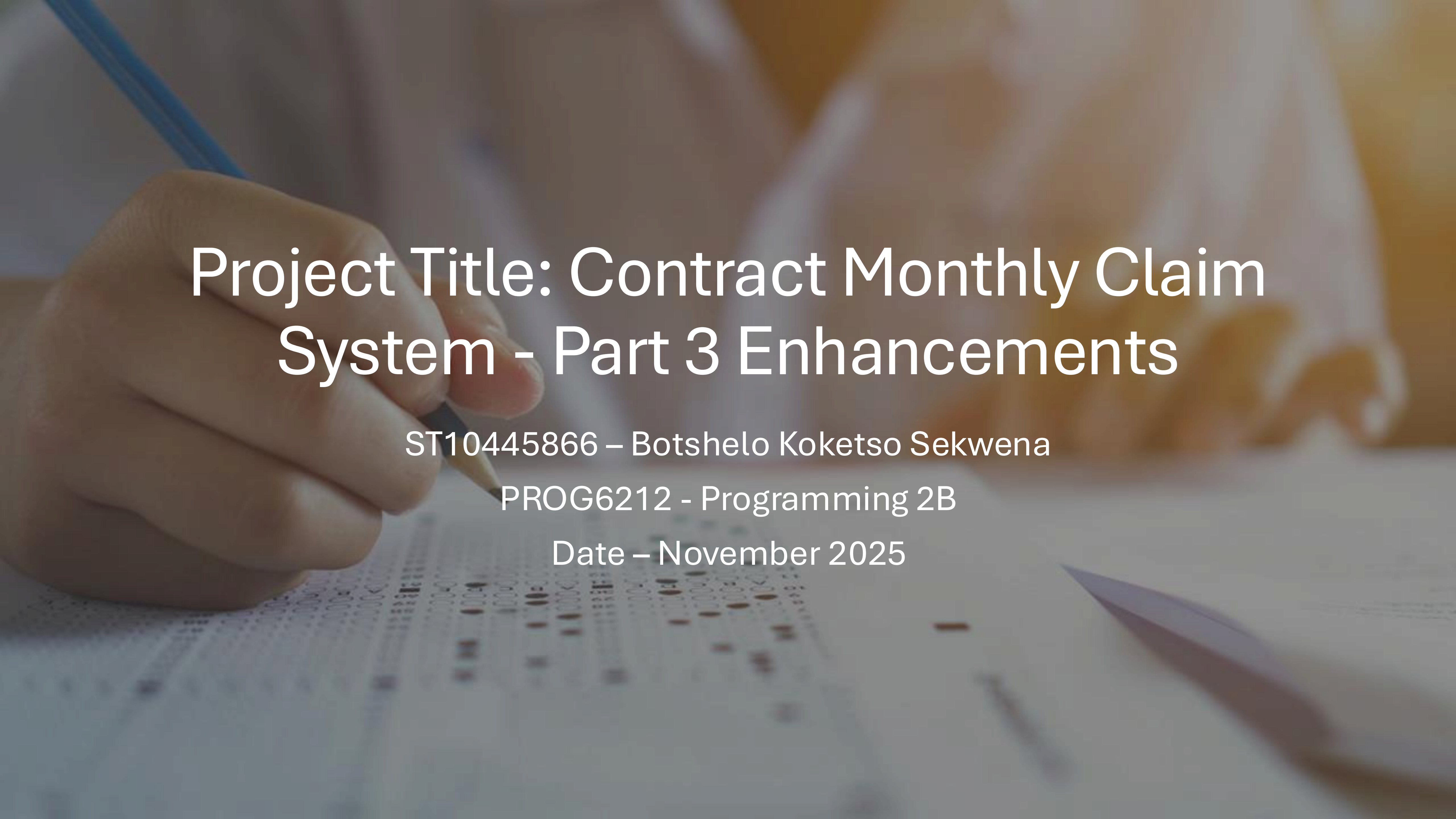
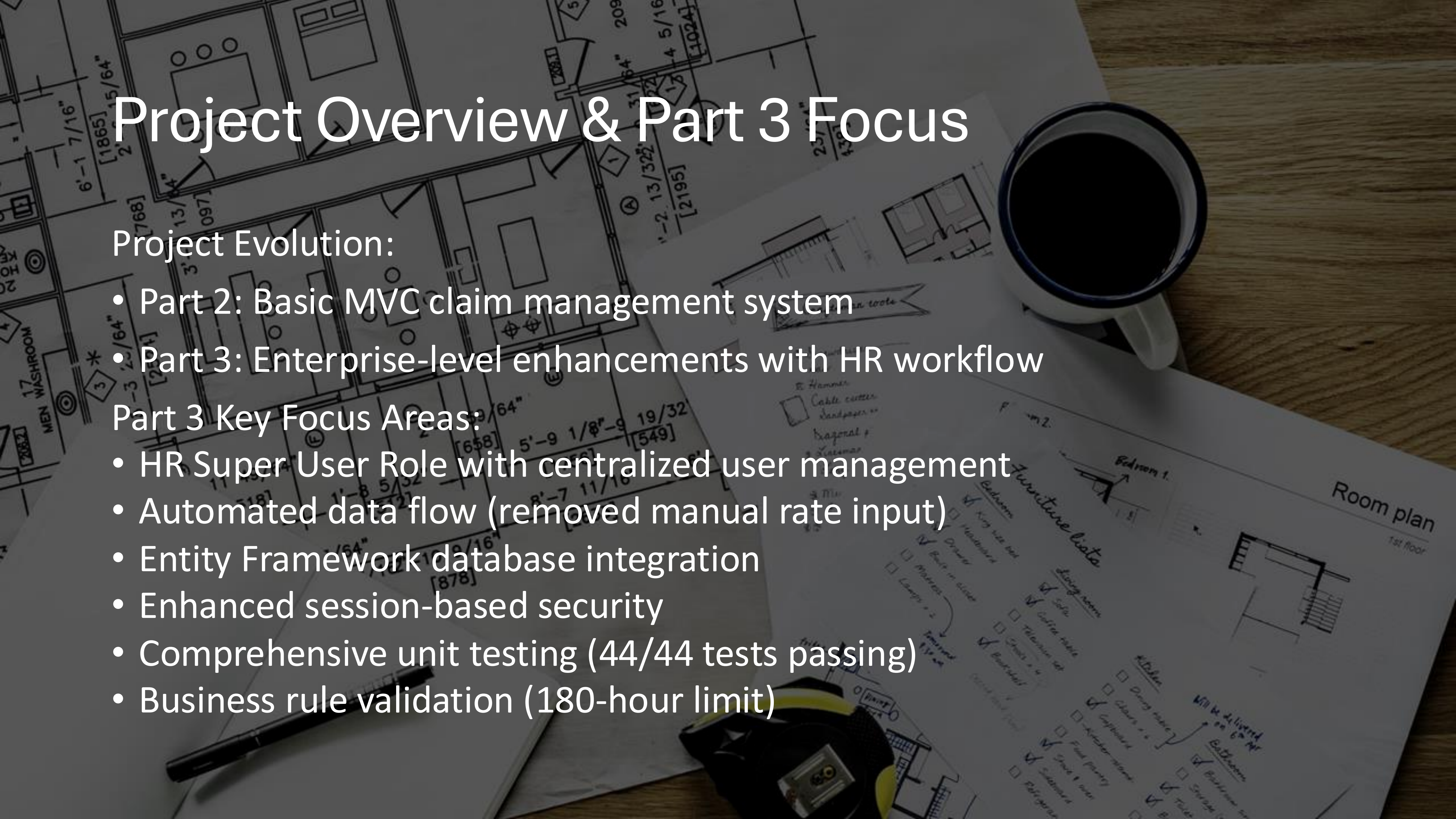


Project Title: Contract Monthly Claim System - Part 3 Enhancements

ST10445866 – Botshelo Koketso Sekwena

PROG6212 - Programming 2B

Date – November 2025



Project Overview & Part 3 Focus

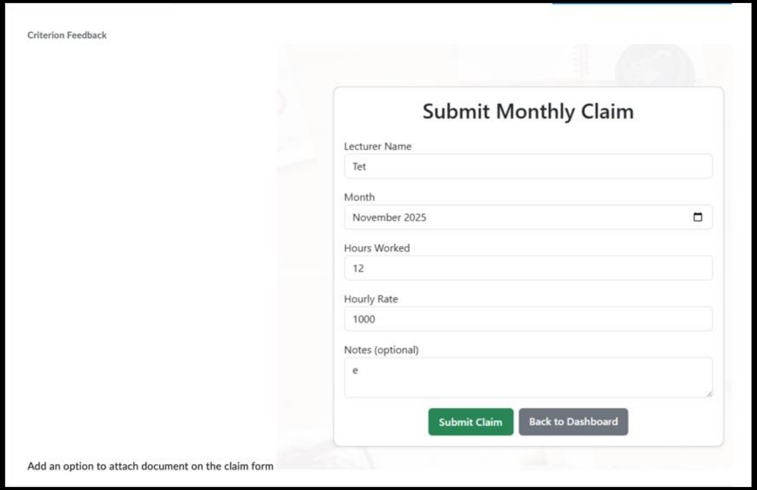
Project Evolution:

