.gitignore

```
# This prevents merge conflicts and conflicting data.
Bucstop/wwwroot/visitcount.json
Bucstop/wwwroot/playcount.json
```

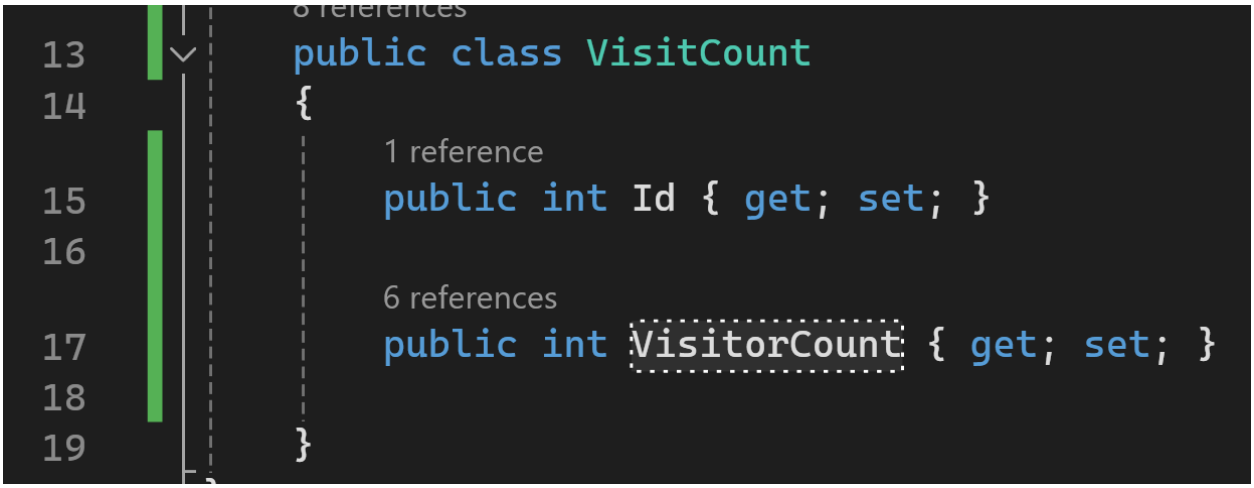
/wwwroot/visitcount.json

```
{"Id":1,"VisitorCount":2}
```

/Microservices/MicroClient.cs

```
/// <summary>
/// Requests the Gateway for a List of Game Information
/// </summary>
/// <returns></returns>
0 references
public async Task<GameInfo[]> GetVisitCountAsync()
{
    try
    {
        var responseMessage = await this.client.GetAsync("/Gateway");

        if (responseMessage != null)
        {
            var stream = await responseMessage.Content.ReadAsStreamAsync();
            return await JsonSerializer.DeserializeAsync<GameInfo[]>(stream, options);
        }
    }
    catch (HttpRequestException ex)
    {
        _logger.LogError(ex.Message);
    }
    return new GameInfo[] { };
}
```

/Models/VisitCount.cs



/Controllers/VisitCountController.cs

```
namespace BucStop.Controllers
{
    [Authorize]
    1 reference
    public class VisitCountController : Controller
    {
        private readonly MicroClient _httpClient;
        private readonly PlayCountManager _playCountManager;
        private readonly GameService _gameService;
        private readonly VisitCountService _visitCountService;
```

```
0 references
public VisitCountController(VisitCountService visitCountService)
{
    _visitCountService = visitCountService;
}

//Takes the user to the index page, passing the games list as an argument
[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
0 references
public async Task<IActionResult> IndexAsync()
{
    List<Game> games = await GetGamesWithInfo();

    //have to update playcounts here since the we are reading it dynamically now instead of from a static list
    foreach(Game game in games)
    {
        game.PlayCount = _playCountManager.GetPlayCount(game.Id);
    }

    return View(games);
}

0 references
public IActionResult Index()
{
    // Fetch the visit count data from the JSON file
    VisitCount? visitCount = _visitCountService.GetVisitCount();

    // Pass the visit count data to the view
    return View(visitCount);
}
```

/Controllers/HomeController.cs

```
3 references
public class HomeController : Controller
{
    private readonly IWebHostEnvironment _webHostEnvironment;

    private readonly ILogger<HomeController> _logger;
    private readonly GameService _gameService;
    private readonly VisitCountService _visitCountService;
    private readonly VisitCountManager _visitCountManager;

    0 references
    public HomeController(ILogger<HomeController> logger, GameService gameService, VisitCountManager visitCountManager)
    {
        _logger = logger;
        _gameService = gameService;

        _visitCountManager = visitCountManager;
    }
```

```
//Sends the user to the deprecated Index page.
0 references
public IActionResult Index()
{
    // Retrieve the updated visit count
    int currentVisitCount = _visitCountManager.GetVisitCounts();

    // Pass the updated visit count to the view
    ViewData["VisitCount"] = currentVisitCount;

    // Fetch and pass game data
    var games = _gameService.GetGames();
    return View(games);
}

0 references
public override void OnActionExecuting(ActionExecutingContext context)
{
    // Load the visit count
    ViewData["VisitCount"] = _visitCountManager.GetVisitCounts();
    base.OnActionExecuting(context);
}
```

