

PROG6212 POE Part3

Student Number: ST10443516

Presentation Topic

Contract Monthly Claim System

Digital workflow for lecturer Claim Management

What this Presentation covers

- Key system updates from part 2 to part 3
- Full walk through of the CMCS user journey
- Visual demonstration of all major features
- UX/UI Improvements
- The system value and impact

Part 2

Core Features of Part2

- **Lecturer Functionality**
 - Submit monthly claims
 - Upload supporting documents
 - View (Track) list of submitted claims
- **Coordinator Functionality**
 - View pending claims
 - Approve or reject lecturer claims
- **Manager Functionality**
 - View coordinator-approved claims
 - Approve or reject them
- **Database & Models Built**
 - Claims, Documents, Users, User Roles
- **Basic UI & Bootstrap Layout**
 - Tables, forms, and buttons functioning but minimal design

Part 3

Core Features of Part3

- Added HR who is a super user
- Login & Sessions
- Auto-calculation for lecturers
- Secure role based access
- PDF GENERATOR
- Removal of manual rate entry

System Architecture Overview

User Roles & Workflow:

Lecturer → Coordinator → Manager → HR → PDF Reports

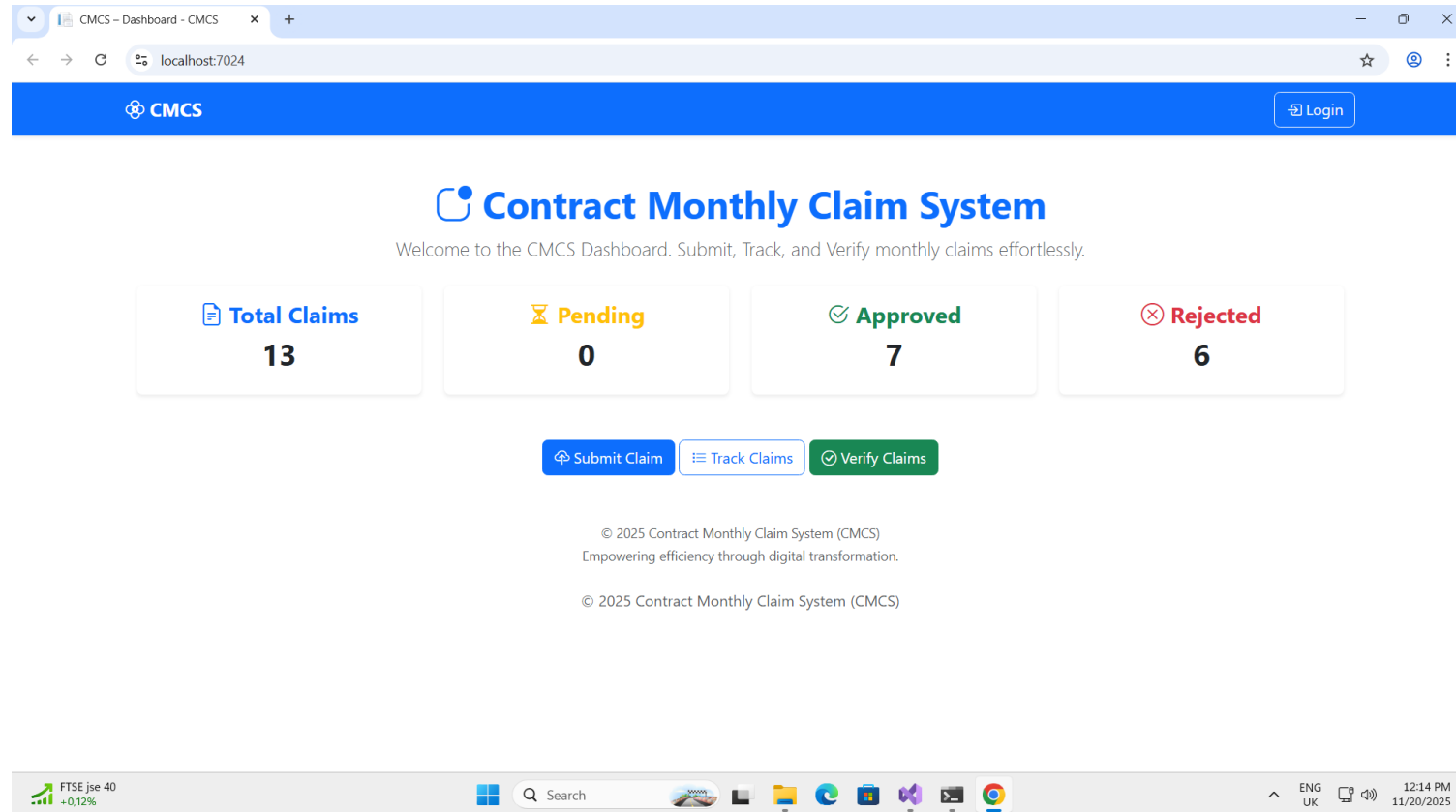
Functional Flow:

