ASP.NET MVC hosting Full Calendar

Full Calendar site:

<https://fullcalendar.io/>

Contents

[New project 2](#_Toc134036954)

[DB Preparation 3](#_Toc134036955)

[New folders: 3](#_Toc134036956)

[appsettings.json 3](#_Toc134036957)

[\*\* New classes under Model 3](#_Toc134036958)

[CalDbContext 3](#_Toc134036959)

[AppUser.cs 3](#_Toc134036960)

[AppTeacher 4](#_Toc134036961)

[Appointment.cs 4](#_Toc134036962)

[Modify CalDbContext: 4](#_Toc134036963)

[Program.cs 4](#_Toc134036964)

[Install Nuget Packages 4](#_Toc134036965)

[“Package Manager Console” 4](#_Toc134036966)

[Add data 5](#_Toc134036967)

[Full Calendar Preparation 5](#_Toc134036968)

[Install calendar to client 5](#_Toc134036969)

[Open Pages\Shared\\_Layout.cshtml 6](#_Toc134036970)

[Login/Register 7](#_Toc134036971)

[**Models/** LoginViewModel.cs 7](#_Toc134036972)

[Program.cs – Session 8](#_Toc134036973)

[Controllers 8](#_Toc134036974)

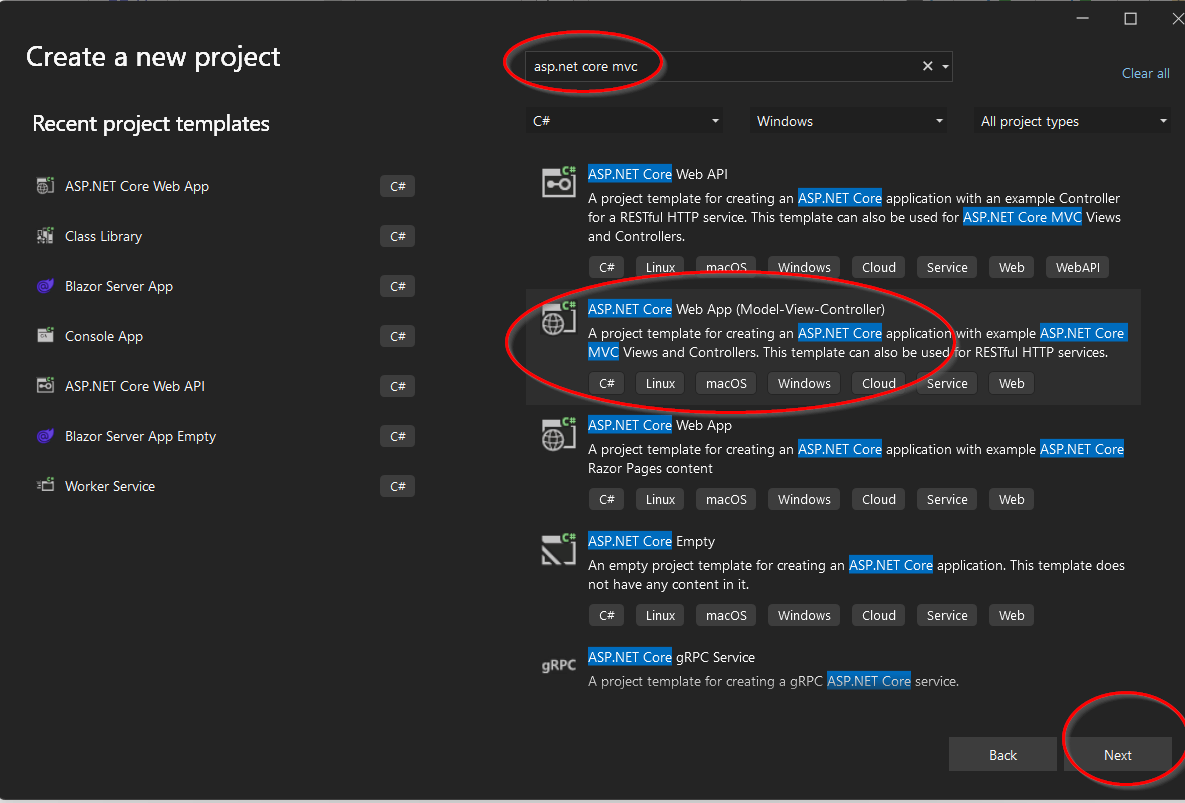
[AuthController 9](#_Toc134036975)

