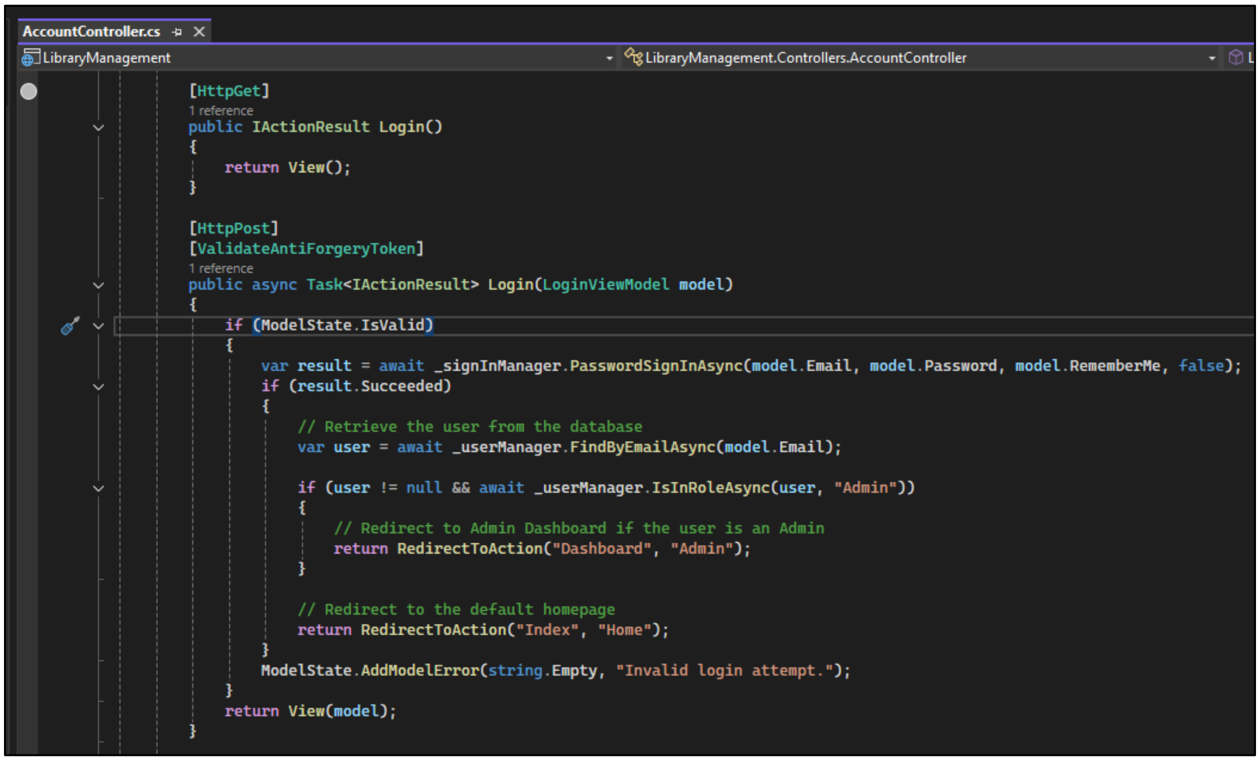
```
using LibraryManagement.Models.Context;
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.Identity;
using LibraryManagement.Services;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

// Configure Entity Framework and Identity services
builder.Services.AddDbContext<LibraryDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DBConnection")));

// Configure Identity services with default settings
builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddEntityFrameworkStores<LibraryDbContext>()
    .AddSignInManager<SignInManager<IdentityUser>>()
    .AddDefaultTokenProviders();

// Register RoleService and BookService for Dependency Injection
builder.Services.AddScoped<IRoleService, RoleService>();
builder.Services.AddScoped<IBookService, BookService>();

// Configure session settings for Identity
builder.Services.ConfigureApplicationCookie(options =>
{
    options.LoginPath = "/Account/Login";
    options.LogoutPath = "/Account/Logout";
    options.ExpireTimeSpan = TimeSpan.FromMinutes(30);
    options.SlidingExpiration = true;
    options.Cookie.HttpOnly = true; // Security setting to prevent XSS attacks
});

// Add Google authentication services
builder.Services.AddAuthentication()
    .AddGoogle(options =>
    {
        // Get these values from Google Developer Console (https://console.developers.google.com)
        options.ClientId = builder.Configuration["Authentication:Google:ClientId"];
        options.ClientSecret = builder.Configuration["Authentication:Google:ClientSecret"];
        options.CallbackPath = "/signin-google";  // This must match the URI registered in Google Clou
    });

var app = builder.Build();

// Seed the admin user and roles
using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    try
    {
        var userManager = services.GetRequiredService<UserManager<IdentityUser>>();
        var roleManager = services.GetRequiredService<RoleManager<IdentityRole>>();
        await SeedAdminUser(userManager, roleManager);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error seeding admin user: {ex.Message}");
    }
}

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see ht
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

// Add authentication and authorization middleware
app.UseAuthentication();
app.UseAuthorization();

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



AccountController.cs

LibraryManagement | LibraryManagement.Controllers.AccountController

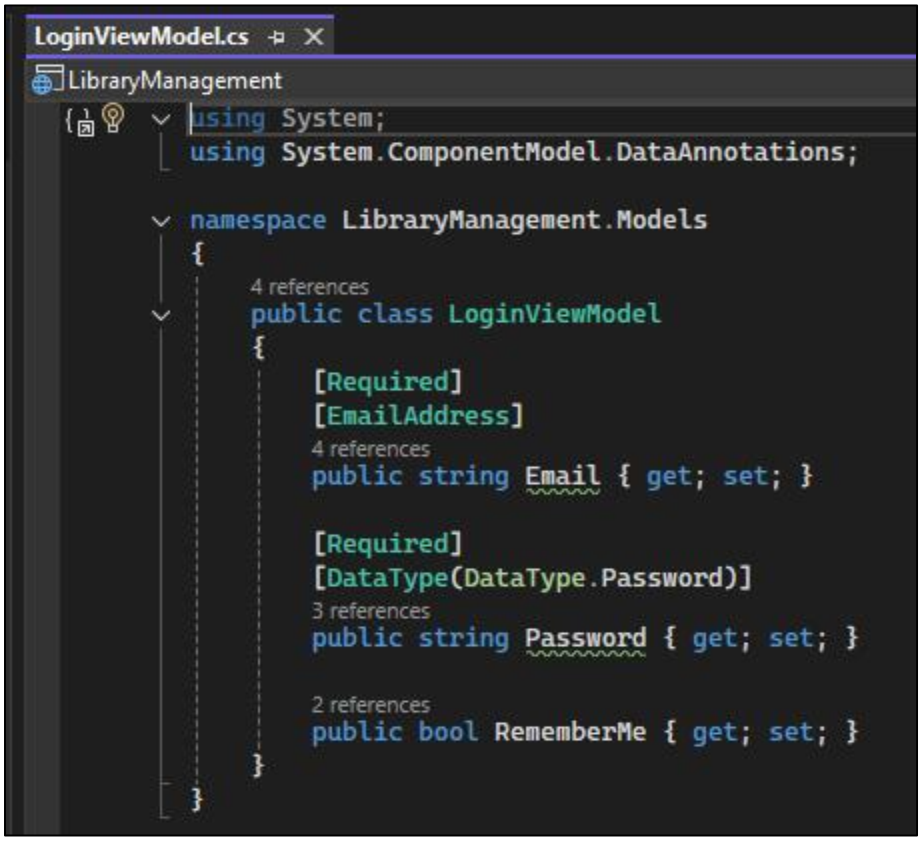
```
using LibraryManagement.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;

namespace LibraryManagement.Controllers
{
    1 reference
    public class AccountController : Controller
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly SignInManager<IdentityUser> _signInManager;

