LAB 4
Caching & Middleware

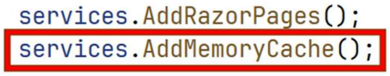
❖ CONTENT

- Implement caching for frequently accessed data
- Create middleware to log user activities

❖ INSTRUCTION

A. Implement caching for frequently accessed data

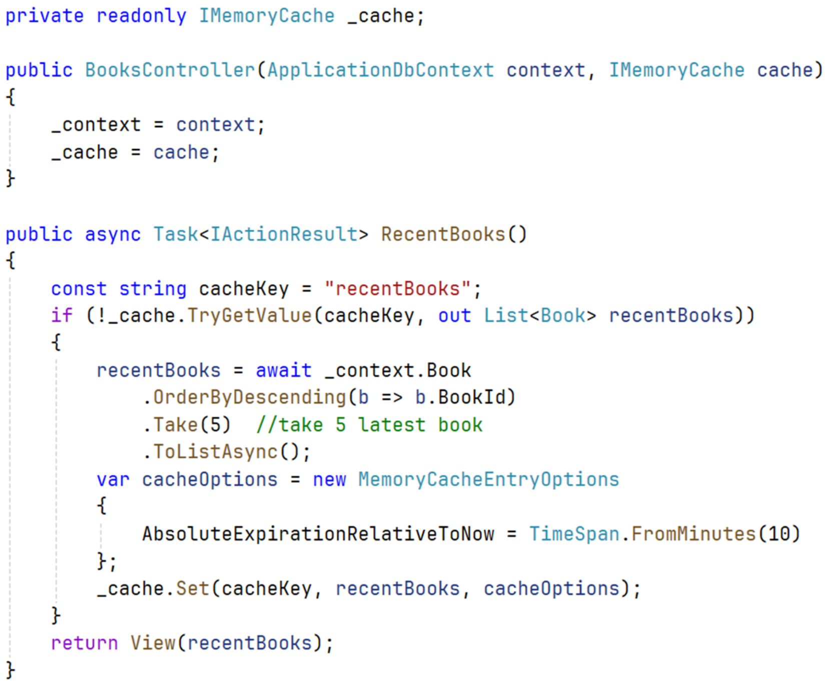
1. Config memory cache in application startup



```
services.AddRazorPages();
services.AddMemoryCache();
```

*Figure 1 – **Startup.cs***

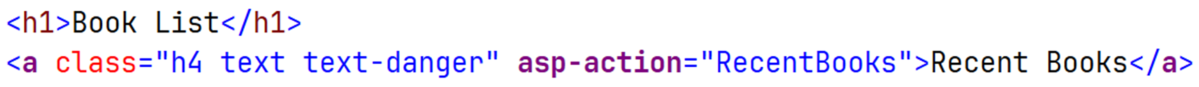
2. Implement caching in Controller



```
private readonly IMemoryCache _cache;

public BooksController(ApplicationDbContext context, IMemoryCache cache)
{
    _context = context;
    _cache = cache;
}

public async Task<IActionResult> RecentBooks()
{
    const string cacheKey = "recentBooks";
    if (!_cache.TryGetValue(cacheKey, out List<Book> recentBooks))
    {
        recentBooks = await _context.Book
            .OrderByDescending(b => b.BookId)
            .Take(5)  //take 5 latest book
            .ToListAsync();
        var cacheOptions = new MemoryCacheEntryOptions
        {
            AbsoluteExpirationRelativeToNow = TimeSpan.FromMinutes(10)
        };
        _cache.Set(cacheKey, recentBooks, cacheOptions);
    }
    return View(recentBooks);
}
```

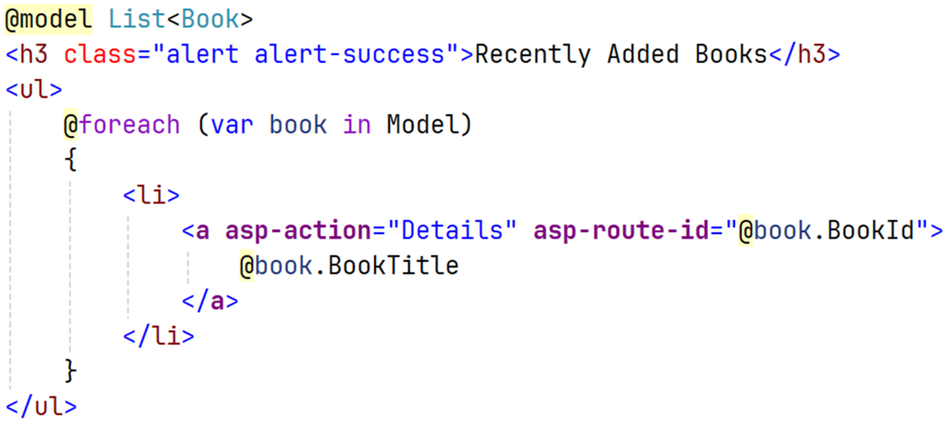
*Figure 2 – **Controllers/BooksController.cs***