- Part 2: Basic MVC claim management system
- Part 3: Enterprise-level enhancements with HR workflow

Part 3 Key Focus Areas:

- HR Super User Role with centralized user management
- Automated data flow (removed manual rate input)
- Entity Framework database integration
- Enhanced session-based security
- Comprehensive unit testing (44/44 tests passing)
- Business rule validation (180-hour limit)

Feedback-Driven Improvements



Criterion Feedback

Submit Monthly Claim

Lecturer Name: Tet

Month: November 2025

Hours Worked: 12

Hourly Rate: 1000

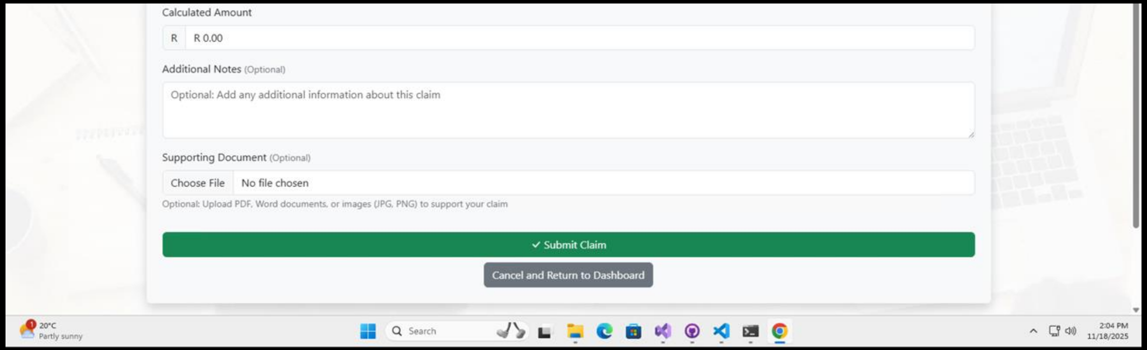
Notes (optional): e

Add an option to attach document on the claim form

Feedback-Driven Improvements

1. Adding an option to attach document on the Claim Form

- Before applying adjustments


Calculated Amount

R R 0.00

Additional Notes (Optional)

Optional: Add any additional information about this claim

Supporting Document (Optional)

Choose File No file chosen

Optional: Upload PDF, Word documents, or images (JPG, PNG) to support your claim

Feedback-Driven Improvements

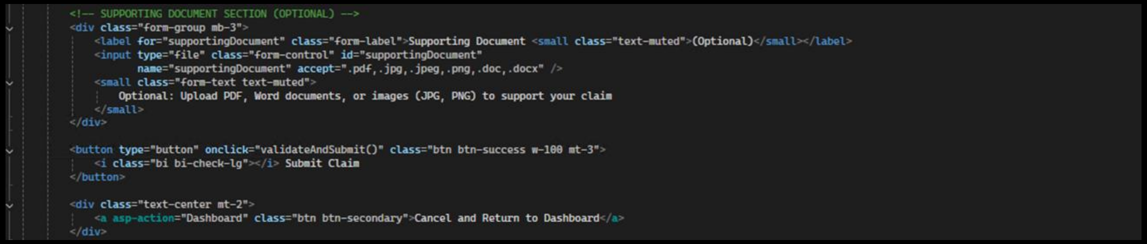
1. Adding an option to attach document on the Claim Form

- After applying adjustment
- Implemented the option to attach a supporting document in the Claim Form

Feedback-Driven Improvements

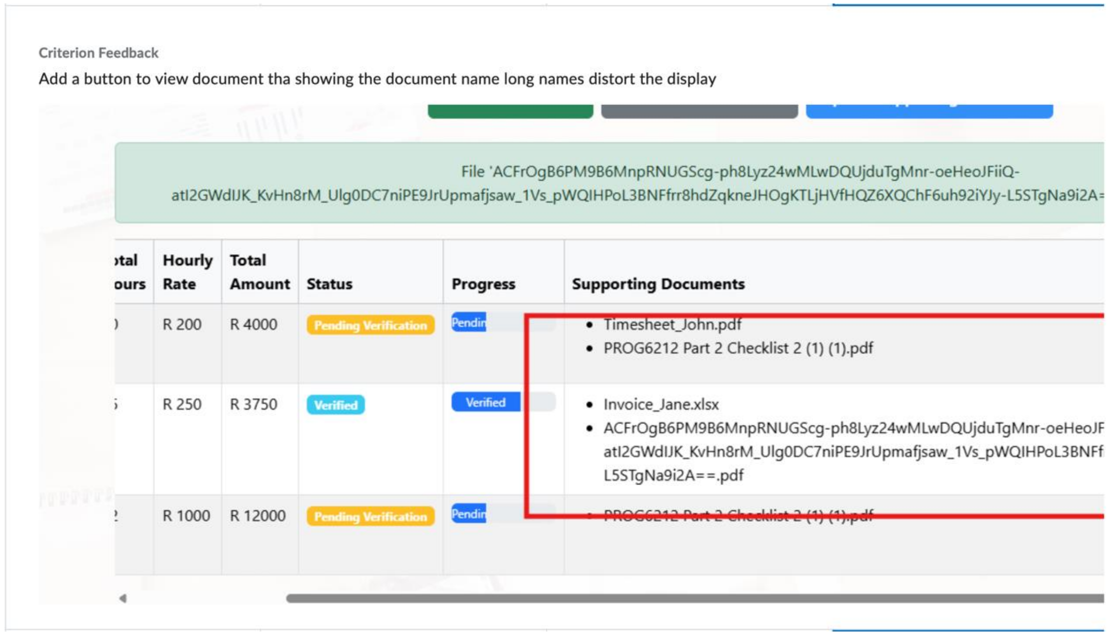
1. Adding an option to attach document on the Claim Form

- After applying adjustment
- Technical Implementation:



```
<!-- SUPPORTING DOCUMENT SECTION (OPTIONAL) -->
<div class="form-group mb-3">
    <label for="supportingDocument" class="form-label">Supporting Document <small class="text-muted">(Optional)</small></label>
    <input type="file" class="form-control" id="supportingDocument"
           name="supportingDocument" accept=".pdf,.jpg,.jpeg,.png,.doc,.docx" />
    <small class="form-text text-muted">
        Optional: Upload PDF, Word documents, or images (JPG, PNG) to support your claim
    </small>
</div>

<button type="button" onclick="validateAndSubmit()" class="btn btn-success w-100 mt-3">
    <i class="bi bi-check-lg"></i> Submit Claim
</button>

<div class="text-center mt-2">
    <a asp-action="Dashboard" class="btn btn-secondary">Cancel and Return to Dashboard</a>
</div>
```

Feedback-Driven Improvements



Criterion Feedback

Add a button to view document tha showing the document name long names distort the display

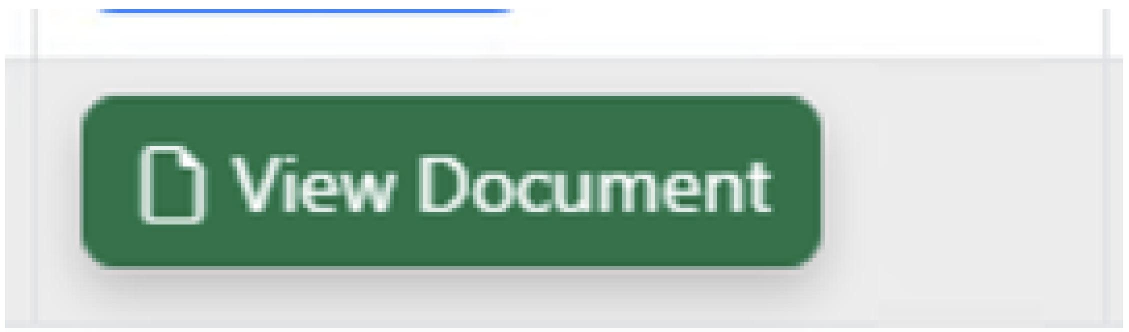
File 'ACFrOgB6PM9B6MnpRNUGScg-ph8Lyz24wMLwDQUjduTgMnr-oeHeoJFiiQ-atI2GWdIJK_KvHn8rM_Ulg0DC7niPE9JrUpmafjsaw_1Vs_pWQIHPoL3BNFfrr8hdZqkneJHOgKTLjHVfHQZ6XQChF6uh92iYJy-L5STgNa9i2A=

tal ours	Hourly Rate	Total Amount	Status	Progress	Supporting Documents
)	R 200	R 4000	Pending Verification	Pendin	• Timesheet_John.pdf • PROG6212 Part 2 Checklist 2 (1) (1).pdf
5	R 250	R 3750	Verified	Verified	• Invoice_Jane.xlsx • ACFrOgB6PM9B6MnpRNUGScg-ph8Lyz24wMLwDQUjduTgMnr-oeHeoJF atI2GWdIJK_KvHn8rM_Ulg0DC7niPE9JrUpmafjsaw_1Vs_pWQIHPoL3BNFf L5STgNa9i2A==.pdf
2	R 1000	R 12000	Pending Verification	Pendin	• PROG6212 Part 2 Checklist 2 (1) (1).pdf

