

```
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     // POST: /Claims/Create
54     [HttpPost]
55     [ValidateAntiForgeryToken]
56     public IActionResult Create(int userId, string title, string      ↵
57         description, int hours = 0, decimal rate = 0, IFormFile file      ↵
58         = null)
59     {
60         try
61         {
62             // Validate required fields
63             if (userId == 0)
64             {
65                 ModelState.AddModelError("UserId", "Lecturer is      ↵
66                     required.");
67                 ViewBag.Users = _context.Users.Where(u => u.      ↵
68                     Role == "Lecturer" && u.IsActive).ToList();
69                 return View();
70             }
71             if (string.IsNullOrWhiteSpace(title))
72             {
73                 ModelState.AddModelError("Title", "Title is      ↵
74                     required.");
75                 ViewBag.Users = _context.Users.Where(u => u.      ↵
76                     Role == "Lecturer" && u.IsActive).ToList();
77                 return View();
78             }
79             // Get the selected user to use their hourly rate if      ↵
80             // rate is 0
81             var selectedUser = _context.Users.Find(userId);
82             if (selectedUser != null && rate == 0)
83             {
84                 rate = selectedUser.HourlyRate;
85             }
86             // Validate file if provided
87             string filePath = null;
88             if (file != null && file.Length > 0)
89             {
90                 // File size limit: 10MB
91                 if (file.Length > 10 * 1024 * 1024)
92                 {
93                     ModelState.AddModelError("file", "File size      ↵
94                         must be less than 10MB.");
95                     ViewBag.Users = _context.Users.Where(u =>      ↵
96                         u.Role == "Lecturer" && u.IsActive).ToList();
97                     return View();
98                 }
99             }
100            // Allowed file types
101            var allowedExtensions = new[] { ".pdf", ".docx",      ↵
102                ".xlsx", ".doc", ".xls" };
```

```
96             var fileExtension = Path.GetExtension  
97                 (file.FileName).ToLowerInvariant();  
98             if (!allowedExtensions.Contains(fileExtension))  
99             {  
100                 ModelState.AddModelError("file", "Only PDF,      ↵  
101                     Word (.docx, .doc), and Excel (.xlsx, .xls)    ↵  
102                     files are allowed.");  
103             ViewBag.Users = _context.Users.Where(u =>  
104                 u.Role == "Lecturer" && u.IsActive).ToList();  
105             return View();  
106         }  
107  
108         // Secure file storage  
109         var uploadsRoot = Path.Combine(_env.WebRootPath ??    ↵  
110             "wwwroot", "uploads");  
111         Directory.CreateDirectory(uploadsRoot);  
112         // Generate secure filename  
113         var safeFileName = Path.GetFileName(file.FileName);  
114         var uniqueFileName = $"{System.Guid.NewGuid()}_    ↵  
115             {safeFileName}";  
116         var relativePath = Path.Combine("uploads",  
117             uniqueFileName);  
118         var absolutePath = Path.Combine(_env.  
119             WebRootPath ?? "wwwroot", relativePath);  
120  
121         using (var stream = new FileStreamAbsolutePath,  
122             FileMode.Create))  
123         {  
124             file.CopyTo(stream);  
125             filePath = relativePath.Replace  
126                 (Path.DirectorySeparatorChar, '/');  
127         }  
128  
129         // Create claim  
130         var claim = new Claim  
131         {  
132             UserId = userId,  
133             Title = title.Trim(),  
134             Description = string.IsNullOrWhiteSpace  
135                 (description) ? "No description provided" :  
136                 description.Trim(),  
137             Hours = hours,  
138             Rate = rate,  
139             FilePath = filePath,  
140             Status = "Pending",  
141             CreatedDate = DateTime.UtcNow,  
142             ModifiedDate = DateTime.UtcNow  
143         };  
144  
145         _context.Claims.Add(claim);  
146         _context.SaveChanges();  
147     }
```

```
137             return RedirectToAction(nameof(Index));
138         }
139         catch (System.Exception ex)
140     {
141         System.Diagnostics.Debug.WriteLine($"Error:
142             {ex.Message}");
