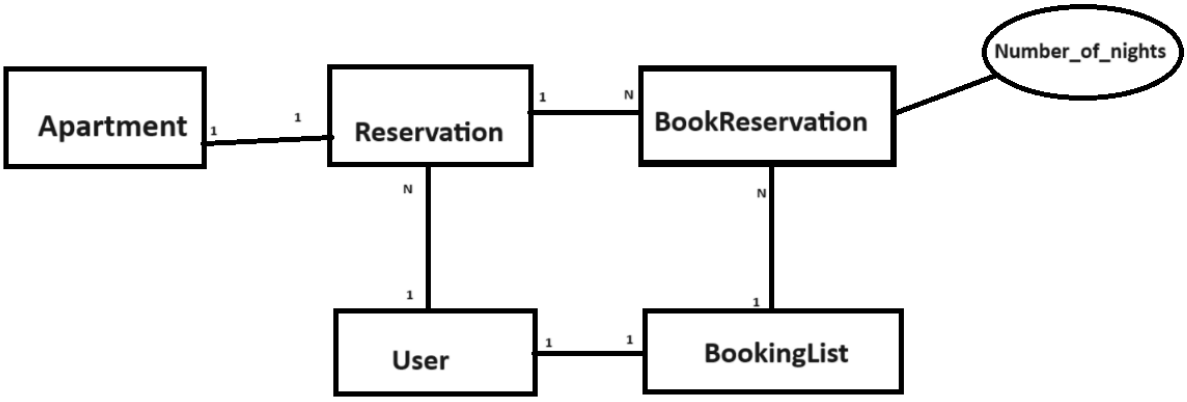
Лабораториска вежба 2 (Група Б) / Laboratory exercise 2 (Group B)

Апликацијата ги опфаќа следниве карактеристики:

- Додавање на Апартман.
- Правење Резервација за даден апартман.

Дополнително, апликацијата треба да ги опфаќа следните карактеристики:

- Нов контролер: BookingListController, такашто треба да соддржи функција "BookNow()" со цел да ја испразни листата и да направи нарачка за резервација.
- Додавање и бришење на Резервации во BookingList.
- Преглед на Резервации во BookingList со целосната сума, пресметана како цена за ноќевање * број на ноќевања.



INTEGRATED SYSTEMS CHECKPOINTS

Presented By Berat

STEP 1: MODELS

```
public class Reservation
{
    [Key]
    public Guid Id { get; set; }
    [Required]
    public DateTime Check_in_date { get; set; }
    public Guid ApartmentId { get; set; }
    public Apartment? Apartment { get; set; }
    public virtual BookingApplicationUser? User { get; set; }
    public string? UserId { get; set; }
}
```

```
public class BookReservation
{
    [Key]
    public Guid Id { get; set; }

    [Required]
    public int Number_Of_Nights { get; set; }

    public Reservation? Reservation { get; set; }
    public Guid? ReservationId { get; set; }

    public BookingList? BookingList { get; set; }
    public Guid? BookingListId { get; set; }

    public BookingApplicationUser? User { get; set; }
    public string? UserId { get; set; }

}
```

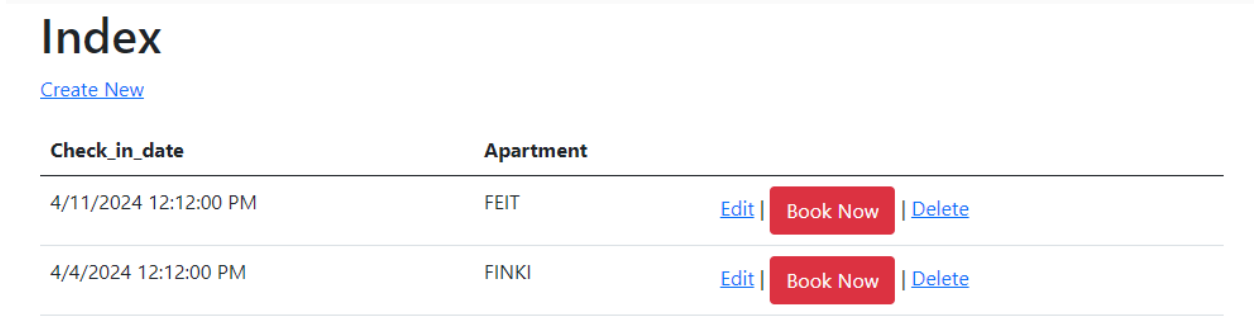
```
    public class BookingList
    {
        [Key]
        public Guid Id { get; set; }
        [Required]
        public ICollection<BookReservation>? BookReservations { get; set; }
        [Required]
        public int Full_Price { get; set; }
        public BookingApplicationUser? User { get; set; }
        public string? UserId { get; set; }
    }
```