- Lecturer submits claim + uploads evidence
- Coordinator validates & approves/rejects
- Manager reviews coordinator-approved claims
- HR completes administrative processing + manages user accounts
- System generates PDF reports for auditing

Technologies Used:

- ASP.NET Core MVC
- SQL / EF Core / Azure Storage
- Bootstrap 5
- QuestPDF for reporting

Home Dashboard.



Home Dashboard

Features of the Home dashboard

- Instant overview of user activity
- Login button
- Summary cards:
 - Total Claims
 - Pending Claims
 - Approved Claims
 - Rejected Claims
- Personalized greeting for logged-in user
- Clean Bootstrap card styling

Login screen

[Login](#)

Login

Enter your username and password

Username

Password



Login

Login Details (for marker):

HR: Username: [HR1](#), Password: [123](#)

Lecturer: Username: [L2001](#), Password: [123](#)

Coordinate: Username: [COORD1](#), Password: [123](#)

Manager: Username: [MANAGER1](#), Password: [123](#)

© 2025 Contract Monthly Claim System (CMCS)

© 2025 Contract Monthly Claim System (CMCS)

Login Screen

Features of the login page

- Clean, professional login layout
- **“Show/Hide Password”** eye icon for accessibility
- User role-based redirection after login
- Validation messages for incorrect credentials
- Secure handling of sessions

Sessions + Role-Based Access Control

- Users can only access screens meant for their role.
- HR → User management
Lecturer → Submit + Track
Coordinator → Verify
Manager → Approve & Pay
- Explain Under Screenshot
- Each controller action is restricted
- If a user attempts to access another role's page → access denied

```
namespace CMCS.Controllers
{
    [AuthorizeRole("Coordinator")]
    public class CoordinatorController : Controller
    {
        private readonly ApplicationDbContext _db;
        public CoordinatorController(ApplicationDbContext db) => _db = db;
    }
}
```

```
namespace CMCS.Controllers
{
    [AuthorizeRole("HR")]
    public class HRController : Controller
    {
        private readonly ApplicationDbContext _db;
        public HRController(ApplicationDbContext db) => _db = db;
    }
}
```

```
namespace CMCS.Controllers
{
    [AuthorizeRole("Lecturer")]
    public class LecturerController : Controller
    {
        private readonly ApplicationDbContext _db;
        private readonly IClaimService _service;
    }
}
```

```
namespace CMCS.Controllers
{
    [AuthorizeRole("Manager")]
    public class ManagerController : Controller
    {
        private readonly ApplicationDbContext _db;
        public ManagerController(ApplicationDbContext db) => _db = db;
    }
}
```

Lecturer Page

Submit Claim Page

CMCS

SubmitTrack

L2001 (Lecturer)

Login

Logout

Submit Your Claim

Month

Hours Worked

Hourly Rate

January 0001

0

500.00

Lecturer Name

Lecturer ID

Total Amount

Namjoon Rm

2001

0.00

Notes (Optional)

Upload Supporting Documents

Choose FilesNo file chosen

Submit Claim

© 2025 Contract Monthly Claim System (CMCS)

Track Claim page

CMCS

SubmitTrack

L2001 (Lecturer)