2. Document Display Issue

- Before applying adjustments

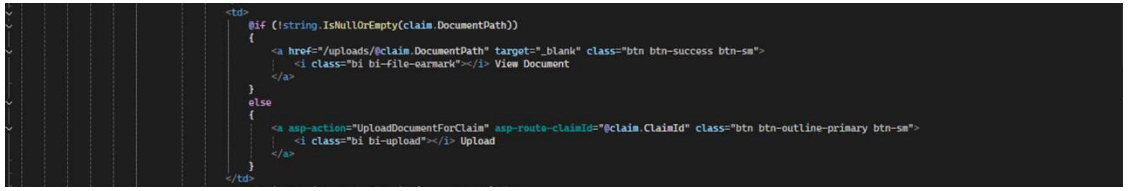
Feedback-Driven Improvements



2. Document Display Issue

- After applying adjustments
- Implemented a View Document Button to view the supporting document submitted in the Claim submission and and viewed in a new tab

Feedback-Driven Improvements



```
<td>
    @if (!string.IsNullOrEmpty(claim.DocumentPath))
    {
        <a href="/uploads/@claim.DocumentPath" target="_blank" class="btn btn-success btn-sm">
            <i class="bi bi-file-earmark"></i> View Document
        </a>
    }
    else
    {
        <a asp-action="UploadDocumentForClaim" asp-route-claimId="@claim.ClaimId" class="btn btn-outline-primary btn-sm">
            <i class="bi bi-upload"></i> Upload
        </a>
    }
</td>
```

2. Document Display Issue

- After applying adjustments
- Technical Implementation

HR Super User Implementation

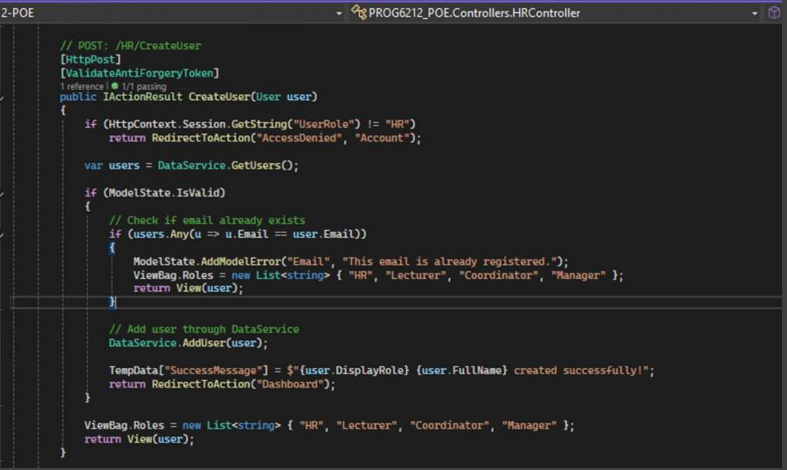
HR Super User Implementation

HR Super User Role

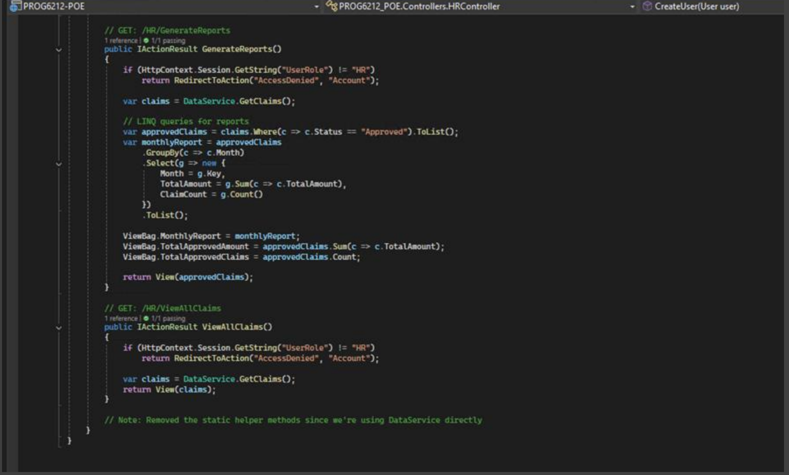
- Centralized user profile management
- No public registration system
- Complete CRUD operations for users
- Automated report generation

Benefit:

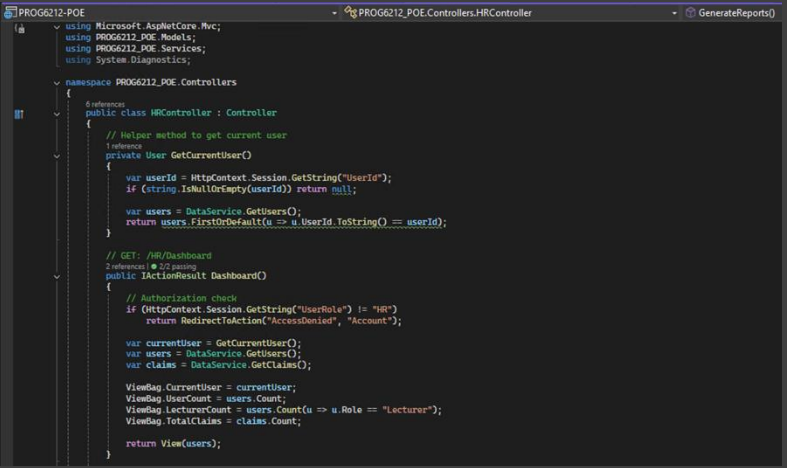
- Controlled, secure user management system



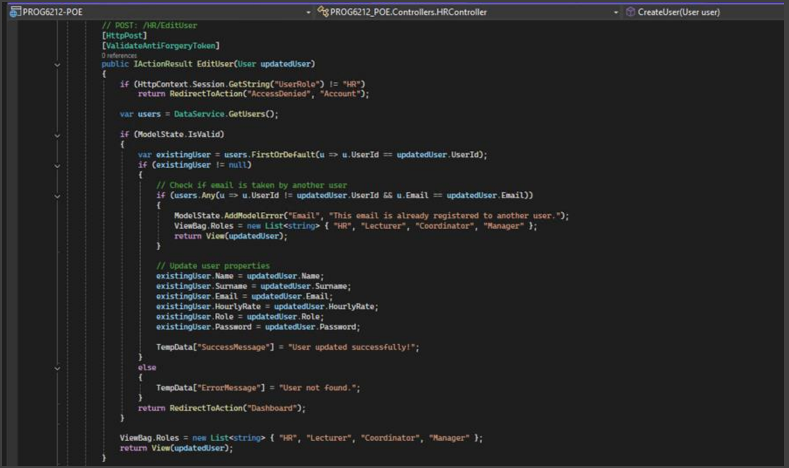
```
// POST: /HR/CreateUser
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult CreateUser(User user)
{
    if (HttpContext.Session.GetString("UserRole") != "HR")
        return RedirectToAction("AccessDenied", "Account");

    var users = DataService.GetUsers();

    if (ModelState.IsValid)
    {
        // Check if email already exists
        if (users.Any(u => u.Email == user.Email))
        {
            ModelState.AddModelError("Email", "This email is already registered.");
            ViewBag.Roles = new List<string> { "HR", "Lecturer", "Coordinator", "Manager" };
            return View(user);
        }

        // Add user through DataService
        DataService.AddUser(user);

        TempData["SuccessMessage"] = $"{user.DisplayRole} {user.FullName} created successfully!";
        return RedirectToAction("Dashboard");
    }

    ViewBag.Roles = new List<string> { "HR", "Lecturer", "Coordinator", "Manager" };
    return View(user);
}
```