        0 references
        public AccountController(UserManager<IdentityUser> userManager, SignInManager<IdentityUser> signInManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
        }
```



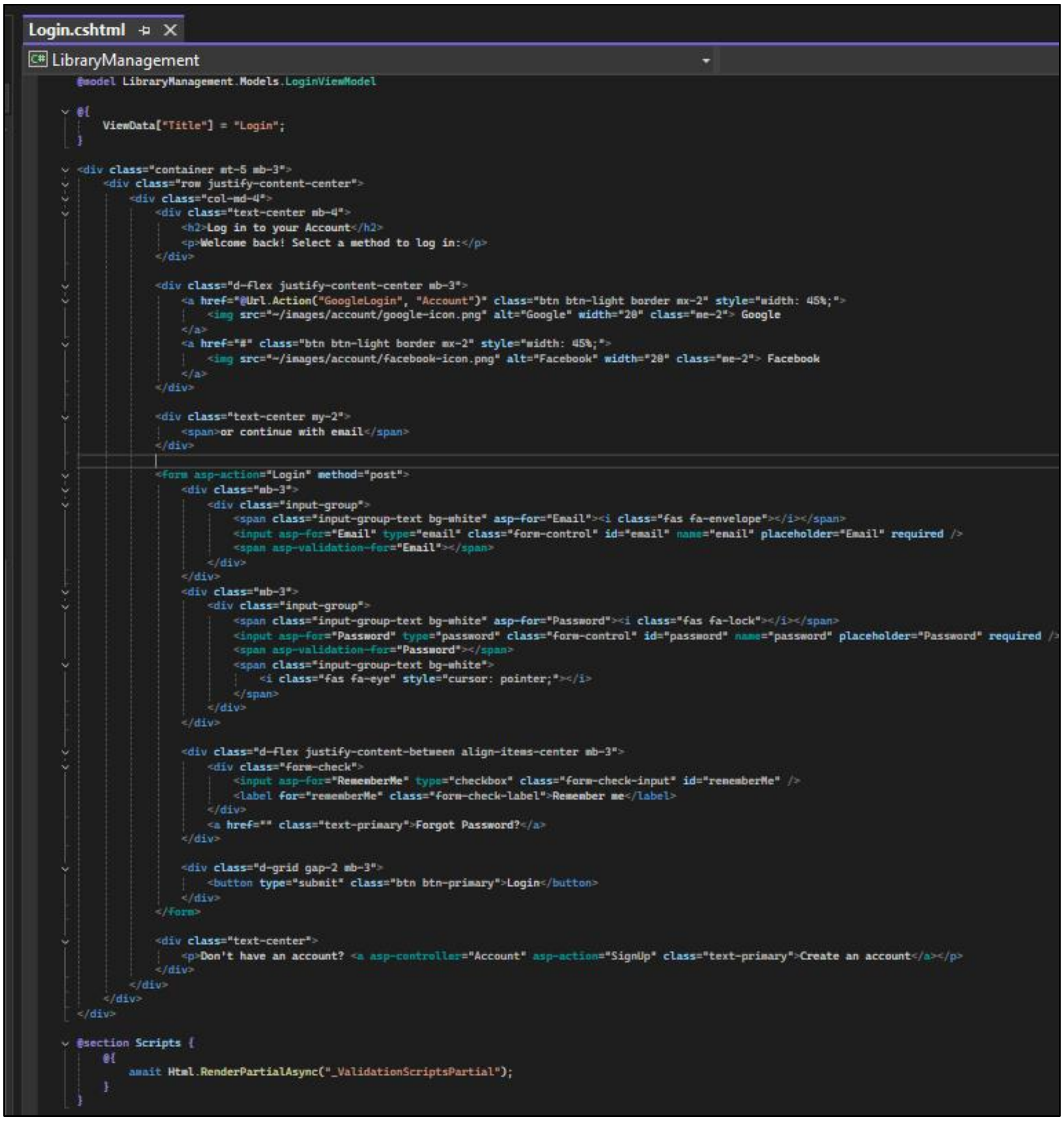
AccountController.cs

LibraryManagement | LibraryManagement.Controllers.AccountController

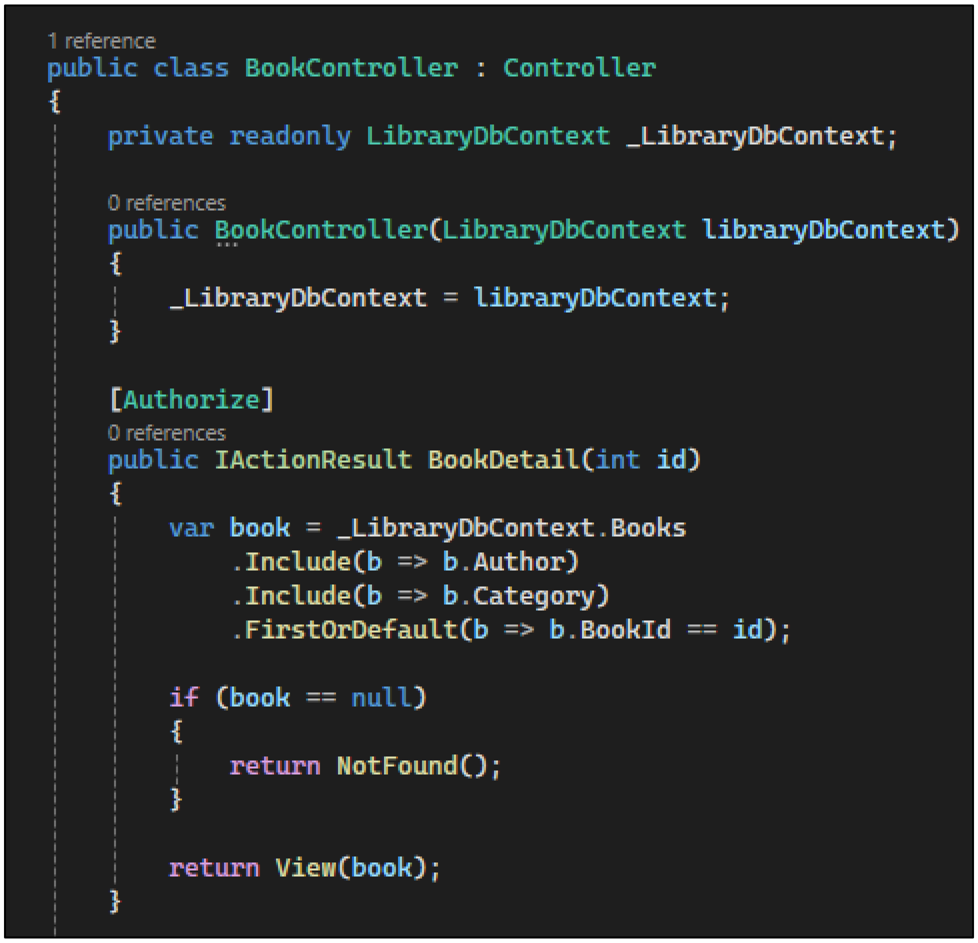
```
        [HttpGet]
        1 reference
        public IActionResult Login()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        1 reference
        public async Task<IActionResult> Login(LoginViewModel model)
        {
            if (ModelState.IsValid)
            {
                var result = await _signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);
                if (result.Succeeded)
                {
                    // Retrieve the user from the database
                    var user = await _userManager.FindByEmailAsync(model.Email);

                    if (user != null && await _userManager.IsInRoleAsync(user, "Admin"))
                    {
                        // Redirect to Admin Dashboard if the user is an Admin
                        return RedirectToAction("Dashboard", "Admin");
                    }

                    // Redirect to the default homepage
                    return RedirectToAction("Index", "Home");
                }
                ModelState.AddModelError(string.Empty, "Invalid login attempt.");
            }
            return View(model);
        }
```



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

namespace LibraryManagement.Models
{
    4 references
    public class LoginViewModel
    {
        [Required]
        [EmailAddress]
        4 references
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        3 references
        public string Password { get; set; }

        2 references
        public bool RememberMe { get; set; }
    }
}
```

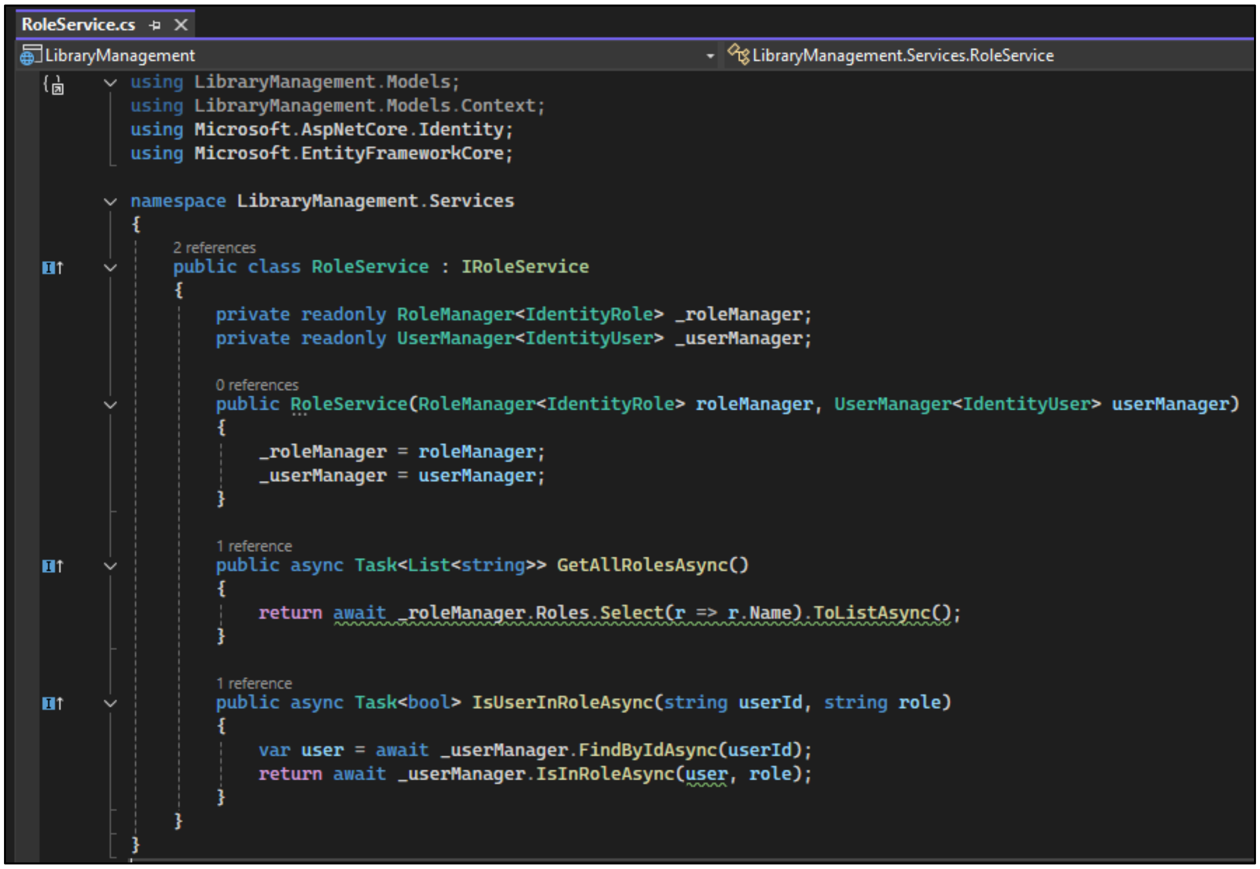


```
@model LibraryManagement.Models.LoginViewModel

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

<div class="container mt-5 mb-3">
    <div class="row justify-content-center">
        <div class="col-md-4">
            <div class="text-center mb-4">
                <h2>Log in to your Account</h2>
                <p>Welcome back! Select a method to log in:</p>
            </div>

            <div class="d-flex justify-content-center mb-3">
                <a href="@Url.Action("GoogleLogin", "Account")" class="btn btn-light border mx-2" style="width: 45%;">
                    <img src="~/images/account/google-icon.png" alt="Google" width="20" class="me-2"> Google
                </a>
                <a href="#" class="btn btn-light border mx-2" style="width: 45%;">
                    <img src="~/images/account/facebook-icon.png" alt="Facebook" width="20" class="me-2"> Facebook
                </a>
            </div>

            <div class="text-center my-2">
                <span>or continue with email</span>
            </div>

            <form asp-action="Login" method="post">
                <div class="mb-3">
                    <div class="input-group">
                        <span class="input-group-text bg-white" asp-for="Email"><i class="fas fa-envelope"></i></span>
                        <input asp-for="Email" type="email" class="form-control" id="email" name="email" placeholder="Email" required />
                        <span asp-validation-for="Email"></span>
                    </div>
                </div>
                <div class="mb-3">
                    <div class="input-group">
                        <span class="input-group-text bg-white" asp-for="Password"><i class="fas fa-lock"></i></span>
                        <input asp-for="Password" type="password" class="form-control" id="password" name="password" placeholder="Password" required /
                        <span asp-validation-for="Password"></span>
                        <span class="input-group-text bg-white">
                            <i class="fas fa-eye" style="cursor: pointer;"></i>
                        </span>
                    </div>
                </div>

                <div class="d-flex justify-content-between align-items-center mb-3">
                    <div class="form-check">
                        <input asp-for="RememberMe" type="checkbox" class="form-check-input" id="rememberMe" />
                        <label for="rememberMe" class="form-check-label">Remember me</label>
                    </div>
                    <a href="" class="text-primary">Forgot Password?</a>
                </div>

                <div class="d-grid gap-2 mb-3">
                    <button type="submit" class="btn btn-primary">Login</button>
                </div>
            </form>

            <div class="text-center">
                <p>Don't have an account? <a asp-controller="Account" asp-action="SignUp" class="text-primary">Create an account</a></p>
            </div>
        </div>
    </div>
</div>

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