Login

Logout

My Submitted Claims

Month	Hours Worked	Hourly Rate	Lecturer Name	Lecturer ID	Total	Status	Notes	Documents	Actions
2025-01	100.00	500.00	Namjoon Rm	L2001	50,000.00	Pending	bjbbbbjlnln	f890f889-23fb-40db-9b9d-3841305043e9_09f8387a2ccc43539938b7383dec0da9.pdf	Delete
2025-12	60.00	500.00	Namjoon Rm	L2001	30,000.00	Approved	kanknj	cake.jpg	Delete
2026-01	70.00	500.00	Namjoon Rm	L2001	35,000.00	Rejected	janniczlnk	cupcakes.jpg	Delete
2025-12	100.00	500.00	Namjoon Rm	L2001	50,000.00	Approved	sjhojnzkoniojskzjk	7e358a5b-db2f-47ce-91d1-dd24fe54c711_6fba3ee674ed42b38126ad01410162a.pdf	Delete
2026-01	80.00	500.00	Namjoon Rm	L2001	40,000.00	Approved	jih6ftgbhn	7e1b3cd2-f86f-4535-a51c-9a4b3c09aa4c_09f8387a2ccc43539938b7383dec0da9.pdf	Delete

© 2025 Contract Monthly Claim System (CMCS)

Lecturer Page

Submit Claim

What the Page Shows:

- Input for Month, Hours Worked, Notes
- File upload for evidence (JPEG/PDF/PNG)
- Auto-total calculation
- Rate is added on the HR page
- Name is auto added from HR page
- Limits only to 180 hours per month

Track Claim Page

What the Page Shows:

- Interactive table with claims
- Status badges (Pending, Approved, Rejected)
- Downloadable evidence links
- Delete option for pending claims
- Accurate totals per month

Lecturer Data Pull

LecturerController → SubmitClaim()

Explaintation of Screenshot

- ❑ System fetches Lecturer HourlyRate, Name, ID from DB
- ❑ Only HoursWorked + Notes are entered manually

```
public async Task<ActionResult> Submit(SubmitClaimVm vm)
{
    if (!ModelState.IsValid)
    {
        return View(vm);
    }

    try
    {
        var claim = new Claim
        {
            LecturerName = vm.LecturerName,
            LecturerId = HttpContext.Session.GetString("Username"),
            Month = vm.Month,
            HoursWorked = vm.HoursWorked,
            HourlyRate = vm.HourlyRate,
            Notes = vm.Notes,
            Status = ClaimStatus.Pending
        };

        await _service.SubmitClaimAsync(claim, vm.FileUploads);

        TempData["Success"] = "Claim submitted successfully!";
        return RedirectToAction("Track");
    }
    catch (Exception ex)
    {
        ModelState.AddModelError(string.Empty, ex.Message);
        return View(vm);
    }
}
```