3. Add hyperlink and proper view to display recent added items (books)



```
<h1>Book List</h1>
<a class="h4 text text-danger" asp-action="RecentBooks">Recent Books</a>
```

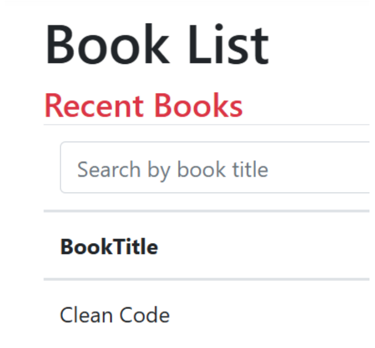
*Figure 3 - **Views/Books/Index.cshtml***



```
@model List<Book>
<h3 class="alert alert-success">Recently Added Books</h3>
<ul>
    @foreach (var book in Model)
    {
        <li>
            <a asp-action="Details" asp-route-id="@book.BookId">
                @book.BookTitle
            </a>
        </li>
    }
</ul>
```

*Figure 4 - **Views/Books/RecentBooks.cshtml***

❖ Result

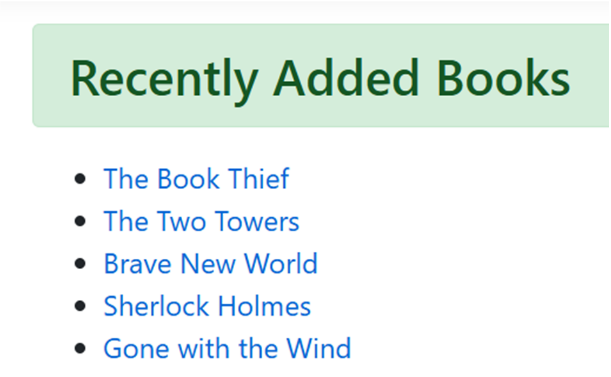


Book List

Recent Books

Search by book title

BookTitle
Clean Code

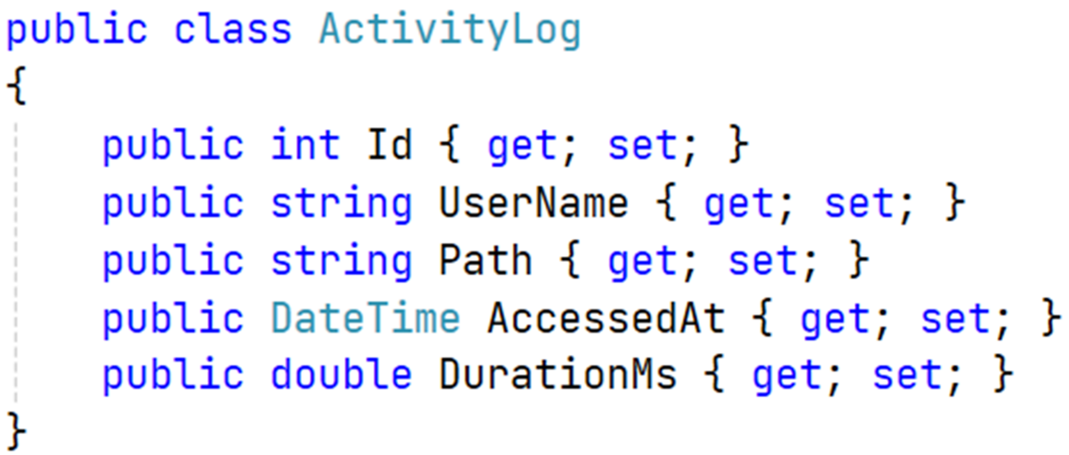


Recently Added Books

- The Book Thief
- The Two Towers
- Brave New World
- Sherlock Holmes
- Gone with the Wind

