**HomeController:**  
  
using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

using ClaimSystem.Models;

using System.Collections.Generic;

using System.Linq;

using System.IO;

namespace ClaimSystem.Controllers

{

public class HomeController : Controller

{

// Static list of claims to acts as database storage

private static List<Claim> \_claims = new()

{

new Claim { ClaimID = 1, Lecturer = "John Doe", Hours = 20, HourlyRate = 100, Notes = "Prog class", Status = "Pending", LastUpdated = System.DateTime.Now },

new Claim { ClaimID = 2, Lecturer = "Jane Smith", Hours = 15, HourlyRate = 150, Notes = "Tutoring services", Status = "Pending", LastUpdated = System.DateTime.Now }

};

// Index action - displays welcome screen

public IActionResult Index()

{

return View();

}

// ClaimSubmission action - displays claim submission form

public IActionResult ClaimSubmission()

{

return View();

}

// SubmitClaim action - handles POST requests to submit claims

[HttpPost]

public IActionResult SubmitClaim(string lecturer, decimal hoursWorked, decimal hourlyRate, string notes, IFormFile document)

{

// Validate file upload

string filePath = null;

string originalFileName = null;

if (document != null && document.Length > 0)

{

// File size validation (limit: 5MB)

if (document.Length > 5 \* 1024 \* 1024)

{

ModelState.AddModelError("File", "The file size cannot exceed 5MB.");

return View("ClaimSubmission");

}

// File type validation

var allowedExtensions = new[] { ".pdf", ".docx", ".xlsx" };

var extension = Path.GetExtension(document.FileName).ToLowerInvariant();

if (!allowedExtensions.Contains(extension))

{

ModelState.AddModelError("File", "Only .pdf, .docx, and .xlsx files are allowed.");

return View("ClaimSubmission");

}

// Sav file to server (in wwwroot/uploads)

var uploadsDirectory = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "uploads");

if (!Directory.Exists(uploadsDirectory))

{

Directory.CreateDirectory(uploadsDirectory);

}

var uniqueFileName = Path.GetRandomFileName() + extension;

filePath = Path.Combine(uploadsDirectory, uniqueFileName);

using (var stream = new FileStream(filePath, FileMode.Create))

{

document.CopyTo(stream);

}

originalFileName = document.FileName;

}

// Create new claim based on submitted data

var newClaim = new Claim

{

ClaimID = \_claims.Count + 1,

Lecturer = lecturer, // Lecturer can now input own name

Hours = hoursWorked,

HourlyRate = hourlyRate, // Hourly rate

Notes = notes, // Notes added

Status = "Pending",

LastUpdated = System.DateTime.Now,

SupportingDocumentPath = filePath, // Store file path in claim

OriginalFileName = originalFileName // Store original file name

};

// Add new claim to list

\_claims.Add(newClaim);

// Redirect to ClaimStatus action after a submission

return RedirectToAction("ClaimStatus");

}

// GET: Login page

public IActionResult Login()

{

return View();

}

// POST: Login process

[HttpPost]

public IActionResult Login(string username, string password)

{

// Role login

if (username == "coordinator" && password == "123")

{

HttpContext.Session.SetString("UserRole", "Coordinator");

return RedirectToAction("ClaimApproval");

}

else if (username == "manager" && password == "123")

{

HttpContext.Session.SetString("UserRole", "Manager");

return RedirectToAction("ClaimApproval");

}

else

{

ViewBag.ErrorMessage = "Invalid username or password.";

return View();

}

}

// Logout action

public IActionResult Logout()

{

HttpContext.Session.Clear(); // Clear session

return RedirectToAction("Login");

}

// ClaimApproval action - shows pending claims for approval / rejection

public IActionResult ClaimApproval()

{

// Filter and give only pending claims

if (HttpContext.Session.GetString("UserRole") == null)

{

return RedirectToAction("Login");

}

return View(\_claims.Where(c => c.Status == "Pending").ToList());

}

// ApproveClaim action - approves claim based on claim ID

public IActionResult ApproveClaim(int id)

{

// Find claim by its ID

var claim = \_claims.FirstOrDefault(c => c.ClaimID == id);

if (claim != null)