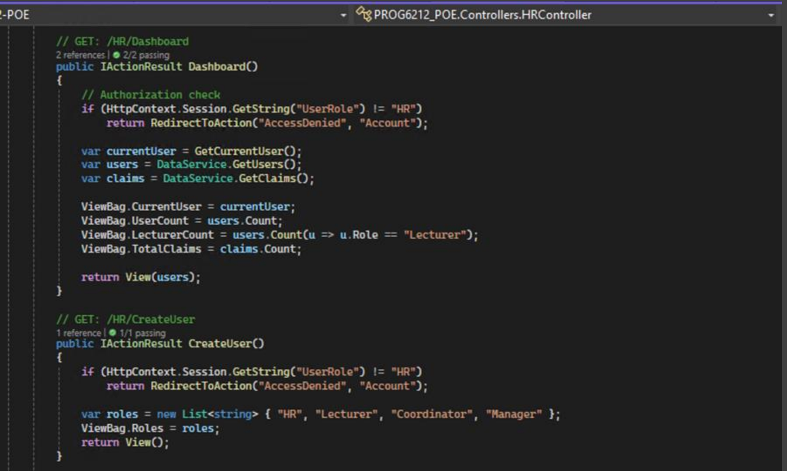
```
        // GET: /HR/GenerateReports
        public IActionResult GenerateReports()
        {
            if (HttpContext.Session.GetString("UserRole") != "HR")
                return RedirectToAction("AccessDenied", "Account");

            var claims = DataService.GetClaims();

            // LINQ queries for reports
            var approvedClaims = claims.Where(c => c.Status == "Approved").ToList();
            var monthlyReport = approvedClaims
                .GroupBy(c => c.Month)
                .Select(g => new {
                    Month = g.Key,
                    TotalAmount = g.Sum(c => c.TotalAmount),
                    ClaimCount = g.Count()
                })
                .ToList();

            ViewBag.MonthlyReport = monthlyReport;
            ViewBag.TotalApprovedAmount = approvedClaims.Sum(c => c.TotalAmount);
            ViewBag.TotalApprovedClaims = approvedClaims.Count;

            return View(approvedClaims);
        }

        // GET: /HR/ViewAllClaims
        public IActionResult ViewAllClaims()
        {
            if (HttpContext.Session.GetString("UserRole") != "HR")
                return RedirectToAction("AccessDenied", "Account");

            var claims = DataService.GetClaims();
            return View(claims);
        }

        // Note: Removed the static helper methods since we're using DataService directly
    }
}
```



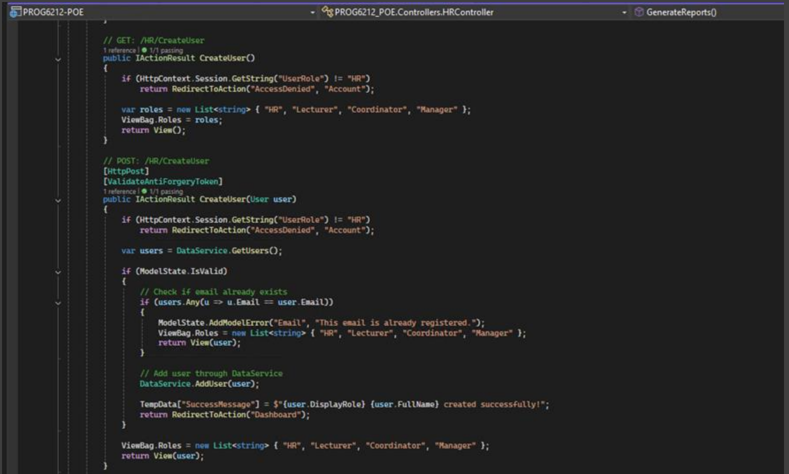
```
using Microsoft.AspNetCore.Mvc;
using PROG6212_POE.Models;
using PROG6212_POE.Services;
using System.Diagnostics;

namespace PROG6212_POE.Controllers
{
    public class HRController : Controller
    {
        // Helper method to get current user
        private User GetCurrentUser()
        {
            var userId = HttpContext.Session.GetString("UserId");
            if (string.IsNullOrEmpty(userId)) return null;

            var users = DataService.GetUsers();
            return users.FirstOrDefault(u => u.UserId.ToString() == userId);
        }

        // GET: /HR/Dashboard
        public IActionResult Dashboard()
        {
            // Authorization check
            if (HttpContext.Session.GetString("UserRole") != "HR")
                return RedirectToAction("AccessDenied", "Account");

            var currentUser = GetCurrentUser();
            var users = DataService.GetUsers();
            var claims = DataService.GetClaims();

            ViewBag.CurrentUser = currentUser;
            ViewBag.UserCount = users.Count;
            ViewBag.LecturerCount = users.Count(u => u.Role == "Lecturer");
            ViewBag.TotalClaims = claims.Count;

            return View(users);
        }
```



```
        // POST: /HR/EditUser
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult EditUser(User updatedUser)
        {
            if (HttpContext.Session.GetString("UserRole") != "HR")
                return RedirectToAction("AccessDenied", "Account");

            var users = DataService.GetUsers();

            if (ModelState.IsValid)
            {
                var existingUser = users.FirstOrDefault(u => u.UserId == updatedUser.UserId);
                if (existingUser != null)
                {
                    // Check if email is taken by another user
                    if (users.Any(u => u.UserId != updatedUser.UserId && u.Email == updatedUser.Email))
                    {
                        ModelState.AddModelError("Email", "This email is already registered to another user.");
                        ViewBag.Roles = new List<string> { "HR", "Lecturer", "Coordinator", "Manager" };
                        return View(updatedUser);
                    }

                    // Update user properties
                    existingUser.Name = updatedUser.Name;
                    existingUser.Surname = updatedUser.Surname;
                    existingUser.Email = updatedUser.Email;
                    existingUser.HourlyRate = updatedUser.HourlyRate;
                    existingUser.Role = updatedUser.Role;
                    existingUser.Password = updatedUser.Password;

                    TempData["SuccessMessage"] = "User updated successfully!";
                }
                else
                {
                    TempData["ErrorMessage"] = "User not found.";
                }
                return RedirectToAction("Dashboard");
            }

            ViewBag.Roles = new List<string> { "HR", "Lecturer", "Coordinator", "Manager" };
            return View(updatedUser);
        }
```



```
// GET: /HR/Dashboard
public IActionResult Dashboard()
{
    // Authorization check
    if (HttpContext.Session.GetString("UserRole") != "HR")
        return RedirectToAction("AccessDenied", "Account");

    var currentUser = GetCurrentUser();
    var users = DataService.GetUsers();
    var claims = DataService.GetClaims();

    ViewBag.CurrentUser = currentUser;
    ViewBag.UserCount = users.Count;
    ViewBag.LecturerCount = users.Count(u => u.Role == "Lecturer");
    ViewBag.TotalClaims = claims.Count;

    return View(users);
}

// GET: /HR/CreateUser
public IActionResult CreateUser()
{
    if (HttpContext.Session.GetString("UserRole") != "HR")
        return RedirectToAction("AccessDenied", "Account");

    var roles = new List<string> { "HR", "Lecturer", "Coordinator", "Manager" };
    ViewBag.Roles = roles;
    return View();
}
```



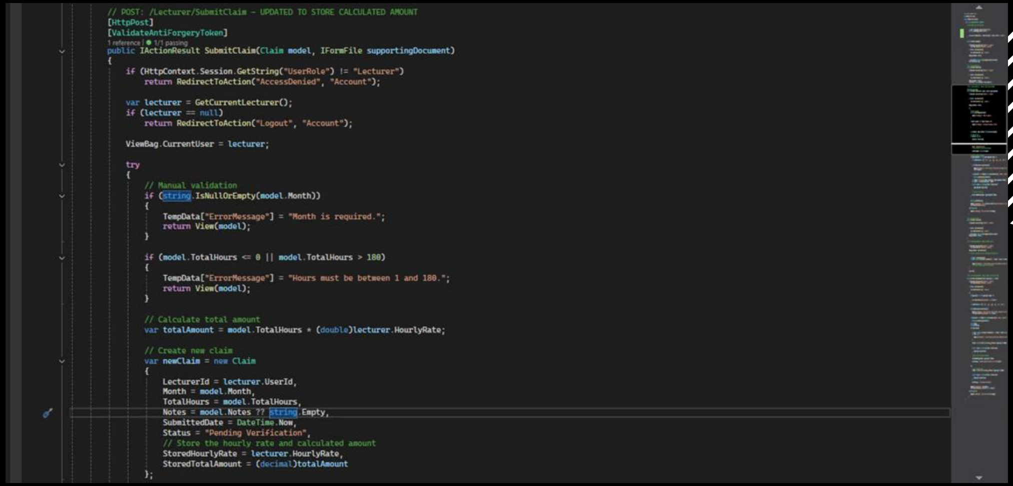
```
        // GET: /HR/CreateUser
        public IActionResult CreateUser()
        {
            if (HttpContext.Session.GetString("UserRole") != "HR")
                return RedirectToAction("AccessDenied", "Account");

            var roles = new List<string> { "HR", "Lecturer", "Coordinator", "Manager" };
            ViewBag.Roles = roles;
            return View();
        }

        // POST: /HR/CreateUser
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult CreateUser(User user)
        {
            if (HttpContext.Session.GetString("UserRole") != "HR")
                return RedirectToAction("AccessDenied", "Account");

            var users = DataService.GetUsers();

            if (ModelState.IsValid)
            {
                // Check if email already exists
                if (users.Any(u => u.Email == user.Email))
                {
                    ModelState.AddModelError("Email", "This email is already registered.");
                    ViewBag.Roles = new List<string> { "HR", "Lecturer", "Coordinator", "Manager" };
                    return View(user);
                }

                // Add user through DataService
                DataService.AddUser(user);

                TempData["SuccessMessage"] = $"{user.DisplayRole} {user.FullName} created successfully!";
                return RedirectToAction("Dashboard");
            }

            ViewBag.Roles = new List<string> { "HR", "Lecturer", "Coordinator", "Manager" };
            return View(user);
        }
```