B. Create middleware for logging user activities

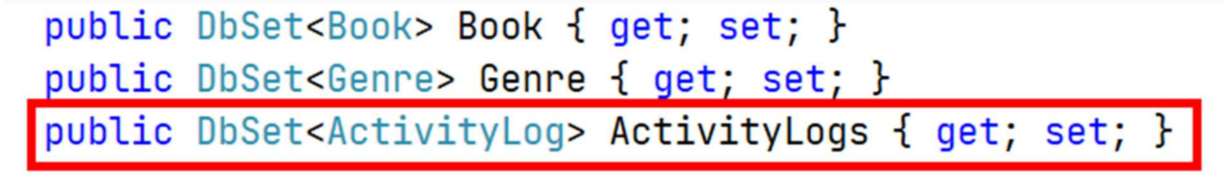
1. Add new model *ActivityLog* to save user activities



```
public class ActivityLog
{
    public int Id { get; set; }
    public string UserName { get; set; }
    public string Path { get; set; }
    public DateTime AccessedAt { get; set; }
    public double DurationMs { get; set; }
}
```

*Figure 5 - **Models/ActivityLog.cs***

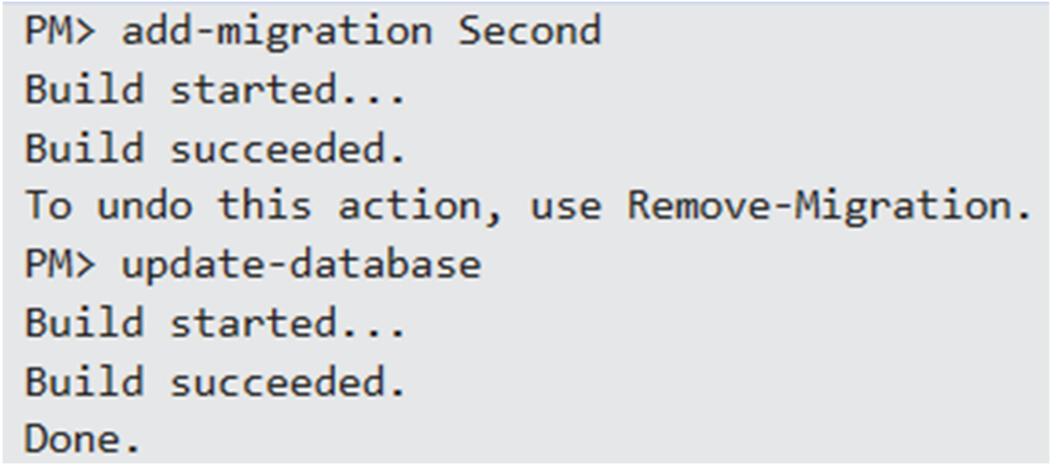
2. Add **ActivityLog** to **Database Context** to enable in database



```
public DbSet<Book> Book { get; set; }
public DbSet<Genre> Genre { get; set; }
public DbSet<ActivityLog> ActivityLogs { get; set; }
```

