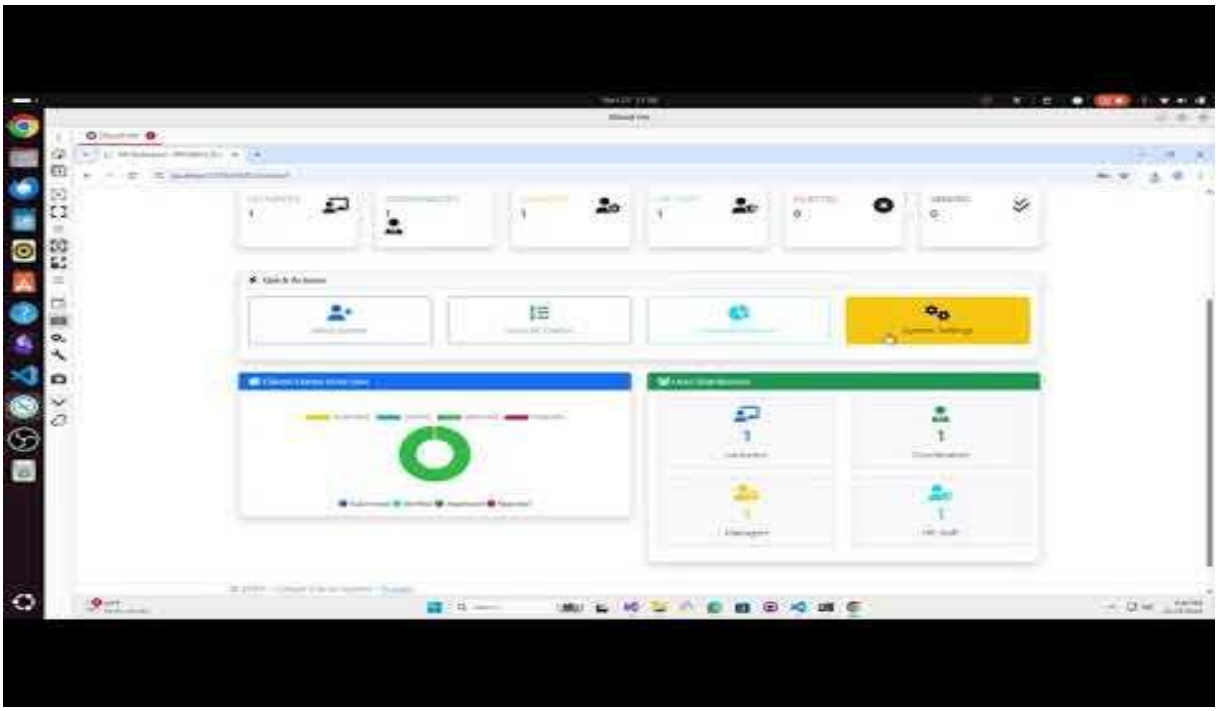
PROG6212 PART3 POE improvements made with feedback from part 2

