- Arul
- Asuvath
- Gokul
- Gokulraj
- Iswarya
- Kavin Kumar
- Kirubaharan
- Prathima
- Ragunath
- Sobhana
- Sruthi

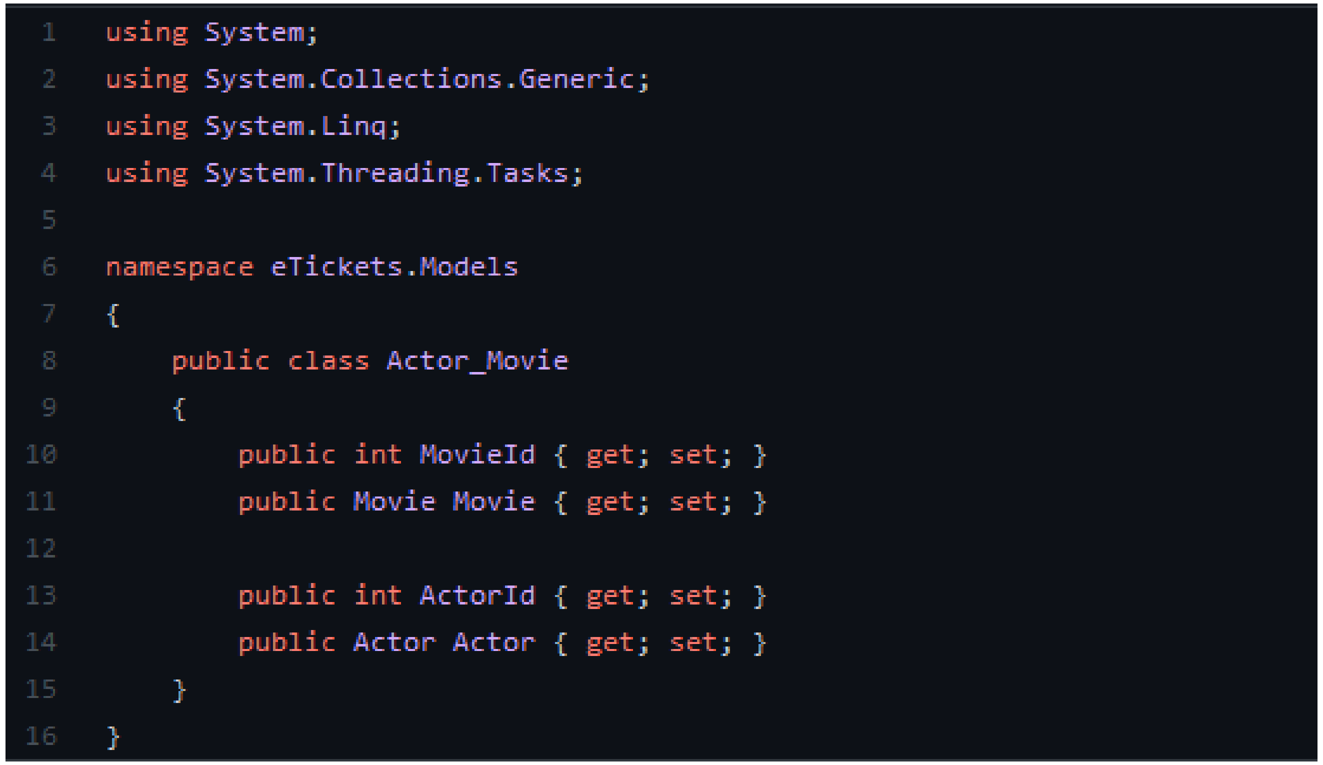
SOLID PRINCIPLES

- EUPHORIA♥☺

aspire SYSTEMS

attention.
always.