*Figure 6 - **Data/ApplicationDbContext.cs***

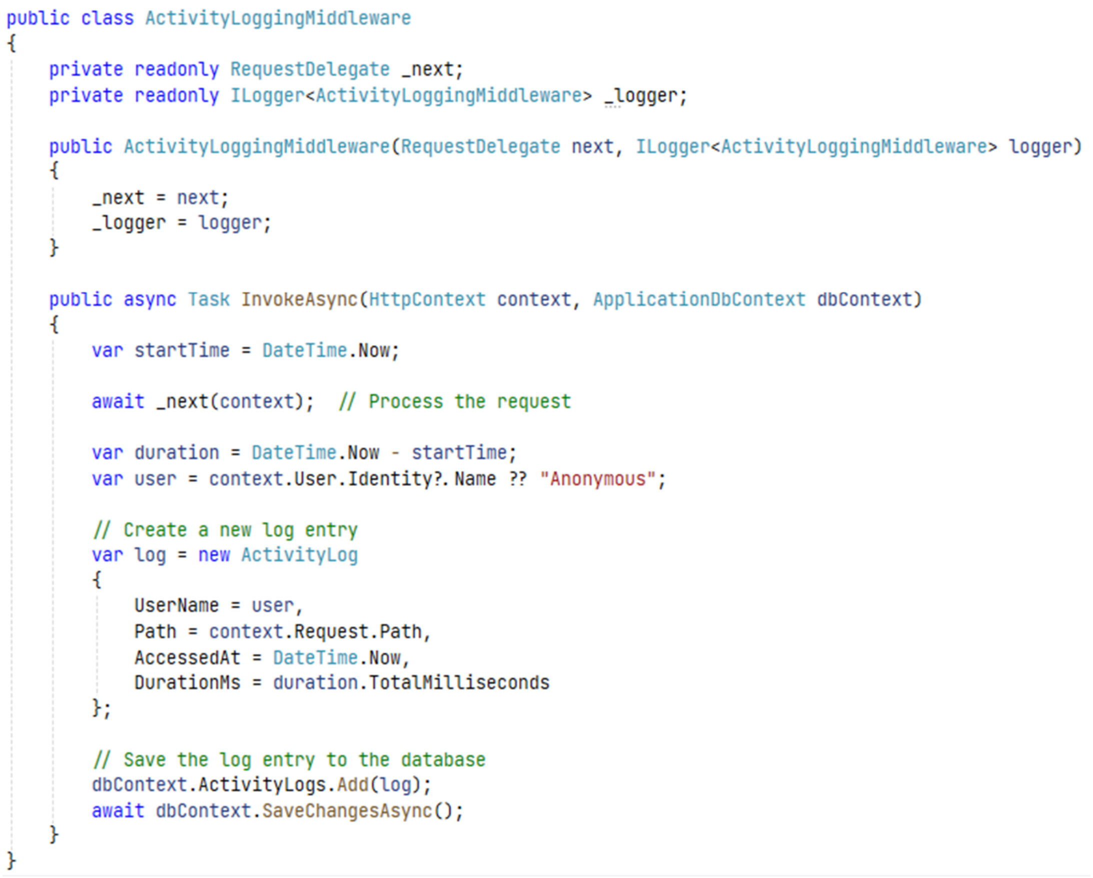
3. Add migration then update database



```
PM> add-migration Second
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> update-database
Build started...
Build succeeded.
Done.
```

*Figure 7 - **Package Manager Console***

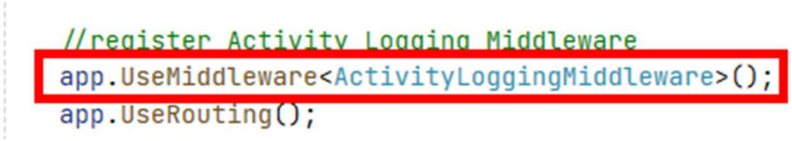
4. Create custom middleware to save user activity logs



```
public class ActivityLoggingMiddleware
{
    private readonly RequestDelegate _next;
    private readonly ILogger<ActivityLoggingMiddleware> _logger;

    public ActivityLoggingMiddleware(RequestDelegate next, ILogger<ActivityLoggingMiddleware> logger)
    {
        _next = next;
        _logger = logger;
    }

    public async Task InvokeAsync(HttpContext context, ApplicationDbContext dbContext)
    {
        var startTime = DateTime.Now;

        await _next(context);  // Process the request

        var duration = DateTime.Now - startTime;
        var user = context.User.Identity?.Name ?? "Anonymous";

        // Create a new log entry
        var log = new ActivityLog
        {
            UserName = user,
            Path = context.Request.Path,
            AccessedAt = DateTime.Now,
            DurationMs = duration.TotalMilliseconds
        };

        // Save the log entry to the database
        dbContext.ActivityLogs.Add(log);
        await dbContext.SaveChangesAsync();
    }
}
```

*Figure 8 - **Middlewares/ActivityLoggingMiddleware.cs***

5. Register middleware in the pipeline to log every request

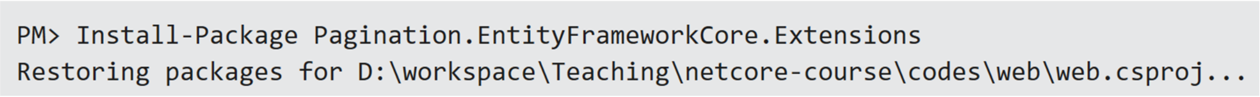


```
//register Activity Logging Middleware
app.UseMiddleware<ActivityLoggingMiddleware>();
app.UseRouting();
```

