Start from page number 3

BookingController

```
using eCommerceAndBooking.Data;
using eCommerceAndBooking.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Threading.Tasks;

namespace eCommerceAndBooking.Controllers
{
    public class BookingController : Controller
    {
        private readonly ApplicationDbContext context;
        private readonly UserManager<ApplicationUser> userManager;

        public BookingController(
            ApplicationDbContext context, UserManager<ApplicationUser> userManager)
        {
            this.context = context;
            this.userManager = userManager;
        }

        public IActionResult Create()
        {
            ViewBag.TimeSlots = new List<string> {
                "9:00", "9:30", "10:00", "10:30", "11:00", "11:30", "12:00"
            };

            return View();
        }

        [HttpPost]
        public async Task<IActionResult> Create(BookingDTO bookingDTO)
        {

            ViewBag.TimeSlots = new List<string> {
                "9:00", "9:30", "10:00", "10:30", "11:00", "11:30", "12:00"
            };

            if (!ModelState.IsValid)
            {
                return View(bookingDTO);
            }

            var dateTime = DateTime.Parse($"{bookingDTO.Date:yyyy-MM-dd}
{bookingDTO.Time}");

            var user = await userManager.GetUserAsync(User);
            Booking newBooking = new Booking
            {
                BookingDateTime = dateTime,
                Status = "Pending",
                UserId = user.Id
            };
```

```
            // Add the new booking to the database
            context.Bookings.Add(newBooking);

            // Save changes to actually store it
            await context.SaveChangesAsync();

            return RedirectToAction("CustomerBookings");

        }

        //Customer Booking Status Page
        public async Task<IActionResult> CustomerBookings()
        {
            var user = await userManager.GetUserAsync(User);

            var bookings = await context.Bookings
                .Where(b => b.UserId == user.Id)
                .ToListAsync();

            return View(bookings);
        }



    }
}
```

Create Booking Model

```
public class Booking
{
    public int Id { get; set; }

    public DateTime BookingDateTime { get; set; }

    public string Status { get; set; } // Pending, Confirmed, Cancelled

    public bool IsSlotBlocked { get; set; }

    // User information
    public string UserId { get; set; } // foreign key
    public ApplicationUser User { get; set; } // navigation property
}
```

Add Booking table to ApplicationDbContext

```
using eCommerceAndBooking.Models;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace eCommerceAndBooking.Data
{
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext(DbContextOptions options) : base(options)
        {

        }

        public DbSet<Product> Products { get; set; }

        public DbSet<Cart> Carts { get; set; }
        public DbSet<CartItem> CartItems { get; set; }
        public DbSet<Booking> Bookings { get; set; }

    }
}
```

add-migrationBookingsTable
update-database

Create BookingDTO

```
public class BookingDTO
 {
     [Required]
     [DataType(DataType.Date)]
     public DateTime Date { get; set; }

     [Required]
     public string Time { get; set; } // "09:00", "09:30"
 }
```

Create BookingController

```
namespace eCommerceAndBooking.Controllers
{
    public class BookingController : Controller
    {

    }
}
```

Create a constructor in the controller and pass as parameters
`ApplicationDbContext context, UserManager<ApplicationUser> userManager`

Create and assign fields

```
    public class BookingController : Controller
    {
        private readonly ApplicationDbContext context;
        private readonly UserManager<ApplicationUser> userManager;

        public BookingController(
            ApplicationDbContext context, UserManager<ApplicationUser> userManager)
        {
            this.context = context;
            this.userManager = userManager;
        }

    }
```

Create an action method called **Create**
The action method should have a **list of time slots**. Store the list of time slots in `ViewBag.TimeSlots` to pass to the **view**.

```
public IActionResult Create()
 {
     ViewBag.TimeSlots = new List<string> {

         "9:00", "9:30", "10:00", "10:30", "11:00", "11:30", "12:00", "12:30"
     };

     return View();
 }
```