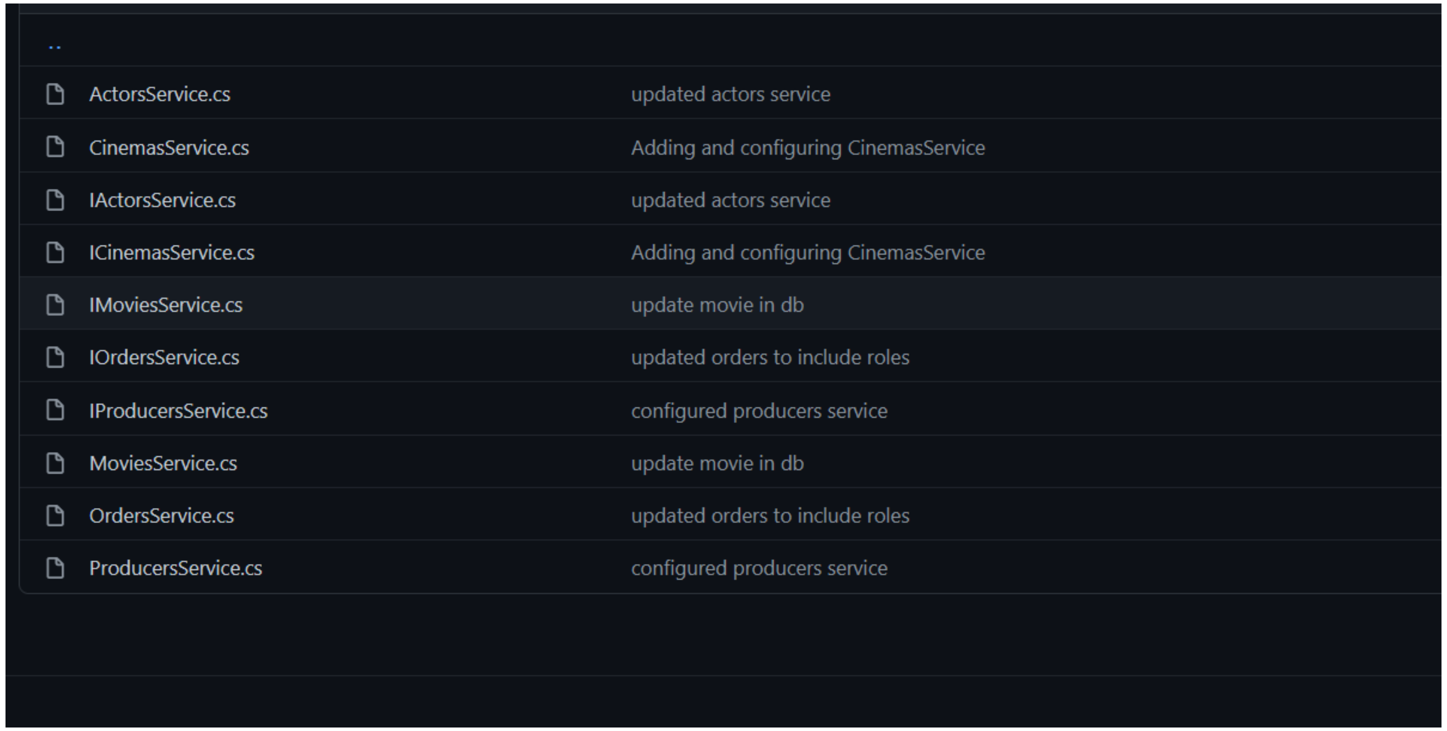
1.Single responsibility Principle



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace eTickets.Models
{
    public class Actor_Movie
    {
        public int MovieId { get; set; }
        public Movie Movie { get; set; }

