

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.

The screenshot shows the CMCS Home Dashboard running on a local host at port 7024. The dashboard features a blue header with the CMCS logo and a 'Login' button. Below the header, the title 'Contract Monthly Claim System' is displayed with a gear icon. A welcome message reads: 'Welcome to the CMCS Dashboard. Submit, Track, and Verify monthly claims effortlessly.' The dashboard displays four main statistics in cards:

- Total Claims**: 13 (blue icon)
- Pending**: 0 (orange icon)
- Approved**: 7 (green icon)
- Rejected**: 6 (red icon)

Below these cards are three buttons: 'Submit Claim' (blue), 'Track Claims' (white), and 'Verify Claims' (green). At the bottom of the dashboard, there is a footer with copyright information and a system status bar.

© 2025 Contract Monthly Claim System (CMCS)
Empowering efficiency through digital transformation.

© 2025 Contract Monthly Claim System (CMCS)

FTSE jse 40 +0.12% 12:14 PM 11/20/2025

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

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

```
1  namespace CMCS.Controllers
2  {
3      [AuthorizeRole("Coordinator")]
4      1 reference
5      public class CoordinatorController : Controller
6      {
7          private readonly AppDbContext _db;
8          0 references
9          public CoordinatorController(AppDbContext db) => _db = db;
10     }
11 }
```

```
10  namespace CMCS.Controllers
11  {
12      [AuthorizeRole("HR")]
13      1 reference
14      public class HRController : Controller
15      {
16          private readonly AppDbContext _db;
17          0 references
18          public HRController(AppDbContext db) => _db = db;
19      }
20 }
```

```
 1  namespace CMCS.Controllers
 2  {
 3      [AuthorizeRole("Lecturer")]
 4      2 references
 5      public class LecturerController : Controller
 6      {
 7          private readonly AppDbContext _db;
 8          private readonly IClaimService _service;
 9          1 reference
10     }
11 }
```

```
 1  namespace CMCS.Controllers
 2  {
 3      [AuthorizeRole("Manager")]
 4      1 reference
 5      public class ManagerController : Controller
 6      {
 7          private readonly AppDbContext _db;
 8          0 references
 9          public ManagerController(AppDbContext db) => _db = db;
10     }
11 }
```

Lecturer Page

Submit Claim Page

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		
<input type="file"/> Choose Files No file chosen		
Submit Claim		

© 2025 Contract Monthly Claim System (CMCS)

Track Claim page

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	bjbbbbbjlnln	f890fd89-23fb-40db-9b9d-3841305043e9_09f8387a2ccc43539938b7383dec0da9.pdf	Delete
2025-12	60.00	500.00	Namjoon Rm	L2001	30.000.00	Approved	kanknjk	cake.jpg	Delete
2026-01	70.00	500.00	Namjoon Rm	L2001	35.000.00	Rejected	jannkzlkknk	cupcakes.jpg	Delete
2025-12	100.00	500.00	Namjoon Rm	L2001	50.000.00	Approved	sjhojnzkoniojsizjk	7e358a5b-db2f-47ce-91d1-dd24fe54c711_6fea3e614e42b38126ad0f1f410162e.pdf	Delete
2026-01	80.00	500.00	Namjoon Rm	L2001	40.000.00	Approved	jih6ftgbhn	7e@b3cd2-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()