0 references

```
public IActionResult Submit()
{
    var fullName = HttpContext.Session.GetString("FullName");
    var lecturerId = HttpContext.Session.GetInt32("UserId");
    var rate = HttpContext.Session.GetString("HourlyRate");

    if (fullName == null || lecturerId == null)
        return RedirectToAction("Login", "Account");

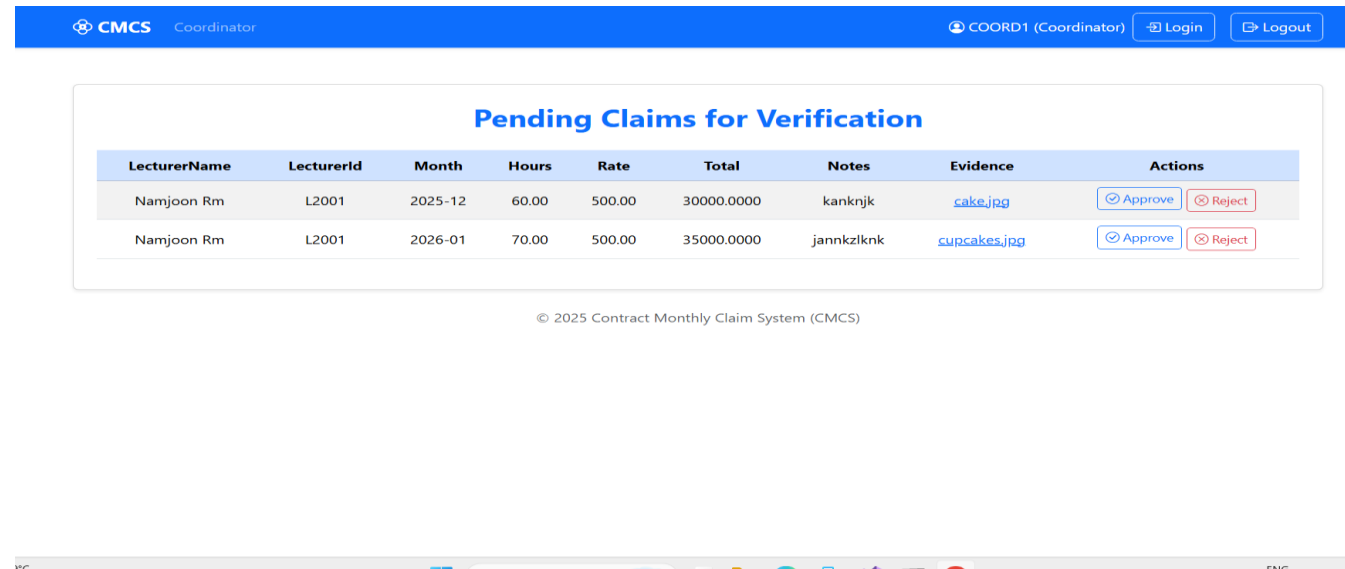
    var vm = new SubmitClaimVm
    {
        LecturerName = fullName,
        LecturerId = lecturerId.Value.ToString(),
        HourlyRate = decimal.Parse(rate ?? "0")
    };

    ModelState.Clear();
    return View(vm);
}
```

Coordinator Page

Features :

- Holds the pending claims from the Lecturer
- Approve or Reject
- Notes & document preview
- Clearly labelled tables



The screenshot displays the 'Coordinator Page' of the 'CMCS' (Contract Monthly Claim System). The page features a blue header bar with the 'CMCS' logo and 'Coordinator' text on the left, and a user profile 'COORD1 (Coordinator)' with 'Login' and 'Logout' buttons on the right. The main content area is titled 'Pending Claims for Verification' and contains a table with the following data:


LecturerName	LecturerId	Month	Hours	Rate	Total	Notes	Evidence	Actions
Namjoon Rm	L2001	2025-12	60.00	500.00	30000.0000	kanknjc	cake.jpg	Approve Reject
Namjoon Rm	L2001	2026-01	70.00	500.00	35000.0000	jannkzlnk	cupcakes.jpg	Approve Reject

At the bottom of the page, there is a copyright notice: '© 2025 Contract Monthly Claim System (CMCS)'.


Manager: Coordinator-Approved Claims


Features


- Hold the approved claims from the coordinator
- Approve & Pay
- Reject with a click
- Access to documents and notes


 CMCS



Manager

 MANAGER1 (Manager)

 Login

 Logout

 **Manager — Coordinator Approved**

Month	Hours	Lecturer Name	Lecturer ID	Rate	Total	Notes	Evidence	Actions
2025-12	60.00	Namjoon Rm	L2001	500.00	30,000.00	kanknjik	Open	<div> Approve & Pay</div> <div> Reject</div>

© 2025 Contract Monthly Claim System (CMCS)

HR Page

HR Index Page

CMCSHR – Users

HR1 (HR)LoginLogout

User Management

Download ReportAdd User

Username	Full Name	Email	Hourly Rate	Role	Actions
hazelshaka	hazel shaka	hazelshaka@gmail.com	200.00	Lecturer	<div><div></div><div></div><div></div></div>
Kim009	kim Yam	jackharlow@gmail.com	70.00	Lecturer	<div><div></div><div></div><div></div></div>
L2001	Namjoon Rm	namjoonrm@gmail.com	500.00	Lecturer	<div><div></div><div></div><div></div></div>
COORD1	john Baptist	nickiminaj@gmail.com	0.00	Coordinator	<div><div></div><div></div><div></div></div>
MANAGER1	Boss Baby	euphoriajung@gmail.com	0.00	Manager	<div><div></div><div></div><div></div></div>
HR1	Queen Bey	bambamgot7@gmail.com	0.00	HR	<div><div></div><div></div><div></div></div>