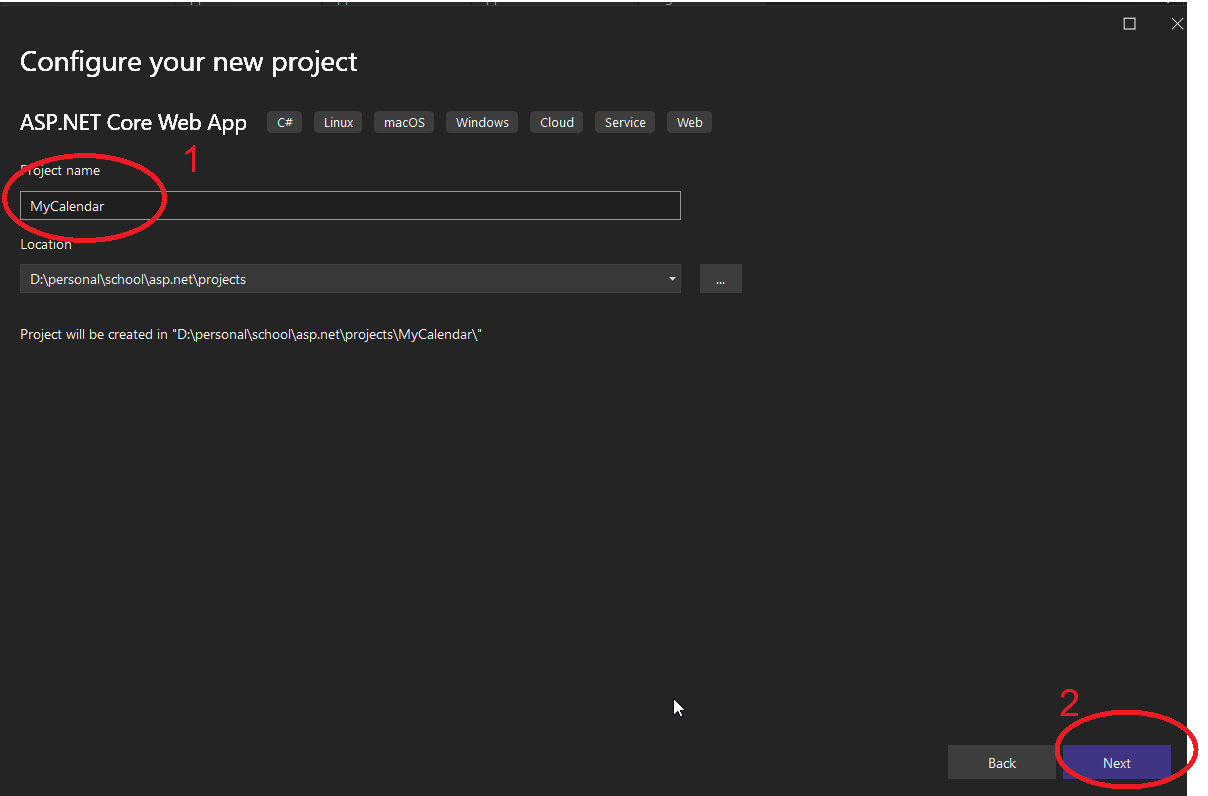
[Login View 10](#_Toc134036976)

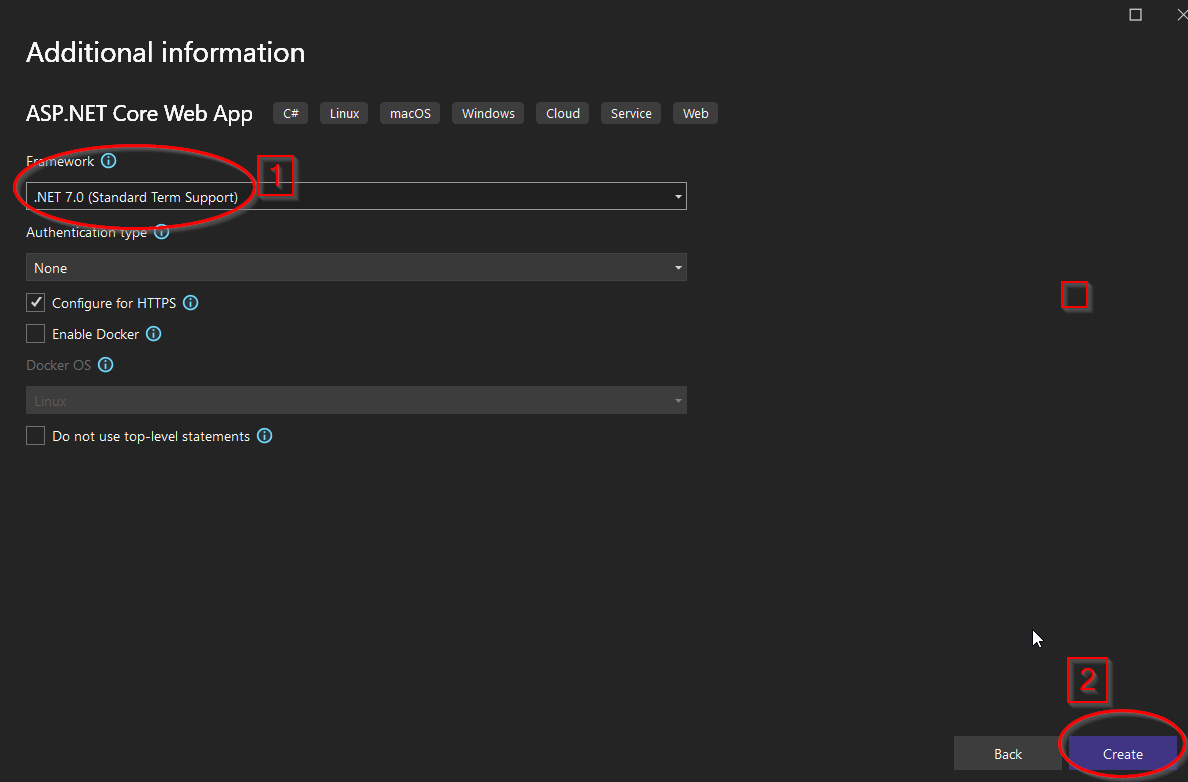
[CalController 12](#_Toc134036977)