{

// Update status - 'Approved'

claim.Status = "Approved";

claim.LastUpdated = System.DateTime.Now;

}

// Go back to ClaimApproval view

return RedirectToAction("ClaimApproval");

}

// RejectClaim action - reject claim based on claim ID

public IActionResult RejectClaim(int id)

{

// Find claim by ID

var claim = \_claims.FirstOrDefault(c => c.ClaimID == id);

if (claim != null)

{

// Update status to 'Rejected'

claim.Status = "Rejected";

claim.LastUpdated = System.DateTime.Now;

}

// Back to ClaimApproval view

return RedirectToAction("ClaimApproval");

}

// ClaimStatus action - displays status of submitted claims

public IActionResult ClaimStatus()

{

// Pass list of claims to view

return View(\_claims);

}

// Error action

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = System.Diagnostics.Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

**Claim:**

using System;

namespace ClaimSystem.Models

{

public class Claim

{

// Unique ID for claims

public int ClaimID { get; set; }

// Lecturer name - submitting claim

public string Lecturer { get; set; } = string.Empty; // Initialize as empty - avoid null problems

// Hours worked (int)

public decimal Hours { get; set; }

// Hourly rate for claim

public decimal HourlyRate { get; set; }

// Additional notes for claim

public string Notes { get; set; } = string.Empty;

// Current claim status - (Pending, Approved, Rejected)

public string Status { get; set; } = "Pending"; // Default - Pending

// Timestamp for last update for claim

public DateTime LastUpdated { get; set; } = DateTime.Now;

// Path for the supporting document

public string? SupportingDocumentPath { get; set; } // Nullable - allows no file submission

// Original file name - seen by managers

public string? OriginalFileName { get; set; }

}

}

**ErrorViewModel:**

namespace ClaimSystem.Models

{

public class ErrorViewModel

{

// Stores request ID to manage issues

public string? RequestId { get; set; }

// True if valid Request ID is present

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

}

**ClaimApproval:**

@model IEnumerable<ClaimSystem.Models.Claim>

@{

Layout = "~/Views/Shared/\_Layout.cshtml"; // Use the app wide layout

ViewData["Title"] = "Claim Approval"; // Page title

}

<div class="container mt-5">

<h2>Claim Approval</h2>

<table class="table table-bordered">

<thead>

<tr>

<th>Claim ID</th>

<th>Lecturer</th>

<th>Hours</th>

<th>Hourly Rate</th>

<th>Document</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<!-- Loop through claims in model - show detials -->

@foreach (var claim in Model)

{

<tr>

<td>@claim.ClaimID</td>

<td>@claim.Lecturer</td>

<td>@claim.Hours</td>

<td>@claim.HourlyRate</td>

<td>

@if (!string.IsNullOrEmpty(claim.SupportingDocumentPath))

{

<!-- Display original file name - visible only to managers -->

<a href="@Url.Content("~/uploads/" + System.IO.Path.GetFileName(claim.SupportingDocumentPath))" target="\_blank">

@claim.OriginalFileName

</a>

}

else

{

<span>No document</span>

}

</td>

<td>

<!-- Buttons to approve / reject claim -->

<a href="@Url.Action("ApproveClaim", "Home", new { id = claim.ClaimID })" class="btn btn-success" style="width: 200px; height: 40px;">Approve</a>

<a href="@Url.Action("RejectClaim", "Home", new { id = claim.ClaimID })" class="btn btn-danger" style="width: 200px; height: 40px;">Reject</a>

</td>

</tr>

}

</tbody>

</table>

</div>

**ClaimStatus:**

@model IEnumerable<ClaimSystem.Models.Claim>

@{

Layout = "~/Views/Shared/\_Layout.cshtml"; // Use the site wide layout

ViewData["Title"] = "Claim Status"; // Set page title

}

<div class="container mt-5">

<h2>Claim Status</h2>

<table class="table table-bordered">

<thead>

<tr>

<th>Claim ID</th>

<th>Lecturer</th>

<th>Hours</th>

<th>Hourly Rate</th>

<th>Notes</th>

<th>Status</th>

<th>Last Updated</th>

<th>Document</th> <!-- Document column -->

</tr>

</thead>

<tbody>

<!-- Loop through claims - display status -->

@foreach (var claim in Model)

