ДОДАТОК А

Ілюстрації програмного застосунку

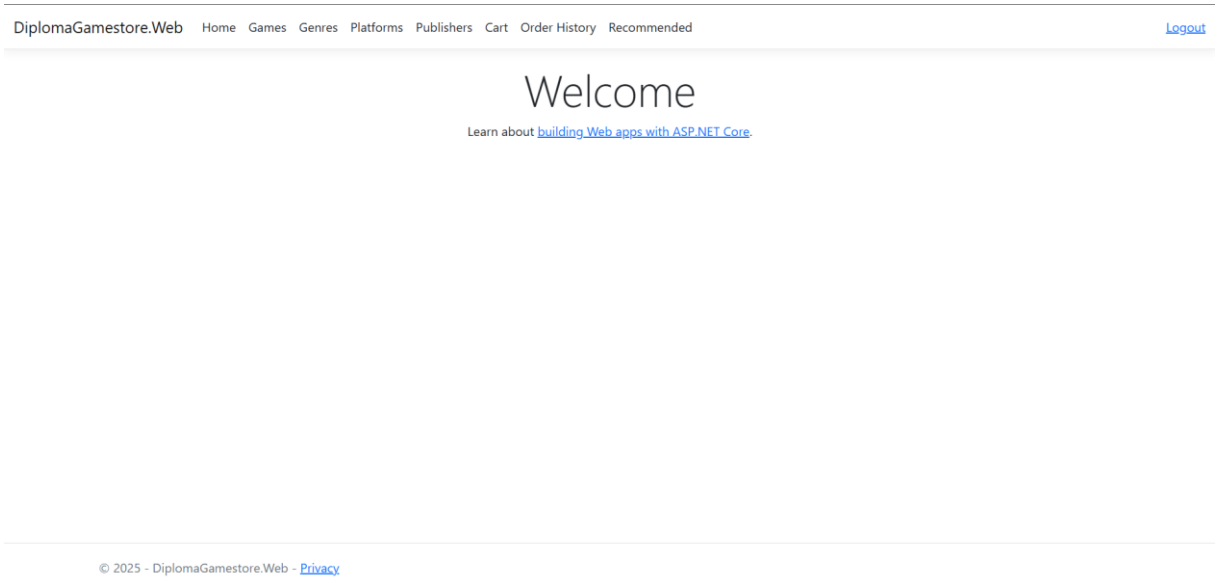

Рисунок А.1 – Головна сторінка

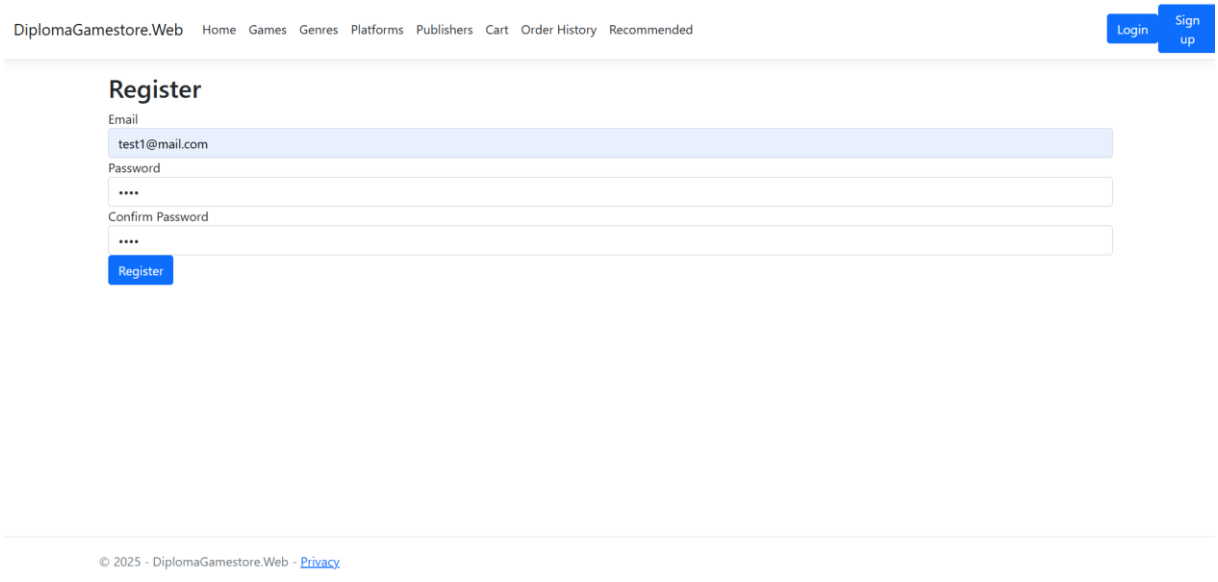

Рисунок А.2 – Сторінка реєстрації

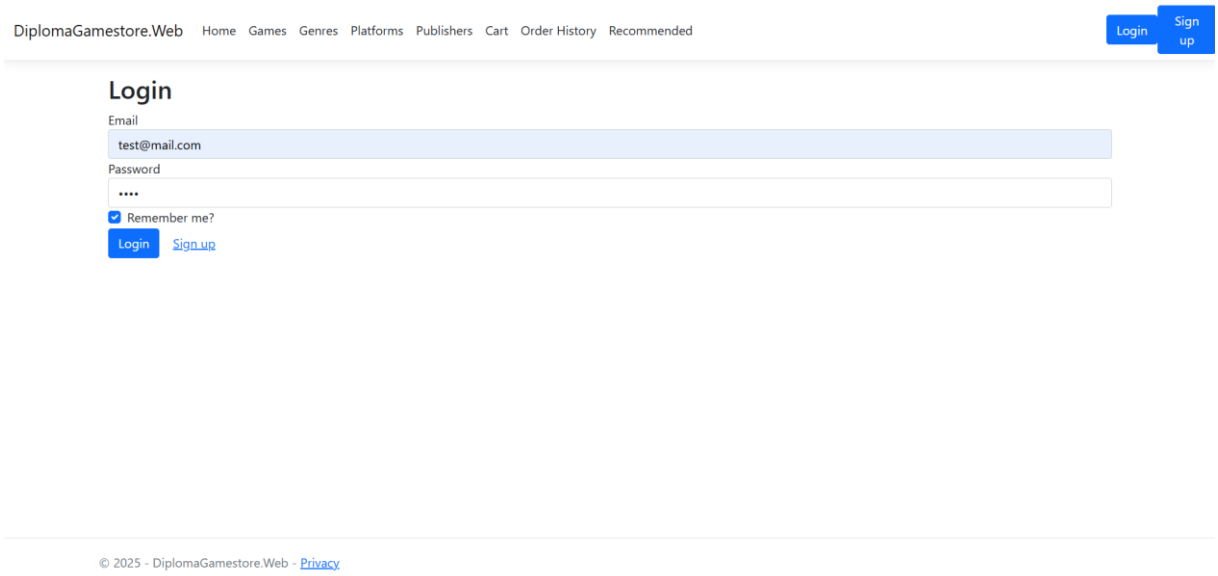

Рисунок А.3 – Сторінка логіну

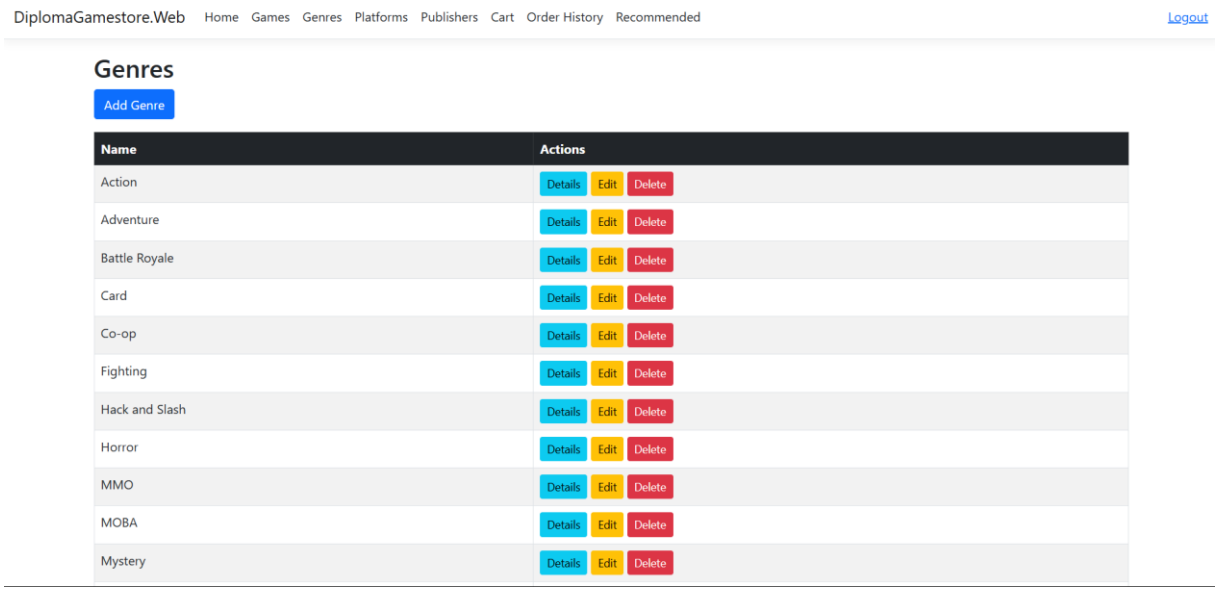

Рисунок А.4 – Сторінка перегляду жанрів

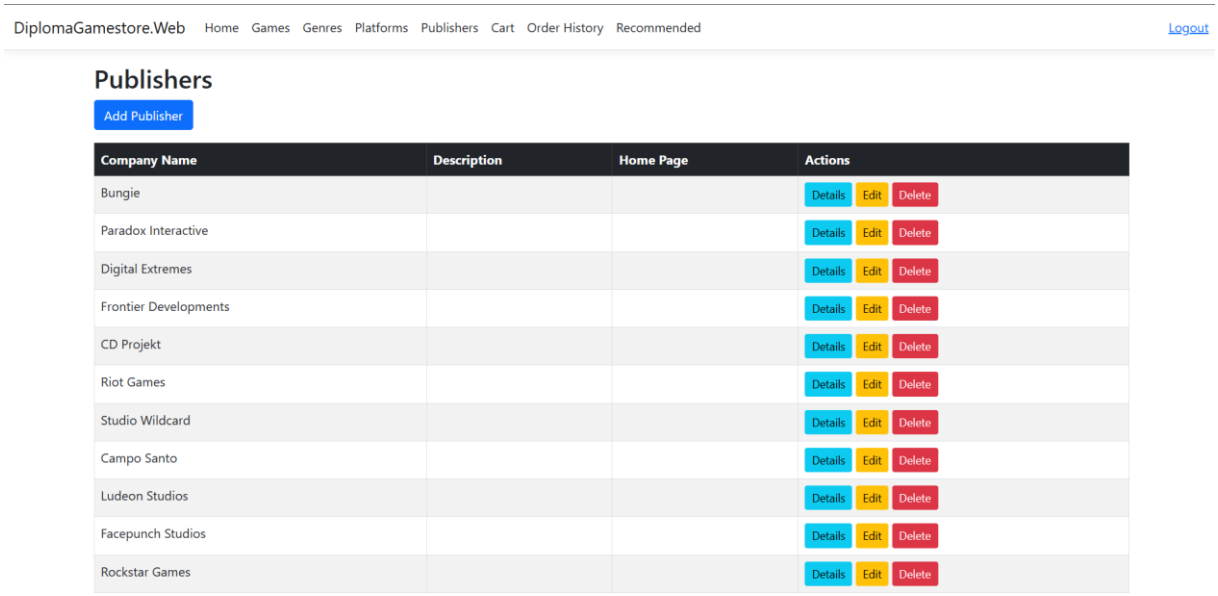

Рисунок А.5 – Сторінка перегляду видавців

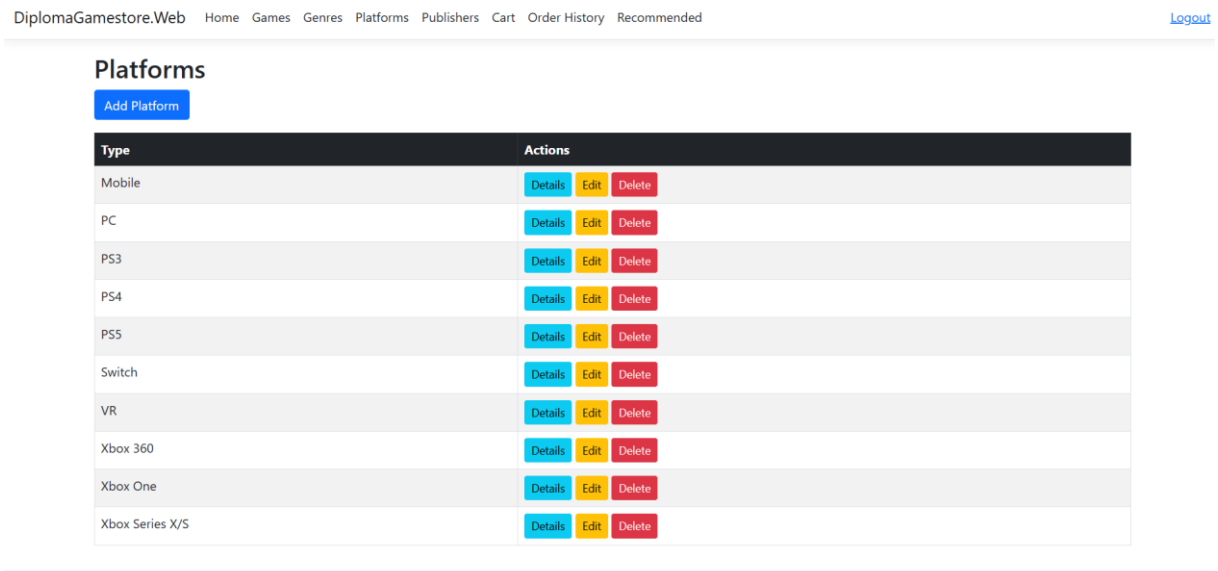

Рисунок А.6 – Сторінка перегляду платформ

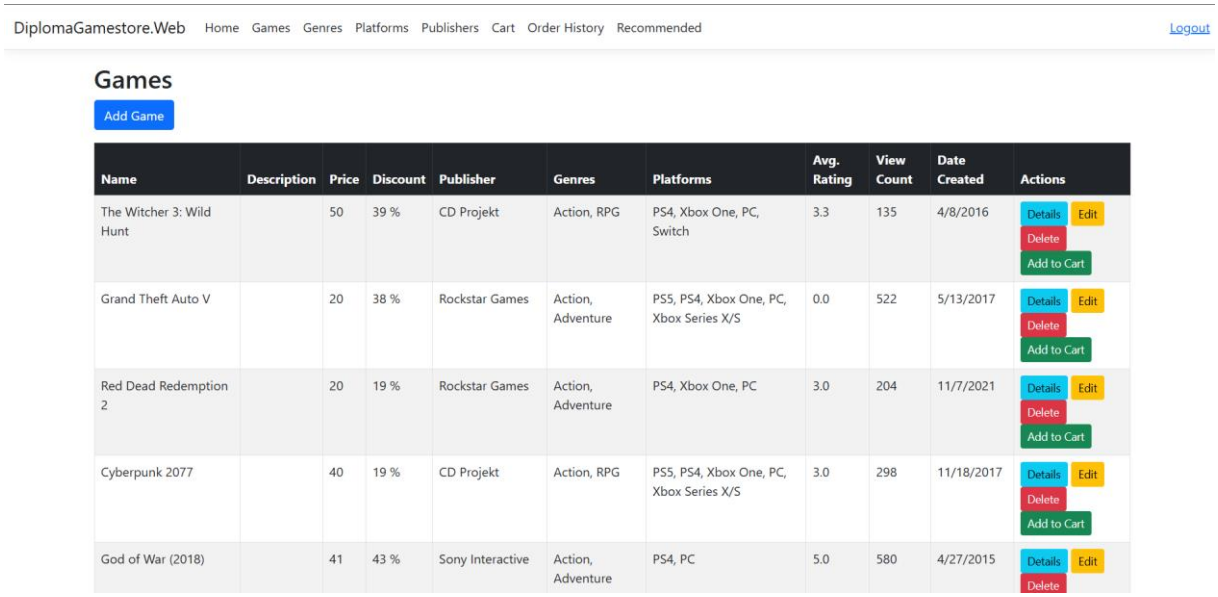