© 2025 Contract Monthly Claim System (CMCS)

HR Create New User Page

CMCSHR – Users

HR1 (HR)LoginLogout

Add New User

Username

Password

FirstName

LastName

Email

HourlyRate

0.00

Role

Lecturer

-- Select Role --

Lecturer

Coordinator

Manager

HR

Save

Cancel

© 2025 Contract Monthly Claim System (CMCS)

HR Page

HR index page

- Add users
- Edit users
- Delete accounts
- Access PDF reports

HR Create New User

- Create username
- Add password
- Add the users rate
- The users email
- Adding the role

HR Creates & Manages All Users

Screenshot your HRController.cs →
Create() method

```
[HttpGet]
0 references
public IActionResult CreateUser()
{
    return View(new AppUser());
}

[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> CreateUser(AppUser model)
{
    if (!ModelState.IsValid)
        return View(model);

    if (string.IsNullOrEmpty(model.PasswordHash))
    {
        ModelState.AddModelError("PasswordHash", "Password is required.");
        return View(model);
    }

    _db.Users.Add(model);
    await _db.SaveChangesAsync();

    TempData["Success"] = "User created successfully!";
    return RedirectToAction(nameof(Index));
}
```

Screenshot the EF
Model AppUser.cs

```
1  using System.ComponentModel.DataAnnotations;
2
3  namespace CMCS.Models
4  {
5      16 references
6      public class AppUser
7      {
8
9          [Key]
10         11 references
11         public int Id { get; set; }
12
13         [Required, MaxLength(50)]
14         15 references
15         public string Username { get; set; } = string.Empty;
16
17         [Required, MaxLength(200)]
18         8 references
19         public string PasswordHash { get; set; } = string.Empty;
20
21         [Required, MaxLength(50)]
22         12 references
23         public string FirstName { get; set; } = string.Empty;
24
25         [Required, MaxLength(50)]
26         12 references
27         public string LastName { get; set; } = string.Empty;
28
29         [Required, EmailAddress]
30         12 references
31         public string Email { get; set; } = string.Empty;
32
33         [Range(0, 2000)]
34         12 references
35         public decimal HourlyRate { get; set; }
36
37         [Required]
38         15 references
39         public UserRole Role { get; set; }
40     }
41 }
```

Explaintation of Screenshot

- ☐ Shows HR creating users with name, username, role, hourly rate
- ☐ Password hashing or default password assignment
- ☐ Role assignment logic

Pdf generated by HR

≡ User Report

1 / 1 | - 100% + | [Icon] [Icon] | [Icon] [Icon]

↓ [Icon] [Icon]



1

CMCS - User Report

ID	Username	First Name	Last Name	Email	Rate
1	L2001	Namjoon	Rm	namjoonrm@gmail.c om	500.00
9	Kim009	kim	Yam	jackharlow@gmail.c om	70.00
10	hazelshaka	hazel	shaka	hazelshaka@gmail.c om	200.00

HR Generates Reports (PDF)

The GeneratePdf() method

```
public async Task<IActionResult> GeneratePdf()
{
    var lecturers = await _db.Users
        .Where(u => u.Role == UserRole.Lecturer)
        .Select(u => new AppUser
        {
            Id = u.Id,
            Username = u.Username,
            Email = u.Email,
            Role = u.Role,

            FirstName = u.FirstName,
            LastName = u.LastName,
            HourlyRate = u.HourlyRate
        })
        .ToListAsync();

    var vm = new ReportVm
    {
        Users = lecturers
    };

    var pdfBytes = new ReportDocument(vm).GeneratePdf();

    return File(pdfBytes, "application/pdf", "LecturerReport.pdf");
}
```