```
1 reference
public class BookController : Controller
{
    private readonly LibraryDbContext _LibraryDbContext;

    0 references
    public BookController(LibraryDbContext libraryDbContext)
    {
        _LibraryDbContext = libraryDbContext;
    }

    [Authorize]
    0 references
    public IActionResult BookDetail(int id)
    {
        var book = _LibraryDbContext.Books
            .Include(b => b.Author)
            .Include(b => b.Category)
            .FirstOrDefault(b => b.BookId == id);

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

        return View(book);
    }
```

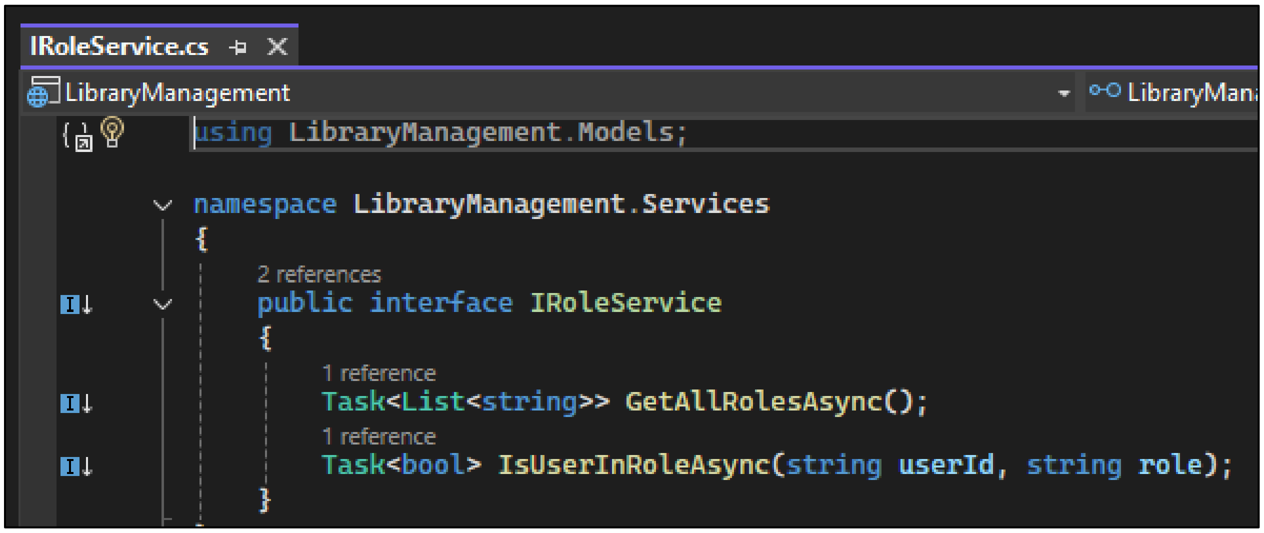
Exercise 2: Implement Role-Based Authorization for Admin Access in MVC



RoleService.cs

LibraryManagement | LibraryManagement.Services.RoleService

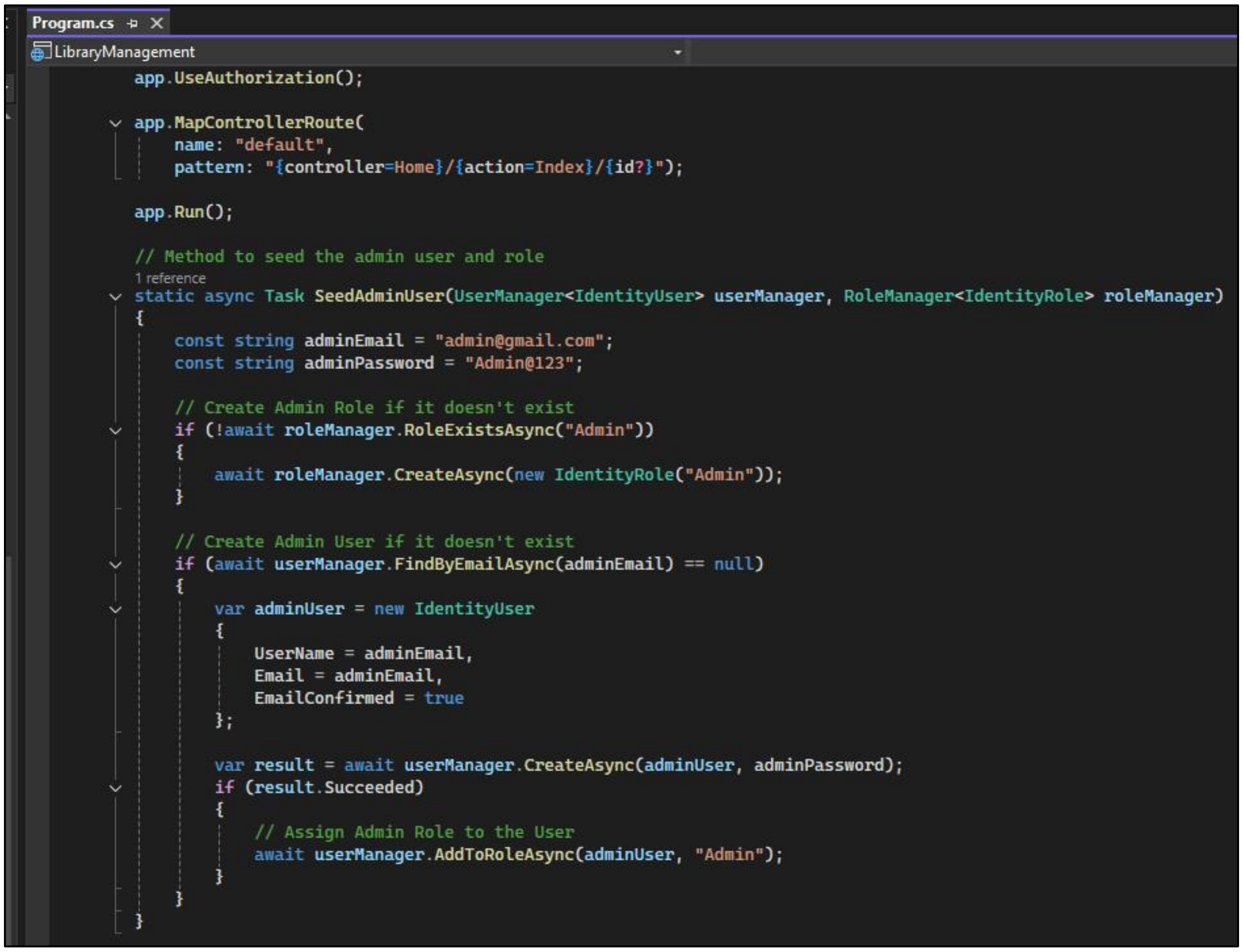
```
using LibraryManagement.Models;
using LibraryManagement.Models.Context;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;

namespace LibraryManagement.Services
{
    2 references
    public class RoleService : IRoleService
    {
        private readonly RoleManager<IdentityRole> _roleManager;
        private readonly UserManager<IdentityUser> _userManager;

        0 references
        public RoleService(RoleManager<IdentityRole> roleManager, UserManager<IdentityUser> userManager)
        {
            _roleManager = roleManager;
            _userManager = userManager;
        }

        1 reference
        public async Task<List<string>> GetAllRolesAsync()
        {
            return await _roleManager.Roles.Select(r => r.Name).ToListAsync();
        }

        1 reference
        public async Task<bool> IsUserInRoleAsync(string userId, string role)
        {
            var user = await _userManager.FindByIdAsync(userId);
            return await _userManager.IsInRoleAsync(user, role);
        }
    }
}
```



IRoleService.cs

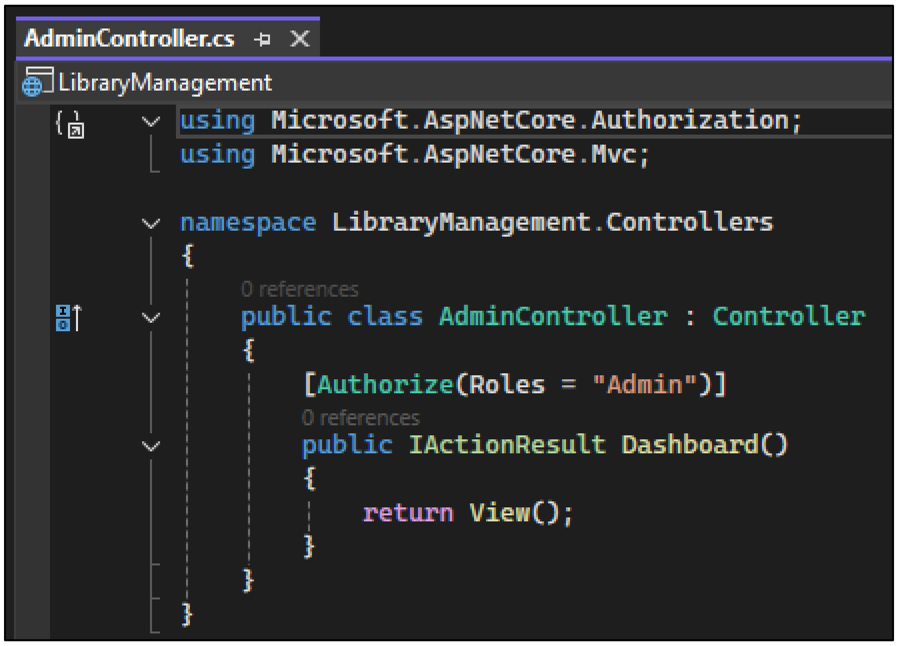
LibraryManagement | LibraryMana