Рисунок А.7 – Сторінка перегляду ігор

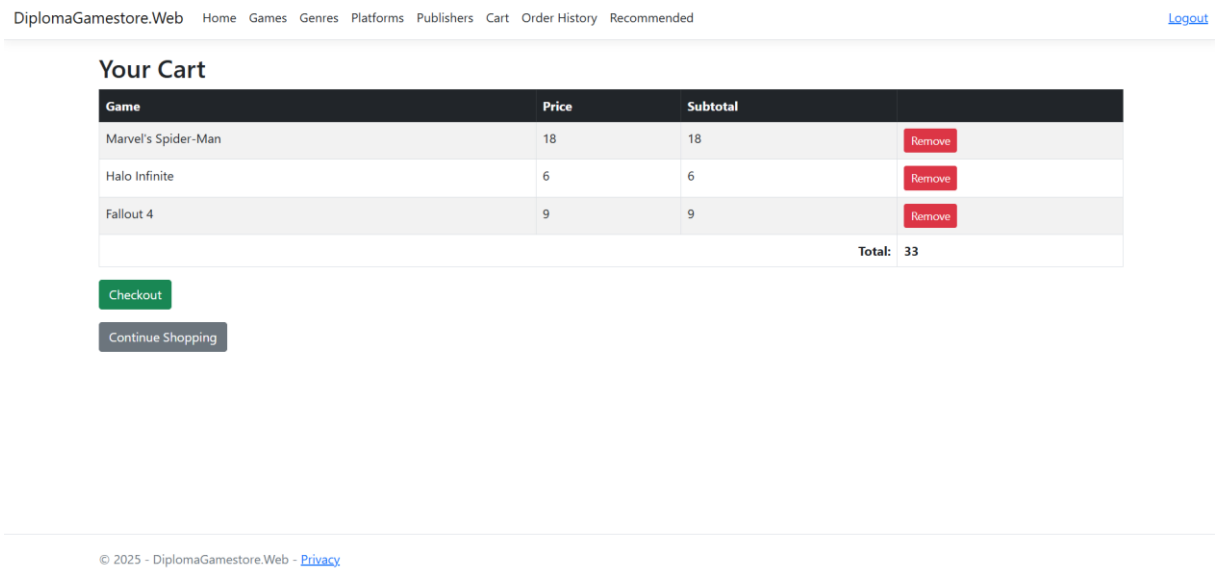

Рисунок А.8 – Сторінка перегляду корзини

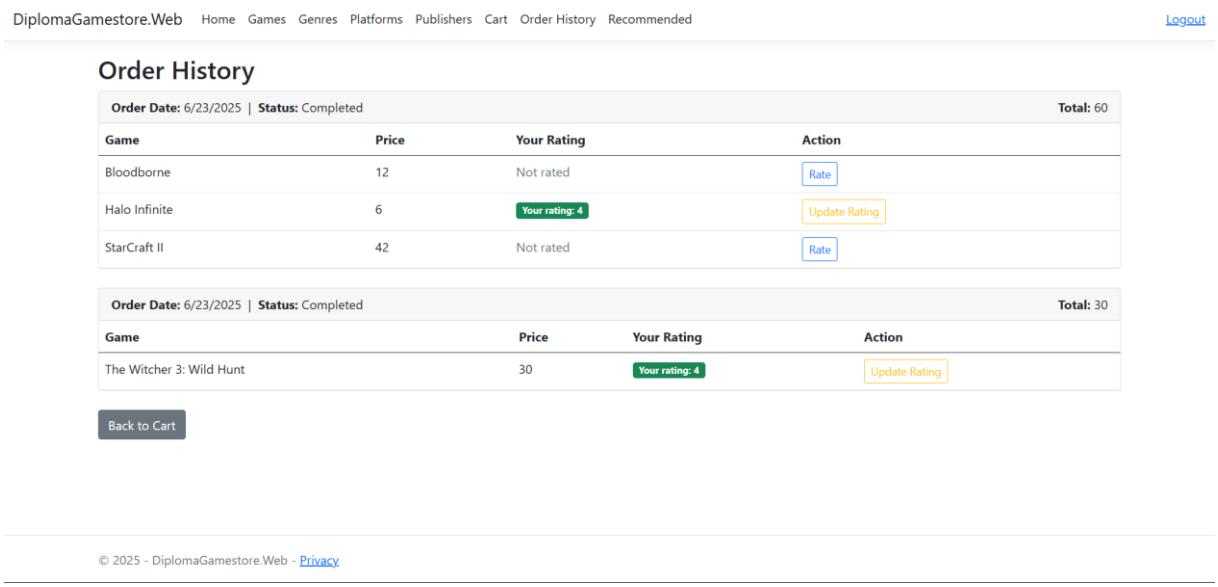

Рисунок А.9 – Сторінка перегляду історії замовлень

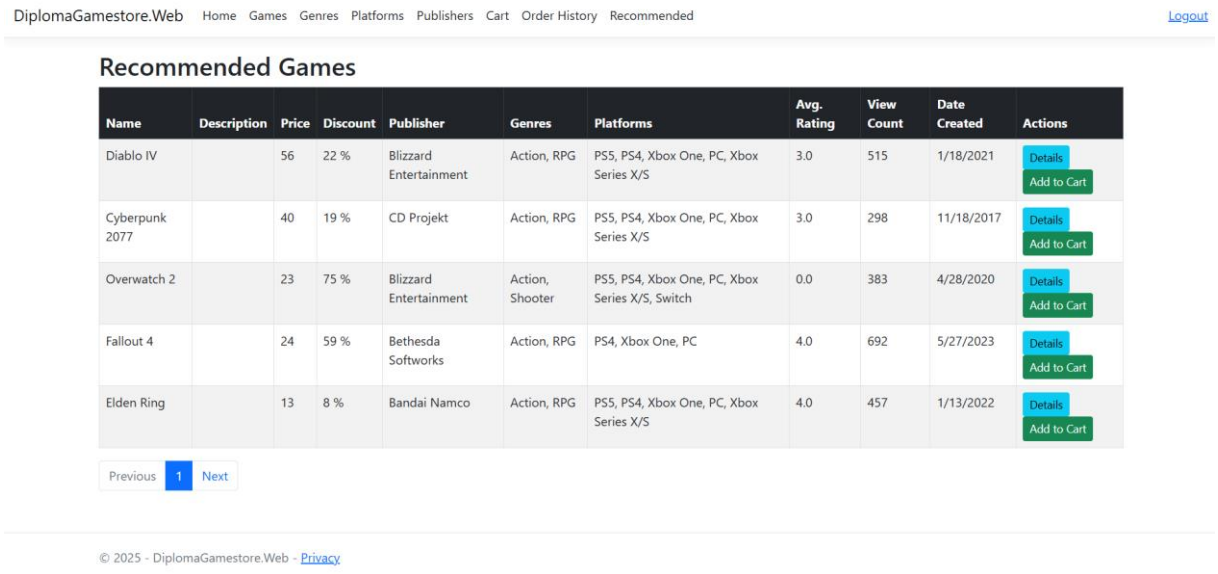

Рисунок А.10 – Сторінка рекомендованих ігор

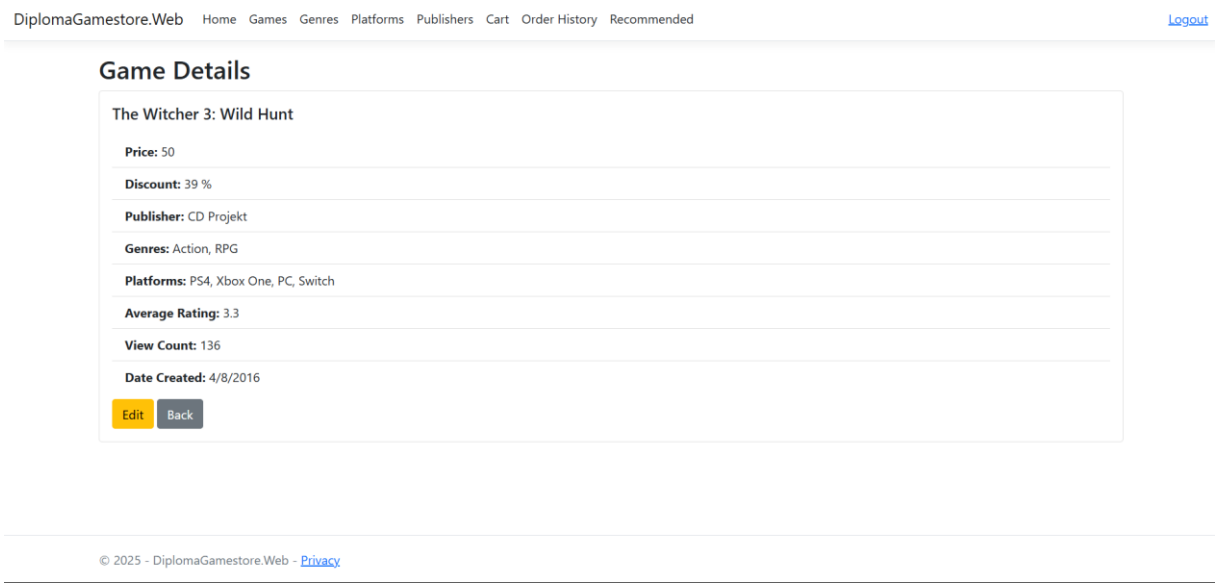

Рисунок А.11 – Сторінка перегляду деталей гри

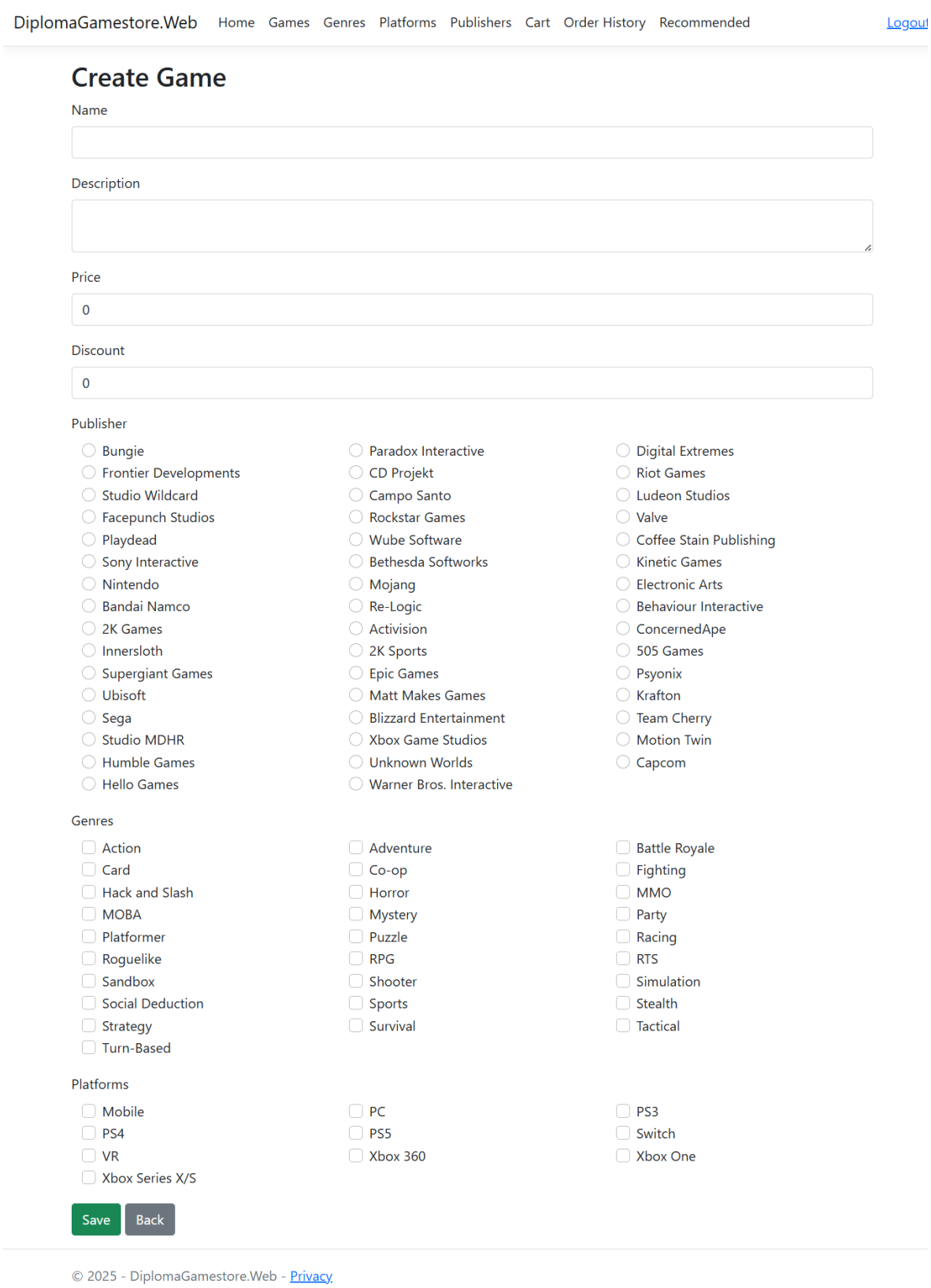

Рисунок А.12 – Сторінка створення гри

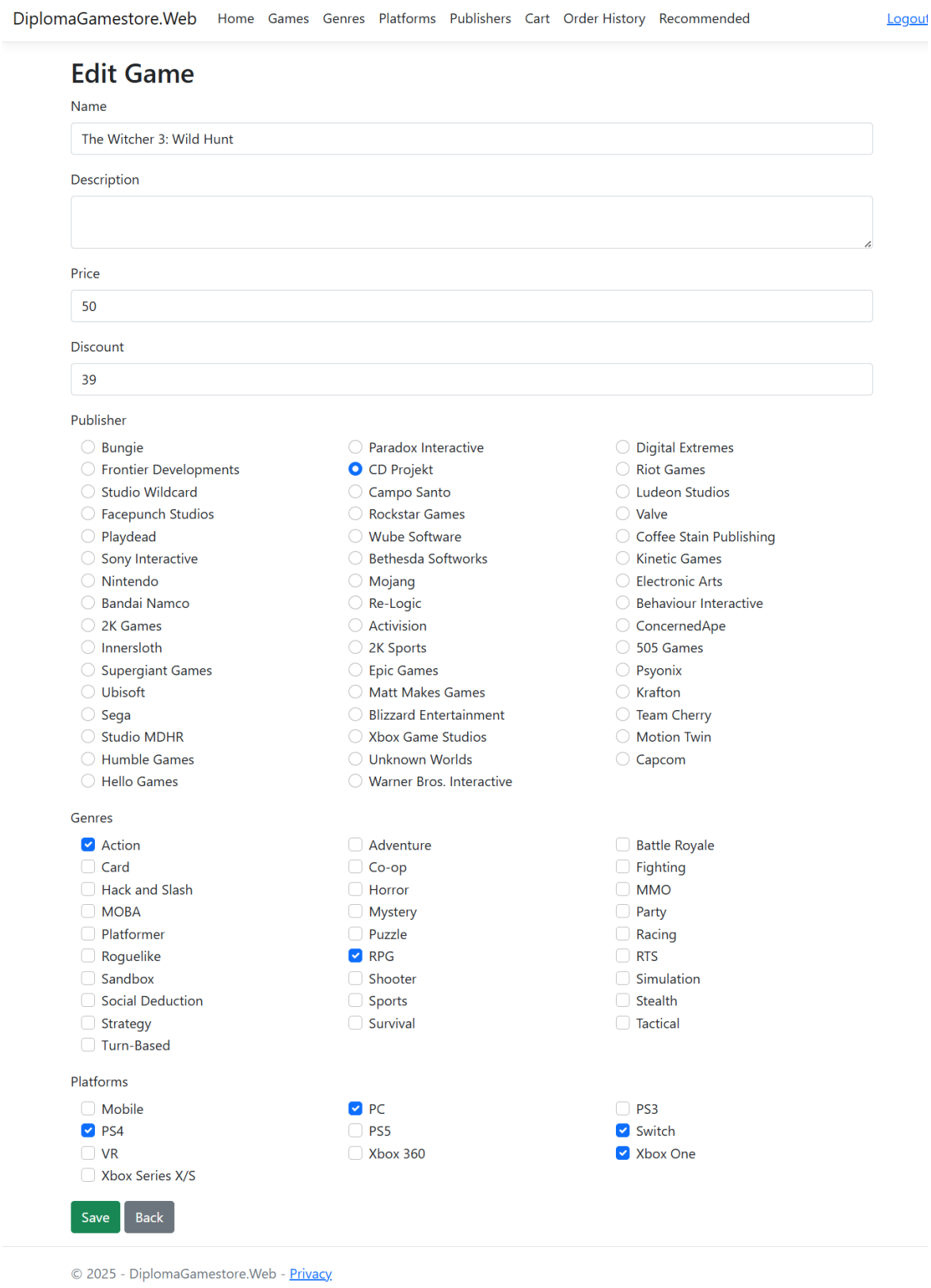

Рисунок А.13 – Сторінка редагування гри