        public int ActorId { get; set; }
        public Actor Actor { get; set; }
    }
}
```

a t t e n t i o n.
a l w a y s.

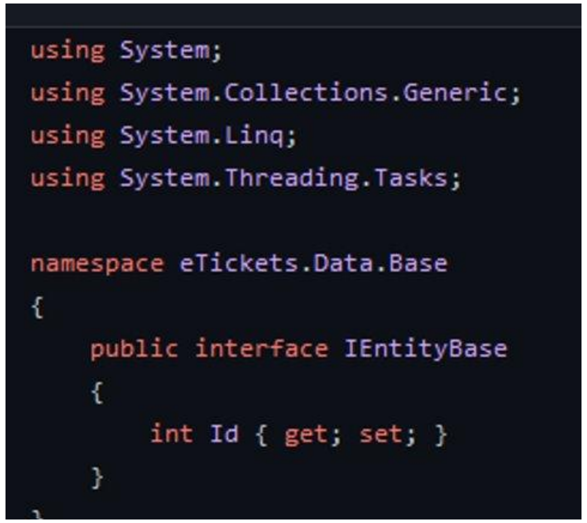
2.Open/Closed Prin



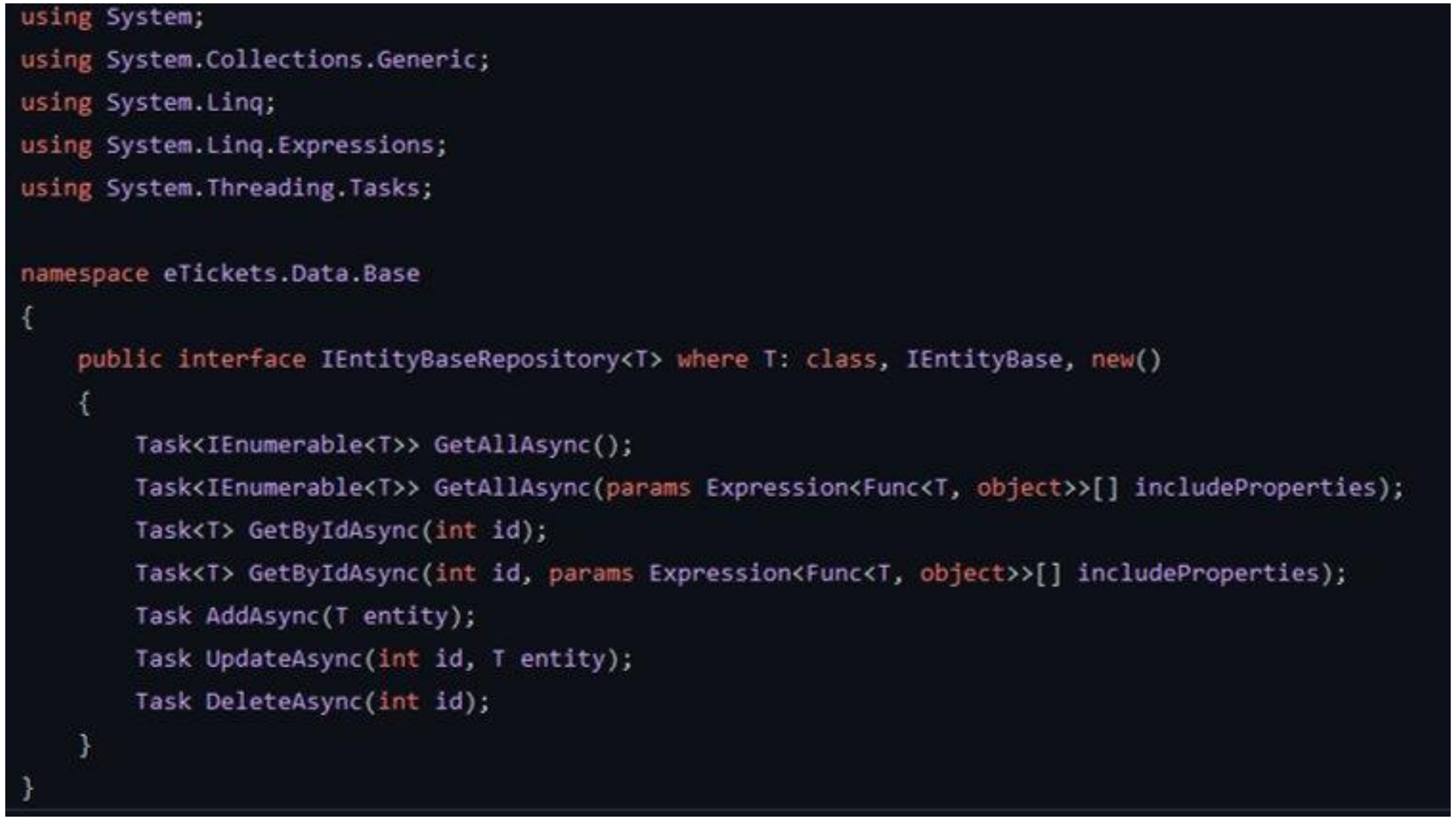
..	
ActorsService.cs	updated actors service
CinemasService.cs	Adding and configuring CinemasService
IActorsService.cs	updated actors service
ICinemasService.cs	Adding and configuring CinemasService
IMoviesService.cs	update movie in db