[Final Code: 15](#_Toc134036978)

# New project







# DB Preparation

## New folders:

Create new folder called App\_Data under the project

## appsettings.json

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;AttachDBFilename=[DataDirectory]\\App\_Data\\cal.mdf;Initial Catalog=cal;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

## \*\* New classes under Model

### CalDbContext

public class CalDbContext : DbContext

{

public CalDbContext(DbContextOptions<CalDbContext> options) : base(options)

{

}

}

### AppUser.cs

public int Id { get; set; }

public string Name { get; set; }

public string Email { get; set; }

public string Password { get; set; }

public int AppTeacherId { get; set; }

public virtual List<Appointment> Appointments { get; set; }

}

### AppTeacher

public class AppTeacher

{

public int Id { get; set; }

public string Name { get; set; }

public string Email { get; set; }

public string Password { get; set; }

//public List<int> UserIds{ get; set; }

public virtual List<AppUser> Users { get; set; }

}

### Appointment.cs

public class Appointment

{

public int Id { get; set; }

public DateTime Start{ get; set; }

public DateTime End{ get; set; }

public int AppUserId { get; set; }

}

### Modify CalDbContext:

public DbSet<AppUser> AppUsers { get; set; }

public DbSet<AppTeacher> AppTeachers { get; set; }

public DbSet<Appointment> Appointments { get; set; }

### Program.cs

string path = Directory.GetCurrentDirectory();

builder.Services.AddDbContext<CalDbContext>(options =>

options.UseSqlServer(

builder.Configuration.GetConnectionString("DefaultConnection")

.Replace("[DataDirectory]", path)));

### Install Nuget Packages

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.Design

Microsoft.EntityFrameworkCore.SqlServer

Microsoft.EntityFrameworkCore.Tools

### “Package Manager Console”

add-migration "init"

update-database

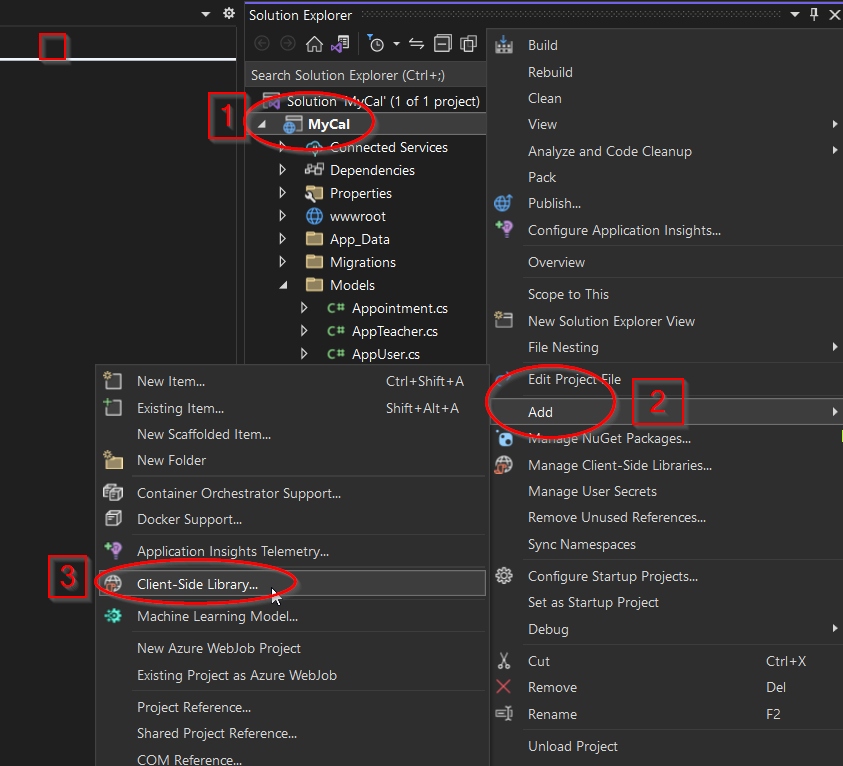
### Add data

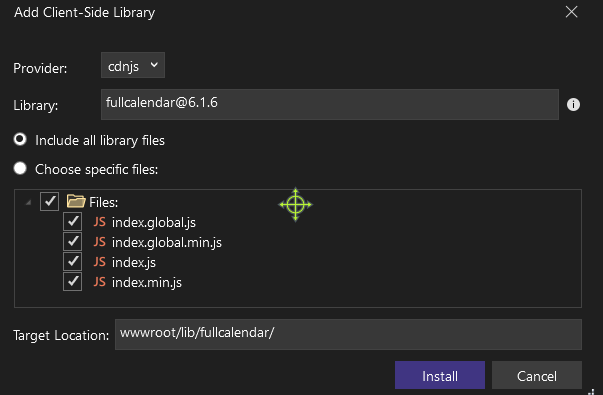
We are not focusing on Auth (Login/Register/Logout)…Thus we want to create manual entries in the DB

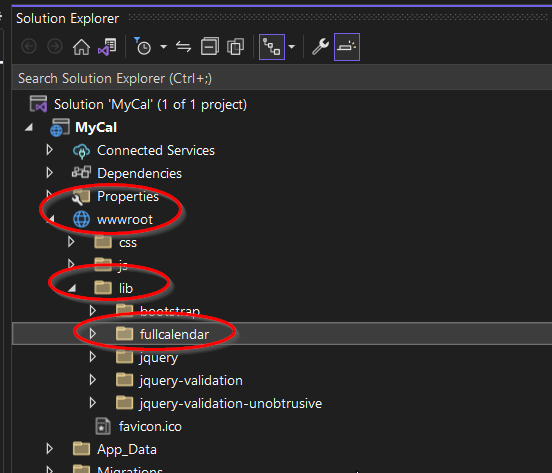
(2 teachers, each has 2 users) to the DB

