

```
1 using CMCS.Mvc.Models;
2 using CMCS.Mvc.Data;
3 using Microsoft.AspNetCore.Hosting;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.AspNetCore.Http;
6 using Microsoft.EntityFrameworkCore;
7 using System.IO;
8 using System.Linq;
9
10 namespace CMCS.Mvc.Controllers
11 {
12     public class ClaimsController : Controller
13     {
14         private readonly CMCSDbContext _context;
15         private readonly IWebHostEnvironment _env;
16
17         public ClaimsController(CMCSDbContext context,
18                                 IWebHostEnvironment env)
19         {
20             _context = context;
21             _env = env;
22         }
23
24         // GET: /Claims
25         public IActionResult Index()
26         {
27             var claims = _context.Claims
28                 .Include(c => c.User) // Changed from Lecturer to User
29                 .ToList();
30             return View(claims);
31         }
32
33         // GET: /Claims/Details/5
34         public IActionResult Details(int id)
35         {
36             var claim = _context.Claims
37                 .Include(c => c.User) // Changed from Lecturer to User
38                 .FirstOrDefault(c => c.Id == id);
39             if (claim == null) return NotFound();
40             return View(claim);
41         }
42
43         // GET: /Claims/Create
44         public IActionResult Create()
45         {
46             ViewBag.Users = _context.Users
47                 .Where(u => u.Role == "Lecturer" && u.IsActive)
48                 .OrderBy(u => u.Name)
49                 .ThenBy(u => u.Surname)
50                 .ToList();
51             return View();
52         }
53     }
54 }
```

```
53 // POST: /Claims/Create
54 [HttpPost]
55 [ValidateAntiForgeryToken]
56 public IActionResult Create(int userId, string title, string
    description, int hours = 0, decimal rate = 0, IFormFile file
    = null)
57 {
58     try
59     {
60         // Validate required fields
61         if (userId == 0)
62         {
63             ModelState.AddModelError("UserId", "Lecturer is
                required.");
64             ViewBag.Users = _context.Users.Where(u => u.
                Role == "Lecturer" && u.IsActive).ToList();
65             return View();
66         }
67
68         if (string.IsNullOrEmpty(title))
69         {
70             ModelState.AddModelError("Title", "Title is
                required.");
71             ViewBag.Users = _context.Users.Where(u => u.
                Role == "Lecturer" && u.IsActive).ToList();
72             return View();
73         }
74
75         // Get the selected user to use their hourly rate if
            rate is 0
76         var selectedUser = _context.Users.Find(userId);
77         if (selectedUser != null && rate == 0)
78         {
79             rate = selectedUser.HourlyRate;
80         }
81
82         // Validate file if provided
83         string filePath = null;
84         if (file != null && file.Length > 0)
85         {
86             // File size limit: 10MB
87             if (file.Length > 10 * 1024 * 1024)
88             {
89                 ModelState.AddModelError("file", "File size
                    must be less than 10MB.");
90                 ViewBag.Users = _context.Users.Where(u =>
                    u.Role == "Lecturer" && u.IsActive).ToList();
91                 return View();
92             }
93
94             // Allowed file types
95             var allowedExtensions = new[] { ".pdf", ".docx",
                ".xlsx", ".doc", ".xls" };
```

```

..._PoE_Prototype_NET8\Controllers\ClaimsController.cs 3
196      var fileExtension = Path.GetExtension
          (file.FileName).ToLowerInvariant();
197      if (!allowedExtensions.Contains(fileExtension))
198      {
199          ModelState.AddModelError("file", "Only PDF,
          Word (.docx, .doc), and Excel (.xlsx, .xls)
          files are allowed.");
200          ViewBag.Users = _context.Users.Where(u =>
          u.Role == "Lecturer" && u.IsActive).ToList();
201          return View();
202      }
203
204      // Secure file storage
205      var uploadsRoot = Path.Combine(_env.WebRootPath ??