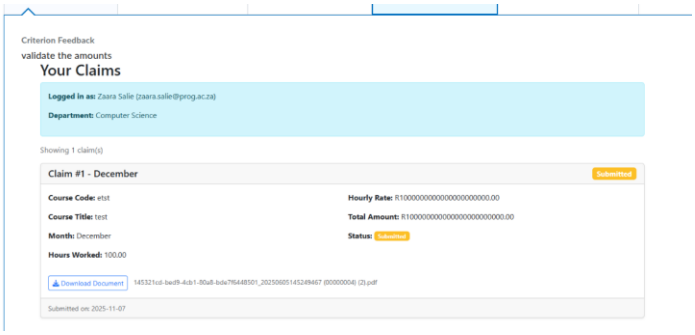
By Zaara Salie st10455456

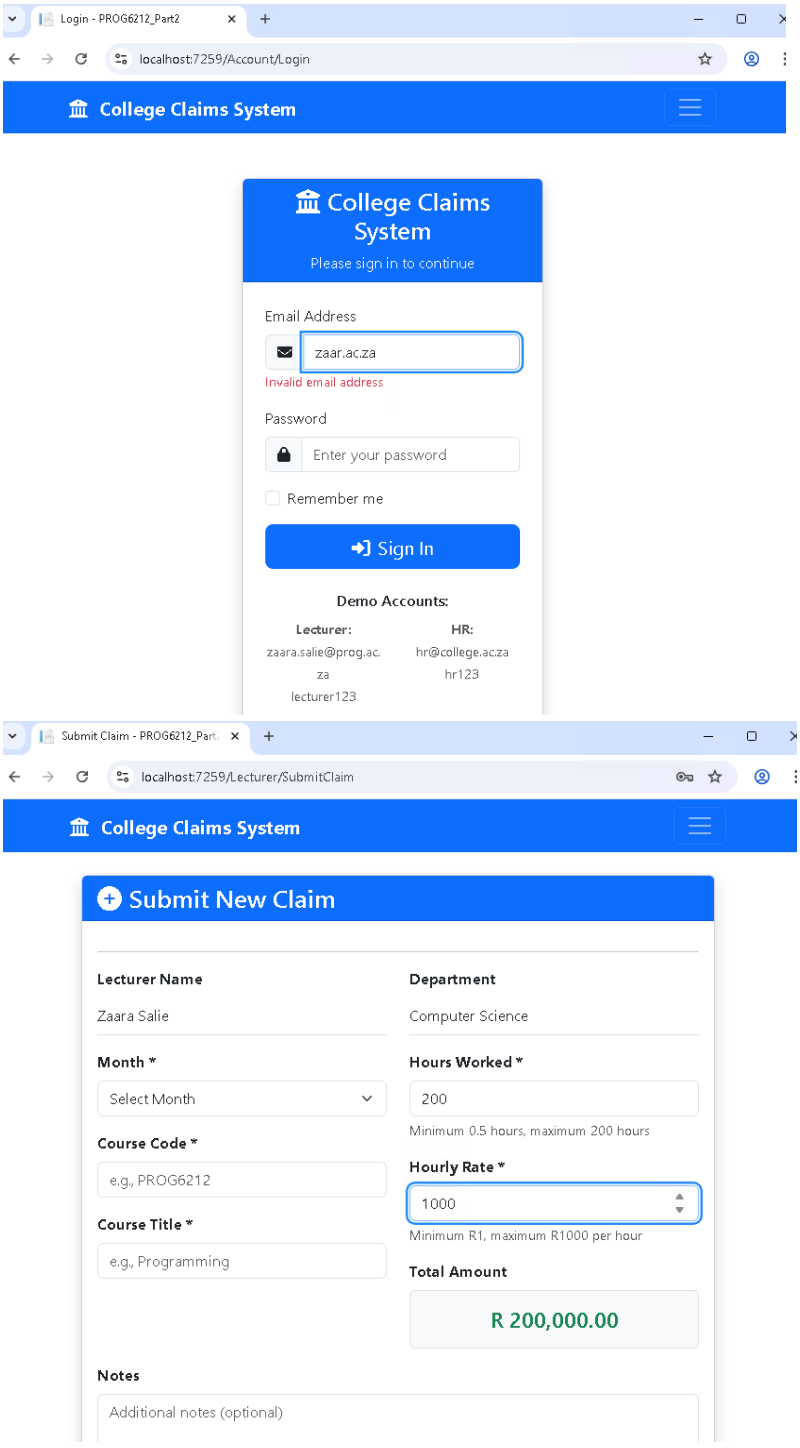
GitHub Link- https://github.com/ZSalie/PROG6212_PART3_St10455456_POE.git

