CMCS Code

Program.cs

```
using CMCS.Mvc; // <-- InMemoryStore namespace

using Microsoft.AspNetCore.Builder;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;


var builder = WebApplication.CreateBuilder(args);


// Add services

builder.Services.AddControllersWithViews();

builder.Services.AddSingleton<InMemoryStore>(); // singleton in-memory store


var app = builder.Build();


if (!app.Environment.IsDevelopment())

{

    app.UseExceptionHandler("/Home/Error");

    app.UseHsts();

}


app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Lecturers}/{action=Index}/{id?}");

app.Run();
```

Controllers

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace CMCS.Mvc.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return RedirectToAction("Index", "Claims");
        }

        public IActionResult Error() => View();
    }
}
```

ClaimsController.cs

```
// File: Controllers/ClaimsController.cs
using CMCS.Mvc.Models;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using System.IO;

namespace CMCS.Mvc.Controllers
{
  public class ClaimsController : Controller
  {
    private readonly InMemoryStore _store;
    private readonly IWebHostEnvironment _env;

    public ClaimsController(InMemoryStore store, IWebHostEnvironment env)
    {
      _store = store ?? throw new System.ArgumentNullException(nameof(store));
      _env = env ?? throw new System.ArgumentNullException(nameof(env));
    }

    // GET: /Claims
    public IActionResult Index()
    {
      var claims = _store.GetAllClaims();
      return View(claims);
```

```
}

// GET: /Claims/Details/5
public IActionResult Details(int id)
{
  var claim = _store.GetClaim(id);
  if (claim == null) return NotFound();
  return View(claim);
}

// GET: /Claims/Create
public IActionResult Create() => View();

// POST: /Claims/Create
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create(Claim claim, IFormFile? file)
{
  if (!ModelState.IsValid) return View(claim);

  // handle file upload if present
  if (file != null && file.Length > 0)
  {
    var uploadsRoot = Path.Combine(_env.WebRootPath ?? "wwwroot", "uploads");
    Directory.CreateDirectory(uploadsRoot);
```

```
            // sanitize filename - minimal approach
            var safeFileName = Path.GetFileName(file.FileName);
            var relativePath = Path.Combine("uploads",
$"{System.Guid.NewGuid()}_{safeFileName}");
            var absolutePath = Path.Combine(_env.WebRootPath ?? "wwwroot", relativePath);

            using (var stream = new FileStream(absolutePath, FileMode.Create))
            {
              file.CopyTo(stream);
            }

            claim.FilePath = relativePath.Replace(Path.DirectorySeparatorChar, '/');
          }

          var added = _store.AddClaim(claim);
          return RedirectToAction(nameof(Index));
        }

        // GET: /Claims/Edit/5
        public IActionResult Edit(int id)
        {
          var claim = _store.GetClaim(id);
          if (claim == null) return NotFound();
          return View(claim);
        }
```

```
    // POST: /Claims/Edit

    [HttpPost]

    [ValidateAntiForgeryToken]

    public IActionResult Edit(Claim claim, IFormFile? file)

    {

      if (!ModelState.IsValid) return View(claim);


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

      {

        var uploadsRoot = Path.Combine(_env.WebRootPath ?? "wwwroot", "uploads");

        Directory.CreateDirectory(uploadsRoot);


        var safeFileName = Path.GetFileName(file.FileName);

        var relativePath = Path.Combine("uploads",
$"{System.Guid.NewGuid()}_{safeFileName}");

        var absolutePath = Path.Combine(_env.WebRootPath ?? "wwwroot", relativePath);


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

        {

          file.CopyTo(stream);

        }


        claim.FilePath = relativePath.Replace(Path.DirectorySeparatorChar, '/');

      }


      var ok = _store.UpdateClaim(claim);
```

```
        if (!ok) return NotFound();

        return RedirectToAction(nameof(Index));
    }

    // GET: /Claims/Delete/5
    public IActionResult Delete(int id)
    {
        var claim = _store.GetClaim(id);
        if (claim == null) return NotFound();
        return View(claim);
    }

    // POST: /Claims/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public IActionResult DeleteConfirmed(int id)
    {
        // optionally attempt to delete stored file (best-effort)
        var claim = _store.GetClaim(id);
        if (claim != null && !string.IsNullOrEmpty(claim.FilePath))
        {
            try
            {
                var filePath = Path.Combine(_env.WebRootPath ?? "wwwroot",
claim.FilePath.Replace('/', Path.DirectorySeparatorChar));
```