Automated Data Flow & Calculations

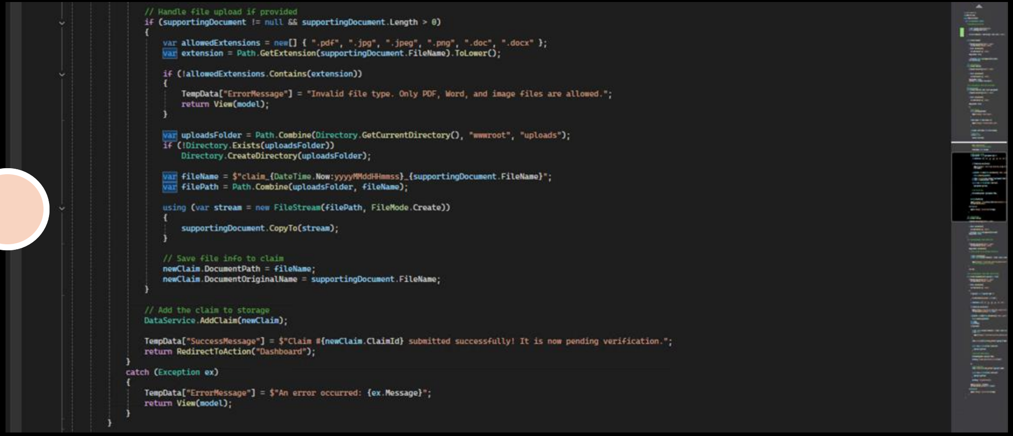
Automated Data Flow & Calculations

Features:

- Hourly rates pulled from HR-managed profiles
- Real-time total amount calculation
- Eliminated manual input errors



```
// POST: /Lecturer/SubmitClaim - UPDATED TO STORE CALCULATED AMOUNT
[HttpPost]
[ValidateAntiForgeryToken]
1 reference | 1/1 passing
public IActionResult SubmitClaim(Claim model, IFormFile supportingDocument)
{
    if (HttpContext.Session.GetString("UserRole") != "Lecturer")
        return RedirectToAction("AccessDenied", "Account");

    var lecturer = GetCurrentLecturer();
    if (lecturer == null)
        return RedirectToAction("Logout", "Account");

    ViewBag.CurrentUser = lecturer;

    try
    {
        // Manual validation
        if (string.IsNullOrEmpty(model.Month))
        {
            TempData["ErrorMessage"] = "Month is required.";
            return View(model);
        }

        if (model.TotalHours <= 0 || model.TotalHours > 180)
        {
            TempData["ErrorMessage"] = "Hours must be between 1 and 180.";
            return View(model);
        }

        // Calculate total amount
        var totalAmount = model.TotalHours * (double)lecturer.HourlyRate;

        // Create new claim
        var newClaim = new Claim
        {
            LecturerId = lecturer.UserId,
            Month = model.Month,
            TotalHours = model.TotalHours,
            Notes = model.Notes ?? string.Empty,
            SubmittedDate = DateTime.Now,
            Status = "Pending Verification",
            // Store the hourly rate and calculated amount
            StoredHourlyRate = lecturer.HourlyRate,
            StoredTotalAmount = (decimal)totalAmount
        };
```



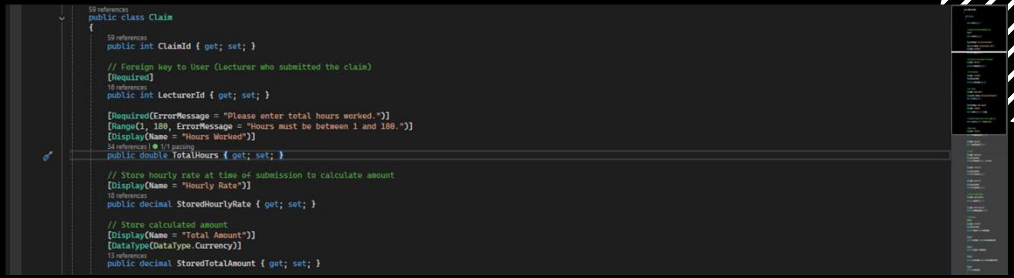
```
        // Handle file upload if provided
        if (supportingDocument != null && supportingDocument.Length > 0)
        {
            var allowedExtensions = new[] { ".pdf", ".jpg", ".jpeg", ".png", ".doc", ".docx" };
            var extension = Path.GetExtension(supportingDocument.FileName).ToLower();

            if (!allowedExtensions.Contains(extension))
            {
                TempData["ErrorMessage"] = "Invalid file type. Only PDF, Word, and image files are allowed.";
                return View(model);
            }

            var uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "uploads");
            if (!Directory.Exists(uploadsFolder))
                Directory.CreateDirectory(uploadsFolder);

            var fileName = $"claim_{DateTime.Now:yyyyMMddHHmmss}_{supportingDocument.FileName}";
            var filePath = Path.Combine(uploadsFolder, fileName);

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

            // Save file info to claim
            newClaim.DocumentPath = fileName;
            newClaim.DocumentOriginalName = supportingDocument.FileName;
        }

        // Add the claim to storage
        DataService.AddClaim(newClaim);

        TempData["SuccessMessage"] = $"Claim #{newClaim.ClaimId} submitted successfully! It is now pending verification.";
        return RedirectToAction("Dashboard");
    }
    catch (Exception ex)
    {
        TempData["ErrorMessage"] = $"An error occurred: {ex.Message}";
        return View(model);
    }
}
```

Enhanced Validation & Business Rules

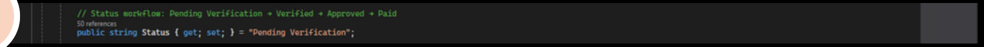
Enhanced Validation & Business Rules

Security Features

- Session Authentication: HttpContext.Session.GetString("UserRole")
- Role-Based Authorization: Each role has specific access
- Access Control: Redirect to AccessDenied for unauthorized users
- Data Isolation: Users only see their own data Maximum 180 hours per month validation



```
59 references
public class Claim
{
    59 references
    public int ClaimId { get; set; }

    // Foreign key to User (Lecturer who submitted the claim)
    [Required]
    16 references
    public int LecturerId { get; set; }

    [Required(ErrorMessage = "Please enter total hours worked.")]
    [Range(1, 180, ErrorMessage = "Hours must be between 1 and 180.")]
    [Display(Name = "Hours Worked")]
    34 references | 1/1 passing
    public double TotalHours { get; set; }

    // Store hourly rate at time of submission to calculate amount
    [Display(Name = "Hourly Rate")]
    18 references
    public decimal StoredHourlyRate { get; set; }

    // Store calculated amount
    [Display(Name = "Total Amount")]
    [DataType(DataType.Currency)]
    13 references
    public decimal StoredTotalAmount { get; set; }
```



```
    // Status workflow: Pending Verification → Verified → Approved → Paid
    50 references
    public string Status { get; set; } = "Pending Verification";
```

Comprehensive Unit Testing

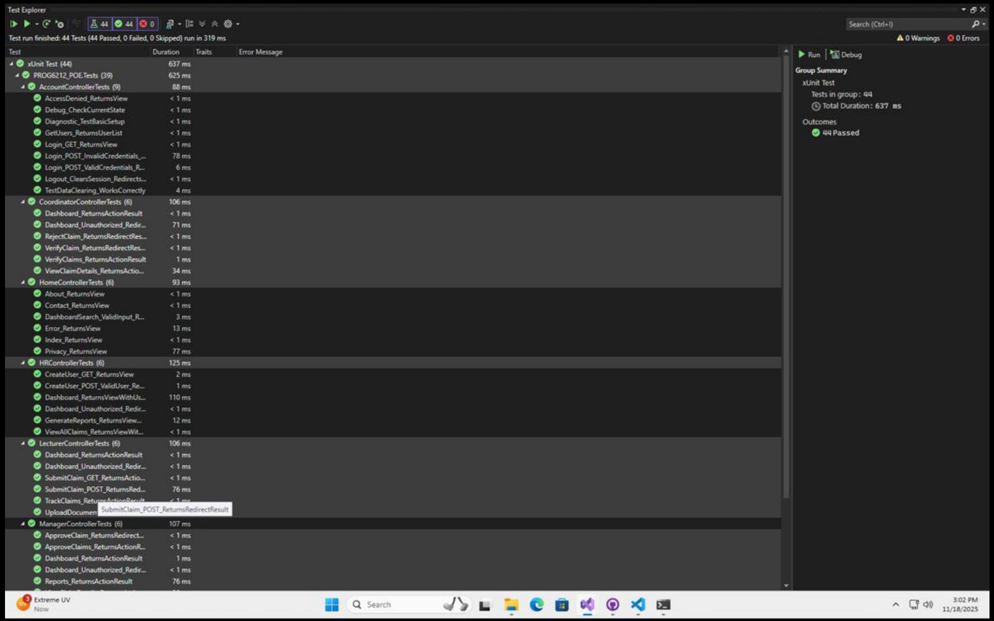
Comprehensive Unit Testing

Test Coverage

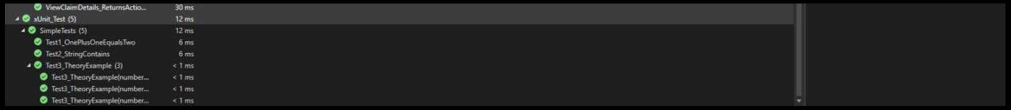
- All Controllers: Account, Lecturer, Coordinator, Manager, HR
- Business Logic: Claim validation and calculations
- Security: Role-based access control
- Session Management: User authentication states
- Data Integrity: Model validation rules