# Full Calendar Preparation

## Install calendar to client



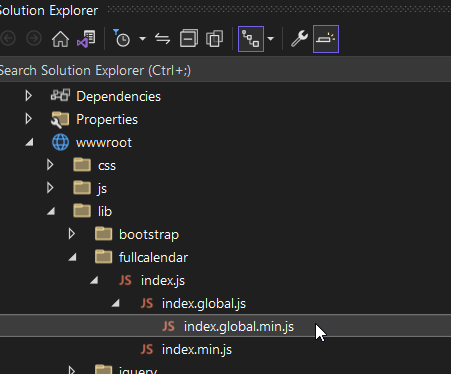


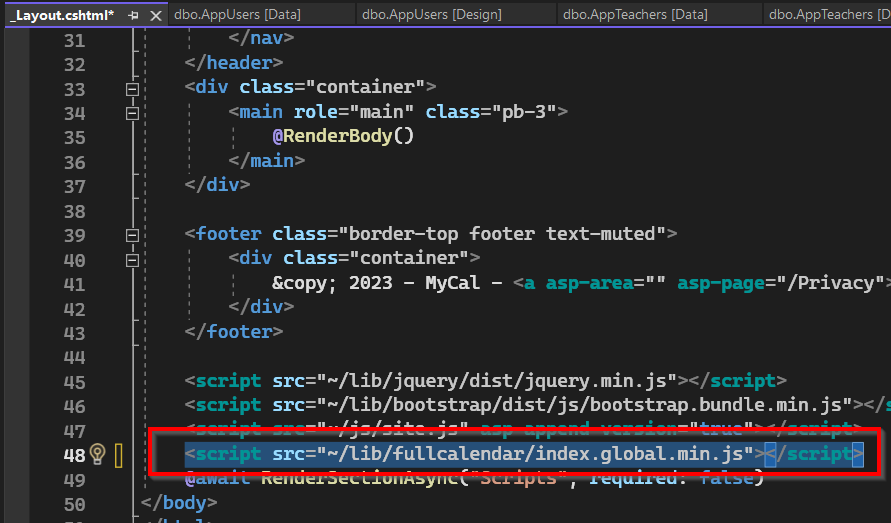


## Open Pages\Shared\\_Layout.cshtml

Scroll to the down

Drag and Drop wwwroot\lib\fullcalendar\index\index.global.js\index.global.min.js





# Login/Register

## **Models/** LoginViewModel.cs

public class LoginViewModel

{

public string Email { get; set; }

public string Password { get; set; }

}

## Program.cs – Session

\*\*\* Before builder.Build();

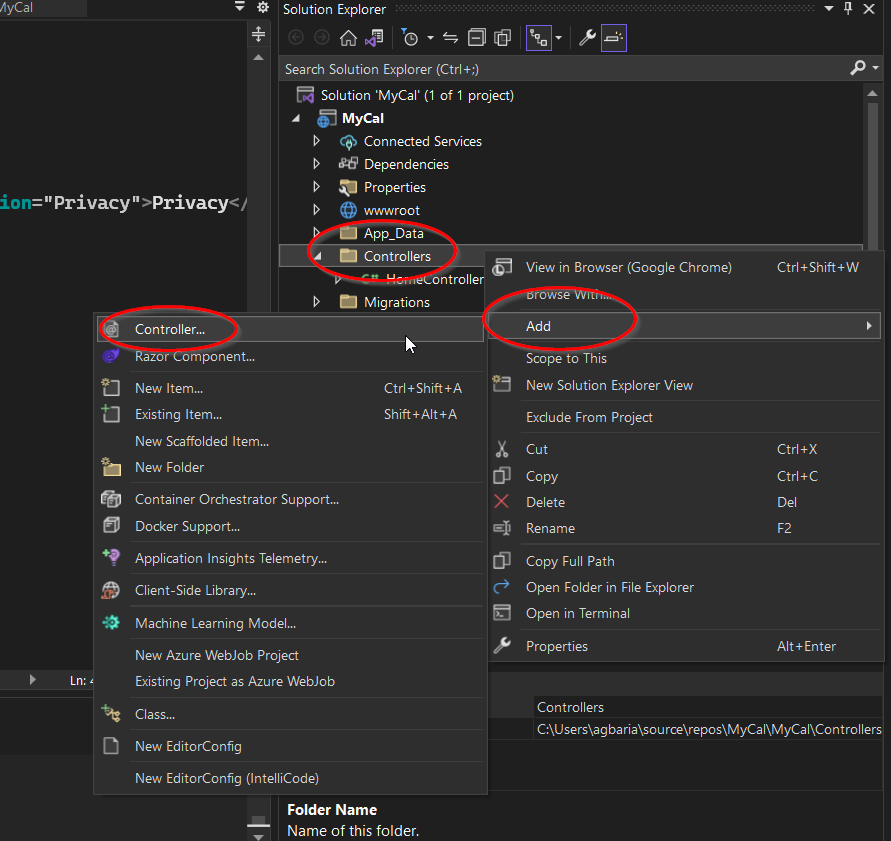
builder.Services.AddSession();