IOrdersService.cs	updated orders to include roles
IProducersService.cs	configured producers service
MoviesService.cs	update movie in db
OrdersService.cs	updated orders to include roles
ProducersService.cs	configured producers service

a t t e n t i o n.
a l w a y s.

3. Liskov Substitution

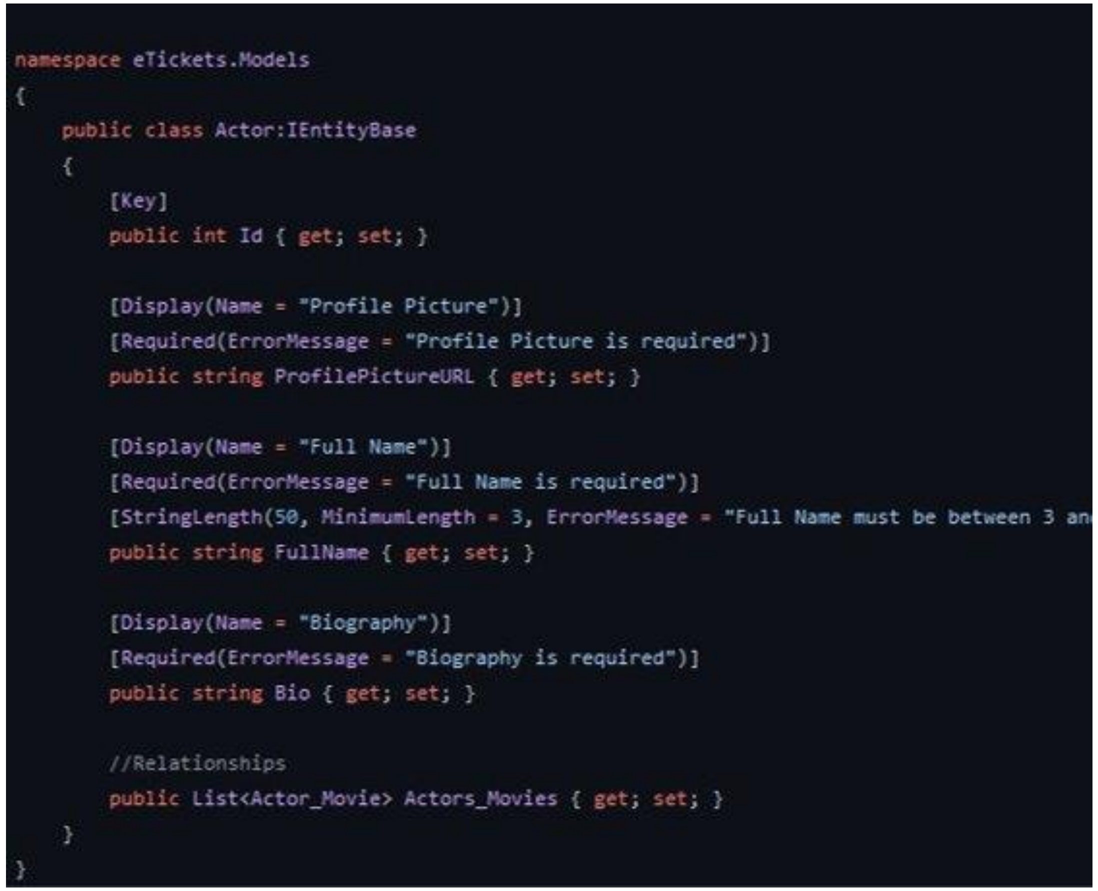


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace eTickets.Data.Base
{
    public interface IEntityBase
    {
        int Id { get; set; }
    }
```