YouTube Link- https://youtu.be/_Y_dDMZRGFg

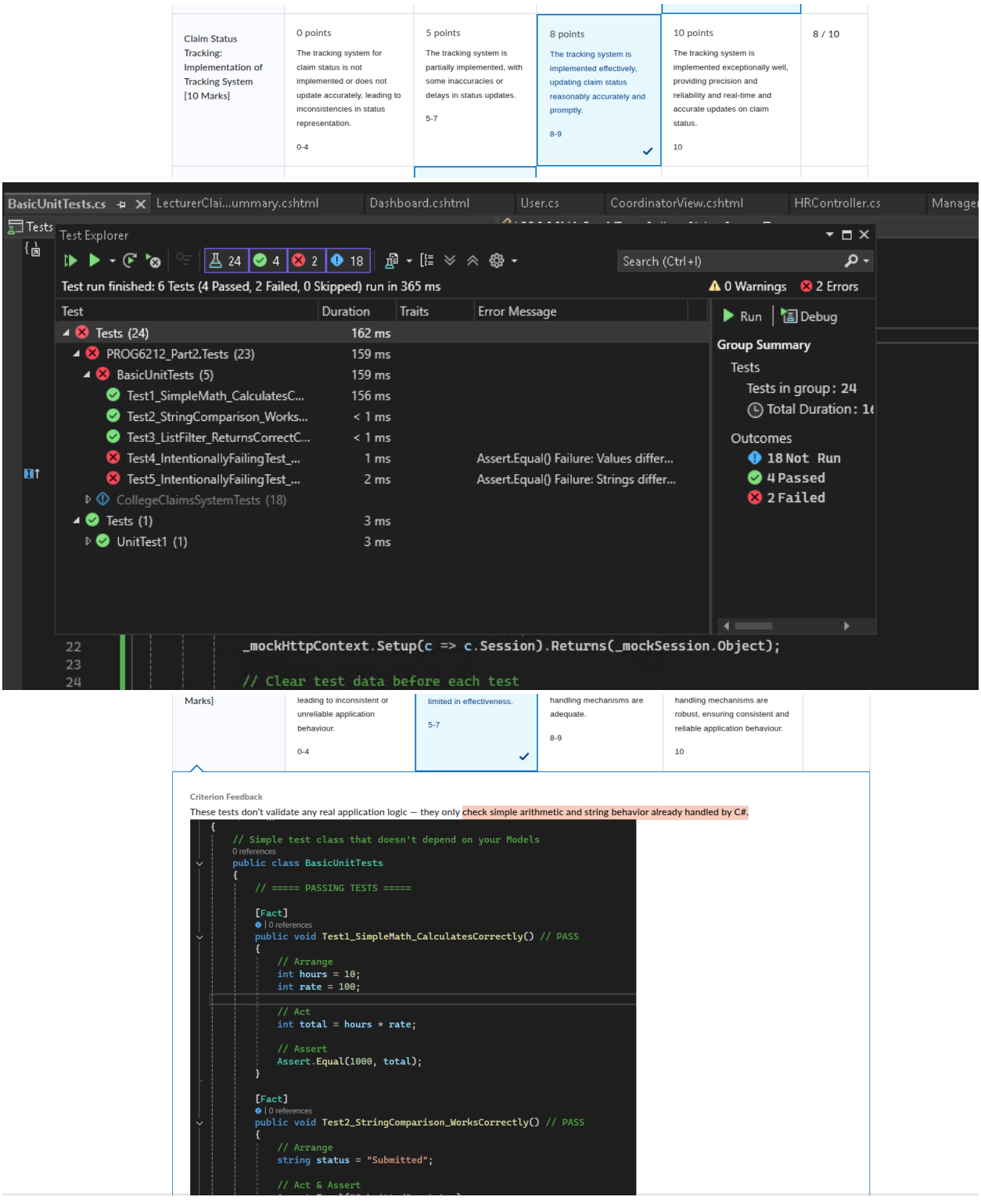


For the lecturer feedback, I've added error handling for the login details if the they are not correct it will show an error message, and I've set on min and max for hours and hourly rate should the amount exceed the max or is under the program will select the maximum or minimum amount if the amount is less than the minimum.