```
using LibraryManagement.Models;

namespace LibraryManagement.Services
{
    2 references
    public interface IRoleService
    {
        1 reference
        Task<List<string>> GetAllRolesAsync();
        1 reference
        Task<bool> IsUserInRoleAsync(string userId, string role);
    }
```


Program.cs

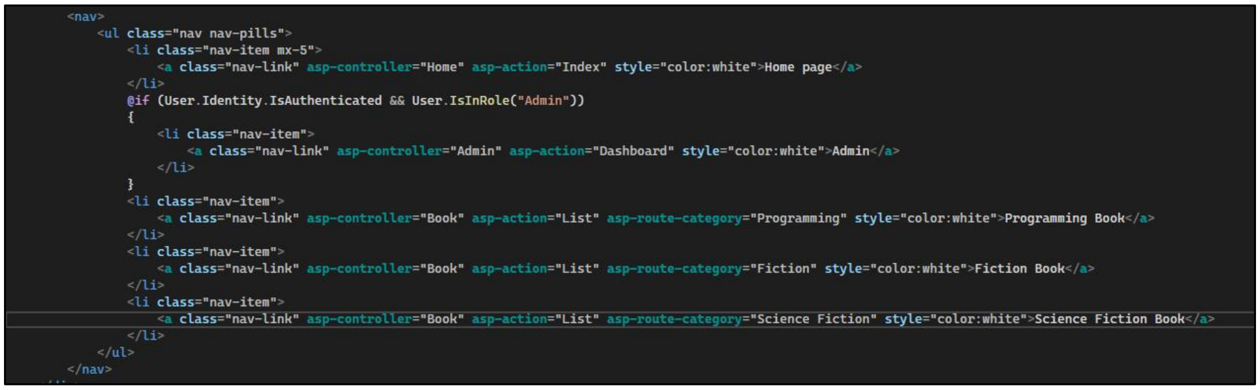
LibraryManagement

```
app.UseAuthorization();

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

app.Run();

// Method to seed the admin user and role
1 reference
static async Task SeedAdminUser(UserManager<IdentityUser> userManager, RoleManager<IdentityRole> roleManager)
{
    const string adminEmail = "admin@gmail.com";
    const string adminPassword = "Admin@123";

    // Create Admin Role if it doesn't exist
    if (!await roleManager.RoleExistsAsync("Admin"))
    {
        await roleManager.CreateAsync(new IdentityRole("Admin"));
    }

    // Create Admin User if it doesn't exist
    if (await userManager.FindByEmailAsync(adminEmail) == null)
    {
        var adminUser = new IdentityUser
        {
            UserName = adminEmail,
            Email = adminEmail,
            EmailConfirmed = true
        };

        var result = await userManager.CreateAsync(adminUser, adminPassword);
        if (result.Succeeded)
        {
            // Assign Admin Role to the User
            await userManager.AddToRoleAsync(adminUser, "Admin");
        }
    }
}
```

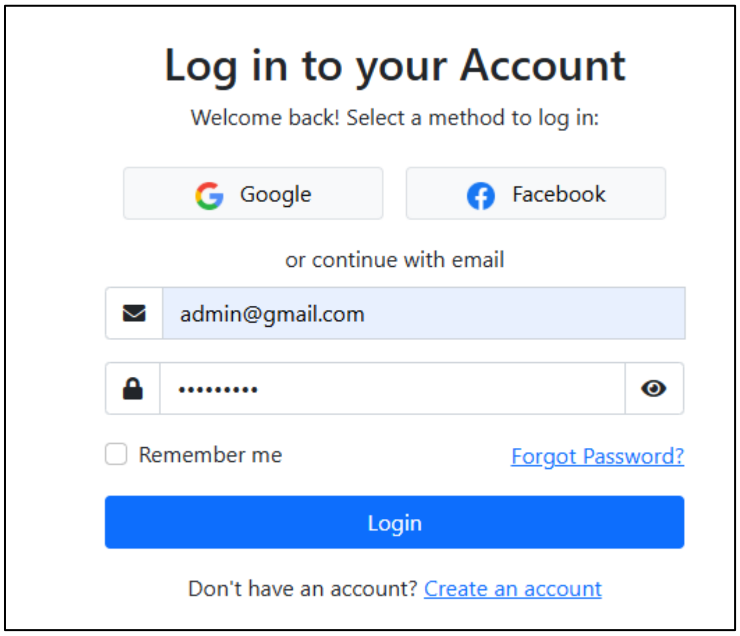

AdminController.cs

LibraryManagement

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace LibraryManagement.Controllers
{
    0 references
    public class AdminController : Controller
    {
        [Authorize(Roles = "Admin")]
        0 references
        public IActionResult Dashboard()
        {
            return View();
        }
    }
}
```



```
<nav>
    <ul class="nav nav-pills">
        <li class="nav-item mx-5">
            <a class="nav-link" asp-controller="Home" asp-action="Index" style="color:white">Home page</a>
        </li>
        @if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))
        {
            <li class="nav-item">
                <a class="nav-link" asp-controller="Admin" asp-action="Dashboard" style="color:white">Admin</a>
            </li>
        }
        <li class="nav-item">
            <a class="nav-link" asp-controller="Book" asp-action="List" asp-route-category="Programming" style="color:white">Programming Book</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" asp-controller="Book" asp-action="List" asp-route-category="Fiction" style="color:white">Fiction Book</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" asp-controller="Book" asp-action="List" asp-route-category="Science Fiction" style="color:white">Science Fiction Book</a>
        </li>
    </ul>
</nav>
```



Log in to your Account

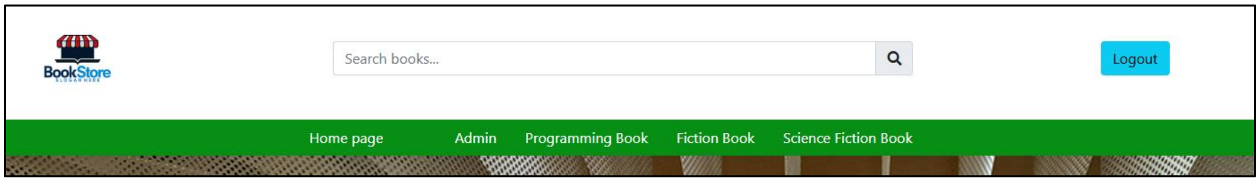
Welcome back! Select a method to log in:

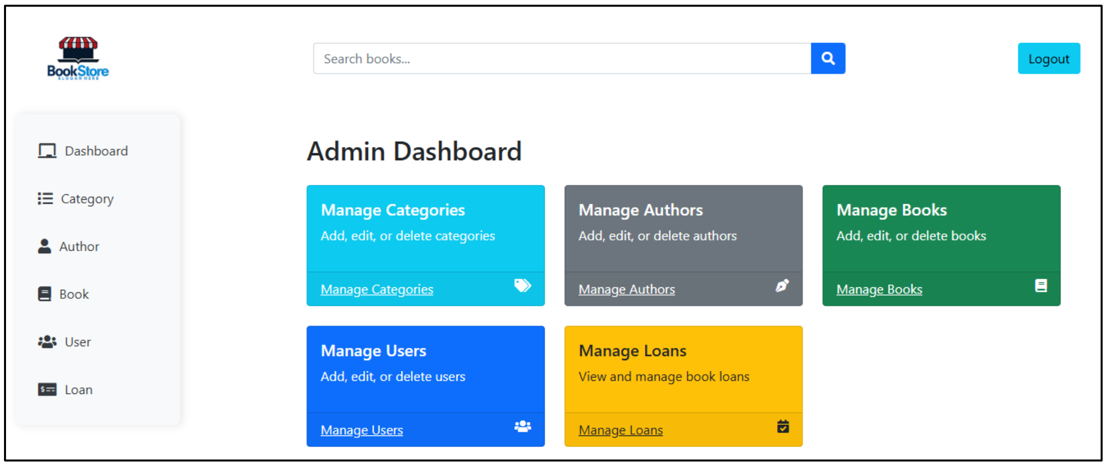
or continue with email

admin@gmail.com

•••••••••

Don't have an account? Create an account

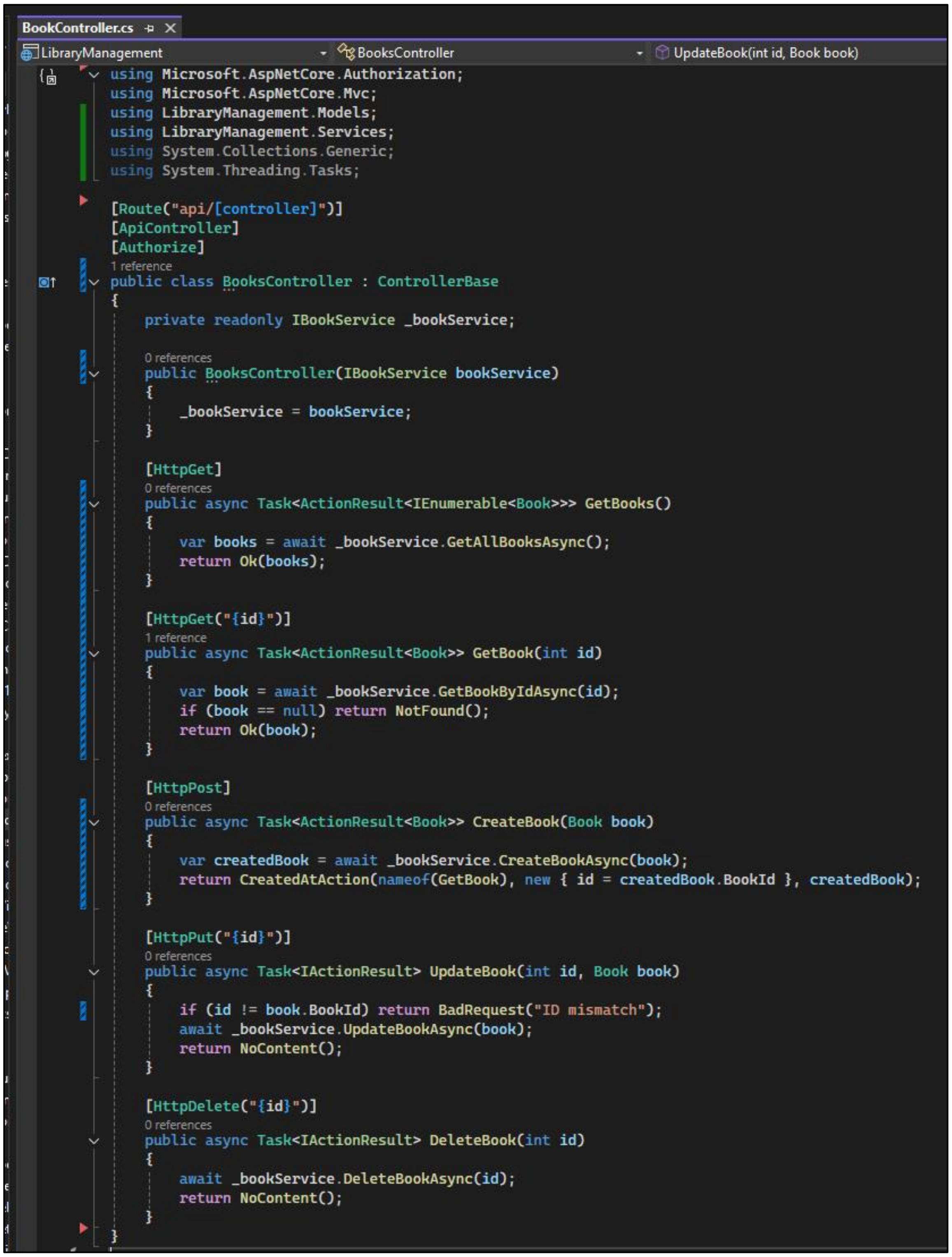




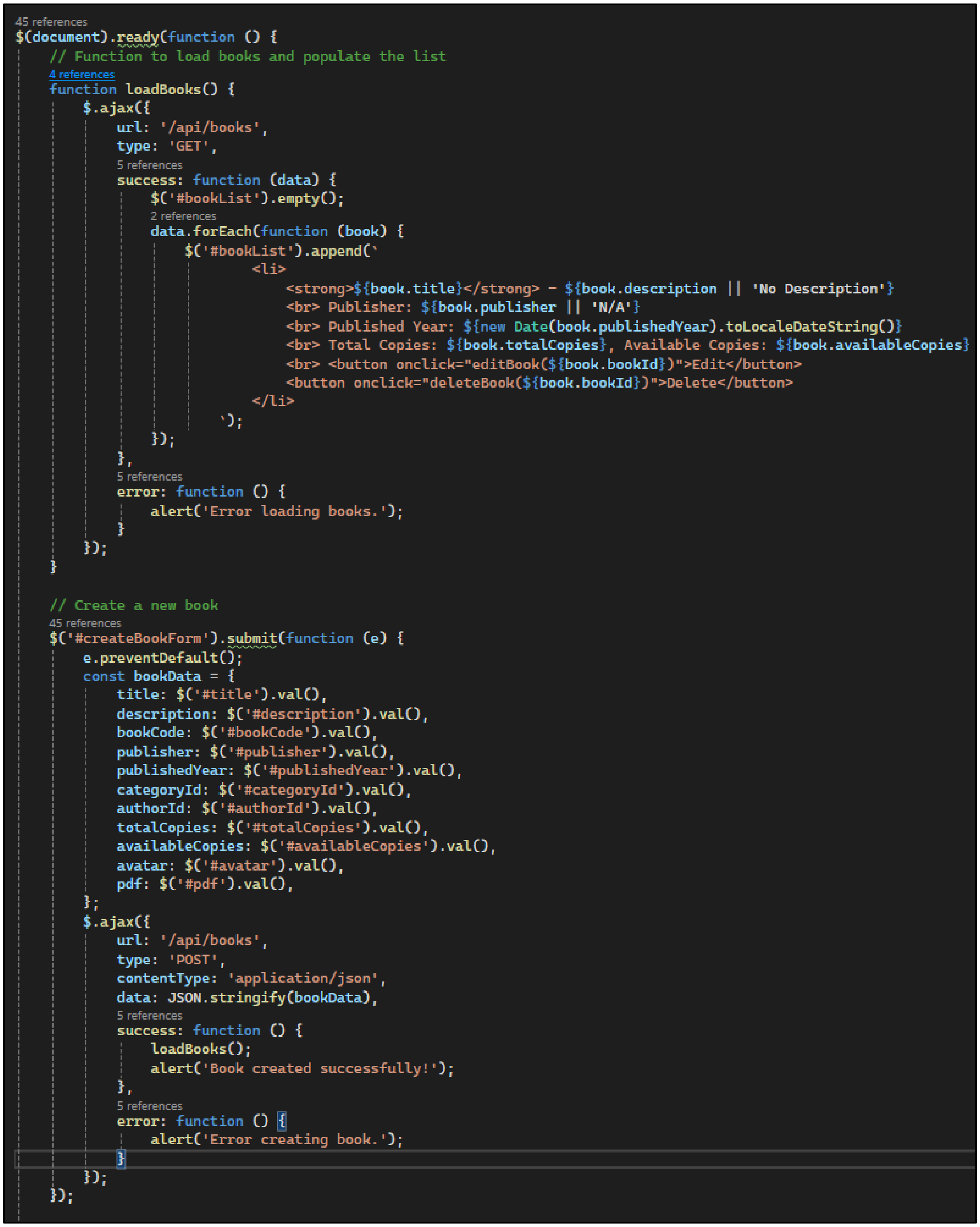
Admin Dashboard

Manage Categories	Add, edit, or delete categories
Manage Authors	Add, edit, or delete authors
Manage Books	Add, edit, or delete books
Manage Users	Add, edit, or delete users
Manage Loans	View and manage book loans

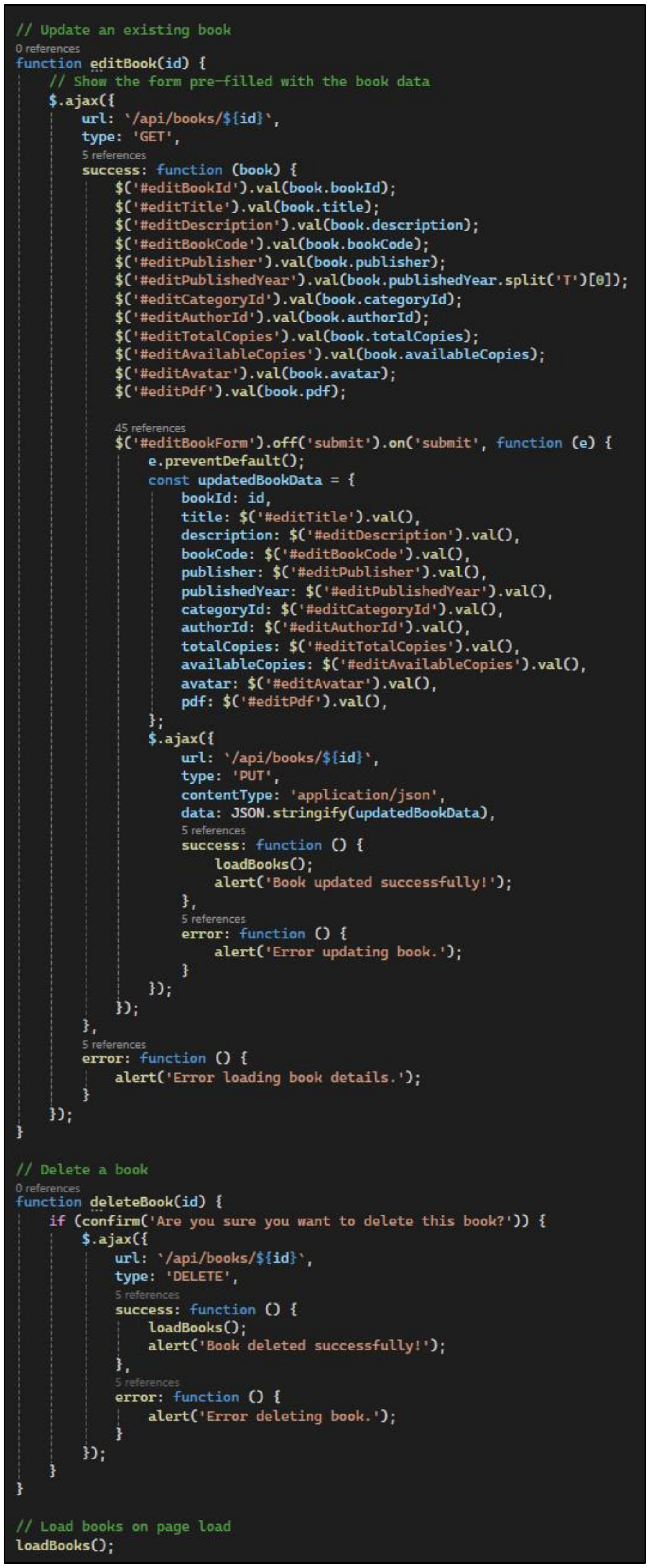
Exercise 3: Create API Endpoints for Book Management and Use AJAX to Interact with them