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Threading.Tasks;

namespace eTickets.Data.Base
{
    public interface IEntityBaseRepository<T> where T: class, IEntityBase, new()
    {
        Task<IEnumerable<T>> GetAllAsync();
        Task<IEnumerable<T>> GetAllAsync(params Expression<Func<T, object>>[] includeProperties);
        Task<T> GetByIdAsync(int id);
        Task<T> GetByIdAsync(int id, params Expression<Func<T, object>>[] includeProperties);
        Task AddAsync(T entity);
        Task UpdateAsync(int id, T entity);
        Task DeleteAsync(int id);
    }
}
```

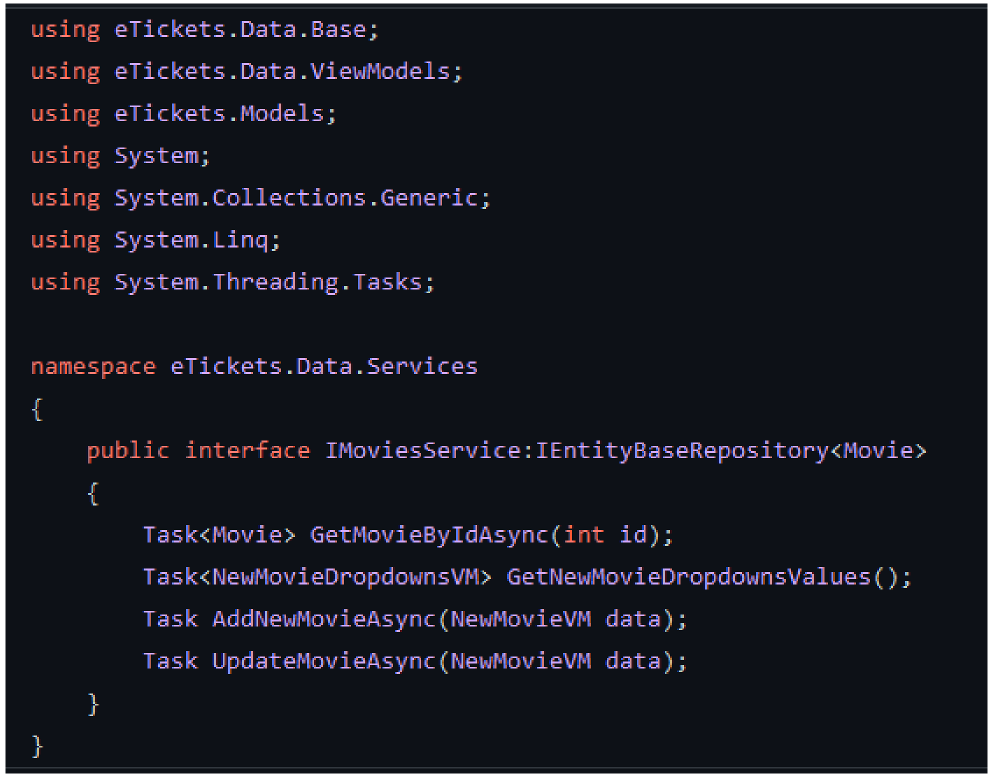
a t t e n t i o n.
a l w a y s.



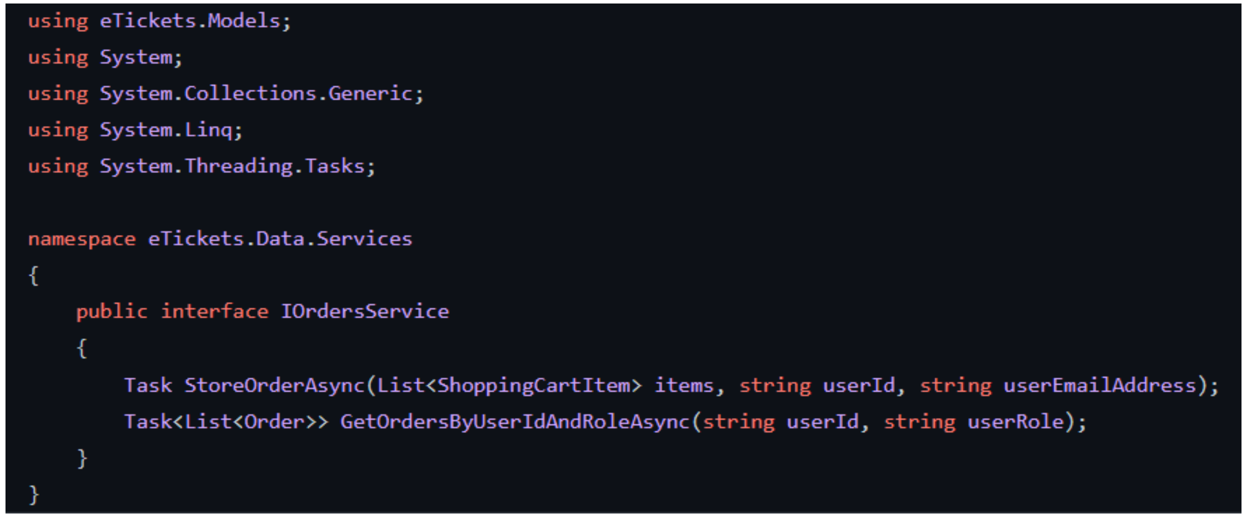
```
namespace eTickets.Models
{
    public class Actor:IEntityBase
    {
        [Key]
        public int Id { get; set; }

        [Display(Name = "Profile Picture")]
        [Required(ErrorMessage = "Profile Picture is required")]
        public string ProfilePictureURL { get; set; }

        [Display(Name = "Full Name")]
        [Required(ErrorMessage = "Full Name is required")]
        [StringLength(50, MinimumLength = 3, ErrorMessage = "Full Name must be between 3 an
        public string FullName { get; set; }

        [Display(Name = "Biography")]
        [Required(ErrorMessage = "Biography is required")]
        public string Bio { get; set; }