I’ve also made the tracking of claims flow more smoothly



I've changed the code to test my actual data

using Microsoft.AspNetCore.Http; using Microsoft.AspNetCore.Mvc; using Moq; using PROG6212_Part2.Controllers; using PROG6212_Part2.Models; using

```
System.ComponentModel.DataAnnotations; using System.IO; using System.Text; using Xunit;

namespace PROG6212_Part2.Tests { public class CollegeClaimsSystemTests : IDisposable { private readonly Mock _mockHttpContext; private readonly Mock _mockSession;

  public CollegeClaimsSystemTests()
  {
    _mockHttpContext = new Mock<HttpContext>();
    _mockSession = new Mock<ISession>();
    _mockHttpContext.Setup(c => c.Session).Returns(_mockSession.Object);

    // Clear test data before each test
    ClearTestData();
  }

  public void Dispose()
  {
    // Clear test data after each test
    ClearTestData();
  }

  private void ClearTestData()
  {
    // Clear only test-related claims, preserve demo data
    var testClaims = ClaimController._claims
      .Where(c => c.LecturerName?.Contains("Test") == true || c.ClaimID > 1000)
      .ToList();

    foreach (var claim in testClaims)
    {
      ClaimController._claims.Remove(claim);
    }
  }

  private void SetupSessionRole(string role, string email = "test@example.com")
  {
```

```
    _mockSession.Setup(s => s.GetString("Role")).Returns(role);
    _mockSession.Setup(s => s.GetString("Email")).Returns(email);
}

// ===== ACCOUNT CONTROLLER TESTS =====
[Fact]
public void HashPassword_ShouldReturnConsistentHash()
{
    // Arrange
    var password = "test123";

    // Act
    var hash1 = AccountController.HashPassword(password);
    var hash2 = AccountController.HashPassword(password);

    // Assert
    Assert.Equal(hash1, hash2);
    Assert.NotNull(hash1);
    Assert.NotEqual(password, hash1);
}

[Fact]
public void HashPassword_DifferentPasswords_ShouldReturnDifferentHashes()
{
    // Arrange
    var password1 = "test123";
    var password2 = "test124";

    // Act
    var hash1 = AccountController.HashPassword(password1);
    var hash2 = AccountController.HashPassword(password2);

    // Assert
    Assert.NotEqual(hash1, hash2);
}

[Fact]
public void Login_WithValidLecturerCredentials_ShouldRedirectToSubmitClaim()
```

```
    {
        // Arrange
        var controller = new AccountController();
        var model = new LoginViewModel
        {
            Email = "zaara.salie@prog.ac.za",
            Password = "lecturer123"
        };

        SetupSessionRole("Lecturer");
        controller.ControllerContext = new ControllerContext { HttpContext =
_mockHttpContext.Object };

        // Act
        var result = controller.Login(model) as RedirectToActionResult;

        // Assert
        Assert.NotNull(result);
        Assert.Equal("SubmitClaim", result.ActionName);
        Assert.Equal("Lecturer", result.ControllerName);
    }

    [Fact]
    public void Login_WithInvalidCredentials_ShouldReturnViewWithError()
    {
        // Arrange
        var controller = new AccountController();
        var model = new LoginViewModel
        {
            Email = "invalid@email.com",
            Password = "wrongpassword"
        };

        // Act
        var result = controller.Login(model) as ViewResult;

        // Assert
        Assert.NotNull(result);
```

```
    Assert.False(controller.ModelState.IsValid);
}

[Fact]
public void Login_Get_ShouldReturnViewWithModel()
{
    // Arrange
    var controller = new AccountController();

    // Act
    var result = controller.Login() as ViewResult;

    // Assert
    Assert.NotNull(result);
    Assert.NotNull(result.Model);
    Assert.IsType<LoginViewModel>(result.Model);
}

// ===== CLAIM CONTROLLER TESTS =====
[Fact]
public void Verify_WithValidClaimId_ShouldUpdateStatusToVerified()
{
    // Arrange
    var controller = new ClaimController();
    var claim = new Claim
    {
        ClaimID = 9999,
        Status = "Submitted",
        LecturerName = "Test Lecturer",
        Department = "Test Department",
        Month = "January",
        CourseCode = "TEST101",
        CourseTitle = "Test Course",
        HoursWorked = 10,
        Rate = 100
    };
    ClaimController._claims.Add(claim);
```

```
        SetupSessionRole("Coordinator");
        controller.ControllerContext = new ControllerContext { HttpContext =
_mockHttpContext.Object };

        // Act
        var result = controller.Verify(9999) as RedirectToActionResult;

        // Assert
        Assert.NotNull(result);