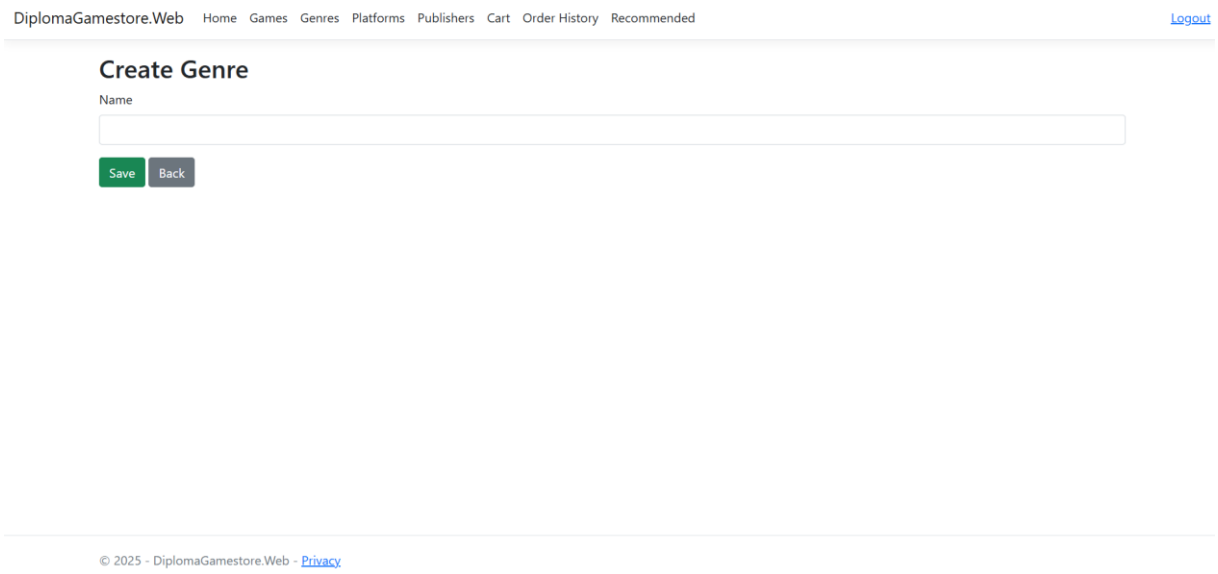

Рисунок А.14 – Сторінка створення жанру

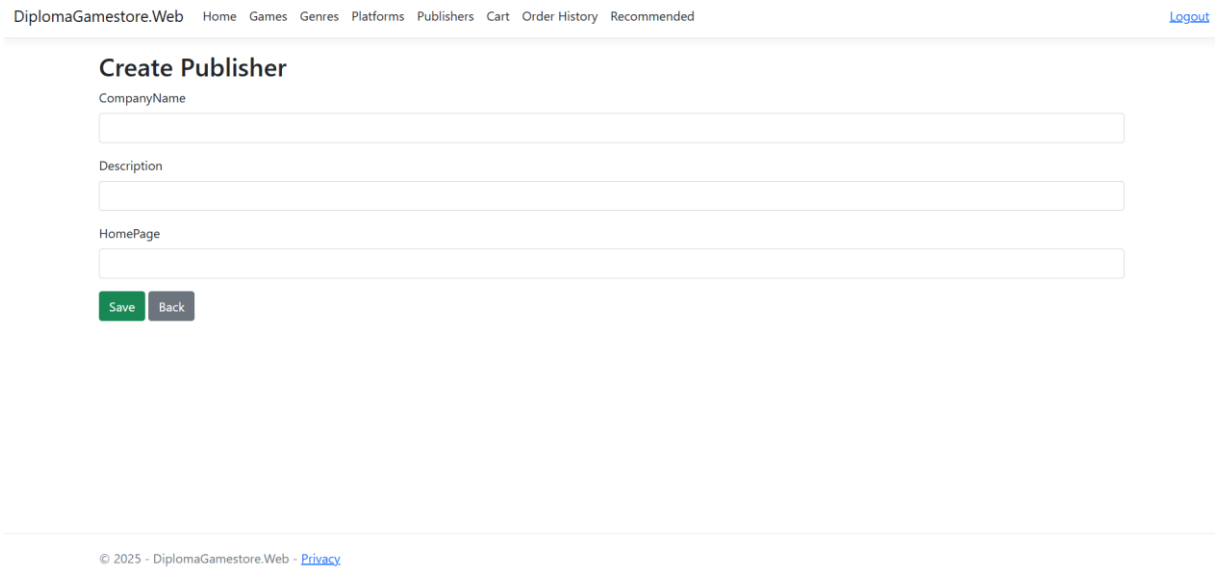

Рисунок А.15 – Сторінка створення видавця

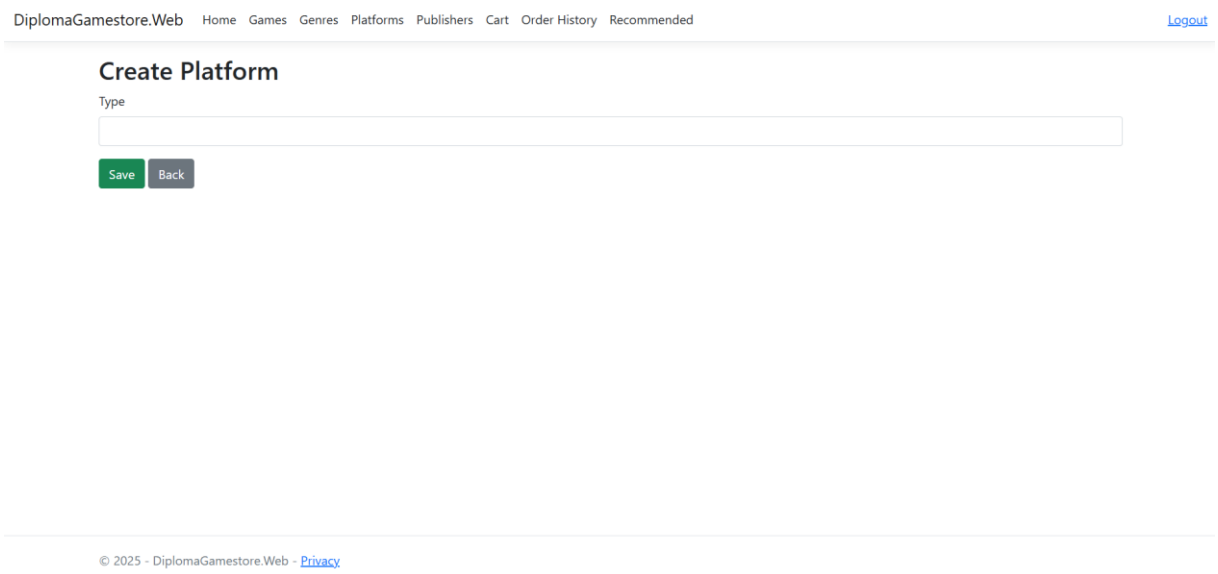

Рисунок А.16 – Сторінка створення платформи

ДОДАТОК Б

Приклади програмної реалізації

Лістинг Б.1 – Реалізація класу DimpomaGamestore.BLL.Configure

```
using DiplomaGamestore.BLL.AzureML;
using DiplomaGamestore.BLL.Mappings;
using DiplomaGamestore.BLL.Services;
using DiplomaGamestore.BLL.Services.Abstracts;
using Microsoft.Extensions.DependencyInjection;
