```
using CMCS.Mvc.Models;
using CMCS.Mvc.Data;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;
using Microsoft.EntityFrameworkCore;
using System.IO;
using System.Linq;

namespace CMCS.Mvc.Controllers
{
    public class ClaimsController : Controller
    {
        private readonly CMCSDbContext _context;
        private readonly IWebHostEnvironment _env;

        public ClaimsController(CMCSDbContext context,
          IWebHostEnvironment env)
        {
            _context = context;
            _env = env;
        }

        // GET: /Claims
        public IActionResult Index()
        {
            var claims = _context.Claims
                .Include(c => c.User)  // Changed from Lecturer to User
                .ToList();
            return View(claims);
        }

        // GET:/Claims/Details/5
        public IActionResult Details(int id)
        {
            var claim = _context.Claims
                .Include(c => c.User)  // Changed from Lecturer to User
                .FirstOrDefault(c => c.Id == id);
            if (claim == null) return NotFound();
            return View(claim);
        }

        // GET: /Claims/Create
        public IActionResult Create()
        {
            ViewBag.Users = _context.Users
                .Where(u => u.Role == "Lecturer" && u.IsActive)
                .OrderBy(u => u.Name)
                .ThenBy(u => u.Surname)
                .ToList();
            return View();
        }

        // POST: /Claims/Create
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Create(int userId, string title, string description, int hours = 0, decimal rate = 0, IFormFile file = null)
        {
            try
            {
                // Validate required fields
                if (userId == 0)
                {
                    ModelState.AddModelError("UserId", "Lecturer is required.");
                    ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer" && u.IsActive).ToList();
                    return View();
                }

                if (string.IsNullOrWhiteSpace(title))
                {
                    ModelState.AddModelError("Title", "Title is required.");
                    ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer" && u.IsActive).ToList();
                    return View();
                }

                // Get the selected user to use their hourly rate if rate is 0
                var selectedUser = _context.Users.Find(userId);
                if (selectedUser != null && rate == 0)
                {
                    rate = selectedUser.HourlyRate;
                }

                // Validate file if provided
                string filePath = null;
                if (file != null && file.Length > 0)
                {
                    // File size limit: 10MB
                    if (file.Length > 10 * 1024 * 1024)
                    {
                        ModelState.AddModelError("file", "File size must be less than 10MB.");
                        ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer" && u.IsActive).ToList();
                        return View();
                    }

                    // Allowed file types
                    var allowedExtensions = new[] { ".pdf", ".docx", ".xlsx", ".doc", ".xls" };
                    var fileExtension = Path.GetExtension(file.FileName).ToLowerInvariant();
                    if (!allowedExtensions.Contains(fileExtension))
                    {
                        ModelState.AddModelError("file", "Only PDF, Word (.docx, .doc), and Excel (.xlsx, .xls) files are allowed.");
                        ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer" && u.IsActive).ToList();
                        return View();
                    }

                    // Secure file storage
                    var uploadsRoot = Path.Combine(_env.WebRootPath ?? "wwwroot", "uploads");
                    Directory.CreateDirectory(uploadsRoot);
                    // Generate secure filename
                    var safeFileName = Path.GetFileName(file.FileName);
                    var uniqueFileName = $"{System.Guid.NewGuid()}_{safeFileName}";
                    var relativePath = Path.Combine("uploads", uniqueFileName);
                    var absolutePath = Path.Combine(_env.WebRootPath ?? "wwwroot", relativePath);

                    using (var stream = new FileStream(absolutePath, FileMode.Create))
                    {
                        file.CopyTo(stream);
                    }
                    filePath = relativePath.Replace(Path.DirectorySeparatorChar, '/');
                }

                // Create claim
                var claim = new Claim
                {
                    UserId = userId,
                    Title = title.Trim(),
                    Description = string.IsNullOrWhiteSpace(description) ? "No description provided" : description.Trim(),
                    Hours = hours,
                    Rate = rate,
                    FilePath = filePath,
                    Status = "Pending",
                    CreatedDate = DateTime.UtcNow,
                    ModifiedDate = DateTime.UtcNow
                };

                _context.Claims.Add(claim);
                _context.SaveChanges();

                return RedirectToAction(nameof(Index));
            }
            catch (System.Exception ex)
            {
                System.Diagnostics.Debug.WriteLine($"Error: {ex.Message}");
                ModelState.AddModelError("", "Error submitting claim. Please try again.");
                ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer" && u.IsActive).ToList();
                return View();
            }
        }

        // GET: /Claims/Edit/5
        public IActionResult Edit(int id)
        {
            var claim = _context.Claims
                .Include(c => c.User)  // Changed from Lecturer to User
                .FirstOrDefault(c => c.Id == id);
            if (claim == null) return NotFound();

            ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer").ToList();  // Changed from Lecturers to Users
            return View(claim);
        }

        // POST:/Claims/Edit
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Edit(int id, int userId, string title, string description, int hours = 0, decimal rate = 0, IFormFile file = null)  // Changed lecturerId to userId
        {
            var existingClaim = _context.Claims.Find(id);
            if (existingClaim == null) return NotFound();

            try
            {
                // Validate required fields
                if (userId == 0)
                {
                    ModelState.AddModelError("UserId", "User is required.");  // Changed from LecturerId to UserId
                    ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer").ToList();
                    return View(existingClaim);
                }
                if (string.IsNullOrWhiteSpace(title))
                {
                    ModelState.AddModelError("Title", "Title is required.");
                    ViewBag.Users = _context.Users.Where(u => u.