```
                if (System.IO.File.Exists(filePath))
                {
                    System.IO.File.Delete(filePath);
                }
            }
            catch
            {
                // ignore file system failures - deletion of the claim still proceeds
            }
        }

        var removed = _store.DeleteClaim(id);
        if (!removed) return NotFound();

        return RedirectToAction(nameof(Index));
    }

    // GET: /Claims/Download/5
    public IActionResult Download(int id)
    {
        var claim = _store.GetClaim(id);
        if (claim == null || string.IsNullOrEmpty(claim.FilePath)) return NotFound();

        var absolutePath = Path.Combine(_env.WebRootPath ?? "wwwroot",
claim.FilePath.Replace('/', Path.DirectorySeparatorChar));
        if (!System.IO.File.Exists(absolutePath)) return NotFound();
```

```
            var fileName = Path.GetFileName(absolutePath);
            return PhysicalFile(absolutePath, "application/octet-stream", fileName);
        }
    }
}
```

LecturesController.cs

```
// File: Controllers/LecturersController.cs
using CMCS.Mvc.Models;
using Microsoft.AspNetCore.Mvc;

namespace CMCS.Mvc.Controllers
{
    public class LecturersController : Controller
    {
        private readonly InMemoryStore _store;

        public LecturersController(InMemoryStore store)
        {
            _store = store;
        }

        // GET: /Lecturers
        public IActionResult Index()
        {
            var lecturers = _store.GetAllLecturers();
```

```
  return View(lecturers);
}

// GET: /Lecturers/Details/5
public IActionResult Details(int id)
{
  var lecturer = _store.GetLecturer(id);
  if (lecturer == null)
    return NotFound();
  return View(lecturer);
}

// GET: /Lecturers/Create
public IActionResult Create() => View();

// POST: /Lecturers/Create
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Create(Lecturer lecturer)
{
  if (!ModelState.IsValid) return View(lecturer);

  _store.AddLecturer(lecturer);
  return RedirectToAction(nameof(Index));
}
```

```
// GET: /Lecturers/Edit/5
public IActionResult Edit(int id)
{
    var lecturer = _store.GetLecturer(id);
    if (lecturer == null)
        return NotFound();
    return View(lecturer);
}

// POST: /Lecturers/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Edit(Lecturer lecturer)
{
    if (!ModelState.IsValid)
        return View(lecturer);

    var ok = _store.UpdateLecturer(lecturer);
    if (!ok) return NotFound();

    return RedirectToAction(nameof(Index));
}

// GET: /Lecturers/Delete/5
public IActionResult Delete(int id)
{
```

```
            var lecturer = _store.GetLecturer(id);
            if (lecturer == null)
                return NotFound();
            return View(lecturer);
        }

        // POST: /Lecturers/DeleteConfirmed/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public IActionResult DeleteConfirmed(int id)
        {
            var ok = _store.DeleteLecturer(id);
            if (!ok) return NotFound();

            return RedirectToAction(nameof(Index));
        }
    }
}
```

Models

Claims.cs

```
// File: Models/Claim.cs
namespace CMCS.Mvc.Models
{
    public class Claim
    {
        public int Id { get; set; }

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

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

        public int Hours { get; set; }

        public decimal Rate { get; set; }

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

        public string Status { get; set; } = "Pending";


        // Computed property for convenience

        public decimal Amount => Hours * Rate;

    }

}
```

ClaimStatus.cs

```
namespace CMCS.Mvc.Models

{

    public enum ClaimStatus

    {

        Draft = 0,

        Submitted = 1,

        Verified = 2,

        SentToManager = 3,

        Approved = 4,

        Rejected = 5,

        Settled = 6

    }

}
```

InMemoryStore.cs

```
// File: InMemoryStore.cs
using CMCS.Mvc.Models;
using System.Collections.Generic;
using System.Linq;

namespace CMCS.Mvc
{
  /// <summary>
  /// Central in-memory store for Claims and Lecturers.
  /// Used instead of a database — data resets when the app restarts.
  /// </summary>
  public class InMemoryStore
  {
    // === CLAIMS ===
    private readonly List<Claim> _claims = new();
    private int _nextClaimId = 1;
    private readonly object _claimsLock = new();

    public List<Claim> GetAllClaims()
    {
      lock (_claimsLock)
        => _claims.Select(CloneClaim).ToList();
    }