namespace DiplomaGamestore.BLL;
public static class Configure
{
    public static void
ConfigureBusinessLogicLayerServices(this
IServiceCollection services)
    {
        services.AddAutoMapper(typeof(MappingProfile));

        services.AddHttpClient<AzureMLClient>();
        services.AddScoped<IRecommendationService,
AzureMlRecommendationService>();

        services.AddScoped<IGameService, GameService>();
        services.AddScoped<IGenreService,
GenreService>();
        services.AddScoped<IPlatformService,
PlatformService>();
        services.AddScoped<IPublisherService,
PublisherService>();
        services.AddScoped<IRatingService,
RatingService>();
        services.AddScoped<IOrderService,
OrderService>();
    }
}
```

Лістинг Б.2 – Інтерфейс репозиторію

```
using System.Linq.Expressions;
using DiplomaGamestore.DAL.Entities.Abstracts;

namespace DiplomaGamestore.DAL.Repositories.Abstracts;

public interface IRepository<T>
    where T : Entity
{
    Task<ICollection<T>> GetAsync(
        Expression<Func<T, bool>>? predicate = null,
        IEnumerable<string>? includes = null,
        Func<IQueryable<T>, IOrderedQueryable<T>>?
orderBy = null,
        bool asNoTracking = false,
        int? pageNumber = null,
        int? pageSize = null);

    Task<T?> GetByIdAsync(
        Guid id,
        IEnumerable<string>? includes = null,
        bool asNoTracking = false);

    Task<T?> GetSingleAsync(
        Expression<Func<T, bool>>? predicate = null,
        IEnumerable<string>? includes = null,
        bool asNoTracking = false);
    Task<T> CreateAsync(T entity);
    Task<T?> DeleteAsync(Guid id);

    void Delete(T entity);

    T Update(T entity);
}
```