{

<tr>

<td>@claim.ClaimID</td>

<td>@claim.Lecturer</td>

<td>@claim.Hours</td>

<td>@claim.HourlyRate</td> <!-- Hourly Rate and Notes added -->

<td>@claim.Notes</td>

<td>

<!-- Status as label -->

@if (claim.Status == "Pending")

{

<span class="badge bg-warning">Pending</span>

}

else if (claim.Status == "Approved")

{

<span class="badge bg-success">Approved</span>

}

else if (claim.Status == "Rejected")

{

<span class="badge bg-danger">Rejected</span>

}

</td>

<td>@claim.LastUpdated.ToString("g")</td>

<td>

@if (!string.IsNullOrEmpty(claim.SupportingDocumentPath))

{

<span>Document Uploaded</span>

}

else

{

<span>No document</span>

}

</td>

</tr>

}

</tbody>

</table>

<!-- Redirect to Login page (For admins) -->

<a href="@Url.Action("Login", "Home")" class="btn btn-primary mt-3">Go to Admin Login</a>

</div>

**ClaimSubmission:**

@{

Layout = "~/Views/Shared/\_Layout.cshtml"; // Use the site wide layout

ViewData["Title"] = "Claim Submission"; // Set page title

}

<div class="container mt-5">

<h2>Claim Submission</h2>

<!-- Form - submit the claim (Updated with hourly rate and notes)-->

<form method="post" enctype="multipart/form-data" action="@Url.Action("SubmitClaim", "Home")">

<!-- Form - lecturer can input name -->

<div class="form-group">

<label for="lecturer">Lecturer Name:</label>

<input type="text" id="lecturer" name="lecturer" class="form-control" placeholder="Enter full name" style="width: 250px; height: 40px;" required />

</div>

<div class="form-group">

<label for="hoursWorked">Hours Worked:</label>

<input type="number" id="hoursWorked" name="hoursWorked" class="form-control" placeholder="Enter hours worked" style="width: 250px; height: 40px;" step="0.01" required />

</div>

<!-- Hourly rate set to 0.01 - allows decimals (will be better later on) -->

<div class="form-group">

<label for="hourlyRate">Hourly Rate:</label>

<input type="number" id="hourlyRate" name="hourlyRate" class="form-control" placeholder="Enter hourly rate" style="width: 250px; height: 40px;" step="0.01" required />

</div>

<div class="form-group">

<label for="notes">Additional Notes:</label>

<textarea id="notes" name="notes" class="form-control" placeholder="Add any additional notes here" style="width: 250px; height: 100px;"></textarea>

</div>

<!-- Supporting Document Upload -->

<div class="form-group">

<label for="document">Supporting Document (PDF, DOCX, XLSX - Max 5MB):</label>

<input type="file" id="document" name="document" class="form-control-file" accept=".pdf,.docx,.xlsx" />

</div>

<!-- Submit button -->

<div>

<button type="submit" class="btn btn-success" style="width: 200px; height: 40px;">Submit Claim</button>

</div>

</form>

<!-- Link - view claim status -->

<div>

<a href="@Url.Action("ClaimStatus", "Home")" class="btn btn-secondary mt-3" style="width: 200px; height: 40px;">Go to Claim Status</a>

</div>

</div>

**Index:**

@{

Layout = "~/Views/Shared/\_Layout.cshtml"; // Use same layout

ViewData["Title"] = "Welcome"; // Set page title

}

<div class="container text-center mt-5">

<h1>Welcome to the Claim System</h1>

<!-- Button - enter app & navigate to claim submission page -->

<button class="btn btn-success mt-5" style="width: 250px; height: 50px;" onclick="location.href='@Url.Action("ClaimSubmission", "Home")'">