*Figure 9 - **Startup.cs***

6. Display all user activity logs

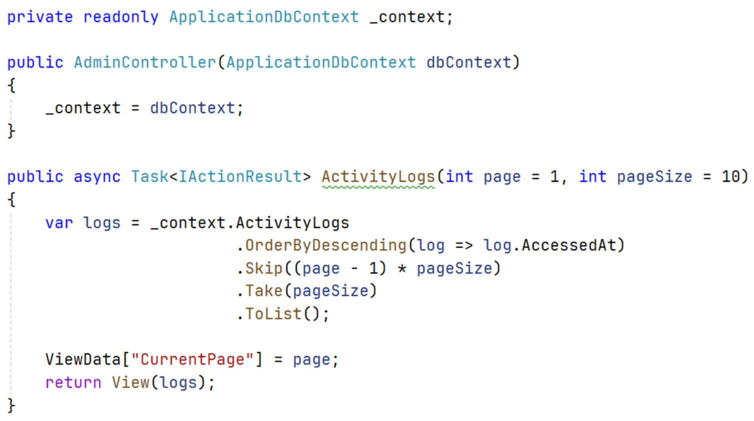
- Install Pagination package (suitable for large data set)



```
PM> Install-Package Pagination.EntityFrameworkCore.Extensions
Restoring packages for D:\workspace\Teaching\netcore-course\codes\web\web.csproj...
```

Figure 10 - Package Manager Console

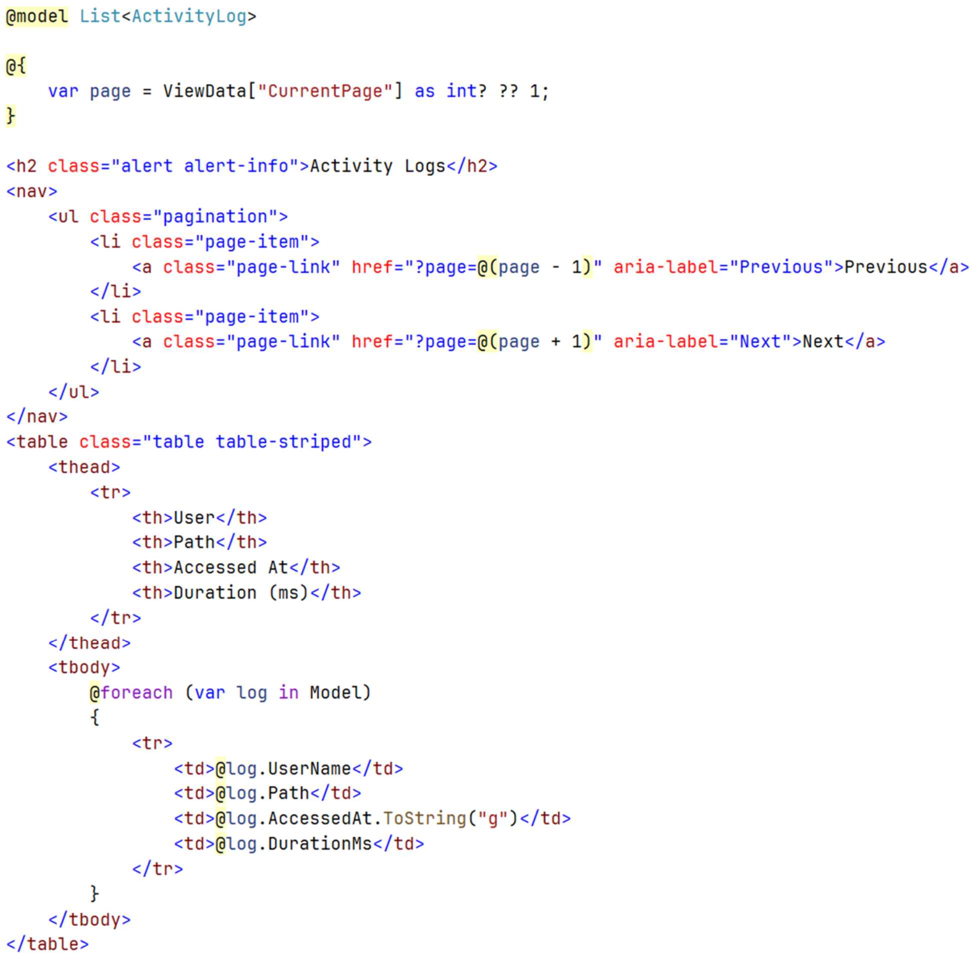
- Update code for Admin Controller



```
private readonly ApplicationDbContext _context;

public AdminController(ApplicationDbContext dbContext)
{
    _context = dbContext;
}

public async Task<IActionResult> ActivityLogs(int page = 1, int pageSize = 10)
{
    var logs = _context.ActivityLogs
                    .OrderByDescending(log => log.AccessedAt)
                    .Skip((page - 1) * pageSize)
                    .Take(pageSize)
                    .ToList();

    ViewData["CurrentPage"] = page;
    return View(logs);
}
```