Comprehensive Unit Testing

- Running Test



```
Test Explorer
44  44  0
Test run finished: 44 Tests (44 Passed, 0 Failed, 0 Skipped) run in 319 ms        Search (Ctrl+I)
                                                                                  0 Warnings  0 Errors
Test                                   Duration  Traits  Error Message
xUnit Test (44)                        637 ms
  PROG6212_POE.Tests (39)              625 ms
    AccountControllerTests (9)         88 ms
      AccessDenied_ReturnsView         < 1 ms
      Debug_CheckCurrentState          < 1 ms
      Diagnostic_TestBasicSetup        < 1 ms
      GetUsers_ReturnsUserList         < 1 ms
      Login_GET_ReturnsView            < 1 ms
      Login_POST_InvalidCredentials_...  78 ms
      Login_POST_ValidCredentials_R...    6 ms
      Logout_ClearsSession_Redirects...  < 1 ms
      TestDataClearing_WorksCorrectly     4 ms
    CoordinatorControllerTests (6)     106 ms
      Dashboard_ReturnsActionResult    < 1 ms
      Dashboard_Unauthorized_Redir...   71 ms
      RejectClaim_ReturnsRedirectRes... < 1 ms
      VerifyClaim_ReturnsRedirectRes... < 1 ms
      VerifyClaims_ReturnsActionResult    1 ms
      ViewClaimDetails_ReturnsActio...   34 ms
    HomeControllerTests (6)             93 ms
      About_ReturnsView                < 1 ms
      Contact_ReturnsView              < 1 ms
      DashboardSearch_ValidInput_R...     3 ms
      Error_ReturnsView                  13 ms
      Index_ReturnsView                < 1 ms
      Privacy_ReturnsView                77 ms
    HRControllerTests (6)              125 ms
      CreateUser_GET_ReturnsView          2 ms
      CreateUser_POST_ValidUser_Re...     1 ms
      Dashboard_ReturnsViewWithUs...    110 ms
      Dashboard_Unauthorized_Redir...  < 1 ms
      GenerateReports_ReturnsView...     12 ms
      ViewAllClaims_ReturnsViewWit...  < 1 ms
    LecturerControllerTests (6)        106 ms
      Dashboard_ReturnsActionResult    < 1 ms
      Dashboard_Unauthorized_Redir...  < 1 ms
      SubmitClaim_GET_ReturnsActio...  < 1 ms
      SubmitClaim_POST_ReturnsRed...     76 ms
      TrackClaims_Retur
      UploadDocumen   SubmitClaim_POST_ReturnsRedirectResult
    ManagerControllerTests (6)         107 ms
      ApproveClaim_ReturnsRedirect...  < 1 ms
      ApproveClaims_ReturnsActionR...  < 1 ms
      Dashboard_ReturnsActionResult       1 ms
      Dashboard_Unauthorized_Redir...  < 1 ms
      Reports_ReturnsActionResult        76 ms

Run | Debug
Group Summary
  xUnit Test
    Tests in group : 44
    Total Duration : 637 ms
  Outcomes
    44 Passed

Extreme UV
Now
Search
3:02 PM
11/18/2025
```



```
      ViewClaimDetails_ReturnsActio...   30 ms
xUnit_Test (5)                           12 ms
  SimpleTests (5)                        12 ms
    Test1_OnePlusOneEqualsTwo             6 ms
    Test2_StringContains                  6 ms
    Test3_TheoryExample (3)            < 1 ms
      Test3_TheoryExample(number...    < 1 ms
      Test3_TheoryExample(number...    < 1 ms
      Test3_TheoryExample(number...    < 1 ms
```

Entity Framework Integration

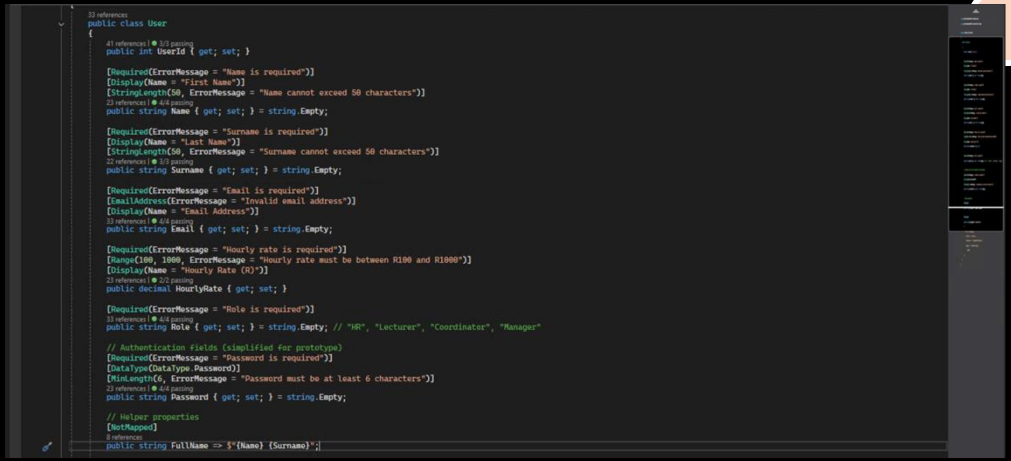
Entity Framework Integration

Data Flow

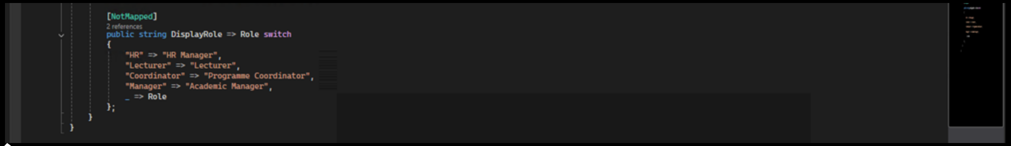
- HR creates users with hourly rates
- Lecturers submit claims with auto-calculated amounts
- System stores rates and amounts permanently
- Admins process claims through workflow

Entity Framework Integration

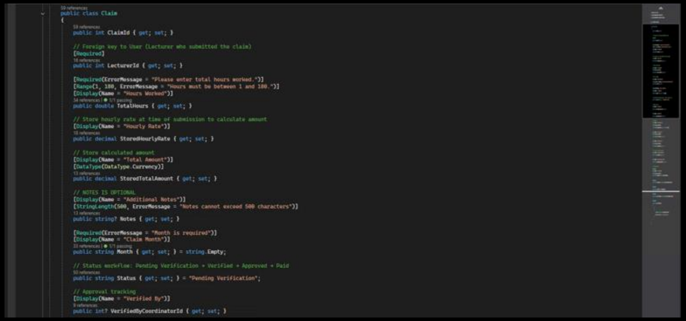
- User Model



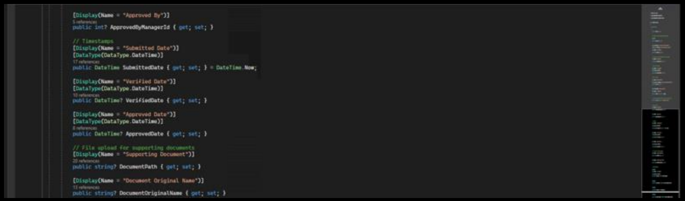
```
33 references
public class User
{
    41 references | 3/3 passing
    public int UserId { get; set; }

    [Required(ErrorMessage = "Name is required")]
    [Display(Name = "First Name")]
    [StringLength(50, ErrorMessage = "Name cannot exceed 50 characters")]
    23 references | 4/4 passing
    public string Name { get; set; } = string.Empty;

    [Required(ErrorMessage = "Surname is required")]
    [Display(Name = "Last Name")]
    [StringLength(50, ErrorMessage = "Surname cannot exceed 50 characters")]
    22 references | 3/3 passing
    public string Surname { get; set; } = string.Empty;

    [Required(ErrorMessage = "Email is required")]
    [EmailAddress(ErrorMessage = "Invalid email address")]
    [Display(Name = "Email Address")]
    33 references | 4/4 passing
    public string Email { get; set; } = string.Empty;

    [Required(ErrorMessage = "Hourly rate is required")]
    [Range(100, 1000, ErrorMessage = "Hourly rate must be between R100 and R1000")]
    [Display(Name = "Hourly Rate (R)")]
    23 references | 2/2 passing
    public decimal HourlyRate { get; set; }

    [Required(ErrorMessage = "Role is required")]
    33 references | 4/4 passing
    public string Role { get; set; } = string.Empty; // "HR", "Lecturer", "Coordinator", "Manager"

    // Authentication fields (simplified for prototype)
    [Required(ErrorMessage = "Password is required")]
    [DataType(DataType.Password)]
    [MinLength(6, ErrorMessage = "Password must be at least 6 characters")]
    23 references | 4/4 passing
    public string Password { get; set; } = string.Empty;

    // Helper properties
    [NotMapped]
    8 references
    public string FullName => $"{Name} {Surname}";
```



```
    [NotMapped]
    2 references
    public string DisplayRole => Role switch
    {
        "HR" => "HR Manager",
        "Lecturer" => "Lecturer",
        "Coordinator" => "Programme Coordinator",
        "Manager" => "Academic Manager",
        _ => Role
    };
}
}
```