          "wwwroot", "uploads");
206      Directory.CreateDirectory(uploadsRoot);
207      // Generate secure filename
208      var safeFileName = Path.GetFileName(file.FileName);
209      var uniqueFileName = $"{System.Guid.NewGuid()}_
          {safeFileName}";
210      var relativePath = Path.Combine("uploads",
          uniqueFileName);
211      var absolutePath = Path.Combine(_env.
          WebRootPath ?? "wwwroot", relativePath);
212
213      using (var stream = new FileStream(absolutePath,
          FileMode.Create))
214      {
215          file.CopyTo(stream);
216      }
217      filePath = relativePath.Replace
          (Path.DirectorySeparatorChar, '/');
218
219
220      // Create claim
221      var claim = new Claim
222      {
223          UserId = userId,
224          Title = title.Trim(),
225          Description = string.IsNullOrEmpty
          (description) ? "No description provided" :
          description.Trim(),
226          Hours = hours,
227          Rate = rate,
228          FilePath = filePath,
229          Status = "Pending",
230          CreatedDate = DateTime.UtcNow,
231          ModifiedDate = DateTime.UtcNow
232      };
233
234      _context.Claims.Add(claim);
235      _context.SaveChanges();
236

```

```
137         return RedirectToAction(nameof(Index));
138     }
139     catch (System.Exception ex)
140     {
141         System.Diagnostics.Debug.WriteLine($"Error: {ex.Message}");
142         ModelState.AddModelError("", "Error submitting claim. Please try again.");
143         ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer" && u.IsActive).ToList();
144         return View();
145     }
146 }
147
148 // GET: /Claims/Edit/5
149 public IActionResult Edit(int id)
150 {
151     var claim = _context.Claims
152         .Include(c => c.User) // Changed from Lecturer to User
153         .FirstOrDefault(c => c.Id == id);
154     if (claim == null) return NotFound();
155
156     ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer").ToList(); // Changed from Lecturers to Users
157     return View(claim);
158 }
159
160 // POST: /Claims/Edit
161 [HttpPost]
162 [ValidateAntiForgeryToken]
163 public IActionResult Edit(int id, int userId, string title, string description, int hours = 0, decimal rate = 0, IFormFile file = null) // Changed lecturerId to userId
164 {
165     var existingClaim = _context.Claims.Find(id);
166     if (existingClaim == null) return NotFound();
167
168     try
169     {
170         // Validate required fields
171         if (userId == 0)
172         {
173             ModelState.AddModelError("UserId", "User is required."); // Changed from LecturerId to UserId
174             ViewBag.Users = _context.Users.Where(u => u.Role == "Lecturer").ToList();
175             return View(existingClaim);
176         }
177         if (string.IsNullOrEmpty(title))
178         {
179             ModelState.AddModelError("Title", "Title is required.");
180             ViewBag.Users = _context.Users.Where(u => u.
```

```
        Role == "Lecturer").ToList();
181     return View(existingClaim);
182 }
183
184 // Validate file if provided
185 string filePath = existingClaim.FilePath;
186 if (file != null && file.Length > 0)
187 {
188     // File size limit: 10MB
189     if (file.Length > 10 * 1024 * 1024)
190     {
191         ModelState.AddModelError("file", "File size
192             must be less than 10MB.");
193         ViewBag.Users = _context.Users.Where(u =>
194             u.Role == "Lecturer").ToList();
195         return View(existingClaim);
196     }
197
198 // Allowed file types
199 var allowedExtensions = new[] { ".pdf", ".docx",
200     ".xlsx", ".doc", ".xls" };
201 var fileExtension = Path.GetExtension
202     (file.FileName).ToLowerInvariant();
203 if (!allowedExtensions.Contains(fileExtension))
204 {
205     ModelState.AddModelError("file", "Only PDF,
206         Word (.docx, .doc), and Excel (.xlsx, .xls)
207         files are allowed.");
208     ViewBag.Users = _context.Users.Where(u =>
209         u.Role == "Lecturer").ToList();
210     return View(existingClaim);
211 }
212
213 // Delete old file if exists
214 if (!string.IsNullOrEmpty(existingClaim.FilePath))
215 {
216     try
217     {
218         var oldFilePath = Path.Combine
219             (_env.WebRootPath ?? "wwwroot",
220                 existingClaim.FilePath.Replace('/',
221                     Path.DirectorySeparatorChar));
222         if (System.IO.File.Exists(oldFilePath))
223         {
224             System.IO.File.Delete(oldFilePath);
225         }
226     }
227     catch
228     {
229         // Ignore deletion errors
230     }
231 }
```

```
224 // Secure file storage
225 var uploadsRoot = Path.Combine(_env.WebRootPath ??