//**Reservation** acts as a model that shows users what they can book
//**BookingReservation** acts as a shopping cart, users can add to cart their bookings
//**BookingList** acts as a order, when users checkout, all their momentarily bookings from cart get listed here

STEP 2 : VIEW MODIFICATIONS
After the models have been migrated and database updated, create controllers by scaffolding and than modify the **Shared/Layout.cshtml** to show button for the new models in the nav-bar

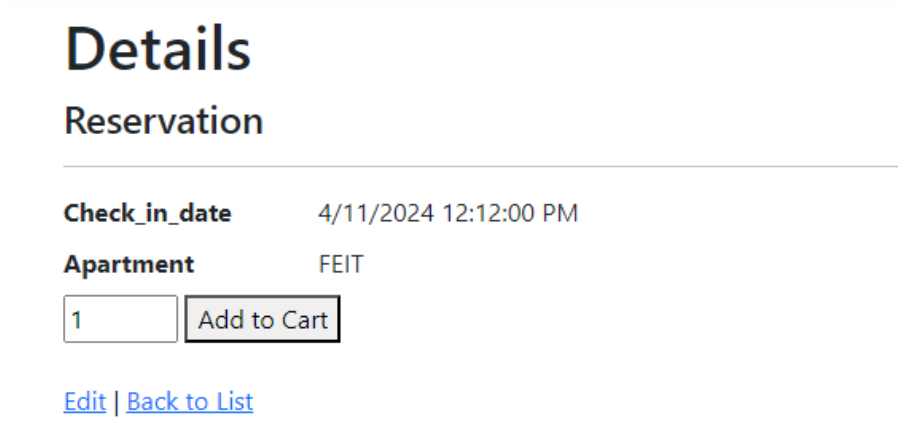
```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Reservations"
asp-action="Index">Reservations</a>
</li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area=""
asp-controller="BookReservations" asp-action="Index">Book Reservations</a>
</li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="BookingLists"
asp-action="Index">Checkout</a>
</li>
```

Now modify the Reservation View to display a ‘Book Now’ button, to do this use the button that displays Details page, and we modify it below for our use




```
                    <a asp-action="Details" asp-route-id="@item.Id">
<button type="submit" class="btn btn-danger">Book Now</button></a> |
```

Now button redirects to Details view, so modify the details view and add input for how many nights a user wants to book apartment



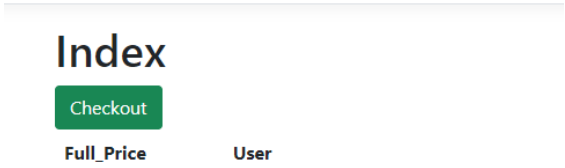

```
 <form asp-action="AddToCart" method="post">
     <input type="hidden" name="ReservationId" value="@Model.Id" />
     <input type="number" name="Number_Of_Nights" value="1" min="1"
max="99" required />
     <button type="submit">Add to Cart</button>
 </form>
```

STEP 3: CONTROLLERS

Now create a AddToCart method which will take ReservationId and Number_Of_Nights from the form

```
//POST: Reservations/AddToCart/5
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> AddToCart(Guid ReservationId, int Number_Of_Nights)
{
    //Find the reservation by ID
    var reservation = await _context.Reservations.FindAsync(ReservationId);

    // Get the ID of the currently authenticated user
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
    if(userId == null)
    {
        return RedirectToAction("Login", "Account");
       // Redirect to login if user is not authenticated
    }

    //Make a BookReservation
    var BookReservation = new BookReservation
    {
        Id = Guid.NewGuid(),
        ReservationId = reservation.Id,
        UserId = userId,
        Number_Of_Nights = Number_Of_Nights
    };

    //Add to the database
    _context.Add(BookReservation);

    //Save changed and redirect
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

Congrats, the cart system should work, now before moving to the Checkout system, create button for it on BookingList Index view




```
<!-- Button for CHECKOUT Post method controller-->
<form asp-action="Checkout" method="post">
    <button type="submit" class="btn btn-success">Checkout</button>
 </form>
```

Now create the `Checkout` method on BookingListController

```
// POST: BookingLists/Checkout
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Checkout()
{
    // Find logged-in user authenticated
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
    if (userId == null)
    {
        return RedirectToAction("Login", "Account"); // Redirect login if user not authenticated
    }
        var bookingList = new BookingList
        {
            Id = Guid.NewGuid(),
            UserId = userId.ToString(),
            BookReservations = _context.BookReservation.Where(b => b.UserId == userId).ToList(),
            Full_Price = _context.BookReservation.Where(b => b.UserId == userId).Sum(b =>
b.Reservation.Apartment.Price_per_night * b.Number_Of_Nights)
        };
        _context.Add(bookingList);

    // Delete the booking reservations of the user
    _context.BookReservation.RemoveRange(_context.BookReservation.Where(b => b.UserId == userId));

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```


That's all Folks!