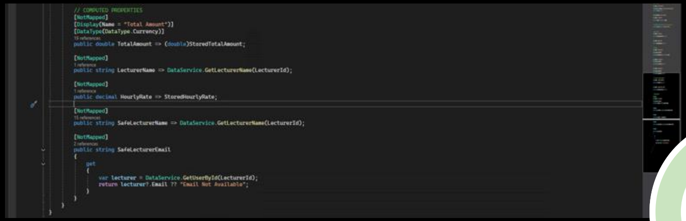
```csharp
59 references
public class Claim
{
    59 references
    public int ClaimId { get; set; }

    // Foreign key to User (Lecturer who submitted the claim)
    [Required]
    18 references
    public int LecturerId { get; set; }

    [Required(ErrorMessage = "Please enter total hours worked.")]
    [Range(1, 180, ErrorMessage = "Hours must be between 1 and 180.")]
    [Display(Name = "Hours Worked")]
    34 references | 1/1 passing
    public double TotalHours { get; set; }

    // Store hourly rate at time of submission to calculate amount
    [Display(Name = "Hourly Rate")]
    18 references
    public decimal StoredHourlyRate { get; set; }

    // Store calculated amount
    [Display(Name = "Total Amount")]
    [DataType(DataType.Currency)]
    13 references
    public decimal StoredTotalAmount { get; set; }

    // NOTES IS OPTIONAL
    [Display(Name = "Additional Notes")]
    [StringLength(500, ErrorMessage = "Notes cannot exceed 500 characters")]
    13 references
    public string? Notes { get; set; }

    [Required(ErrorMessage = "Month is required")]
    [Display(Name = "Claim Month")]
    33 references | 1/1 passing
    public string Month { get; set; } = string.Empty;

    // Status workflow: Pending Verification → Verified → Approved → Paid
    50 references
    public string Status { get; set; } = "Pending Verification";

    // Approval tracking
    [Display(Name = "Verified By")]
    9 references
    public int? VerifiedByCoordinatorId { get; set; }
```



```csharp
    [Display(Name = "Approved By")]
    5 references
    public int? ApprovedByManagerId { get; set; }

    // Timestamps
    [Display(Name = "Submitted Date")]
    [DataType(DataType.DateTime)]
    17 references
    public DateTime SubmittedDate { get; set; } = DateTime.Now;

    [Display(Name = "Verified Date")]
    [DataType(DataType.DateTime)]
    10 references
    public DateTime? VerifiedDate { get; set; }

    [Display(Name = "Approved Date")]
    [DataType(DataType.DateTime)]
    8 references
    public DateTime? ApprovedDate { get; set; }

    // File upload for supporting documents
    [Display(Name = "Supporting Document")]
    20 references
    public string? DocumentPath { get; set; }

    [Display(Name = "Document Original Name")]
    13 references
    public string? DocumentOriginalName { get; set; }
```

Entity Framework Integration

- Claim Model



```csharp
    // COMPUTED PROPERTIES
    [NotMapped]
    [Display(Name = "Total Amount")]
    [DataType(DataType.Currency)]
    19 references
    public double TotalAmount => (double)StoredTotalAmount;

    [NotMapped]
    1 reference
    public string LecturerName => DataService.GetLecturerName(LecturerId);

    [NotMapped]
    1 reference
    public decimal HourlyRate => StoredHourlyRate;

    [NotMapped]
    15 references
    public string SafeLecturerName => DataService.GetLecturerName(LecturerId);

    [NotMapped]
    2 references
    public string SafeLecturerEmail
    {
        get
        {
            var lecturer = DataService.GetUserById(LecturerId);
            return lecturer?.Email ?? "Email Not Available";
        }
    }
}
}
```

Data Validation & Business Rules

Data Validation & Business Rules

Business Logic Enforcement:

- Hourly Rate Control: HR sets rates (R100-R1000 range)
- Work Hour Limits: Maximum 180 hours per month
- Role Management: Strict role-based system
- Data Integrity: Required field validation
- Audit Trail: Permanent storage of calculated values

Data Validation & Business Rules

1. Required Field Validation

- [Required(ErrorMessage = "Name is required")]
  [Required(ErrorMessage = "Email is required")]
  [Required(ErrorMessage = "Role is required")]

2. Business Rule Enforcement

- // HOURLY RATE BUSINESS RULE
  [Range(100, 1000, ErrorMessage = "Hourly rate must be between R100 and R1000")] public decimal HourlyRate { get; set; }
  // ROLE MANAGEMENT BUSINESS RULE public string Role { get; set; } = string.Empty; // Strict role system: "HR", "Lecturer", "Coordinator", "Manager"

Data Validation & Business Rules

3. Data Format Validation

- [EmailAddress(ErrorMessage = "Invalid email address")] [StringLength(50, ErrorMessage = "Name cannot exceed 50 characters")] [DataType(DataType.Password)] [MinLength(6, ErrorMessage = "Password must be at least 6 characters")]

4. User Experience Enhancements

- [Display(Name = "First Name")] [Display(Name = "Hourly Rate (R)")] [Display(Name = "Email Address")]

Automated Calculation System

Automated Calculation System

Key Features:

- HR-Managed Rates from User profiles
- Real-time calculation eliminates manual errors
- Permanent audit trail of calculated values
- Data integrity through automated workflow

Automated Calculation System

Implementation:

```
// AUTO-CALCULATION FROM HR-MANAGED DATA [HttpPost] public IActionResult SubmitClaim(Claim
model)
{
        var lecturer = GetCurrentLecturer();
        // REAL-TIME CALCULATION
        var totalAmount = model.TotalHours * (double)lecturer.HourlyRate;
         // STORE FOR AUDIT TRAIL
        var newClaim = new Claim
        {
                StoredHourlyRate = lecturer.HourlyRate,
                StoredTotalAmount = (decimal)totalAmount,
                Status = "Pending Verification"
        };

}
```

Technical Architecture

Technical Architecture

Technology Stack:

- ASP.NET Core MVC Framework
- Entity Framework Core (Code-First)
- xUnit Testing Framework
- Session-Based Authentication
- Bootstrap UI Framework

Architecture Pattern:

- Model-View-Controller (MVC)
- Repository Pattern with DataService
- Layered Security Architecture
- Role-Based Access Control

Scalability

- Enterprise-ready solution

Version Control

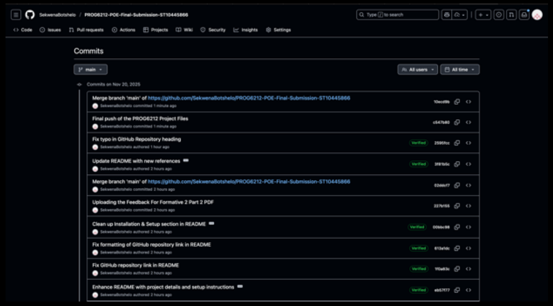
Version Control

GitHub Development Practices

- Frequent, descriptive commits
- Feedback-driven iterations
- Professional workflow management
- Regular pushes to repository

Version Control

- GitHub Commits:



SekwenaBotshelo / PROG6212-POE-Final-Submission-ST10445866

Commits

main | All users | All time

Commits on Nov 20, 2025

Merge branch 'main' of https://github.com/SekwenaBotshelo/PROG6212-POE-Final-Submission-ST10445866
SekwenaBotshelo committed 1 minute ago — 10ecd9b

Final push of the PROG6212 Project Files
SekwenaBotshelo committed 1 minute ago — c547b80

Fix typo in GitHub Repository heading
SekwenaBotshelo authored 1 hour ago — Verified — 2595fcc

Update README with new references
SekwenaBotshelo authored 2 hours ago — Verified — 3f81b5c

Merge branch 'main' of https://github.com/SekwenaBotshelo/PROG6212-POE-Final-Submission-ST10445866
SekwenaBotshelo committed 2 hours ago — 02ddcf7