Add Booking to the navbar

```
<li class="nav-item">
     <a class="nav-link" asp-area="" asp-controller="Booking"
       asp-action="Create">Booking</a>
 </li>
```

Create the View to display the UI for booking

```
@model BookingDTO
@{
    var slots = ViewBag.TimeSlots as List<string>;
}

<h3>Book Appointment</h3>

<form method="post">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>

    <label>Date</label>
    <input asp-for="Date" type="date" class="form-control" />

    <label class="mt-3">Time</label>
    <select asp-for="Time" class="form-control">
        @foreach (var slot in slots)
        {
            <option value="@slot">@slot</option>
        }
    </select>

    <button class="btn btn-primary mt-3">Book</button>
</form>
```

Create [HttpPost] method to handle booking details submitted by the user

- Repopulate the time slots because ViewBag data does not persist between requests. Else the dropdown (or list) in the view will be empty.
- Check if the submitted `BookingDTO` passes validation, and if not, return the same view with the entered data so validation errors can be displayed.

```
[HttpPost]
public async Task<IActionResult> Create(BookingDTO bookingDTO)
{
    ViewBag.TimeSlots = new List<string>
    {
        "09:00", "09:30", "10:00", "10:30",
        "11:00", "11:30", "12:00"
    };

    if (!ModelState.IsValid)

        return View(bookingDTO);

}
```

Combine the separate Date and Time values into one complete DateTime value that can be checked from the database.

- `Date: 2026-02-15`
- `Time: "09:30"`

Combined

2026-02-15 09:30:00

```
var dateTime = DateTime.Parse($"{bookingDTO.Date:yyyy-MM-dd} {bookingDTO.Time}");
```

Get the logged-in user's details using `userManager.GetUserAsync(User);`

```
var user = await userManager.GetUserAsync(User);
```

Create a new booking record for the logged-in user

```
Booking newBooking = new Booking
 {
      BookingDateTime = dateTime,
      Status = "Pending",
      UserId = user.Id

  };
```

- `BookingDateTime = dateTime`: sets when the booking will happen
- `Status = "Pending"`: marks the booking as not yet confirmed
- `UserId = user.Id`: links the booking to the currently logged-in user

```
await context.SaveChangesAsync();

return RedirectToAction("CustomerBookings");
```

```
[HttpPost]
public async Task<IActionResult> Create(BookingDTO bookingDTO)
{

    ViewBag.TimeSlots = new List<string> {
        "9:00", "9:30", "10:00", "10:30", "11:00", "11:30", "12:00"
    };

    if (!ModelState.IsValid)
    {
        return View(bookingDTO);
    }

    var dateTime = DateTime.Parse($"{bookingDTO.Date:yyyy-MM-dd} {bookingDTO.Time}");

    var user = await userManager.GetUserAsync(User);


    Booking newBooking = new Booking
    {
        BookingDateTime = dateTime,
        Status = "Pending",
        UserId = user.Id
    };

    await context.SaveChangesAsync();

    return RedirectToAction("CustomerBookings");

}
```

This code gets the currently logged-in user, retrieves all bookings for that user from the database, and passes them to the view to be displayed.

Create **CustomerBookings()** method to get the currently logged-in user details.

Retrieve all bookings for that user from the database, and pass them to the view to be displayed.

```
        //Customer Booking Status Page
        public async Task<IActionResult> CustomerBookings()
        {
            var user = await userManager.GetUserAsync(User);

            var bookings = await context.Bookings
                .Where(b => b.UserId == user.Id)
                .ToListAsync();

            return View(bookings);

        }
```

Create the View

```
@model List<Booking>

<h3>My Bookings</h3>

<table class="table">
    <tr>
        <th>Date</th>
        <th>Time</th>
        <th>Status</th>
    </tr>

    @foreach (var b in Model)
    {
        <tr>
            <td>@b.BookingDateTime.ToShortDateString()</td>
            <td>@b.BookingDateTime.ToShortTimeString()</td>
            <td>@b.Status</td>
        </tr>
    }
</table>
```