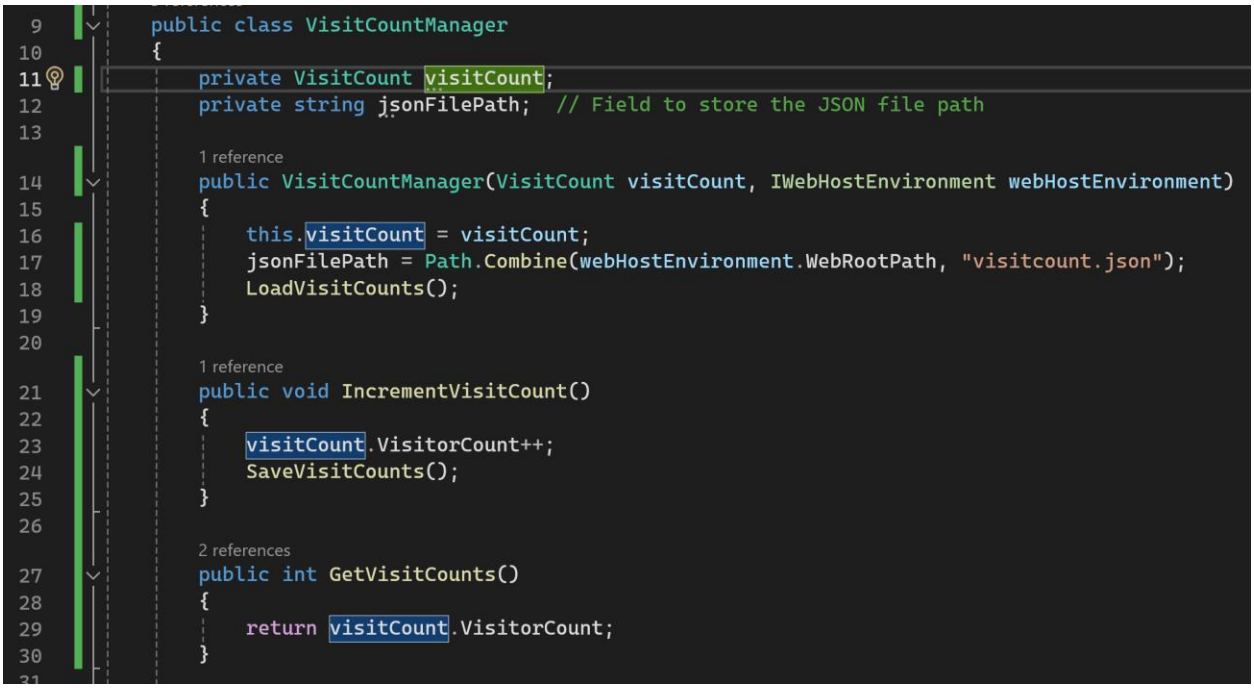
/Services/VisitCountService.cs

```cs
public class VisitCountService
{
    private string path = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "visitcount.json");

    1 reference
    public VisitCount? GetVisitCount()
    {
        string json = File.ReadAllText(path);
        VisitCount? visitcount = JsonSerializer.Deserialize<VisitCount>(json);
        return visitcount;
    }
}
```