        //Relationships
        public List<Actor_Movie> Actors_Movies { get; set; }
    }
}
```

a t t e n t i o n.
a l w a y s.

4.Interface Separation

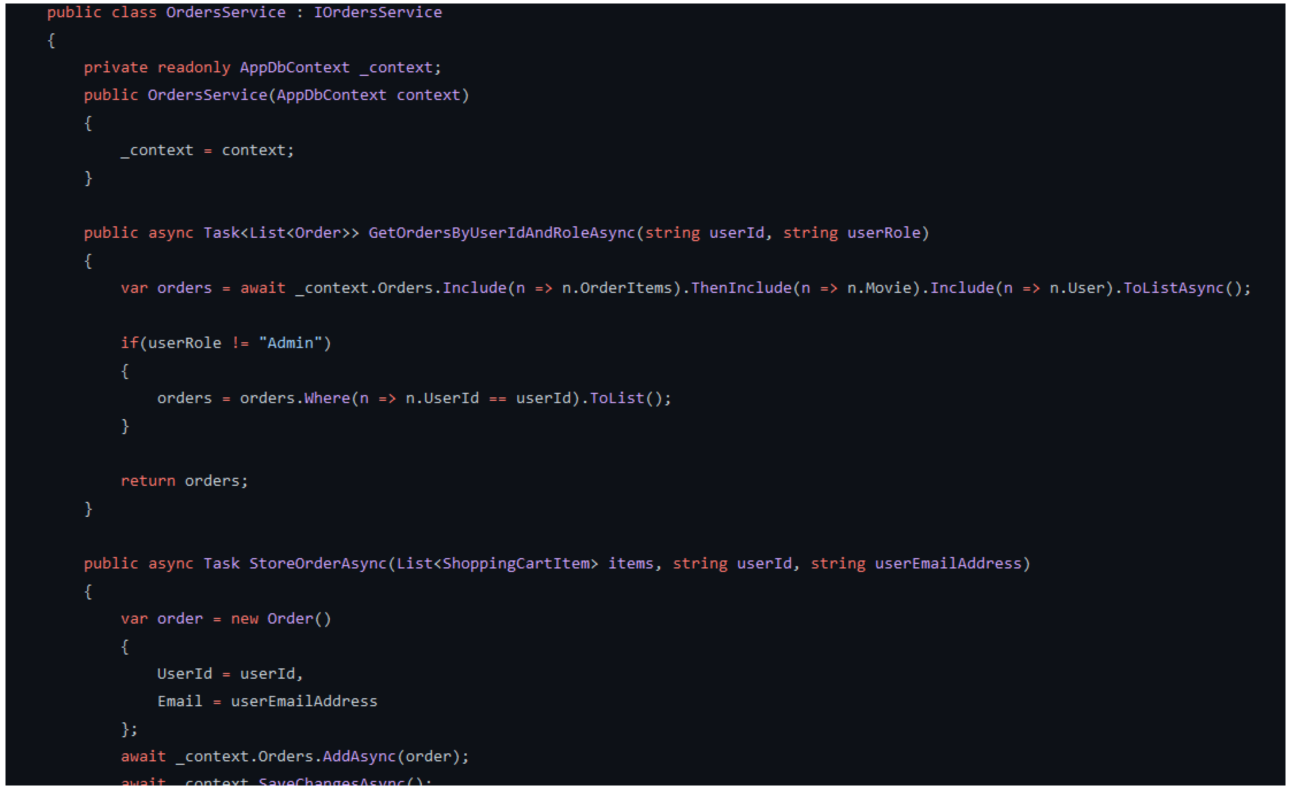