Explaintion of Screenshot

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

```
Technico
public async Task<IActionResult> 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

CMCS Coordinator

COORD1 (Coordinator) [Login](#) [Logout](#)

Pending Claims for Verification

LecturerName	LecturerId	Month	Hours	Rate	Total	Notes	Evidence	Actions
Namjoon Rm	L2001	2025-12	60.00	500.00	30000.0000	kanknjk	cake.jpg	<input checked="" type="checkbox"/> Approve <input type="checkbox"/> Reject
Namjoon Rm	L2001	2026-01	70.00	500.00	35000.0000	jannkzlnknk	cupcakes.jpg	<input checked="" type="checkbox"/> Approve <input type="checkbox"/> Reject

© 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



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	kanknjk	Open	<button>Approve & Pay</button> <button>Reject</button>

© 2025 Contract Monthly Claim System (CMCS)

HR Page

HR Index Page

The screenshot shows the CMCS HR - Users interface. At the top, there's a blue header bar with the CMCS logo, the title "HR - Users", and navigation links for "HR1 (HR)", "Login", and "Logout". Below the header is a section titled "User Management" with a table. The table has columns for "Username", "Full Name", "Email", "Hourly Rate", "Role", and "Actions". The data in the table is as follows:

Username	Full Name	Email	Hourly Rate	Role	Actions
hazelshaka	hazel shaka	hazelshaka@gmail.com	200.00	Lecturer	
Kim009	kim Yam	jackcharlow@gmail.com	70.00	Lecturer	
L2001	Namjoon Rm	namjoonrm@gmail.com	500.00	Lecturer	
COORD1	john Baptist	nickiminaj@gmail.com	0.00	Coordinator	
MANAGER1	Boss Baby	euphoriahjung@gmail.com	0.00	Manager	
HR1	Queen Bey	bambangot7@gmail.com	0.00	HR	

At the bottom left, there's a copyright notice: "© 2025 Contract Monthly Claim System (CMCS)".

HR Create New User Page

The screenshot shows the CMCS HR - Users interface with a form titled "+ Add New User". The form fields are: "Username" (text input), "Password" (text input), "FirstName" (text input), "LastName" (text input), "Email" (text input), "HourlyRate" (text input with value "0.00"), and "Role" (dropdown menu). The dropdown menu is open, showing options: "Lecturer" (selected), "Coordinator", "Manager", and "HR". At the bottom left of the form is a "Save" button and at the bottom right is a "Cancel" button. A copyright notice "© 2025 Contract Monthly Claim System (CMCS)" is at the bottom center.

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]
public IActionResult CreateUser()
{
    return View(new AppUser());
}

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

    if (string.IsNullOrWhiteSpace(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

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

Explaintion 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% + | ☰ 🔍 | ⌂ ⌂

⤵ ⌂ ⌂



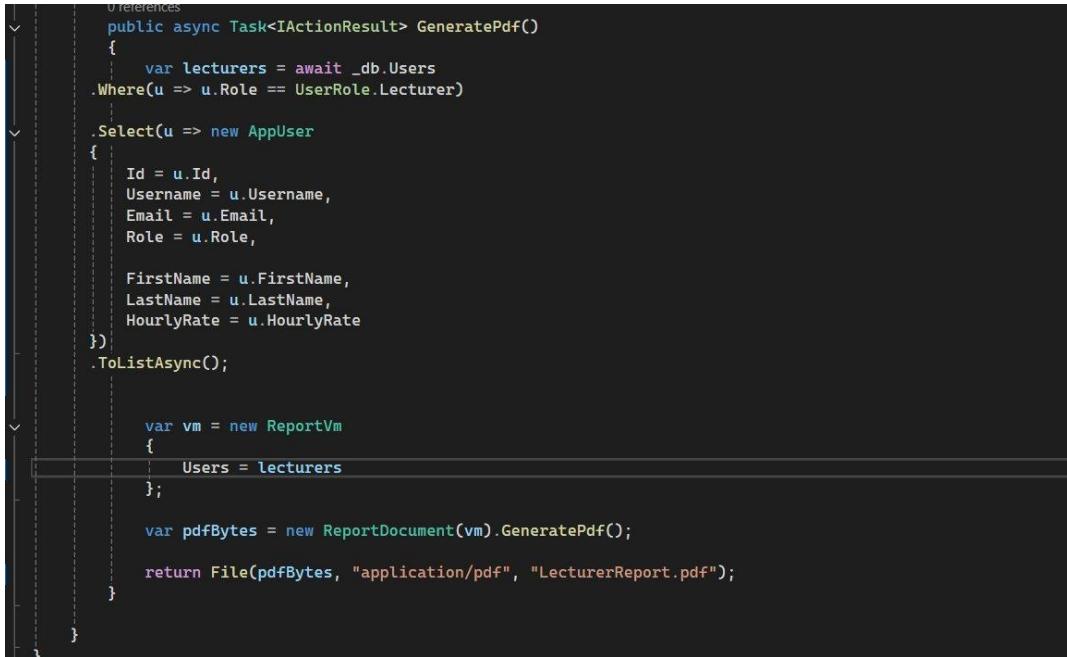
1

CMCS - User Report

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

HR Generates Reports (PDF)

The GeneratePdf() method



A screenshot of a code editor showing a C# method named `GeneratePdf()`. The code uses Entity Framework to query users from a database, filters them by role (Lecturer), and then maps them to a `ReportVm` object. Finally, it generates a PDF document from the `ReportVm` and returns it as a file response.