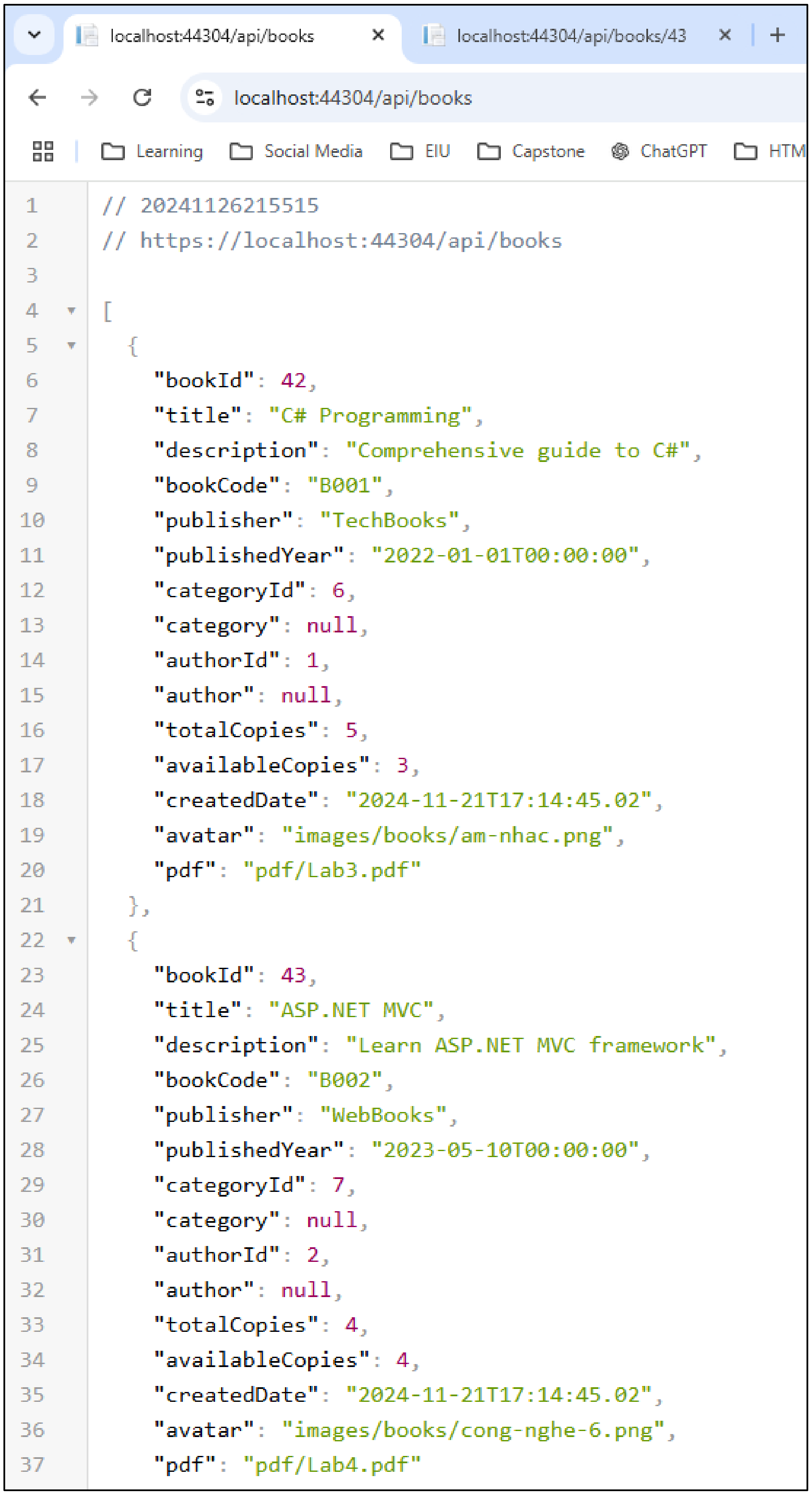
```csharp
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using LibraryManagement.Models;
using LibraryManagement.Services;
using System.Collections.Generic;
using System.Threading.Tasks;

[Route("api/[controller]")]
[ApiController]
[Authorize]
1 reference
public class BooksController : ControllerBase
{
    private readonly IBookService _bookService;

    0 references
    public BooksController(IBookService bookService)
    {
        _bookService = bookService;
    }

    [HttpGet]
    0 references
    public async Task<ActionResult<IEnumerable<Book>>> GetBooks()
    {
        var books = await _bookService.GetAllBooksAsync();
        return Ok(books);
    }

    [HttpGet("{id}")]
    1 reference
    public async Task<ActionResult<Book>> GetBook(int id)
    {
        var book = await _bookService.GetBookByIdAsync(id);
        if (book == null) return NotFound();
        return Ok(book);
    }

    [HttpPost]
    0 references
    public async Task<ActionResult<Book>> CreateBook(Book book)
    {
        var createdBook = await _bookService.CreateBookAsync(book);
        return CreatedAtAction(nameof(GetBook), new { id = createdBook.BookId }, createdBook);
    }

    [HttpPut("{id}")]
    0 references
    public async Task<IActionResult> UpdateBook(int id, Book book)
    {
        if (id != book.BookId) return BadRequest("ID mismatch");
        await _bookService.UpdateBookAsync(book);
        return NoContent();
    }

    [HttpDelete("{id}")]
    0 references
    public async Task<IActionResult> DeleteBook(int id)
    {
        await _bookService.DeleteBookAsync(id);
        return NoContent();
    }
}
```



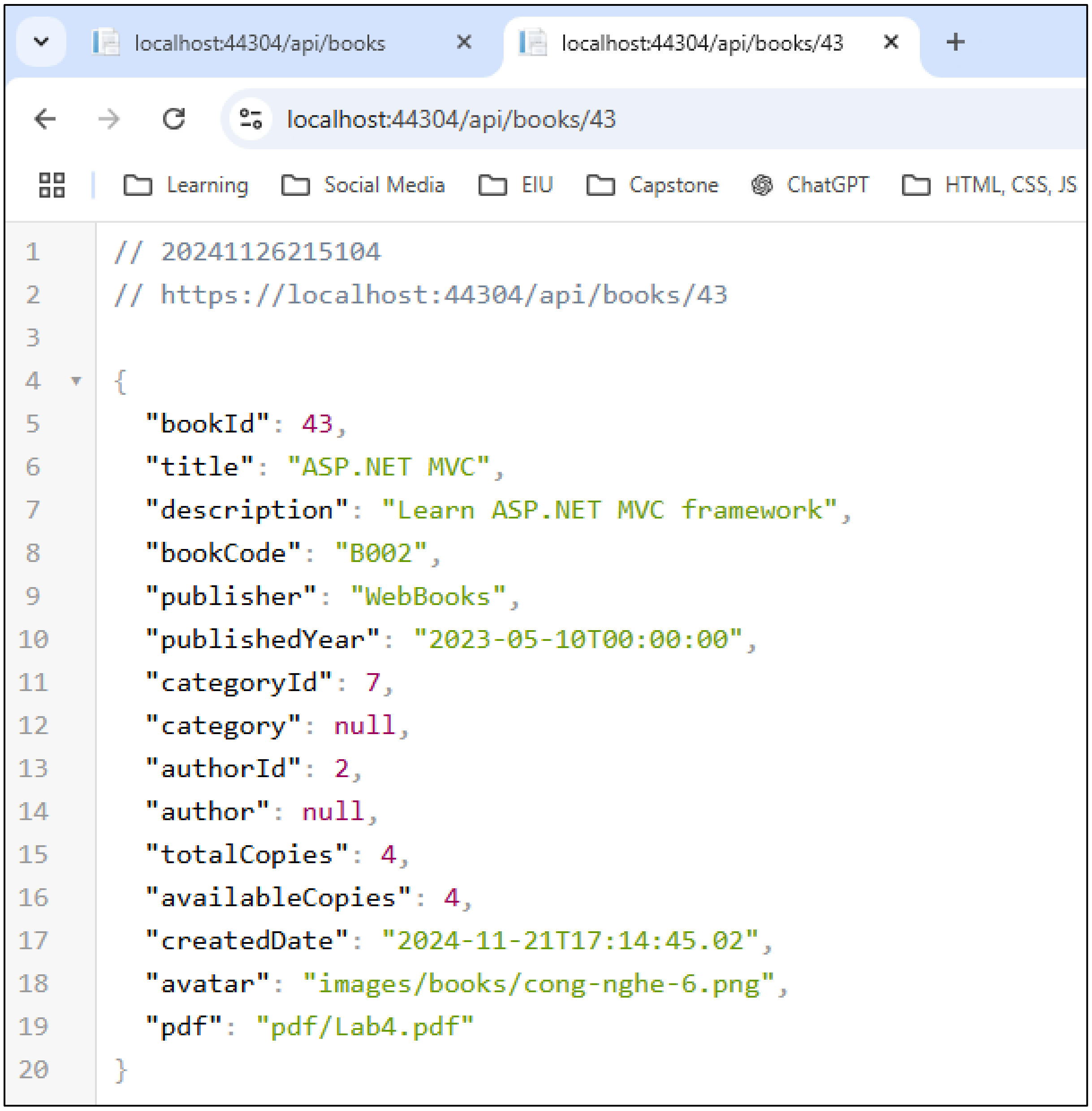
```
45 references
$(document).ready(function () {
    // Function to load books and populate the list
    4 references
    function loadBooks() {
        $.ajax({
            url: '/api/books',
            type: 'GET',
            5 references
            success: function (data) {
                $('#bookList').empty();
                2 references
                data.forEach(function (book) {
                    $('#bookList').append(`
                            <li>
                                <strong>${book.title}</strong> - ${book.description || 'No Description'}
                                <br> Publisher: ${book.publisher || 'N/A'}
                                <br> Published Year: ${new Date(book.publishedYear).toLocaleDateString()}
                                <br> Total Copies: ${book.totalCopies}, Available Copies: ${book.availableCopies}
                                <br> <button onclick="editBook(${book.bookId})">Edit</button>
                                <button onclick="deleteBook(${book.bookId})">Delete</button>
                            </li>
                        `);
                });
            },
            5 references
            error: function () {
                alert('Error loading books.');
            }
        });
    }

    // Create a new book
    45 references
    $('#createBookForm').submit(function (e) {
        e.preventDefault();
        const bookData = {
            title: $('#title').val(),
            description: $('#description').val(),
            bookCode: $('#bookCode').val(),
            publisher: $('#publisher').val(),
            publishedYear: $('#publishedYear').val(),
            categoryId: $('#categoryId').val(),
            authorId: $('#authorId').val(),
            totalCopies: $('#totalCopies').val(),
            availableCopies: $('#availableCopies').val(),
            avatar: $('#avatar').val(),
            pdf: $('#pdf').val(),
        };
        $.ajax({
            url: '/api/books',
            type: 'POST',
            contentType: 'application/json',
            data: JSON.stringify(bookData),
            5 references
            success: function () {
                loadBooks();
                alert('Book created successfully!');
            },
            5 references
            error: function () {
                alert('Error creating book.');
            }
        });
    });
```



```
// Update an existing book
0 references
function editBook(id) {
    // Show the form pre-filled with the book data
    $.ajax({
        url: `/api/books/${id}`,
        type: 'GET',
        5 references
        success: function (book) {
            $('#editBookId').val(book.bookId);
            $('#editTitle').val(book.title);
            $('#editDescription').val(book.description);
            $('#editBookCode').val(book.bookCode);
            $('#editPublisher').val(book.publisher);
            $('#editPublishedYear').val(book.publishedYear.split('T')[0]);
            $('#editCategoryId').val(book.categoryId);
            $('#editAuthorId').val(book.authorId);
            $('#editTotalCopies').val(book.totalCopies);
            $('#editAvailableCopies').val(book.availableCopies);
            $('#editAvatar').val(book.avatar);
            $('#editPdf').val(book.pdf);

            45 references
            $('#editBookForm').off('submit').on('submit', function (e) {
                e.preventDefault();
                const updatedBookData = {
                    bookId: id,
                    title: $('#editTitle').val(),
                    description: $('#editDescription').val(),
                    bookCode: $('#editBookCode').val(),
                    publisher: $('#editPublisher').val(),
                    publishedYear: $('#editPublishedYear').val(),
                    categoryId: $('#editCategoryId').val(),
                    authorId: $('#editAuthorId').val(),
                    totalCopies: $('#editTotalCopies').val(),
                    availableCopies: $('#editAvailableCopies').val(),
                    avatar: $('#editAvatar').val(),
                    pdf: $('#editPdf').val(),
                };
                $.ajax({
                    url: `/api/books/${id}`,
                    type: 'PUT',
                    contentType: 'application/json',
                    data: JSON.stringify(updatedBookData),
                    5 references
                    success: function () {
                        loadBooks();
                        alert('Book updated successfully!');
                    },
                    5 references
                    error: function () {
                        alert('Error updating book.');
                    }
                });
            });
        },
        5 references
        error: function () {
            alert('Error loading book details.');
        }
    });
}

// Delete a book
0 references
function deleteBook(id) {
    if (confirm('Are you sure you want to delete this book?')) {
        $.ajax({
            url: `/api/books/${id}`,
            type: 'DELETE',
            5 references
            success: function () {
                loadBooks();
                alert('Book deleted successfully!');
            },
            5 references
            error: function () {
                alert('Error deleting book.');
            }
        });
    }
}

// Load books on page load
loadBooks();
```