```csharp
using eTickets.Data.Base;
using eTickets.Data.ViewModels;
using eTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace eTickets.Data.Services
{
    public interface IMoviesService:IEntityBaseRepository<Movie>
    {
        Task<Movie> GetMovieByIdAsync(int id);
        Task<NewMovieDropdownsVM> GetNewMovieDropdownsValues();
        Task AddNewMovieAsync(NewMovieVM data);
        Task UpdateMovieAsync(NewMovieVM data);
    }
}
```

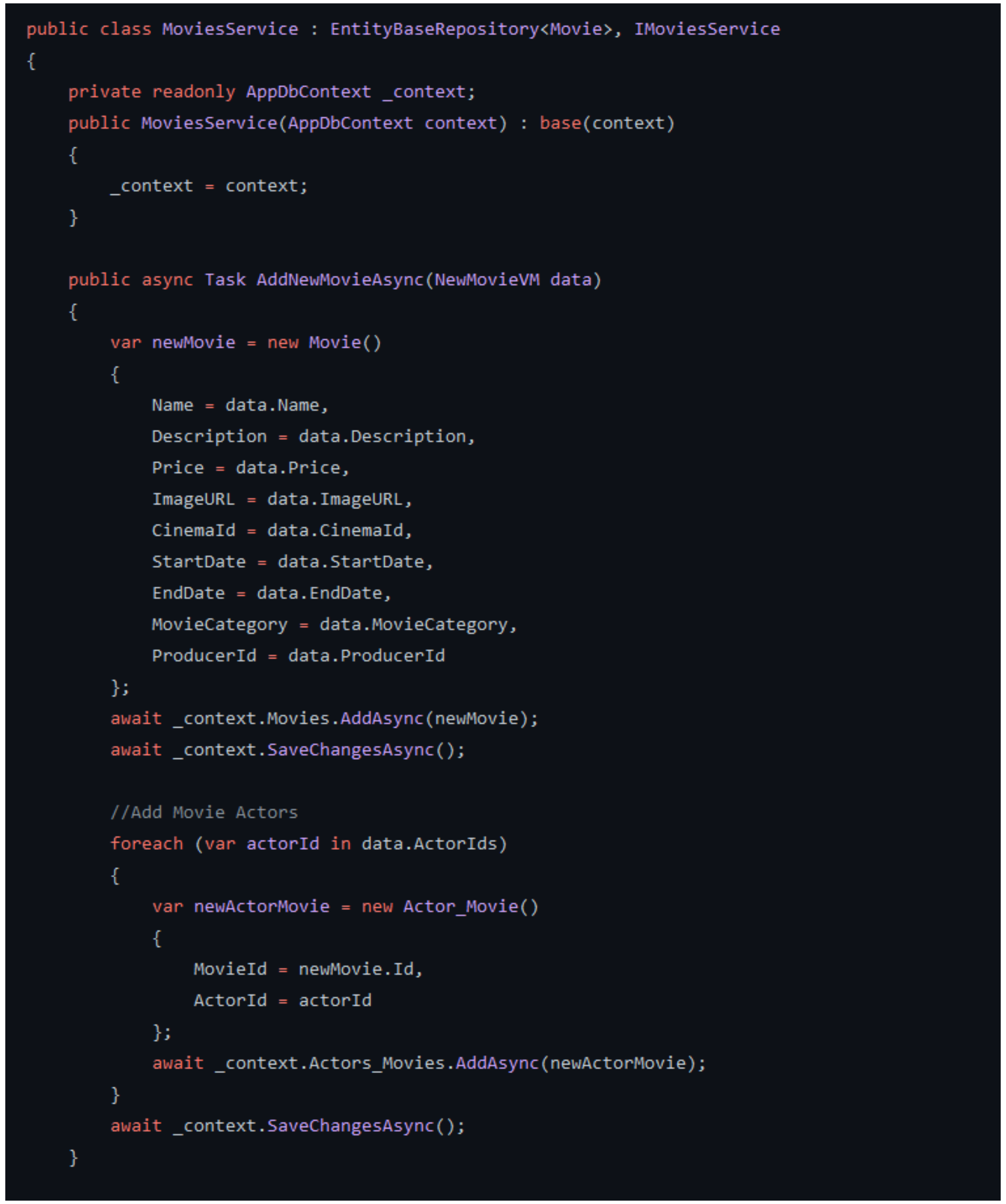


```csharp
using eTickets.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace eTickets.Data.Services
{
    public interface IOrdersService
    {
        Task StoreOrderAsync(List<ShoppingCartItem> items, string userId, string userEmailAddress);
        Task<List<Order>> GetOrdersByUserIdAndRoleAsync(string userId, string userRole);
    }
}
```

a t t e n t i o n.
a l w a y s.



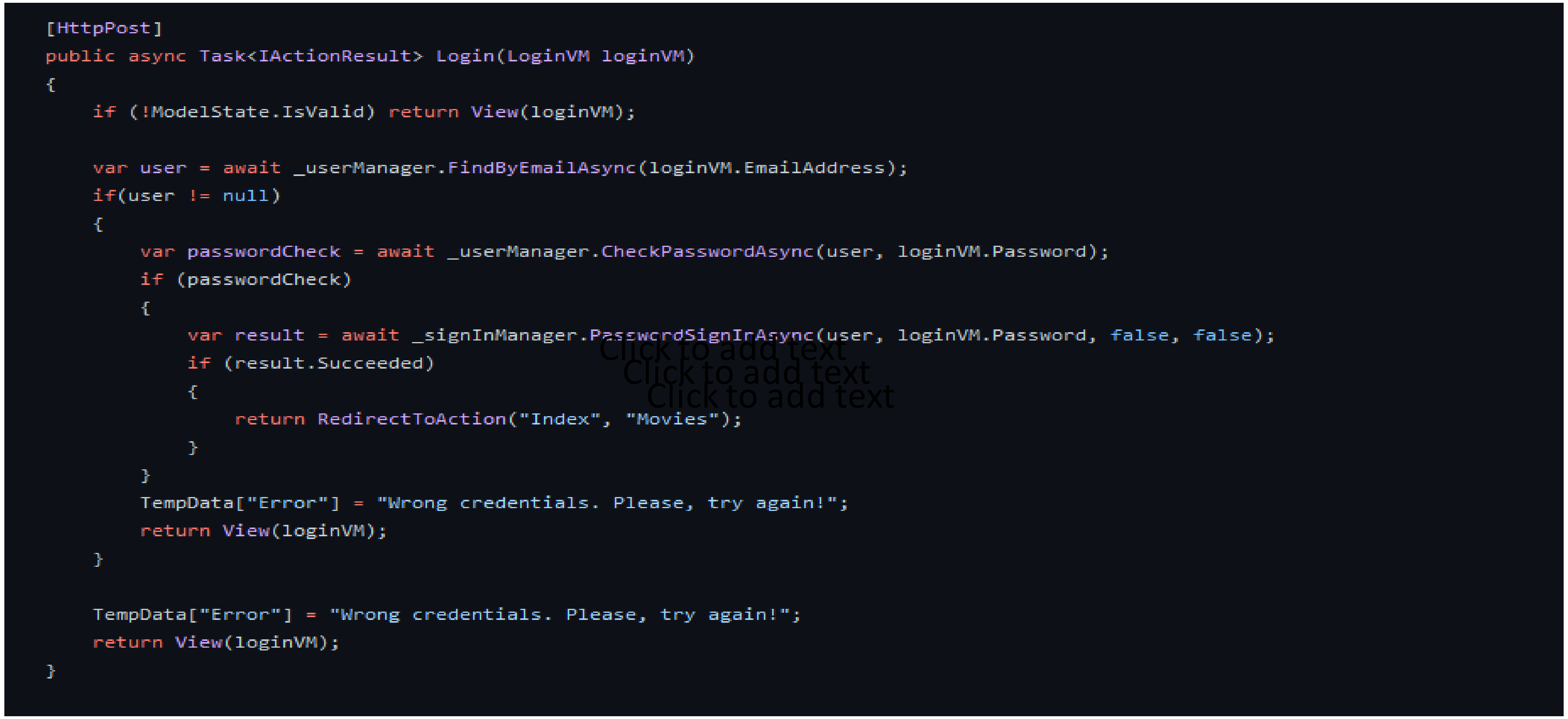
```
public class OrdersService : IOrdersService
{
    private readonly AppDbContext _context;
    public OrdersService(AppDbContext context)
    {
        _context = context;
    }