Лістинг Б.3 – Інтерфейс Unit of Work

```
using DiplomaGamestore.DAL.Entities;
using DiplomaGamestore.DAL.Repositories.Abstracts;

namespace DiplomaGamestore.DAL.UnitOfWorks.Abstracts;

public interface IUnitOfWork : IDisposable
{
    IRepository<Game> GameRepository { get; }
    IRepository<Genre> GenreRepository { get; }
    IRepository<Platform> PlatformRepository { get; }
    IRepository<Publisher> PublisherRepository { get; }
    IRepository<Rating> RatingRepository { get; }
    IRepository<Order> OrderRepository { get; }
    IRepository<OrderDetails> OrderDetailsRepository {
get; }
    IRepository<GameGenre> GameGenreRepository { get; }
    IRepository<GamePlatform> GamePlatformRepository {
get; }

    Task SaveAsync();
}
```

Лістинг Б.4 – Реалізація класу GamestoreDbContext

```
using DiplomaGamestore.DAL.Entities;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace DiplomaGamestore.DAL.Data;

public class GamestoreDbContext :
IdentityDbContext<IdentityUser>
{
```

```
        public
GamestoreDbContext(DbContextOptions<GamestoreDbContext>
options) : base(options)
        {
        }

        public DbSet<Game> Games { get; set; }
        public DbSet<Genre> Genres { get; set; }
        public DbSet<Platform> Platforms { get; set; }
        public DbSet<Publisher> Publishers { get; set; }
        public DbSet<Rating> Ratings { get; set; }
        public DbSet<Order> Orders { get; set; }
        public DbSet<OrderDetails> OrderDetails { get; set; }
        public DbSet<GameGenre> GameGenres { get; set; }
        public DbSet<GamePlatform> GamePlatforms { get; set; }

        protected override void OnModelCreating(ModelBuilder
builder)
        {
            base.OnModelCreating(builder);

            // Genre
            builder.Entity<Genre>().HasIndex(e => e.Name,
"UX_Genre_Name").IsUnique();

            // Platform
            builder.Entity<Platform>().HasIndex(e => e.Type,
"UX_Platform_Type").IsUnique();

            // Publisher
            builder.Entity<Publisher>().HasIndex(e =>
e.CompanyName, "UX_Publisher_CompanyName").IsUnique();

            // GameGenre
            builder.Entity<GameGenre>()
                .HasIndex(
```

```
            e => new
            {
                e.GameId, e.GenreId
            },
            "UX_GameGenre_GenreId_GameId");

    // GameGenre
    builder.Entity<GamePlatform>()
        .HasIndex(
            e => new
            {
                e.GameId, e.PlatformId
            },
            "UX_GameGenre_PlatformId_GameId");

    // One to Many (Genre - Publisher)
    builder.Entity<Game>()
        .HasOne(g => g.Publisher)
        .WithMany(pg => pg.Games)
        .HasForeignKey(g => g.PublisherId)
        .OnDelete(DeleteBehavior.Cascade);

    // Many to Many (Game - Genre)
    builder.Entity<GameGenre>()
        .HasOne(gg => gg.Game)
        .WithMany(g => g.GameGenres)
        .HasForeignKey(gg => gg.GameId)
        .OnDelete(DeleteBehavior.Cascade);

    builder.Entity<GameGenre>()
        .HasOne(gg => gg.Genre)
        .WithMany(g => g.GameGenres)
        .HasForeignKey(gg => gg.GenreId)
        .OnDelete(DeleteBehavior.Cascade);

    // Many to Many (Game - Platform)
```

```
            builder.Entity<GamePlatform>()
                .HasOne(gp => gp.Game)
                .WithMany(g => g.GamePlatforms)
                .HasForeignKey(gp => gp.GameId)
                .OnDelete(DeleteBehavior.Cascade);

            builder.Entity<GamePlatform>()
                .HasOne(gp => gp.Platform)
                .WithMany(p => p.GamePlatforms)
                .HasForeignKey(gp => gp.PlatformId)
                .OnDelete(DeleteBehavior.Cascade);

            // Many to Many (Game - Order)
            builder.Entity<OrderDetails>()
                .HasOne(od => od.Game)
                .WithMany(g => g.OrderDetails)
                .HasForeignKey(od => od.GameId)
                .OnDelete(DeleteBehavior.Cascade);

            builder.Entity<OrderDetails>()
                .HasOne(od => od.Order)
                .WithMany(o => o.OrderDetails)
                .HasForeignKey(od => od.OrderId)
                .OnDelete(DeleteBehavior.Cascade);
        }
    }
```

Лістинг Б.5 – Реалізація класу GameService

```
    public class GameService : IGameService
    {
        private readonly IUnitOfWork _unitOfWork;
        private readonly IMapper _mapper;

        public GameService(IUnitOfWork unitOfWork, IMapper
mapper)
```

```
        {
            _unitOfWork = unitOfWork;
            _mapper = mapper;
        }

        public async Task<GameDto?> GetByIdAsync(Guid id)
        {
            var game = await
_unitOfWork.GameRepository.GetByIdAsync(id, new[] {
"Publisher", "GameGenres.Genre", "GamePlatforms.Platform",
"Ratings" });
            return game == null ? null :
_mapper.Map<GameDto>(game);
        }

        public async Task<ICollection<GameDto>> GetAsync()
        {
            var games = await
_unitOfWork.GameRepository.GetAsync(null, new[] { "Publisher",
"GameGenres.Genre", "GamePlatforms.Platform", "Ratings" });
            return
games.Select(_mapper.Map<GameDto>).ToList();
        }

        public async Task<GameDto> CreateAsync(GameDto
gameDto)
        {
            var game = _mapper.Map<Game>(gameDto);
            await
_unitOfWork.GameRepository.CreateAsync(game);
            await _unitOfWork.SaveAsync();
            return _mapper.Map<GameDto>(game);
        }

        public async Task<GameDto?> UpdateAsync(GameDto
gameDto)
```

```
        {
            var game = _mapper.Map<Game>(gameDto);
            _unitOfWork.GameRepository.Update(game);
            await _unitOfWork.SaveAsync();
            return _mapper.Map<GameDto>(game);
        }

        public async Task<bool> DeleteAsync(Guid id)
        {
            var result = await
_unitOfWork.GameRepository.DeleteAsync(id);
            await _unitOfWork.SaveAsync();
            return result != null;
        }
    }
```

