Introduction

The CMCS automates claim submission, approval, and payment for contract staff, leading to increased efficiency and reduced administrative overhead.

Scope: The system allows users to submit claims for hours worked.

Automatically calculate final payments.

Approve or reject claims based on predetermined policies.

Target users are contract lecturers who file claims.

HR/Coordinators authorize claims.

Administrators are responsible for managing the system.

2. System Features.

Lecturer Features:

Submit Claims: Lecturers can log in and submit claim forms for their hours worked.

Claim Status Tracking: View the status of your claims (pending, approved, or denied).

Personal Information Management: Update contact information.

Coordinator/HR features:

Review Claims: Verify submitted claims, hours, and policies.

Approve genuine claims or reject those that violate policies.

Generate Reports: Summarize accepted claims for payment processing.

Automation features:

Automatically calculate payouts based on hourly rates and hours performed.

Send email notifications to lecturers regarding claim acceptance or rejection.

Validation: Complete all fields before submitting.

3. System Architecture Technology. Stack:

Frontend: Razor Pages (ASP.NET Core MVC).

Backend: ASP.NET Core and Entity Framework Core.

Database: SQL Server.

Authentication: Use ASP.NET Identity for user authentication and role management.

Key Components:

Controllers: Manage the logic and data flow.

ClaimController manages claims.

AccountController is responsible for authentication.

Database: Manages claims, users, and approval statuses.

Views use Razor templates to display forms, data, and reports.

Workflow:

Lecturer submits a claim, validates it, and saves it to the database.

The coordinator evaluates the claim, approves/rejects it, and updates the database.

The system automatically calculates payment and generates invoices for HR processing.

4. Technical Implementation.

Data Structure:

Tables:

Users: User data is stored (for example, lecturers and HR staff).

Claims: Tracks submitted claims using fields such as HoursWorked, HourlyRate, and Status.

User roles are defined as follows: lecturer, coordinator, and administrator.

Claim Submittal Flow:

Lecturer Input:

The form includes three fields: hours worked, hourly rate, and description.

Validation: Backend checks for correct data.

Database Insertion: Claim stored as Pending.

Claim Approval Flow:

Coordinator View: The dashboard shows claims with details.

Decision: Use buttons to approve or reject claims.

Notification: An email was sent to the lecturer.

Auto-Calculations:

Total Payment = Hours Worked x Hourly Rate.

Automatically updates the Total Payment field in the Claims table.

5.

Role	Permissions
Lecturer	Submit claims, see claim status, and update profiles.
Coordinator	Approve/reject claims and read reports.
Admin	Manage users and configure settings.

7. Testing Unit Tests: Verify payment computations with various inputs.

Check form validations for claim submissions.

Integration tests cover the full procedure (submission, approval, and payment).

Manually test all UI forms for usability and correctness.

8. Challenges and Solutions.

Renaming the controller produced routing complications.

Solution: Reverted the names and changed the routes in Startup.cs.

Challenge: The database is not fully configured.

Solution: Added missing entities and performed migrations.