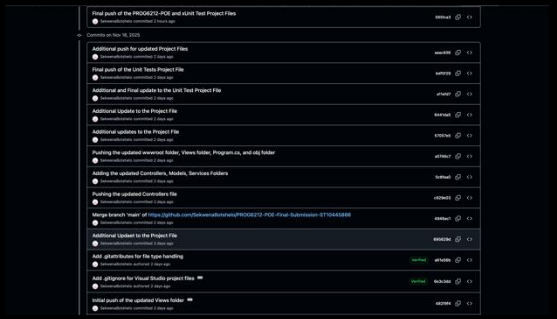
Uploading the Feedback For Formative 2 Part 2 PDF
SekwenaBotshelo committed 2 hours ago — 227b155

Clean up Installation & Setup section in README
SekwenaBotshelo authored 2 hours ago — Verified — 00bbc98

Fix formatting of GitHub repository link in README
SekwenaBotshelo authored 2 hours ago — Verified — 613a1dc

Fix GitHub repository link in README
SekwenaBotshelo authored 2 hours ago — Verified — 110a83c

Enhance README with project details and setup instructions
SekwenaBotshelo authored 2 hours ago — Verified — eb57f77



Final push of the PROG6212-POE and xUnit Test Project Files
SekwenaBotshelo committed 2 hours ago — 565fca3

Commits on Nov 18, 2025

Additional push for updated Project Files
SekwenaBotshelo committed 2 days ago — aaac936

Final push of the Unit Tests Project File
SekwenaBotshelo committed 2 days ago — bdf0f29

Additional and Final update to the Unit Test Project File
SekwenaBotshelo committed 2 days ago — a11efd7

Additional Update to the Project File
SekwenaBotshelo committed 2 days ago — 6441da5

Additional updates to the Project File
SekwenaBotshelo committed 2 days ago — 57057e5

Pushing the updated wwwroot folder, Views folder, Program.cs, and obj folder
SekwenaBotshelo committed 2 days ago — a5786c7

Adding the updated Controllers, Models, Services Folders
SekwenaBotshelo committed 2 days ago — 0c8faa0

Pushing the updated Controllers file
SekwenaBotshelo committed 2 days ago — c929e03

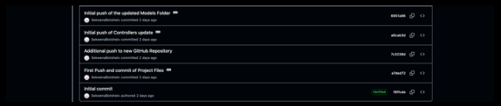
Merge branch 'main' of https://github.com/SekwenaBotshelo/PROG6212-POE-Final-Submission-ST10445866
SekwenaBotshelo committed 2 days ago — 4946ac1

Additional Updaet to the Project File
SekwenaBotshelo committed 2 days ago — 690629d

Add .gitattributes for file type handling
SekwenaBotshelo authored 2 days ago — Verified — a61e58b

Add .gitignore for Visual Studio project files
SekwenaBotshelo authored 2 days ago — Verified — 6e3c3dd

Initial push of the updated Views folder
SekwenaBotshelo committed 2 days ago — 482f9f4



Initial push of the updated Models Folder
SekwenaBotshelo committed 2 days ago — 6951a96

Initial push of Controllers update
SekwenaBotshelo committed 2 days ago — a6cab3d

Additional push to new GitHub Repository
SekwenaBotshelo committed 2 days ago — 7c3238d

First Push and commit of Project Files
SekwenaBotshelo committed 2 days ago — a7ded72

Initial commit
SekwenaBotshelo authored 2 days ago — Verified — f8ffcde

User Workflow Demonstration

User Workflow Demonstration

HR Workflow

- Login → Manage Users → Generate Reports

Lecturer Workflow

- Login → Submit Claim (Auto-calculated) → Track Status

Coordinator Workflow

- Login → Verify Claims → Update Status

Manager Workflow

- Login → Approve Claims → View Reports

Benefits

- Streamlined, role-specific user experiences

Challenges & Solutions

Challenges & Solutions

Challenge 1

- Data consistency across multiple roles

Solution

- Centralized HR user management system

Challenge 2

- Secure session management

Solution

- Custom authorization with role validation

Challenge 3

- Automated calculations

Solution

- Real-time computation with stored HR rates

Challenge 4

- File upload and display

Solution

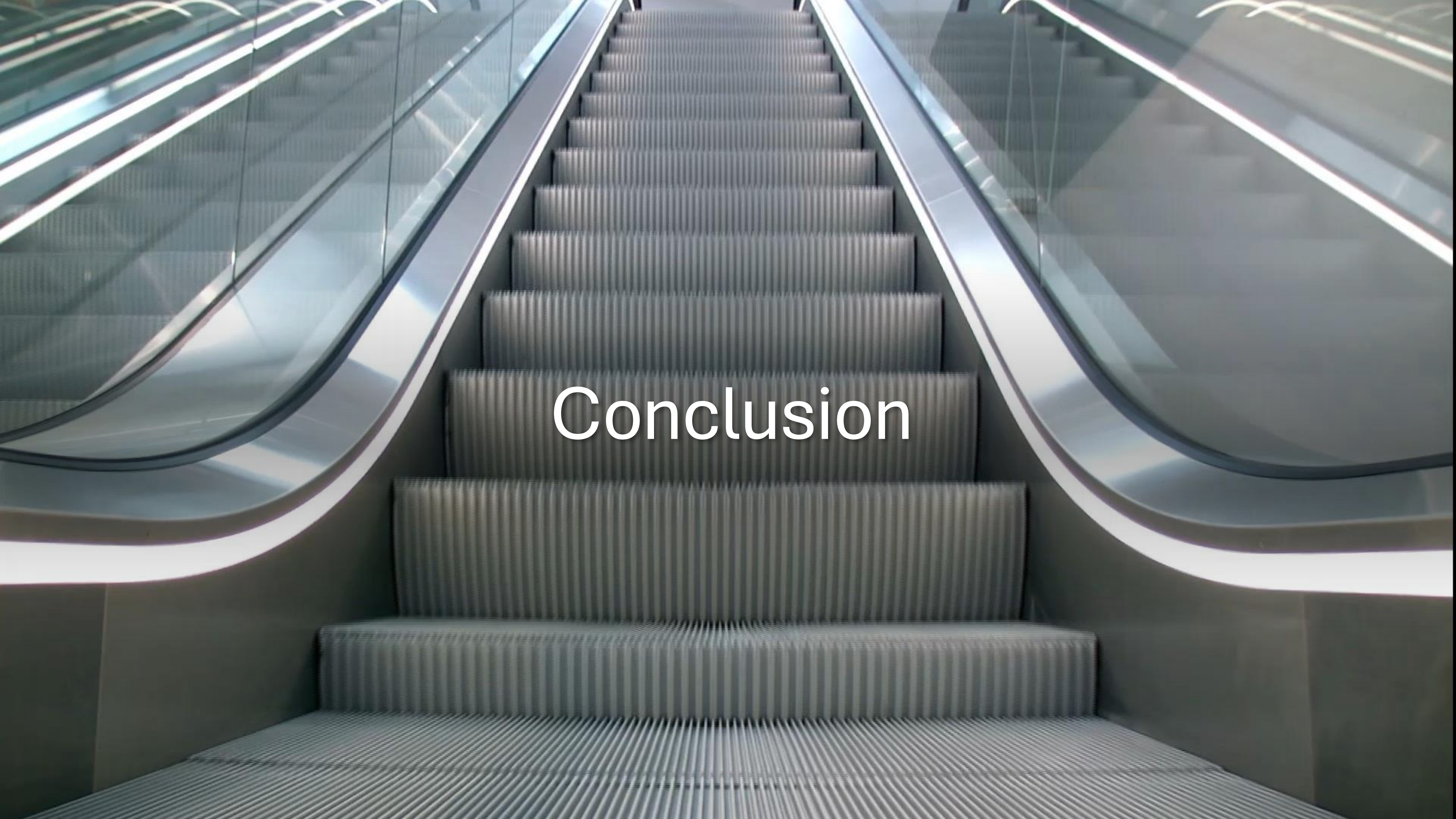
- Button-based document viewing system

Future Enhancement Opportunities

Future Enhancement Opportunities

Potential Extensions

- PDF report generation with LINQ queries
- Email notification system for status updates
- Advanced analytics dashboard with charts
- Mobile-responsive design optimization
- API integration for external systems
- Advanced reporting capabilities



Conclusion

Conclusion

Project Success Summary:

- HR Super User System
  - Centralized user management
- Automated Data Flow
  - Real-time calculations from HR rates
- Enhanced Security
  - Role-based access control & sessions
- Comprehensive Testing
  - 44/44 unit tests passing
- Entity Framework
  - Robust database integration
- Feedback Implementation
  - Document attachment & display fixes
- Business Rules
  - Validation & workflow enforcement
- Professional UI/UX
  - Streamlined user workflows

Q&A

Q&A

Thank You!

Questions?

Contact Information:

- Student Name:
  - Botshelo Koketso Sekwena
- Student Number:
  - ST10445866
- Module Code:
  - PROG6212
  - Programming 2B
- GitHub Repository Link:
  - https://github.com/SekwenaBotshelo/PROG6212-POE-Final-Submission-ST10445866.git