    "wwwroot", "uploads");
226 Directory.CreateDirectory(uploadsRoot);
227
228 // Generate secure filename
229 var safeFileName = Path.GetFileName(file.FileName);
230 var uniqueFileName = $"{System.Guid.NewGuid()}_
    {safeFileName}";
231 var relativePath = Path.Combine("uploads",
    uniqueFileName);
232 var absolutePath = Path.Combine(_env.
    WebRootPath ?? "wwwroot", relativePath);
233
234 using (var stream = new FileStream(absolutePath,
    FileMode.Create))
235 {
236     file.CopyTo(stream);
237 }
238 filePath = relativePath.Replace
    (Path.DirectorySeparatorChar, '/');
239 }
240
241 // Update claim
242 existingClaim.UserId = userId; // Changed from
    LecturerId to UserId
243 existingClaim.Title = title.Trim();
244 existingClaim.Description = string.IsNullOrEmpty
    (description) ? "No description provided" :
    description.Trim();
245 existingClaim.Hours = hours;
246 existingClaim.Rate = rate;
247 existingClaim.FilePath = filePath;
248
249 _context.SaveChanges();
250
251 return RedirectToAction(nameof(Index));
252 }
253 catch (System.Exception ex)
254 {
255     System.Diagnostics.Debug.WriteLine($"Error:
    {ex.Message}");
256     ModelState.AddModelError("", "Error updating claim.
    Please try again.");
257     ViewBag.Users = _context.Users.Where(u => u.Role ==
    "Lecturer").ToList();
258     return View(existingClaim);
259 }
260 }
261
262 // GET: /Claims/Delete/5
263 public IActionResult Delete(int id)
264 {
```

```

265     var claim = _context.Claims
266         .Include(c => c.User) // Changed from Lecturer to User
267         .FirstOrDefault(c => c.Id == id);
268     if (claim == null) return NotFound();
269     return View(claim);
270 }
271
272 // POST: /Claims/Delete/5
273 [HttpPost, ActionName("Delete")]
274 [ValidateAntiForgeryToken]
275 public IActionResult DeleteConfirmed(int id)
276 {
277     var claim = _context.Claims.Find(id);
278     if (claim != null && !string.IsNullOrEmpty(claim.FilePath))
279     {
280         try
281         {
282             var filePath = Path.Combine(_env.WebRootPath ??
283                 "wwwroot",
284                 claim.FilePath.Replace('/',
285                     Path.DirectorySeparatorChar));
286             if (System.IO.File.Exists(filePath))
287             {
288                 System.IO.File.Delete(filePath);
289             }
290         }
291         catch
292         {
293             // Ignore file system failures
294         }
295     }
296
297     if (claim != null)
298     {
299         _context.Claims.Remove(claim);
300         _context.SaveChanges();
301     }
302
303     return RedirectToAction(nameof(Index));
304 }
305
306 // GET: /Claims/Download/5
307 public IActionResult Download(int id)
308 {
309     var claim = _context.Claims.Find(id);
310     if (claim == null || string.IsNullOrEmpty(claim.FilePath))
311         return NotFound();
312
313     var absolutePath = Path.Combine(_env.WebRootPath ??
314         "wwwroot",
315         claim.FilePath.Replace('/',
316             Path.DirectorySeparatorChar));

```

```
314         if (!System.IO.File.Exists(absolutePath)) return NotFound  
315             ();  
316         var fileName = Path.GetFileName(absolutePath);  
317         return PhysicalFile(absolutePath, "application/octet-  
318             stream", fileName);  
319     }  
320 }
```