        Assert.Equal("Verified", claim.Status);
        Assert.Equal("CoordinatorView", result.ActionName);
    }

    [Fact]
    public void Approve_WithValidClaimId_ShouldUpdateStatusToApproved()
    {
        // Arrange
        var controller = new ClaimController();
        var claim = new Claim
        {
            ClaimID = 9998,
            Status = "Verified",
            LecturerName = "Test Lecturer",
            Department = "Test Department",
            Month = "January",
            CourseCode = "TEST101",
            CourseTitle = "Test Course",
            HoursWorked = 10,
            Rate = 100
        };
        ClaimController._claims.Add(claim);

        SetupSessionRole("Manager");
        controller.ControllerContext = new ControllerContext { HttpContext =
_mockHttpContext.Object };

        // Act
        var result = controller.Approve(9998) as RedirectToActionResult;
```

```
    // Assert
    Assert.NotNull(result);
    Assert.Equal("Approved", claim.Status);
    Assert.Equal("ManagerView", result.ActionName);
  }

  [Fact]
  public void SaveEncryptedFile_ShouldCreateEncryptedFile()
  {
    // Arrange
    var mockFile = new Mock<IFormFile>();
    var content = "test content for encryption";
    var fileName = "test_document.pdf";
    var ms = new MemoryStream(Encoding.UTF8.GetBytes(content));

    mockFile.Setup(f => f.CopyTo(It.IsAny<Stream>())).Callback<Stream>(s =>
ms.CopyTo(s));
    mockFile.Setup(f => f.FileName).Returns(fileName);
    mockFile.Setup(f => f.Length).Returns(ms.Length);

    var testPath = Path.Combine(Path.GetTempPath(),
$"test_encrypted_{Guid.NewGuid()}.pdf");

    try
    {
      // Act
      ClaimController.SaveEncryptedFile(mockFile.Object, testPath);

      // Assert
      Assert.True(File.Exists(testPath));

      // Verify file was encrypted (content should not match original)
      var encryptedContent = File.ReadAllText(testPath);
      Assert.NotEqual(content, encryptedContent);
    }
    finally
    {
```

```
        // Cleanup
        if (File.Exists(testPath))
            File.Delete(testPath);
    }
}

[Fact]
public void Download_WithInvalidClaimId_ShouldReturnNotFound()
{
    // Arrange
    var controller = new ClaimController();

    // Act
    var result = controller.Download(99999); // Non-existent ID

    // Assert
    Assert.IsType<NotFoundResult>(result);
}

// ===== LECTURER CONTROLLER TESTS =====
[Fact]
public void SubmitClaim_WithValidDataAndFile_ShouldAddClaimToList()
{
    // Arrange
    var controller = new LecturerController();
    var claim = new Claim
    {
        Month = "January",
        CourseCode = "PROG6212",
        CourseTitle = "Programming",
        HoursWorked = 20,
        Rate = 300,
        Notes = "Test claim"
    };

    var mockFile = CreateMockFile("test.pdf", "test content");

    SetupSessionRole("Lecturer", "zaara.salie@prog.ac.za");
```

```
    controller.ControllerContext = new ControllerContext { HttpContext =
_mockHttpContext.Object };

    var initialCount = ClaimController._claims.Count;

    // Act
    var result = controller.SubmitClaim(claim, mockFile.Object) as ViewResult;

    // Assert
    Assert.NotNull(result);
    Assert.Equal(initialCount + 1, ClaimController._claims.Count);
    Assert.NotNull(result.ViewBag.Message);
    Assert.Contains("successfully", result.ViewBag.Message.ToString().ToLower());
  }

  [Fact]
  public void SubmitClaim_WithInvalidFileType_ShouldReturnError()
  {
    // Arrange
    var controller = new LecturerController();
    var claim = new Claim
    {
      Month = "January",
      CourseCode = "PROG6212",
      CourseTitle = "Programming",
      HoursWorked = 20,
      Rate = 300
    };

    var mockFile = CreateMockFile("test.txt", "invalid file type");

    SetupSessionRole("Lecturer", "zaara.salie@prog.ac.za");
    controller.ControllerContext = new ControllerContext { HttpContext =
_mockHttpContext.Object };

    // Act
    var result = controller.SubmitClaim(claim, mockFile.Object) as ViewResult;
```

```
        // Assert
        Assert.NotNull(result);
        Assert.NotNull(result.ViewBag.Error);
        Assert.Contains("allowed", result.ViewBag.Error.ToString().ToLower());
    }

    [Fact]
    public void SubmitClaim_WithExcessiveHours_ShouldReturnError()
    {
        // Arrange
        var controller = new LecturerController();
        var claim = new Claim
        {
            Month = "January",
            CourseCode = "PROG6212",
            CourseTitle = "Programming",