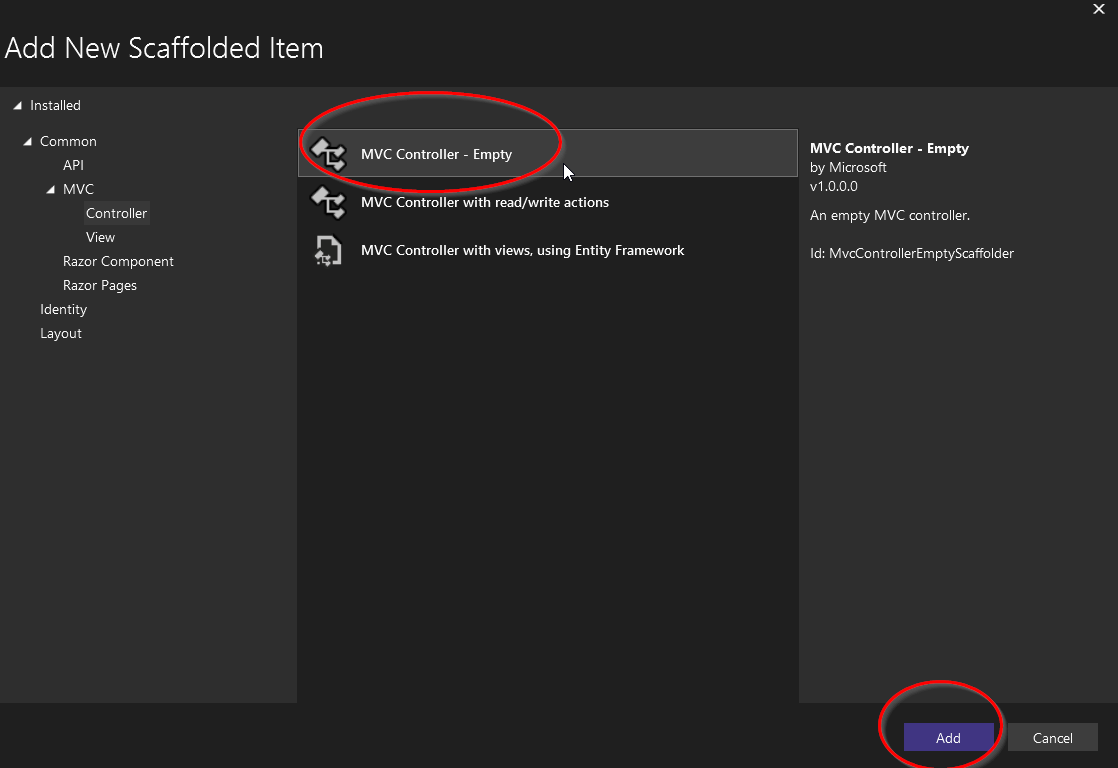
**\*\* Between app.UseRouting and app.** **UseAuthorization**

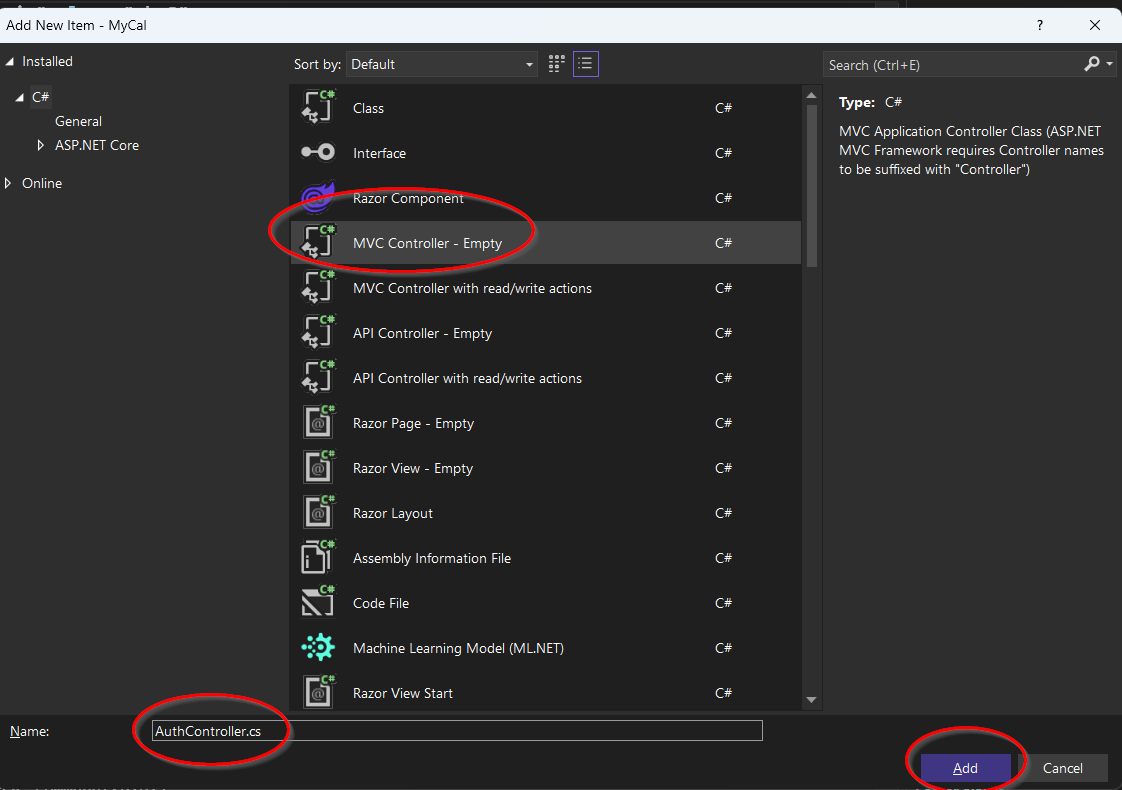
app.UseSession();