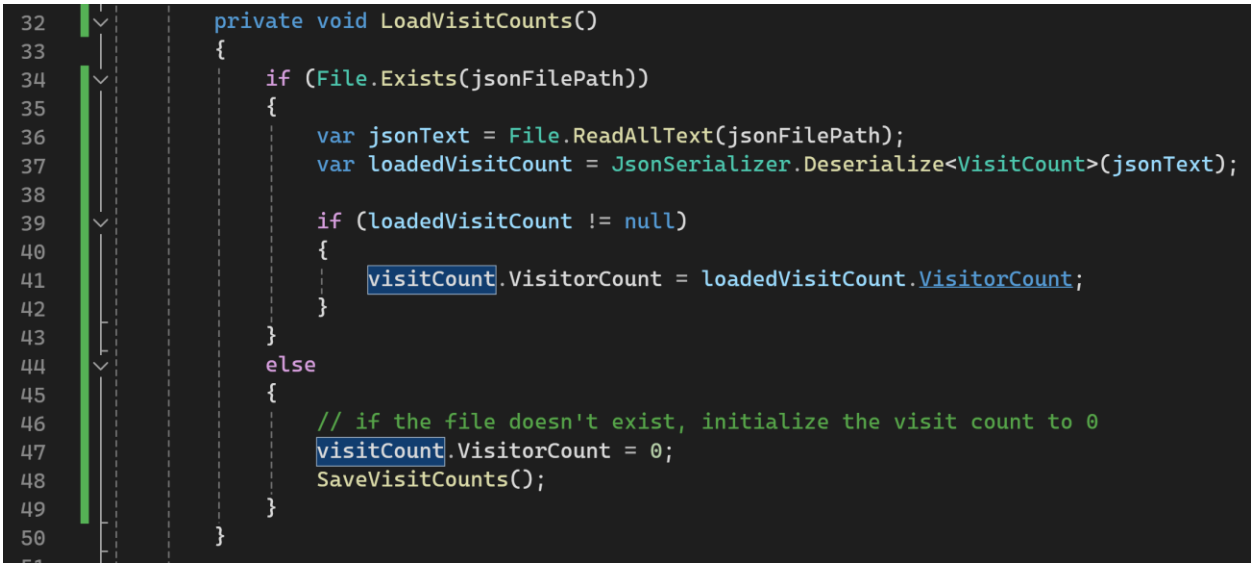
/Views/Shared/_Layout.cshtml

```cshtml
<!-- This block handles the footer, which includes the version number for the project. -->
<footer class="border-top footer text-muted">
    <div class="container">
        &copy; <script>document.write(new Date().getFullYear())</script> - BucStop - <a asp-area="" asp-
          controller="Home" asp-action="Privacy">Privacy</a> - <a href="https://github.com/chrisseals98/BOBBY/
          releases/tag/v2.0.0" style="color:#C6AA76;">Version: 2.0.0</a> - Current Visit Count: @
          (Context.Items["VisitCount"] ?? 0)
    </div>
</footer>
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script> <!-- Adds bootstrap script to the
  page-->
<script src="~/js/site.js" asp-append-version="true"></script>
@await RenderSectionAsync("Scripts", required: false)
```

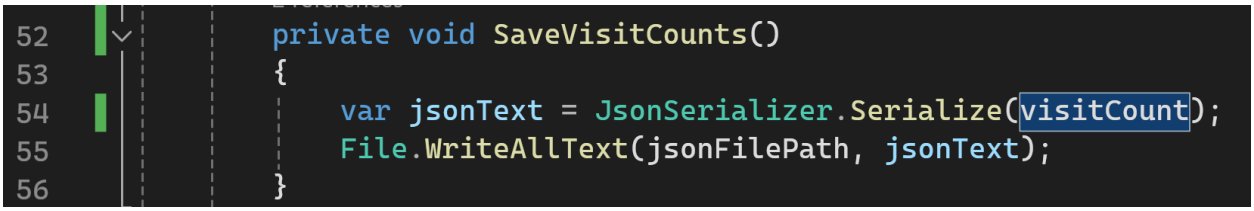
/Services/VisitCountManager.cs



```
public class VisitCountManager
{
    private VisitCount visitCount;
    private string jsonFilePath;  // Field to store the JSON file path

    1 reference
    public VisitCountManager(VisitCount visitCount, IWebHostEnvironment webHostEnvironment)
    {
        this.visitCount = visitCount;
        jsonFilePath = Path.Combine(webHostEnvironment.WebRootPath, "visitcount.json");
        LoadVisitCounts();
    }

    1 reference
    public void IncrementVisitCount()
    {
        visitCount.VisitorCount++;
        SaveVisitCounts();
    }

    2 references
    public int GetVisitCounts()
    {
        return visitCount.VisitorCount;
    }
```



```
    private void LoadVisitCounts()
    {
        if (File.Exists(jsonFilePath))
        {
            var jsonText = File.ReadAllText(jsonFilePath);
            var loadedVisitCount = JsonSerializer.Deserialize<VisitCount>(jsonText);

            if (loadedVisitCount != null)
            {
                visitCount.VisitorCount = loadedVisitCount.VisitorCount;
            }
        }
        else
        {
            // if the file doesn't exist, initialize the visit count to 0
            visitCount.VisitorCount = 0;
            SaveVisitCounts();
        }
    }
```



```
    private void SaveVisitCounts()
    {
        var jsonText = JsonSerializer.Serialize(visitCount);
        File.WriteAllText(jsonFilePath, jsonText);
    }
```

/Program.cs

```
var builder = WebApplication.CreateBuilder(args);

// Register VisitCountManager as a singleton
builder.Services.AddSingleton<VisitCountManager>();

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

var provider=builder.Services.BuildServiceProvider();
var configuration=provider.GetRequiredService<IConfiguration>();
```

```
builder.Services.AddSingleton<GameService>();

var app = builder.Build();

// Add BaseController to the request pipeline
app.UseMiddleware<BaseController>();
```

```
//Handles routing to "separate" game pages by setting the Play page to have subpages depending on ID
app.MapControllerRoute(
    name: "Games",
    pattern: "Games/{action}/{id?}",
    defaults: new { controller = "Games", action = "Index" });

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

app.Run();
```