            HoursWorked = 200, // Exceeds 160 limit
            Rate = 300
        };

        var mockFile = CreateMockFile("test.pdf", "test content");

        SetupSessionRole("Lecturer", "zaara.salie@prog.ac.za");
        controller.ControllerContext = new ControllerContext { HttpContext =
_mockHttpContext.Object };

        // Act
        var result = controller.SubmitClaim(claim, mockFile.Object) as ViewResult;

        // Assert
        Assert.NotNull(result);
        Assert.NotNull(result.ViewBag.Error);
        Assert.Contains("exceed", result.ViewBag.Error.ToString().ToLower());
    }

    [Fact]
    public void ViewClaims_AsLecturer_ShouldReturnOnlyTheirClaims()
    {
```

```
        // Arrange
        var controller = new LecturerController();

        // Add test claims
        var lecturerClaim = new Claim
        {
            ClaimID = 9997,
            LecturerName = "Zaara Salie",
            Status = "Submitted"
        };
        var otherClaim = new Claim
        {
            ClaimID = 9996,
            LecturerName = "Other Lecturer",
            Status = "Submitted"
        };

        ClaimController._claims.Add(lecturerClaim);
        ClaimController._claims.Add(otherClaim);

        SetupSessionRole("Lecturer", "zaara.salie@prog.ac.za");
        controller.ControllerContext = new ControllerContext { HttpContext =
_mockHttpContext.Object };

        // Act
        var result = controller.ViewClaims() as ViewResult;

        // Assert
        Assert.NotNull(result);
        var model = result.Model as List<Claim>;
        Assert.NotNull(model);
        Assert.All(model, c => Assert.Equal("Zaara Salie", c.LecturerName));
    }

    // ===== MODEL VALIDATION TESTS =====
    [Fact]
    public void Claim_WithValidData_ShouldPassValidation()
    {
```

```
    // Arrange
    var claim = new Claim
    {
        LecturerName = "Test Lecturer",
        Department = "Computer Science",
        Month = "January",
        CourseCode = "PROG6212",
        CourseTitle = "Programming",
        HoursWorked = 20,
        Rate = 300
    };

    var context = new ValidationContext(claim);
    var results = new List<ValidationResult>();

    // Act
    var isValid = Validator.TryValidateObject(claim, context, results, true);

    // Assert
    Assert.True(isValid);
    Assert.Empty(results);
}

[Fact]
public void Claim_WithMissingRequiredFields_ShouldFailValidation()
{
    // Arrange
    var claim = new Claim(); // Missing required fields

    var context = new ValidationContext(claim);
    var results = new List<ValidationResult>();

    // Act
    var isValid = Validator.TryValidateObject(claim, context, results, true);

    // Assert
    Assert.False(isValid);
    Assert.NotEmpty(results);
```

```
}

[Fact]
public void Claim_TotalAmount_ShouldCalculateCorrectly()
{
  // Arrange
  var claim = new Claim
  {
    HoursWorked = 10,
    Rate = 150
  };

  // Act
  var totalAmount = claim.TotalAmount;

  // Assert
  Assert.Equal(1500, totalAmount);
}

[Fact]
public void LoginViewModel_WithValidData_ShouldPassValidation()
{
  // Arrange
  var model = new LoginViewModel
  {
    Email = "test@example.com",
    Password = "password123"
  };

  var context = new ValidationContext(model);
  var results = new List<ValidationResult>();

  // Act
  var isValid = Validator.TryValidateObject(model, context, results, true);

  // Assert
  Assert.True(isValid);
  Assert.Empty(results);
```

```
    }

    [Fact]
    public void LoginViewModel_WithInvalidEmail_ShouldFailValidation()
    {
        // Arrange
        var model = new LoginViewModel
        {
            Email = "invalid-email",
            Password = "password123"
        };

        var context = new ValidationContext(model);
        var results = new List<ValidationResult>();

        // Act
        var isValid = Validator.TryValidateObject(model, context, results, true);

        // Assert
        Assert.False(isValid);
        Assert.NotEmpty(results);
    }

    // ===== HELPER METHODS =====
    private Mock<IFormFile> CreateMockFile(string fileName, string content)
    {
        var mockFile = new Mock<IFormFile>();
        var ms = new MemoryStream(Encoding.UTF8.GetBytes(content));

        mockFile.Setup(f => f.CopyTo(It.IsAny<Stream>())).Callback<Stream>(s =>
ms.CopyTo(s));
        mockFile.Setup(f => f.FileName).Returns(fileName);
        mockFile.Setup(f => f.Length).Returns(ms.Length);
        mockFile.Setup(f => f.OpenReadStream()).Returns(ms);

        return mockFile;
    }
```