    public Claim? GetClaim(int id)
```

```
{
   lock (_claimsLock)
      => _claims.FirstOrDefault(c => c.Id == id) is Claim claim ? CloneClaim(claim) : null;
}

public Claim AddClaim(Claim claim)
{
   lock (_claimsLock)
   {
      var copy = CloneClaim(claim);
      copy.Id = _nextClaimId++;
      _claims.Add(copy);
      return CloneClaim(copy);
   }
}

public bool UpdateClaim(Claim claim)
{
   lock (_claimsLock)
   {
      var existing = _claims.FirstOrDefault(c => c.Id == claim.Id);
      if (existing == null) return false;

      existing.Title = claim.Title;
      existing.Description = claim.Description;
      existing.Hours = claim.Hours;
```

```
        existing.Rate = claim.Rate;

        existing.FilePath = claim.FilePath;

        existing.Status = claim.Status;

        return true;

    }

}

public bool DeleteClaim(int id)

{

    lock (_claimsLock)

    {

        var c = _claims.FirstOrDefault(x => x.Id == id);

        if (c == null) return false;

        _claims.Remove(c);

        return true;

    }

}

private static Claim CloneClaim(Claim c) => new()

{

    Id = c.Id,

    Title = c.Title,

    Description = c.Description,

    Hours = c.Hours,

    Rate = c.Rate,

    FilePath = c.FilePath,
    Status = c.Status
};

// === LECTURERS ===
private readonly List<Lecturer> _lecturers = new();
private int _nextLecturerId = 1;
private readonly object _lecturerLock = new();

public List<Lecturer> GetAllLecturers()
{
    lock (_lecturerLock)
        => _lecturers.Select(CloneLecturer).ToList();
}

public Lecturer? GetLecturer(int id)
{
    lock (_lecturerLock)
        => _lecturers.FirstOrDefault(l => l.Id == id) is Lecturer l ? CloneLecturer(l) : null;
}

public Lecturer AddLecturer(Lecturer lecturer)
{
    lock (_lecturerLock)
    {
        var copy = CloneLecturer(lecturer);
        copy.Id = _nextLecturerId++;
        _lecturers.Add(copy);
        return CloneLecturer(copy);
    }
}

public bool UpdateLecturer(Lecturer lecturer)
{
    lock (_lecturerLock)
    {
        var existing = _lecturers.FirstOrDefault(l => l.Id == lecturer.Id);
        if (existing == null) return false;

        existing.Name = lecturer.Name;
        existing.Email = lecturer.Email;
        existing.Department = lecturer.Department;
        existing.Phone = lecturer.Phone;
        return true;
    }
}

public bool DeleteLecturer(int id)
{
    lock (_lecturerLock)
    {
        var existing = _lecturers.FirstOrDefault(l => l.Id == id);
        if (existing == null) return false;
```

```
            _lecturers.Remove(existing);
            return true;
          }
        }

        private static Lecturer CloneLecturer(Lecturer l) => new()
        {
          Id = l.Id,
          Name = l.Name,
          Email = l.Email,
          Department = l.Department,
          Phone = l.Phone
        };
    }
}
```

Lecturer.cs

```
// File: Models/Lecturer.cs
namespace CMCS.Mvc.Models
{
    public class Lecturer
    {
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;
        public string Email { get; set; } = string.Empty;
        public string Department { get; set; } = string.Empty;
        public string Phone { get; set; } = string.Empty;
    }
}
```

ProofDocument.cs

```
using System.ComponentModel.DataAnnotations;
using System.Security.Claims;

namespace CMCS.Mvc.Models
{
    public class ProofDocument
    {
        public int Id { get; set; }
        public string FileName { get; set; } = "";
        public string FilePath { get; set; } = "";
        public DateTime UploadedAt { get; set; } = DateTime.UtcNow;

        // Navigation
        public int? ClaimId { get; set; }
        public Claim? Claim { get; set; }
    }
}
```

Views

Home

Claims.cshtml

```
@{
    Layout = "_Layout";
}
<h2>Claims</h2>

<section class="card">
    <div class="button-row">
        <button type="button" class="btn-primary">Add Claim</button>
    </div>
    <table class="styled-table">
        <thead>
            <tr>
                <th>Lecturer</th>
                <th>Month</th>
                <th>Amount</th>
                <th>Status</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>Jane Doe</td>
                <td>August 2025</td>
                <td>R 4,500.00</td>
                <td>Submitted</td>
                <td><button class="btn-secondary">Verify</button> <button class="btn-
primary">Send to Manager</button></td>
```