```
public async Task<IActionResult> GeneratePdf()
{
    var lecturers = await _db.Users
        .Where(u => u.Role == UserRole.Lecturer)
        .Select(u => new ApplicationUser
    {
        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");
}
```

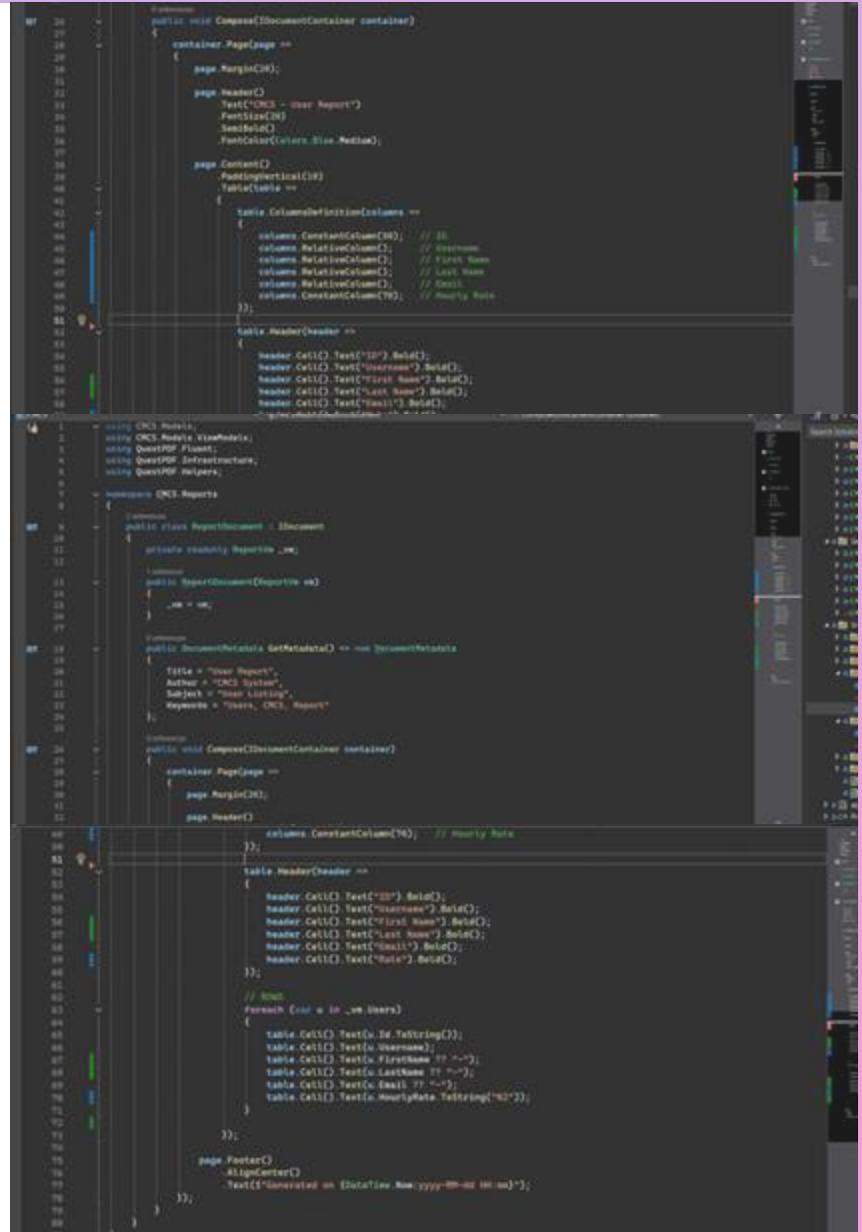
Explaintion of Screenshot

- 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



The screenshot shows a code editor with a dark theme, displaying C# code for generating a PDF report using the QuestPDF library. The code includes a class definition for a report page and a method to compose the document container.

```
using (CNCs.Rports)
{
    public class ReportDocument : Document
    {
        private readonly ReportDocument _report;
        public ReportDocument(ReportDocument report)
        {
            _report = report;
        }

        public DocumentMetadata GetMetadata() => new DocumentMetadata
        {
            Title = "User Report",
            Author = "CNCS System",
            Subject = "User Listing",
            Keywords = "Users, CNCS, Report"
        };

        public void Compose(DocumentContainer container)
        {
            container.Page[0] =
            {
                page.Margin[20];
                page.Header[0]
                Text("CNCS - User Report")
                FontSize(20)
                Bold(true)
                ForeColor(Color.Blue.Medium);
                page.Content[0]
                PaddingHorizontal(10)
                Table[0] ==
                {
                    table.ColumnsDefinition[0] ==
                    {
                        column.ConstantColumn[0]; // ID
                        column.RelativeColumn[1]; // Username
                        column.RelativeColumn[2]; // First Name
                        column.RelativeColumn[3]; // Last Name
                        column.RelativeColumn[4]; // Email
                        column.ConstantColumn[5]; // Hourly Rate
                    };
                    table.Header[0] ==
                    {
                        header.Cell[0].Text("ID").Bold();
                        header.Cell[1].Text("Username").Bold();
                        header.Cell[2].Text("First Name").Bold();
                        header.Cell[3].Text("Last Name").Bold();
                        header.Cell[4].Text("Email").Bold();
                        header.Cell[5].Text("Hourly Rate").Bold();
                    };
                    table.Rows[0] ==
                    {
                        row.Cells[0].Text("1");
                        row.Cells[1].Text("JohnDoe");
                        row.Cells[2].Text("John");
                        row.Cells[3].Text("Doe");
                        row.Cells[4].Text("johndoe@example.com");
                        row.Cells[5].Text("100");
                    };
                };
                page.Footer[0]
                AlignCenter()
                .Text("Generated on " + DateTime.Now.ToString("MM/dd/yyyy HH:mm:ss"));
            };
        }
    }
}
```

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