                      Role == "Lecturer").ToList();
                    return View(existingClaim);
                }

                // Validate file if provided
                string filePath = existingClaim.FilePath;
                if (file != null && file.Length > 0)
                {
                    // File size limit: 10MB
                    if (file.Length > 10 * 1024 * 1024)
                    {
                        ModelState.AddModelError("file", "File size must be less than 10MB.");
                        ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer").ToList();
                        return View(existingClaim);
                    }

                    // Allowed file types
                    var allowedExtensions = new[] { ".pdf", ".docx", ".xlsx", ".doc", ".xls" };
                    var fileExtension = Path.GetExtension(file.FileName).ToLowerInvariant();
                    if (!allowedExtensions.Contains(fileExtension))
                    {
                        ModelState.AddModelError("file", "Only PDF, Word (.docx, .doc), and Excel (.xlsx, .xls) files are allowed.");
                        ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer").ToList();
                        return View(existingClaim);
                    }

                    // Delete old file if exists
                    if (!string.IsNullOrEmpty(existingClaim.FilePath))
                    {
                        try
                        {
                            var oldFilePath = Path.Combine(_env.WebRootPath ?? "wwwroot",
                                existingClaim.FilePath.Replace('/', Path.DirectorySeparatorChar));
                            if (System.IO.File.Exists(oldFilePath))
                            {
                                System.IO.File.Delete(oldFilePath);
                            }
                        }
                        catch
                        {
                            // Ignore deletion errors
                        }
                    }

```

```
                    // Secure file storage
                    var uploadsRoot = Path.Combine(_env.WebRootPath ?? "wwwroot", "uploads");
                    Directory.CreateDirectory(uploadsRoot);

                    // Generate secure filename
                    var safeFileName = Path.GetFileName(file.FileName);
                    var uniqueFileName = $"{System.Guid.NewGuid()}_{safeFileName}";
                    var relativePath = Path.Combine("uploads", uniqueFileName);
                    var absolutePath = Path.Combine(_env.WebRootPath ?? "wwwroot", relativePath);

                    using (var stream = new FileStream(absolutePath, FileMode.Create))
                    {
                        file.CopyTo(stream);
                    }
                    filePath = relativePath.Replace(Path.DirectorySeparatorChar, '/');
                }

                // Update claim
                existingClaim.UserId = userId;  // Changed from LecturerId to UserId
                existingClaim.Title = title.Trim();
                existingClaim.Description = string.IsNullOrWhiteSpace(description) ? "No description provided" : description.Trim();
                existingClaim.Hours = hours;
                existingClaim.Rate = rate;
                existingClaim.FilePath = filePath;

                _context.SaveChanges();

                return RedirectToAction(nameof(Index));
            }
            catch (System.Exception ex)
            {
                System.Diagnostics.Debug.WriteLine($"Error: {ex.Message}");
                ModelState.AddModelError("", "Error updating claim. Please try again.");
                ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer").ToList();
                return View(existingClaim);
            }
        }

        // GET: /Claims/Delete/5
        public IActionResult Delete(int id)
        {
            var claim = _context.Claims
                .Include(c => c.User)  // Changed from Lecturer to User
                .FirstOrDefault(c => c.Id == id);
            if (claim == null) return NotFound();
            return View(claim);
        }

        // POST: /Claims/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public IActionResult DeleteConfirmed(int id)
        {
            var claim = _context.Claims.Find(id);
            if (claim != null && !string.IsNullOrEmpty(claim.FilePath))
            {
                try
                {
                    var filePath = Path.Combine(_env.WebRootPath ?? "wwwroot",
                        claim.FilePath.Replace('/', Path.DirectorySeparatorChar));
                    if (System.IO.File.Exists(filePath))
                    {
                        System.IO.File.Delete(filePath);
                    }
                }
                catch
                {
                    // Ignore file system failures
                }
            }

            if (claim != null)
            {
                _context.Claims.Remove(claim);
                _context.SaveChanges();
            }

            return RedirectToAction(nameof(Index));
        }

        // GET: /Claims/Download/5
        public IActionResult Download(int id)
        {
            var claim = _context.Claims.Find(id);
            if (claim == null || string.IsNullOrEmpty(claim.FilePath))
                return NotFound();

            var absolutePath = Path.Combine(_env.WebRootPath ?? "wwwroot",
                claim.FilePath.Replace('/', Path.DirectorySeparatorChar));

            if (!System.IO.File.Exists(absolutePath)) return NotFound();

            var fileName = Path.GetFileName(absolutePath);
            return PhysicalFile(absolutePath, "application/octet-stream", fileName);
        }
    }
}
```