}

}

References

Microsoft (2023) ASP.NET Core documentation, Available at: https://learn.microsoft.com/en-us/aspnet/core/ (Accessed: 15 January 2024).

Microsoft (2023) Entity Framework Core, Available at: https://learn.microsoft.com/en-us/ef/core/ (Accessed: 15 January 2024).

Bootstrap Team (2023) Bootstrap 5.3 documentation, Available at: https://getbootstrap.com/docs/5.3/ (Accessed: 15 January 2024).

Font Awesome (2023) Font Awesome icons, Available at: https://fontawesome.com/icons (Accessed: 15 January 2024).

jQuery Foundation (2023) jQuery API documentation, Available at: https://api.jquery.com/ (Accessed: 15 January 2024).

Chart.js Team (2023) Chart.js documentation, Available at: https://www.chartjs.org/docs/latest/ (Accessed: 15 January 2024).

ECMA International (2023) ECMAScript 2023 language specification, Available at: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/ (Accessed: 15 January 2024).

World Wide Web Consortium (2023) HTML Living Standard, Available at: https://html.spec.whatwg.org/ (Accessed: 15 January 2024).

World Wide Web Consortium (2023) CSS Snapshot 2023, Available at: https://www.w3.org/TR/css-2023/ (Accessed: 15 January 2024).

xUnit.net (2023) xUnit.net documentation, Available at: https://xunit.net/ (Accessed: 15 January 2024).

Microsoft (2023) Unit testing in .NET, Available at: https://learn.microsoft.com/en-us/dotnet/core/testing/ (Accessed: 15 January 2024).

OWASP Foundation (2023) OWASP Cheat Sheet Series, Available at: https://cheatsheetseries.owasp.org/ (Accessed: 15 January 2024).

Microsoft (2023) ASP.NET Core security documentation, Available at: https://learn.microsoft.com/en-us/aspnet/core/security/ (Accessed: 15 January 2024).

Microsoft (2023) Visual Studio 2022 documentation, Available at: https://learn.microsoft.com/en-us/visualstudio/ide/ (Accessed: 15 January 2024).

.NET Foundation (2023) .NET 7 SDK, Available at: https://dotnet.microsoft.com/en-us/download/dotnet/7.0 (Accessed: 15 January 2024).

Microsoft (2023) ASP.NET Core MVC pattern, Available at: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview (Accessed: 15 January 2024).

Gamma, E. et al. (1994) Design patterns: elements of reusable object-oriented software, Boston: Addison-Wesley.

Martin, R.C. (2008) Clean code: a handbook of agile software craftsmanship, Upper Saddle River: Prentice Hall.

Fowler, M. (2002) Patterns of enterprise application architecture, Boston: Addison-Wesley.

Beck, K. (2002) Test-driven development: by example, Boston: Addison-Wesley.

Evans, E. (2003) Domain-driven design: tackling complexity in the heart of software, Boston: Addison-Wesley.

Internet Engineering Task Force (2023) *HTTP/1.1 specification (RFC 9112)*, Available at: https://httpwg.org/specs/rfc9112.html (Accessed: 15 January 2024).

World Wide Web Consortium (2023) Web Content Accessibility Guidelines (WCAG) 2.2, Available at: https://www.w3.org/TR/WCAG22/ (Accessed: 15 January 2024).

Information Commissioner's Office (2023) Guide to the UK General Data Protection Regulation (UK GDPR), Available at: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/ (Accessed: 15 January 2024).

European Union (2016) General Data Protection Regulation (GDPR), Official Journal of the European Union, L119/1.

Microsoft (2023) C# coding conventions, Available at: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions (Accessed: 15 January 2024).

GitHub (2023) GitHub Flow, Available at: https://docs.github.com/en/get-started/using-github/github-flow (Accessed: 15 January 2024).

Newtonsoft (2023) Json.NET documentation, Available at: https://www.newtonsoft.com/json (Accessed: 15 January 2024).

Moq Project (2023) Moq documentation, Available at: https://github.com/moq/moq (Accessed: 15 January 2024).

University of South Africa (2023) PROG6212: Programming course materials, Pretoria: UNISA.

Stack Overflow (2023) Stack Overflow community knowledge base, Available at: https://stackoverflow.com/ (Accessed: 15 January 2024).

Chacon, S. and Straub, B. (2014) Pro Git, 2nd edn, New York: Apress.

Git (2023) Git documentation, Available at: https://git-scm.com/doc (Accessed: 15 January 2024).