*Figure 11 - **Controllers/AdminController.cs***

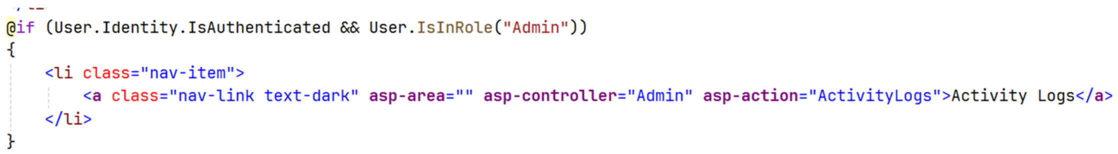
- Create a view to display table of user activities



```
@model List<ActivityLog>

@{
    var page = ViewData["CurrentPage"] as int? ?? 1;
}

<h2 class="alert alert-info">Activity Logs</h2>
<nav>
    <ul class="pagination">
        <li class="page-item">
            <a class="page-link" href="?page=@(page - 1)" aria-label="Previous">Previous</a>
        </li>
        <li class="page-item">
            <a class="page-link" href="?page=@(page + 1)" aria-label="Next">Next</a>
        </li>
    </ul>
</nav>
<table class="table table-striped">
    <thead>
        <tr>
            <th>User</th>
            <th>Path</th>
            <th>Accessed At</th>
            <th>Duration (ms)</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var log in Model)
        {
            <tr>
                <td>@log.UserName</td>
                <td>@log.Path</td>
                <td>@log.AccessedAt.ToString("g")</td>
                <td>@log.DurationMs</td>
            </tr>
        }
    </tbody>
</table>
```

*Figure 12 - **Views/Admin/ActivityLogs.cshtml***

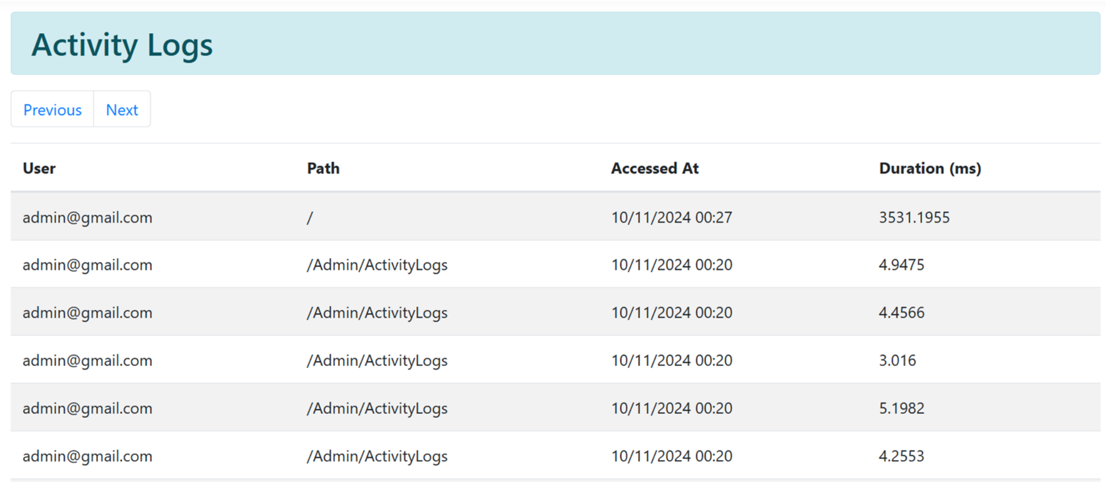
- Add new menu to navigation bar for only admin role



```
@if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))
{
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Admin" asp-action="ActivityLogs">Activity Logs</a>
    </li>
}
```

*Figure 13 - **Views/Shared/_Layout.cshtml***

❖ Result



Activity Logs

User	Path	Accessed At	Duration (ms)
admin@gmail.com	/	10/11/2024 00:27	3531.1955
admin@gmail.com	/Admin/ActivityLogs	10/11/2024 00:20	4.9475
admin@gmail.com	/Admin/ActivityLogs	10/11/2024 00:20	4.4566
admin@gmail.com	/Admin/ActivityLogs	10/11/2024 00:20	3.016
admin@gmail.com	/Admin/ActivityLogs	10/11/2024 00:20	5.1982
admin@gmail.com	/Admin/ActivityLogs	10/11/2024 00:20	4.2553