Enter Application

</button>

</div>

@{

Layout = "~/Views/Shared/\_Layout.cshtml"; // Use same layout

ViewData["Title"] = "Welcome"; // Set page title

}

<div class="container text-center mt-5">

<h1>Welcome to the Claim System</h1>

<!-- Button - enter app & navigate to claim submission page -->

<button class="btn btn-success mt-5" style="width: 250px; height: 50px;" onclick="location.href='@Url.Action("ClaimSubmission", "Home")'">

Enter Application

</button>

</div>

**Login:**

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

ViewData["Title"] = "Admin Login";

}

<div class="container mt-5">

<h2>Admin Login</h2>

<form method="post" action="@Url.Action("Login", "Home")">

<div class="form-group">

<label for="username">Username:</label>

<input type="text" id="username" name="username" class="form-control" placeholder="Enter username" required />

</div>

<div class="form-group">

<label for="password">Password:</label>

<input type="password" id="password" name="password" class="form-control" placeholder="Enter password" required />

</div>

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

</form>

@if (ViewBag.ErrorMessage != null)

{

<div class="alert alert-danger">@ViewBag.ErrorMessage</div>

}

</div>

**Privacy:**

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@{

ViewData["Title"] = "Privacy Policy";

}

<h1>@ViewData["Title"]</h1>

<p>Use this page to detail your site's privacy policy.</p>

**Error:**

@model ClaimSystem.Models.ErrorViewModel

@{

ViewData["Title"] = "Error";

}

<h1 class="text-danger">An error occurred while processing your request.</h1>

@if (Model.ShowRequestId)

{

<p>

<strong>Request ID:</strong> <code>@Model.RequestId</code>

</p>

}

**\_layout:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - Claim System</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" />

</head>

<body>

<div class="container">

@RenderBody()

</div>

</body>

</html>

**\_layout.cshtml.css:**

/\* Please see documentation at https://learn.microsoft.com/aspnet/core/client-side/bundling-and-minification

for details on configuring this project to bundle and minify static web assets. \*/

a.navbar-brand {

white-space: normal;

text-align: center;

word-break: break-all;

}

a {

color: #0077cc;

}

.btn-primary {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

.border-top {

border-top: 1px solid #e5e5e5;

}

.border-bottom {

border-bottom: 1px solid #e5e5e5;

}

.box-shadow {

box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);

}

button.accept-policy {

font-size: 1rem;

line-height: inherit;

}

.footer {

position: absolute;

bottom: 0;

width: 100%;

white-space: nowrap;

line-height: 60px;

}

.progress-bar {

width: 100%;

background-color: #e0e0e0;

border-radius: 5px;

margin-bottom: 10px;

}

.progress {

height: 20px;

border-radius: 5px;

}

.progress-warning {

background-color: #f0ad4e; /\* Pending \*/

}

.progress-success {

background-color: #4caf50; /\* Approved \*/

}

.progress-danger {

background-color: #d9534f; /\* Rejected \*/

}

**Program:**

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

namespace ClaimSystem

{

public class Program

{

// Main of application - entry point

public static void Main(string[] args)

{

// Create, run the web host

CreateHostBuilder(args).Build().Run();

}

// Set web host with basic settings

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>(); // Selects Startup class to be used

});

}

}

**Startup:**

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using System.Globalization;

using Microsoft.AspNetCore.Localization;

namespace ClaimSystem

{

public class Startup

{

// Method to configure

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

services.AddSession(); // Enable session

}

// Method to configure

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

// Set culture of app - uses 'en-US' - this allows '.' to be the decimal operator

var supportedCultures = new[] { new CultureInfo("en-US") };

app.UseRequestLocalization(new RequestLocalizationOptions

{

DefaultRequestCulture = new RequestCulture("en-US"), // Default culture set - "en-US"

SupportedCultures = supportedCultures,

SupportedUICultures = supportedCultures

});

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage(); // Show errors

}

else

{

app.UseExceptionHandler("/Home/Error"); // Go to error page in production

app.UseHsts();

}

app.UseHttpsRedirection(); // Redirect to HTTPS

app.UseStaticFiles();

app.UseRouting();

app.UseSession(); // Use session

app.UseAuthorization();

// Routing for application

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}"); // Way to Home controllers Index (action)

});

}

}

}