## Controllers

New Controller AuthController (responsible for login/logout/register)







## AuthController

public class AuthController : Controller

{

private readonly CalDbContext dbContext;

public AuthController(CalDbContext dbContext)

{

this.dbContext = dbContext;

}

public IActionResult Login()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Login(LoginViewModel model)

{

//check if user exists

if (ModelState.IsValid)

{

var user = await dbContext.AppUsers.Where(u => u.Email == model.Email && u.Password == model.Password).FirstOrDefaultAsync();

if (user == null)

{

var teacer = await dbContext.AppTeachers.Where(u => u.Email == model.Email && u.Password == model.Password).FirstOrDefaultAsync();

if (teacer == null)

{

ModelState.AddModelError("Email", "Email or Password does not exist");

return View();

}

HttpContext.Session.SetString("email", teacer.Email);

HttpContext.Session.SetString("userid", teacer.Id.ToString());

HttpContext.Session.SetString("type", "teacer");

return RedirectToAction("TeacherCal", "Cal");

}

else

{

HttpContext.Session.SetString("email", model.Email);

HttpContext.Session.SetString("userid", user.Id.ToString());

HttpContext.Session.SetString("type", "user");

return RedirectToAction("UserCal", "Cal");

}

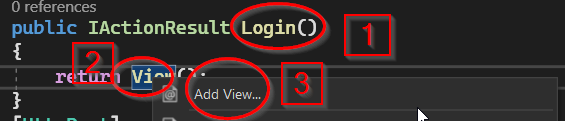
}

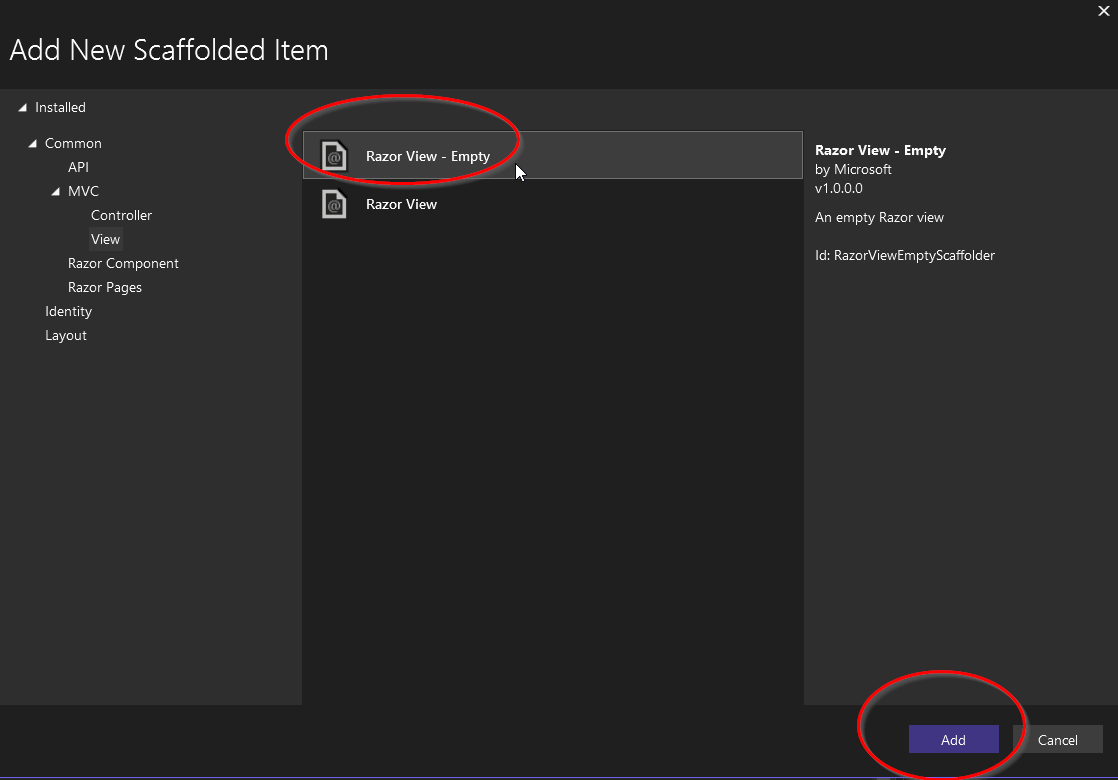
return View();