143         ModelState.AddModelError("", "Error submitting claim.
144             Please try again.");
145         ViewBag.Users = _context.Users.Where(u => u.Role ==
146             "Lecturer" && u.IsActive).ToList();
147         return View();
148     }
149 
150     // GET: /Claims/Edit/5
151     public IActionResult Edit(int id)
152     {
153         var claim = _context.Claims
154             .Include(c => c.User) // Changed from Lecturer to User
155             .FirstOrDefault(c => c.Id == id);
156         if (claim == null) return NotFound();
157 
158         ViewBag.Users = _context.Users.Where(u => u.Role ==
159             "Lecturer").ToList(); // Changed from Lecturers to Users
160         return View(claim);
161     }
162 
163     // POST:/Claims/Edit
164     [HttpPost]
165     [ValidateAntiForgeryToken]
166     public IActionResult Edit(int id, int userId, string title,
167         string description, int hours = 0, decimal rate = 0,
168         IFormFile file = null) // Changed lecturerId to userId
169     {
170         var existingClaim = _context.Claims.Find(id);
171         if (existingClaim == null) return NotFound();
172 
173         try
174         {
175             // Validate required fields
176             if (userId == 0)
177             {
178                 ModelState.AddModelError("UserId", "User is
179                     required."); // Changed from LecturerId to UserId
180                 ViewBag.Users = _context.Users.Where(u => u.
181                     Role == "Lecturer").ToList();
182                 return View(existingClaim);
183             }
184             if (string.IsNullOrWhiteSpace(title))
185             {
186                 ModelState.AddModelError("Title", "Title is
187                     required.");
188                 ViewBag.Users = _context.Users.Where(u => u.
```

```
181             Role == "Lecturer").ToList();
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             }
232         }
233     }
234 }
```

```
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()}_"
231                 {safeFileName}";
232             var relativePath = Path.Combine("uploads",
233                 uniqueFileName);
234             var absolutePath = Path.Combine(_env.
235                 WebRootPath ?? "wwwroot", relativePath);
236
237             using (var stream = new FileStream(absolutePath,
238                 FileMode.Create))
239             {
240                 file.CopyTo(stream);
241             }
242             filePath = relativePath.Replace
243                 (Path.DirectorySeparatorChar, '/');
244
245             // Update claim
246             existingClaim.UserId = userId; // Changed from
247                 LecturerId to UserId
248             existingClaim.Title = title.Trim();
249             existingClaim.Description = string.IsNullOrWhiteSpace
250                 (description) ? "No description provided" :
251                     description.Trim();
252             existingClaim.Hours = hours;
253             existingClaim.Rate = rate;
254             existingClaim.FilePath = filePath;
255
256             _context.SaveChanges();
257
258             return RedirectToAction(nameof(Index));
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 ?? "wwwroot",
283                     claim.FilePath.Replace('/', Path.DirectorySeparatorChar));
284                 if (System.IO.File.Exists(filePath))
285                 {
286                     System.IO.File.Delete(filePath);
287                 }
288             }
289             catch
290             {
291                 // Ignore file system failures
292             }
293         }
294
295         if (claim != null)
296         {
297             _context.Claims.Remove(claim);
298             _context.SaveChanges();
299         }
300
301         return RedirectToAction(nameof(Index));
302     }
303
304     // GET: /Claims/Download/5
305     public IActionResult Download(int id)
306     {
307         var claim = _context.Claims.Find(id);
308         if (claim == null || string.IsNullOrEmpty(claim.FilePath))
309             return NotFound();
310
311         var absolutePath = Path.Combine(_env.WebRootPath ?? "wwwroot",
312             claim.FilePath.Replace('/', Path.DirectorySeparatorChar));
```

..._PoE_Prototype_NET8\Controllers\ClaimsController.cs 8

```
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 }
```