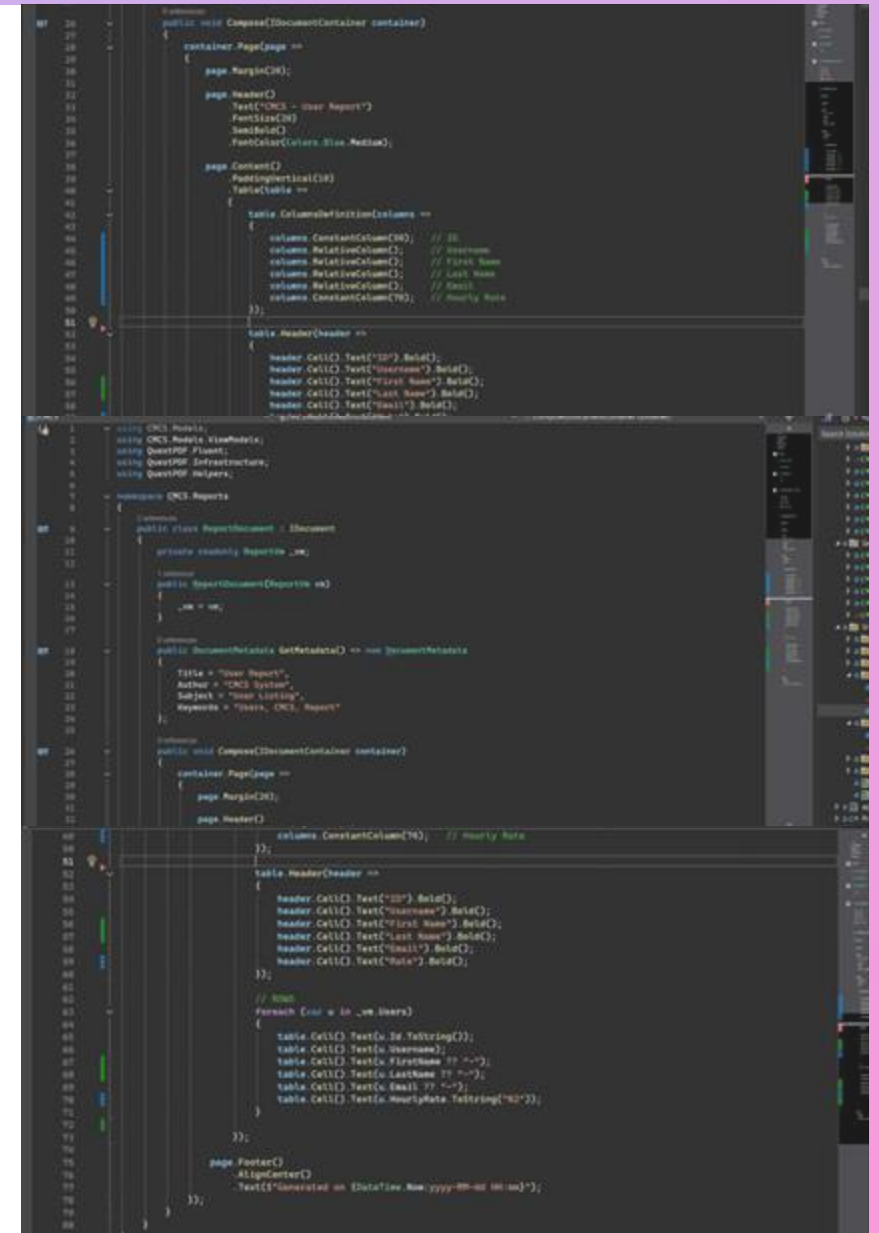
Explantation of Screenshot

- System loads all users or claims
- Renders them into a structured PDF
- HR can download with one click

The QuestPDF document layout class

Explaintation of Screenshot

- System loads all users or claims
- Renders them into a structured PDF
- HR can download with one click



Full System Demonstration Summary

The CMCS System has :

- Full claim workflow from submission to payment
- Multi-role approval chain
- Secure authentication
- Evidence management
- PDF reporting
- Robust search
- Clean, user-friendly interface