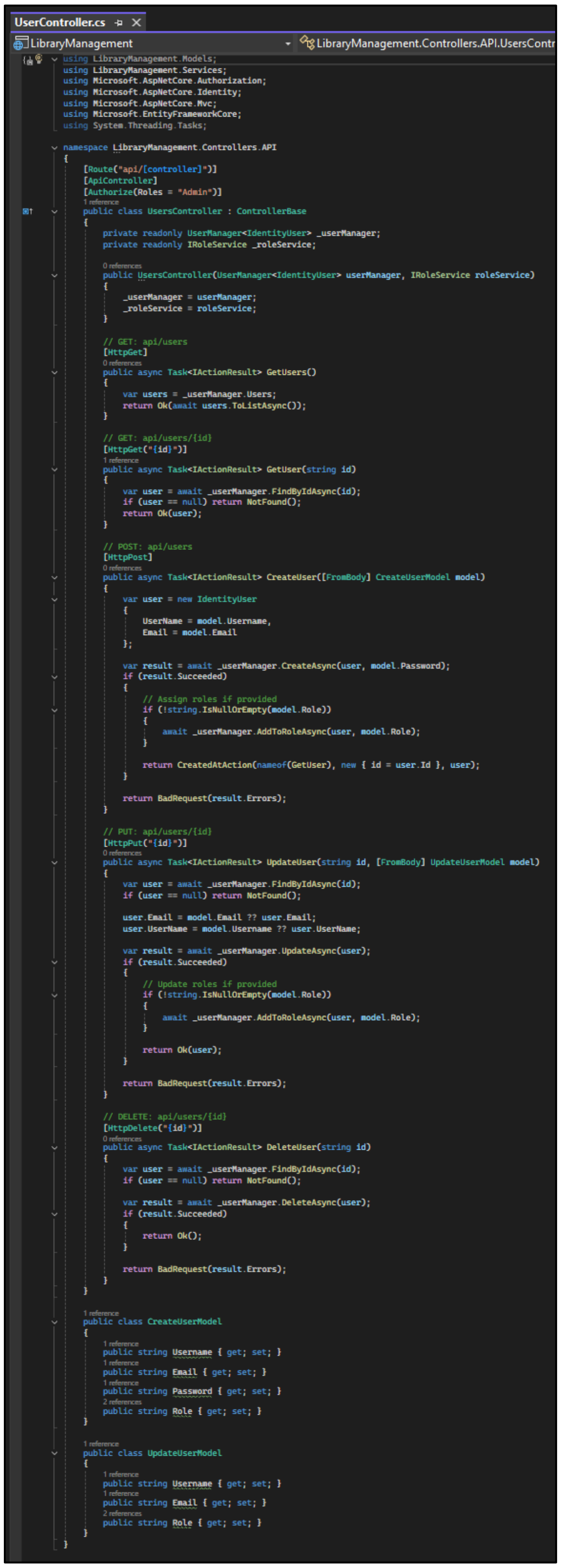

localhost:44304/api/books
localhost:44304/api/books/43
localhost:44304/api/books
Learning
Social Media
EIU
Capstone
ChatGPT
HTM

```
// 20241126215515
// https://localhost:44304/api/books

[
  {
    "bookId": 42,
    "title": "C# Programming",
    "description": "Comprehensive guide to C#",
    "bookCode": "B001",
    "publisher": "TechBooks",
    "publishedYear": "2022-01-01T00:00:00",
    "categoryId": 6,
    "category": null,
    "authorId": 1,
    "author": null,
    "totalCopies": 5,
    "availableCopies": 3,
    "createdDate": "2024-11-21T17:14:45.02",
    "avatar": "images/books/am-nhac.png",
    "pdf": "pdf/Lab3.pdf"
  },
  {
    "bookId": 43,
    "title": "ASP.NET MVC",
    "description": "Learn ASP.NET MVC framework",
    "bookCode": "B002",
    "publisher": "WebBooks",
    "publishedYear": "2023-05-10T00:00:00",
    "categoryId": 7,
    "category": null,
    "authorId": 2,
    "author": null,
    "totalCopies": 4,
    "availableCopies": 4,
    "createdDate": "2024-11-21T17:14:45.02",
    "avatar": "images/books/cong-nghe-6.png",
    "pdf": "pdf/Lab4.pdf"
```

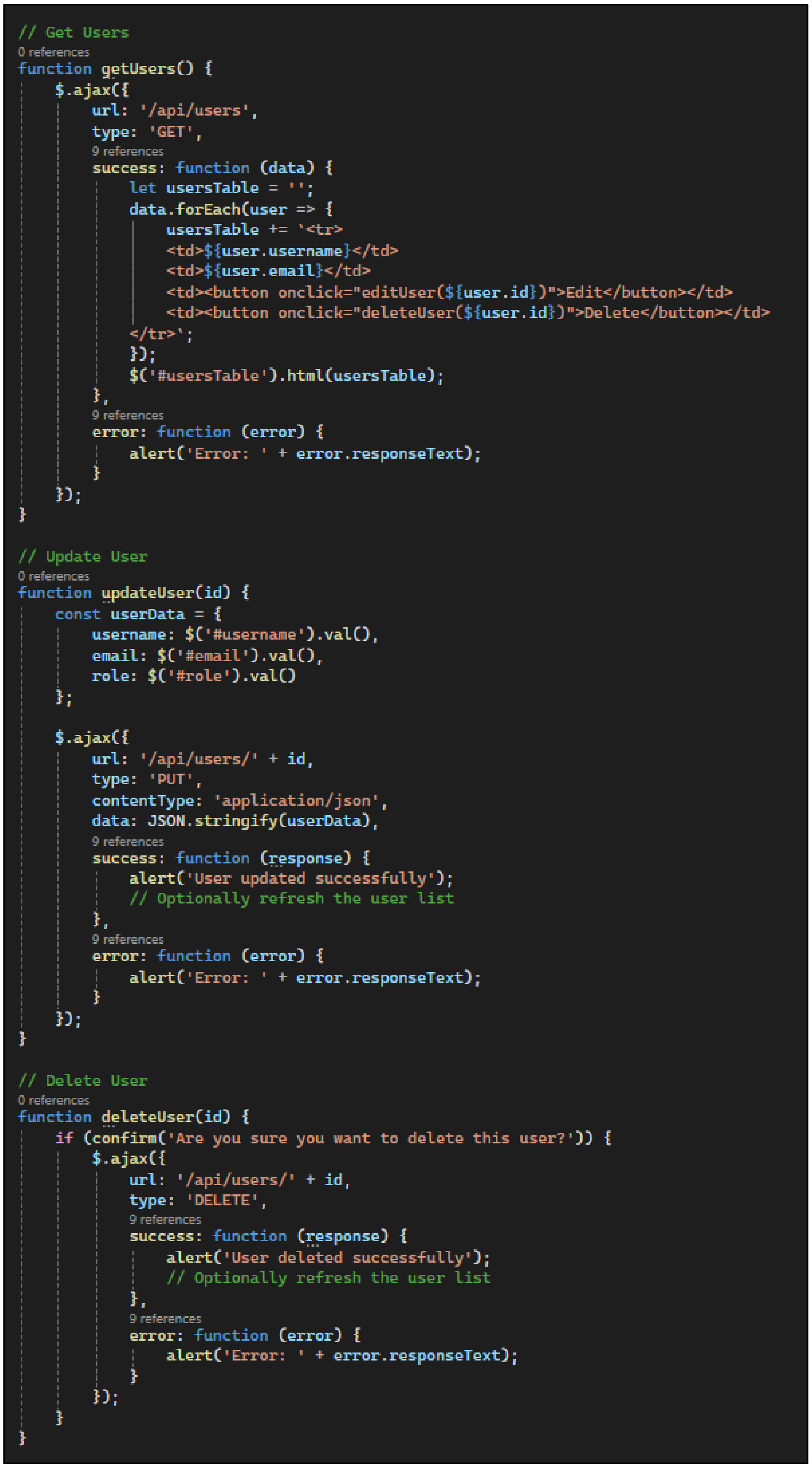

localhost:44304/api/books
localhost:44304/api/books/43
localhost:44304/api/books/43
Learning
Social Media
EIU
Capstone
ChatGPT
HTML, CSS, JS

```
// 20241126215104
// https://localhost:44304/api/books/43

{
  "bookId": 43,
  "title": "ASP.NET MVC",
  "description": "Learn ASP.NET MVC framework",
  "bookCode": "B002",
  "publisher": "WebBooks",
  "publishedYear": "2023-05-10T00:00:00",
  "categoryId": 7,
  "category": null,
  "authorId": 2,
  "author": null,
  "totalCopies": 4,
  "availableCopies": 4,
  "createdDate": "2024-11-21T17:14:45.02",
  "avatar": "images/books/cong-nghe-6.png",
  "pdf": "pdf/Lab4.pdf"
}
```