Лістинг Б.6 – Реалізація класу AzureMLClient

```
    public class AzureMLClient
    {
        private readonly HttpClient _httpClient;
        private readonly string _endpoint;
        private readonly string _apiKey;

        public AzureMLClient(IConfiguration configuration,
HttpClient httpClient)
        {
            _httpClient = httpClient;
            _endpoint = configuration["AzureML:Endpoint"]!;
            _apiKey = configuration["AzureML:ApiKey"]!;
            _httpClient.DefaultRequestHeaders.Authorization =
new AuthenticationHeaderValue("Bearer", _apiKey);
        }

        public async Task<List<Guid>>
GetRecommendedGameIdsAsync(string userId, int count = 10)
```

```
        {
            var requestPayload = new { UserId = userId,
RecommendationCount = count };
            var content = new
StringContent(JsonSerializer.Serialize(requestPayload),
Encoding.UTF8, "application/json");
            var response = await
_httpClient.PostAsync(_endpoint, content);

            if (!response.IsSuccessStatusCode)
                throw new RequestFailedException($"Azure ML
request failed: {response.StatusCode}");

            var responseContent = await
response.Content.ReadAsStringAsync();
            var result =
JsonSerializer.Deserialize<AzureMLResponse>(responseContent);
            return result?.RecommendedGameIds ?? new
List<Guid>();
        }

        private class AzureMLResponse
        {
            public List<Guid> RecommendedGameIds { get; set; }
        }
}
```

Лістинг Б.6 – Реалізація класу GameController

```
using AutoMapper;
using DiplomaGamestore.BLL.DTOs;
using DiplomaGamestore.BLL.Services.Abstracts;
using DiplomaGamestore.Web.Models;
using Microsoft.AspNetCore.Mvc;

namespace DiplomaGamestore.Web.Controllers;
```

```
public class GamesController : Controller
{
    private readonly IGameService _gameService;
    private readonly IGenreService _genreService;
    private readonly IMapper _mapper;
    private readonly IPlatformService _platformService;
    private readonly IPublisherService _publisherService;

    public GamesController(
        IGameService gameService,
        IGenreService genreService,
        IPlatformService platformService,
        IPublisherService publisherService,
        IMapper mapper)
    {
        _gameService = gameService;
        _genreService = genreService;
        _platformService = platformService;
        _publisherService = publisherService;
        _mapper = mapper;
    }

    public async Task<IActionResult> Index()
    {
        var games = await _gameService.GetAsync();
        var vms = _mapper.Map<List<GameViewModel>>(games);
        return View(vms);
    }

    public async Task<IActionResult> Details(Guid id)
    {
        var game = await _gameService.GetByIdAsync(id);
        if (game == null) return NotFound();
        var vm = _mapper.Map<GameViewModel>(game);
        return View(vm);
    }
```

```
    public async Task<IActionResult> Create()
    {
        var vm = new GameCreateEditViewModel();
        await PopulateOptions(vm);
        return View(vm);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult>
Create(GameCreateEditViewModel vm)
    {
        if (!ModelState.IsValid)
        {
            await PopulateOptions(vm);
            return View(vm);
        }

        var dto = _mapper.Map<GameDto>(vm);
        dto.PublisherId = vm.PublisherId;
        dto.Genres = vm.SelectedGenreIds.Select(id => new
GenreDto { Id = id }).ToList();
        dto.Platforms = vm.SelectedPlatformIds.Select(id
=> new PlatformDto { Id = id }).ToList();
        await _gameService.CreateAsync(dto);
        return RedirectToAction(nameof(Index));
    }

    public async Task<IActionResult> Edit(Guid id)
    {
        var game = await _gameService.GetByIdAsync(id);
        if (game == null) return NotFound();
        var vm =
_mapper.Map<GameCreateEditViewModel>(game);
        await PopulateOptions(vm);
```

```
            return View(vm);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(Guid id,
GameCreateEditViewModel vm)
        {
            if (id != vm.Id) return BadRequest();
            if (!ModelState.IsValid)
            {
                await PopulateOptions(vm);
                return View(vm);
            }

            var dto = _mapper.Map<GameDto>(vm);
            dto.PublisherId = vm.PublisherId;
            dto.Genres = vm.SelectedGenreIds.Select(gid => new
GenreDto { Id = gid }).ToList();
            dto.Platforms = vm.SelectedPlatformIds.Select(pid
=> new PlatformDto { Id = pid }).ToList();
            await _gameService.UpdateAsync(dto);
            return RedirectToAction(nameof(Index));
        }

        public async Task<IActionResult> Delete(Guid id)
        {
            var game = await _gameService.GetByIdAsync(id);
            if (game == null) return NotFound();
            var vm = _mapper.Map<GameViewModel>(game);
            return View(vm);
        }

        [HttpPost]
        [ActionName("Delete")]
        [ValidateAntiForgeryToken]
```

```
    public async Task<IActionResult> DeleteConfirmed(Guid
id)
    {
        await _gameService.DeleteAsync(id);
        return RedirectToAction(nameof(Index));
    }

    private async Task
PopulateOptions(GameCreateEditViewModel vm)
    {
        vm.AllPublishers = (await
_publisherService.GetAsync())
            .Select(p => new PublisherViewModel { Id =
p.Id, CompanyName = p.CompanyName }).ToList();
        vm.AllGenres = (await _genreService.GetAsync())
            .Select(g => new GenreViewModel { Id = g.Id,
Name = g.Name }).ToList();
        vm.AllPlatforms = (await
_platformService.GetAsync())
            .Select(p => new PlatformViewModel { Id =
p.Id, Type = p.Type }).ToList();
    }
}
```

Лістинг Б.7 – Реалізація класу GameViewModel

```
using System.ComponentModel.DataAnnotations;

namespace DiplomaGamestore.Web.Models;

public class GameViewModel
{
    public Guid Id { get; set; }

    [Required] [StringLength(50)] public string Name {
get; set; }
```

```
    public string? Description { get; set; }

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

    [Range(0, 100)] public int Discount { get; set; }

    public long ViewCount { get; set; }
    public DateTime DateCreated { get; set; }

    public Guid PublisherId { get; set; }
    public string PublisherName { get; set; }

    public List<Guid> SelectedGenreIds { get; set; } =
new();
    public List<Guid> SelectedPlatformIds { get; set; } =
new();

    public List<string> GenreNames { get; set; } = new();
    public List<string> PlatformTypes { get; set; } =
new();

    public double AverageRating { get; set; }
}
```

Лістинг Б.8 – Реалізація класу GameCreateEditViewModel

```
using System.ComponentModel.DataAnnotations;

namespace DiplomaGamestore.Web.Models;

public class GameCreateEditViewModel
{
    public Guid Id { get; set; }
```

```
    [Required] [StringLength(50)] public string Name {
get; set; }

    public string? Description { get; set; }

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

    [Range(0, 100)] public int Discount { get; set; }

    [Required] public Guid PublisherId { get; set; }

    [MinLength(1, ErrorMessage = "At least one genre must
be selected.")]
    public List<Guid> SelectedGenreIds { get; set; } =
new();

    [MinLength(1, ErrorMessage = "At least one platform
must be selected.")]
    public List<Guid> SelectedPlatformIds { get; set; } =
new();

    // For rendering options
    public List<PublisherViewModel> AllPublishers { get;
set; } = new();
    public List<GenreViewModel> AllGenres { get; set; } =
new();
    public List<PlatformViewModel> AllPlatforms { get;
set; } = new();
}
```

ДОДАТОК В

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	82 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Тітаренко К.О.			Розробка системи рекомендацій ігор у вебзастосунку «Магазин ігор» з використанням Microsoft Azure ML	Шифр групи	Код напр./спец.
Перевірив		Волощук О.Б.				ІТШІ-21-4	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					