```
        </tr>

        <tr>

          <td>John Smith</td>

          <td>August 2025</td>

          <td>R 3,200.00</td>

          <td>Verified</td>

          <td><button class="btn-primary">Send to Manager</button></td>

        </tr>

      </tbody>

  </table>

</section>
```

Index.cshtml

```
@{

  Layout = "_Layout";

}


<h2>Welcome</h2>

<p>This is the Contract Monthly Claim System (CMCS).</p>


<section class="card">

  <h3>Quick Claim Submission</h3>

  <form method="post" action="/Home/SubmitClaim" class="form-grid">

    <label>

      Name

      <input name="name" placeholder="Lecturer name" />

    </label>
```

```
        <label>
            Month
            <input name="month" placeholder="e.g. August 2025" />
        </label>
        <button type="submit" class="btn-primary">Submit Claim</button>
    </form>
    @if (ViewBag.Message != null)
    {
        <p class="info">@ViewBag.Message</p>
        <p><strong>Submitted by:</strong> @ViewBag.Name for @ViewBag.Month</p>
    }
</section>
```

LectureSubmit.cshtml

```
@{
    Layout = "_Layout";
}
<h2>Lecturer - Submit Claim</h2>

<section class="card">
    <form class="form-grid">
        <label>
            Lecturer Name
            <input placeholder="Full name" />
        </label>
        <label>
            Month
```

```
      <input placeholder="e.g. August 2025" />
    </label>
    <label>
      Hours Worked
      <input placeholder="Number of hours" />
    </label>
    <label>
      Hourly Rate
      <input placeholder="Rate per hour" />
    </label>
    <label>
      Claim Amount
      <input placeholder="Calculated amount" readonly />
    </label>
    <div class="button-row">
      <button type="button" class="btn-secondary">Attach Supporting Docs</button>
      <button type="button" class="btn-primary">Submit Claim</button>
    </div>
  </form>
</section>
```

ManagerApprove.cshtml

```
@{
  Layout = "_Layout";
}
<h2>Academic Manager - Approve Claims</h2>
```

```
<section class="card">
  <div class="claim">
    <h4>Jane Doe — August 2025</h4>
    <p><strong>Amount:</strong> R 4,500.00</p>
    <p><strong>Supporting Docs:</strong> Attached</p>
    <div class="button-row">
      <button class="btn-primary">Approve</button>
      <button class="btn-danger">Reject</button>
    </div>
  </div>
</section>
```

StatusTracking.cshtml

```
@{
  Layout = "_Layout";
}
<h2>Claim Status Tracking</h2>

<section class="card">
  <ol class="status-list">
    <li>Submitted — <strong>Aug 5, 2025</strong></li>
    <li>Verified — <strong>Aug 7, 2025</strong></li>
    <li>Approved — <strong>Aug 9, 2025</strong></li>
    <li>Settled — <strong>Aug 12, 2025</strong></li>
  </ol>
</section>
```

UploadProof

```
@{
   Layout = "_Layout";
}
<h2>Upload Supporting Documents</h2>

<section class="card">
   <form enctype="multipart/form-data" class="form-grid">
      <label>
         Select file
         <input type="file" />
      </label>
      <div class="button-row">
         <button type="button" class="btn-primary">Attach</button>
      </div>
   </form>
</section>
```

Shared

_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width" />
   <title>CMCS</title>
   <link rel="stylesheet" href="~/css/site.css" />
```

```
</head>
<body>
  <header class="site-header">
    <div class="container">
      <h1 class="logo">Contract Monthly Claim System (CMCS)</h1>
      <nav class="navbar">
        <a href="/">Home</a>
        <a href="/Home/LecturerSubmit">Lecturer Submit</a>
        <a href="/Home/Claims">Claims</a>
        <a href="/Home/ManagerApprove">Academic Manager</a>
        <a href="/Home/UploadProof">Upload Proof</a>
        <a href="/Home/StatusTracking">Status Tracking</a>
      </nav>
    </div>
  </header>

  <main class="container">
    @RenderBody()
  </main>

  <footer class="site-footer">
    <div class="container">
      <small>Contract Monthly Claim System © 2025</small>
    </div>
  </footer>
</body>
```

</html>