}

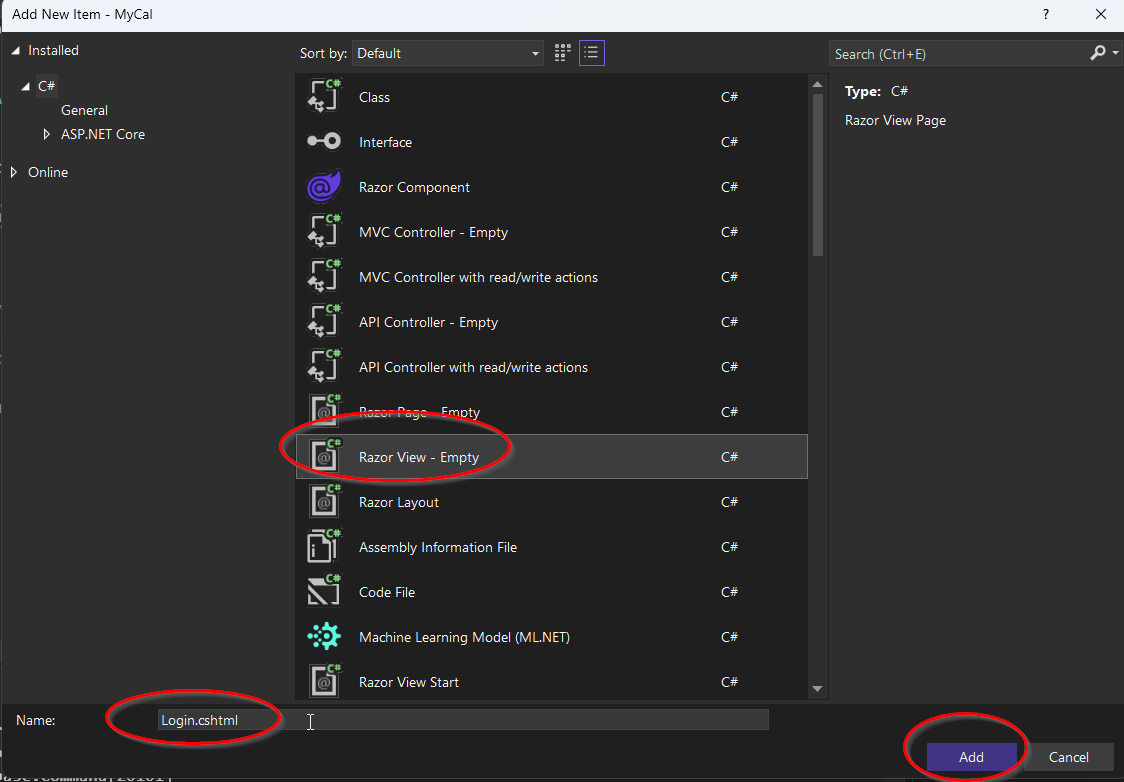
}

## Login View





**Login.cshtml**



@model LoginViewModel

@{

ViewData["Title"] = "Login";

}

<h1>Login</h1>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="Login">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

<div class="form-group">

<**label** **asp-for**="Email" class="control-label"></**label**>

<**input** **asp-for**="Email" class="form-control" />

<**span** **asp-validation-for**="Email" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Password" class="control-label"></**label**>

<**input** **asp-for**="Password" class="form-control" />

<**span** **asp-validation-for**="Password" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Login" class="btn btn-primary" />

<input type="reset" value="Clear" class="btn btn-secondary" />

</div>

</**form**>

</div>

</div>

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}

## CalController

CalController.cs

using Microsoft.AspNetCore.Mvc;

using MyCal.Models;

namespace MyCal.Controllers

{

public class CalController : Controller

{

private readonly CalDbContext dbContext;

public CalController(CalDbContext dbContext)

{

this.dbContext = dbContext;

}

public IActionResult TeacherCal()

{

return View();

}

public IActionResult UserCal()

{

return View();

}

}

}

\*\* New Views

UserCal.cshtml

<div id='calendar'></div>

site.css

#calendar {

max-width: 1100px;

margin: 40px auto;

}

//Back to the cshrml

// <https://fullcalendar.io/demos>

@section Scripts

{

<script>

var calendarEl = document.getElementById('calendar');

var calendar = new FullCalendar.Calendar(calendarEl, {

//selectable: true,

initialView: "timeGridWeek",

headerToolbar: {

left: "prev,next today",

center: "title",

right: "dayGridMonth,timeGridWeek,timeGridDay,listWeek"

},

eventTimeFormat: {

hour: "numeric",

minute: "2-digit",

timeZoneName: "short"

},

dateClick: async function (info) {

let uri = '/Cal/SetupMeeting';

try {

const response = await fetch(uri,{

method: 'POST',

headers: {

'Accept': 'application/json',

'Content-Type': 'application/json'

},

body: JSON.stringify(info.dateStr)

});

const jsonData = await response.json(); //string

const data = JSON.parse(jsonData);

console.log(data);

//alert('clicked ' + info.dateStr);

addMeetingEvent("Event#"+data.id, data.start, data.end)

}

catch (err) {

console.error('Unable to update item.', error);

alert(error);

}

},

select: function (info) {

alert('selected ' + info.startStr + ' to ' + info.endStr);

}

});

calendar.render();

function addMeetingEvent(title, start, end) {

const sdate = new Date(start);

const edate = new Date(end);

if (!isNaN(sdate.valueOf()) && !isNaN(edate.valueOf())) {

calendar.addEvent({ title: title, start: sdate, end: edate });

}else{

console.error("Invalid dates: " + start + " OR " + end)

}

} </script>

}

**//Back to the CalController.cs**

[HttpPost]

public async Task<IActionResult> SetupMeeting([FromBody] string dt)