    public async Task<List<Order>> GetOrdersByUserIdAndRoleAsync(string userId, string userRole)
    {
        var orders = await _context.Orders.Include(n => n.OrderItems).ThenInclude(n => n.Movie).Include(n => n.User).ToListAsync();

        if(userRole != "Admin")
        {
            orders = orders.Where(n => n.UserId == userId).ToList();
        }

        return orders;
    }

    public async Task StoreOrderAsync(List<ShoppingCartItem> items, string userId, string userEmailAddress)
    {
        var order = new Order()
        {
            UserId = userId,
            Email = userEmailAddress
        };
        await _context.Orders.AddAsync(order);
        await _context.SaveChangesAsync();
```



```
public class MoviesService : EntityBaseRepository<Movie>, IMoviesService
{
    private readonly AppDbContext _context;
    public MoviesService(AppDbContext context) : base(context)
    {
        _context = context;
    }

    public async Task AddNewMovieAsync(NewMovieVM data)
    {
        var newMovie = new Movie()
        {
            Name = data.Name,
            Description = data.Description,
            Price = data.Price,
            ImageURL = data.ImageURL,
            CinemaId = data.CinemaId,
            StartDate = data.StartDate,
            EndDate = data.EndDate,
            MovieCategory = data.MovieCategory,
            ProducerId = data.ProducerId
        };
        await _context.Movies.AddAsync(newMovie);
        await _context.SaveChangesAsync();

        //Add Movie Actors
        foreach (var actorId in data.ActorIds)
        {
            var newActorMovie = new Actor_Movie()
            {
                MovieId = newMovie.Id,
                ActorId = actorId
            };
            await _context.Actors_Movies.AddAsync(newActorMovie);
        }
        await _context.SaveChangesAsync();
    }
```

a t t e n t i o n.
a l w a y s.

5. Dependency Inversion



```
[HttpPost]
public async Task<IActionResult> Login(LoginVM loginVM)
{
    if (!ModelState.IsValid) return View(loginVM);

    var user = await _userManager.FindByEmailAsync(loginVM.EmailAddress);
    if(user != null)
    {
        var passwordCheck = await _userManager.CheckPasswordAsync(user, loginVM.Password);
        if (passwordCheck)
        {
            var result = await _signInManager.PasswordSignInAsync(user, loginVM.Password, false, false);
            if (result.Succeeded)
            {
                return RedirectToAction("Index", "Movies");
            }
        }
        TempData["Error"] = "Wrong credentials. Please, try again!";
        return View(loginVM);
    }

    TempData["Error"] = "Wrong credentials. Please, try again!";
    return View(loginVM);
}
```

Click to add text
Click to add text
Click to add text

a t t e n t i o n.
a l w a y s.

Thank you

for your time

aspire SYSTEMS | attention. always.