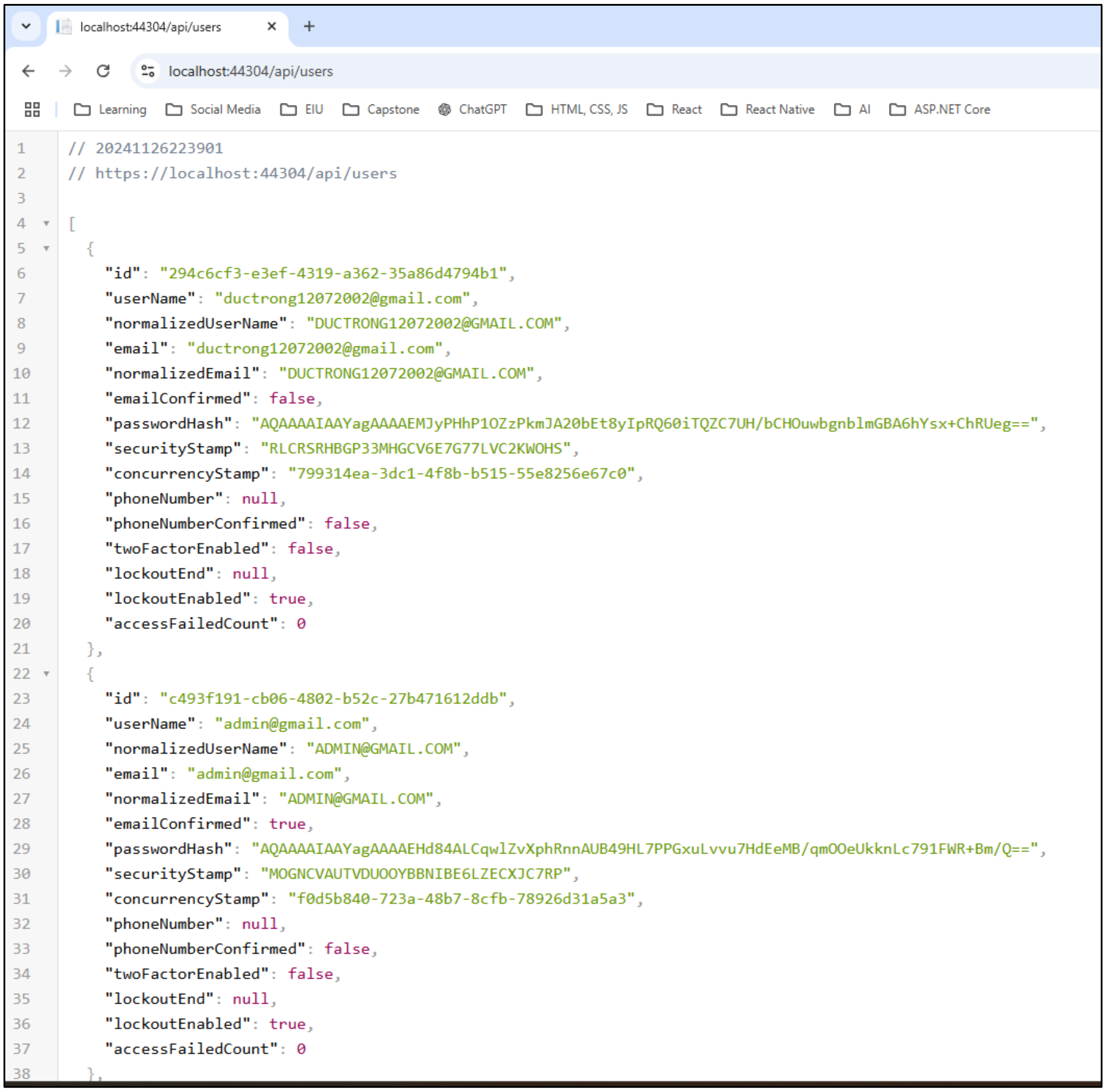
Exercise 4: Implement API Endpoints and AJAX CRUD for User Management with RoleBased Authorization



```csharp
using LibraryManagement.Models;
using LibraryManagement.Services;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Threading.Tasks;

namespace LibraryManagement.Controllers.API
{
    [Route("api/[controller]")]
    [ApiController]
    [Authorize(Roles = "Admin")]
    1 reference
    public class UsersController : ControllerBase
    {
        private readonly UserManager<IdentityUser> _userManager;
        private readonly IRoleService _roleService;

        0 references
        public UsersController(UserManager<IdentityUser> userManager, IRoleService roleService)
        {
            _userManager = userManager;
            _roleService = roleService;
        }

        // GET: api/users
        [HttpGet]
        0 references
        public async Task<IActionResult> GetUsers()
        {
            var users = _userManager.Users;
            return Ok(await users.ToListAsync());
        }

        // GET: api/users/{id}
        [HttpGet("{id}")]
        1 reference
        public async Task<IActionResult> GetUser(string id)
        {
            var user = await _userManager.FindByIdAsync(id);
            if (user == null) return NotFound();
            return Ok(user);
        }

        // POST: api/users
        [HttpPost]
        0 references
        public async Task<IActionResult> CreateUser([FromBody] CreateUserModel model)
        {
            var user = new IdentityUser
            {
                UserName = model.Username,
                Email = model.Email
            };

            var result = await _userManager.CreateAsync(user, model.Password);
            if (result.Succeeded)
            {
                // Assign roles if provided
                if (!string.IsNullOrEmpty(model.Role))
                {
                    await _userManager.AddToRoleAsync(user, model.Role);
                }

                return CreatedAtAction(nameof(GetUser), new { id = user.Id }, user);
            }

            return BadRequest(result.Errors);
        }

        // PUT: api/users/{id}
        [HttpPut("{id}")]
        0 references
        public async Task<IActionResult> UpdateUser(string id, [FromBody] UpdateUserModel model)
        {
            var user = await _userManager.FindByIdAsync(id);
            if (user == null) return NotFound();

            user.Email = model.Email ?? user.Email;
            user.UserName = model.Username ?? user.UserName;

            var result = await _userManager.UpdateAsync(user);
            if (result.Succeeded)
            {
                // Update roles if provided
                if (!string.IsNullOrEmpty(model.Role))
                {
                    await _userManager.AddToRoleAsync(user, model.Role);
                }

                return Ok(user);
            }

            return BadRequest(result.Errors);
        }

        // DELETE: api/users/{id}
        [HttpDelete("{id}")]
        0 references
        public async Task<IActionResult> DeleteUser(string id)
        {
            var user = await _userManager.FindByIdAsync(id);
            if (user == null) return NotFound();

            var result = await _userManager.DeleteAsync(user);
            if (result.Succeeded)
            {
                return Ok();
            }

            return BadRequest(result.Errors);
        }
    }

    1 reference
    public class CreateUserModel
    {
        1 reference
        public string Username { get; set; }
        1 reference
        public string Email { get; set; }
        1 reference
        public string Password { get; set; }
        2 references
        public string Role { get; set; }
    }

    1 reference
    public class UpdateUserModel
    {
        1 reference
        public string Username { get; set; }
        1 reference
        public string Email { get; set; }
        2 references
        public string Role { get; set; }
    }
}
```



```
// Get Users
0 references
function getUsers() {
    $.ajax({
        url: '/api/users',
        type: 'GET',
        9 references
        success: function (data) {
            let usersTable = '';
            data.forEach(user => {
                usersTable += `<tr>
                <td>${user.username}</td>
                <td>${user.email}</td>
                <td><button onclick="editUser(${user.id})">Edit</button></td>
                <td><button onclick="deleteUser(${user.id})">Delete</button></td>
            </tr>`;
            });
            $('#usersTable').html(usersTable);
        },
        9 references
        error: function (error) {
            alert('Error: ' + error.responseText);
        }
    });
}

// Update User
0 references
function updateUser(id) {
    const userData = {
        username: $('#username').val(),
        email: $('#email').val(),
        role: $('#role').val()
    };

    $.ajax({
        url: '/api/users/' + id,
        type: 'PUT',
        contentType: 'application/json',
        data: JSON.stringify(userData),
        9 references
        success: function (response) {
            alert('User updated successfully');
            // Optionally refresh the user list
        },
        9 references
        error: function (error) {
            alert('Error: ' + error.responseText);
        }
    });
}

// Delete User
0 references
function deleteUser(id) {
    if (confirm('Are you sure you want to delete this user?')) {
        $.ajax({
            url: '/api/users/' + id,
            type: 'DELETE',
            9 references
            success: function (response) {
                alert('User deleted successfully');
                // Optionally refresh the user list
            },
            9 references
            error: function (error) {
                alert('Error: ' + error.responseText);
            }
        });
    }
}
```



```json
// 20241126223901
// https://localhost:44304/api/users

[
  {
    "id": "294c6cf3-e3ef-4319-a362-35a86d4794b1",
    "userName": "ductrong12072002@gmail.com",
    "normalizedUserName": "DUCTRONG12072002@GMAIL.COM",
    "email": "ductrong12072002@gmail.com",
    "normalizedEmail": "DUCTRONG12072002@GMAIL.COM",
    "emailConfirmed": false,
    "passwordHash": "AQAAAAIAAYagAAAAEMJyPHhP1OZzPkmJA20bEt8yIpRQ60iTQZC7UH/bCHOuwbgnblmGBA6hYsx+ChRUeg==",
    "securityStamp": "RLCRSRHBGP33MHGCV6E7G77LVC2KWOHS",
    "concurrencyStamp": "799314ea-3dc1-4f8b-b515-55e8256e67c0",
    "phoneNumber": null,
    "phoneNumberConfirmed": false,
    "twoFactorEnabled": false,
    "lockoutEnd": null,
    "lockoutEnabled": true,
    "accessFailedCount": 0
  },
  {
    "id": "c493f191-cb06-4802-b52c-27b471612ddb",
    "userName": "admin@gmail.com",
    "normalizedUserName": "ADMIN@GMAIL.COM",
    "email": "admin@gmail.com",
    "normalizedEmail": "ADMIN@GMAIL.COM",
    "emailConfirmed": true,
    "passwordHash": "AQAAAAIAAYagAAAAEHd84ALCqwlZvXphRnnAUB49HL7PPGxuLvvu7HdEeMB/qmOOeUkknLc791FWR+Bm/Q==",
    "securityStamp": "MOGNCVAUTVDUOOYBBNIBE6LZECXJC7RP",
    "concurrencyStamp": "f0d5b840-723a-48b7-8cfb-78926d31a5a3",
    "phoneNumber": null,
    "phoneNumberConfirmed": false,
    "twoFactorEnabled": false,
    "lockoutEnd": null,
    "lockoutEnabled": true,
    "accessFailedCount": 0
  },
```