{

var datetime = DateTime.Parse(dt);

string uid = HttpContext.Session.GetString("userid");

if (int.TryParse(uid, out int id))

{

var user = await dbContext.AppUsers.Where(u => u.Id == id).FirstOrDefaultAsync();

if (user == null)

{

return BadRequest("User not found!");

}

//send data to db

Appointment eventData = new Appointment()

{

AppUserId = user.Id,

Start = datetime,

End = datetime.AddMinutes(30)

};

dbContext.Appointments.Add(eventData);

dbContext.SaveChanges();

return Ok(System.Text.Json.JsonSerializer.Serialize(eventData).ToLower());

}

return BadRequest("User not registered!");

}

}

# Final Code:

**AuthController:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using MyCal.Models;

using System.ComponentModel.DataAnnotations;

namespace MyCal.Controllers

{

public class AuthController : Controller

{

private readonly CalDbContext dbContext;

public AuthController(CalDbContext dbContext)

{

this.dbContext = dbContext;

}

public IActionResult Login()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Login(LoginViewModel model)

{

//check if user exists

if (ModelState.IsValid)

{

var user = await dbContext.AppUsers.Where(u => u.Email == model.Email && u.Password == model.Password).FirstOrDefaultAsync();

if (user == null)

{

var teacer = await dbContext.AppTeachers.Where(u => u.Email == model.Email && u.Password == model.Password).FirstOrDefaultAsync();

if (teacer == null)

{

ModelState.AddModelError("Email", "Email or Password does not exist");

return View();

}

HttpContext.Session.SetString("email", teacer.Email);

HttpContext.Session.SetString("userid", teacer.Id.ToString());

HttpContext.Session.SetString("type", "teacer");

return RedirectToAction("TeacherCal", "Cal");

}

else

{

HttpContext.Session.SetString("email", model.Email);

HttpContext.Session.SetString("userid", user.Id.ToString());

HttpContext.Session.SetString("type", "user");

return RedirectToAction("UserCal", "Cal");

}

}

return View();

}

}

}

CalController

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using MyCal.Models;

using System.Text.Json.Serialization;

namespace MyCal.Controllers

{

public class CalController : Controller

{

private readonly CalDbContext dbContext;

public CalController(CalDbContext dbContext)

{

this.dbContext = dbContext;

}

public IActionResult TeacherCal()

{

return View();

}

public IActionResult UserCal()

{

string sid = HttpContext.Session.GetString("userid");

if(string.IsNullOrEmpty(sid))

{

return BadRequest();

}

int uid = int.Parse(sid);

dbContext.Appointments.ToList(); //load them

var user = dbContext.AppUsers.Find(uid);

return View(user);

}

[HttpPost]

public async Task<IActionResult> SetupMeeting([FromBody] string dt)

{

//Add to DB

var datetime = DateTime.Parse(dt);

string uid = HttpContext.Session.GetString("userid");

if (int.TryParse(uid, out int id))

{

var user = await dbContext.AppUsers.Where(u => u.Id == id).FirstOrDefaultAsync();

if (user == null)

{

return BadRequest("User not found!");

}

//send data to db

Appointment eventData = new Appointment()

{

AppUserId = user.Id,

Start = datetime,

End = datetime.AddMinutes(30)

};

dbContext.Appointments.Add(eventData);

dbContext.SaveChanges();

return Ok(System.Text.Json.JsonSerializer.Serialize(eventData).ToLower());

}

return BadRequest("User not registered!");

}

}

}

UserCal.cshtml

@model AppUser

<div id='calendar'></div>

@section Scripts

{

<script>

var appointments = JSON.parse('@Json.Serialize(@Model.Appointments)');

let events = [];

if (appointments) {

for (let i = 0; i < appointments.length; i++) {

const p = appointments[i];

events.push({title:"id#"+p.id, start: p.start, end:p.end});

console.log(p);

}

}

var calendarEl = document.getElementById('calendar');

var calendar = new FullCalendar.Calendar(calendarEl, {

//selectable: true,

initialView: "timeGridWeek",

headerToolbar: {

left: "prev,next today",

center: "title",

right: "dayGridMonth,timeGridWeek,timeGridDay,listWeek"

},

events: events,

eventTimeFormat: {

hour: "numeric",

minute: "2-digit",

timeZoneName: "short"

},

dateClick: async function (info) {

let uri = '/Cal/SetupMeeting';

try {

const response = await fetch(uri,{

method: 'POST',

headers: {

'Accept': 'application/json',

'Content-Type': 'application/json'

},

body: JSON.stringify(info.dateStr)

});

const jsonData = await response.json(); //string

const data = JSON.parse(jsonData);

console.log(data);

//alert('clicked ' + info.dateStr);

addMeetingEvent("Event#"+data.id, data.start, data.end)

}

catch (err) {

console.error('Unable to update item.', error);

alert(error);

}

},

select: function (info) {

alert('selected ' + info.startStr + ' to ' + info.endStr);

}

});

calendar.render();

function addMeetingEvent(title, start, end) {

const sdate = new Date(start);

const edate = new Date(end);

if (!isNaN(sdate.valueOf()) && !isNaN(edate.valueOf())) {

calendar.addEvent({ title: title, start: sdate, end: edate });

}else{

console.error("Invalid dates: " + start + " OR " + end)

}

}

//});

</script>

}