```
 1  using Microsoft.AspNetCore.Mvc;
 2  using Microsoft.EntityFrameworkCore;
 3  using WebApplication1;
 4  using WebApplication1.DbModels;
 5  using Microsoft.AspNetCore.Authorization;
 6  using System.Security.Claims;
 7  using WebApplication1.Models;
 8  using WebApplication1.Services;
 9
10  public class BlogsController : Controller
11  {
12      private readonly AppDbContext _context;
13      private readonly UserService _userService;
14
15
16      public BlogsController(AppDbContext context,UserService
          userService)
17      {
18          _context = context;
19          _userService = userService;
20      }
21
22      // GET: Blogs/Create
23      public IActionResult Create()
24      {
25          ViewBag.userId = _userService.GetUserId();
26          return View();
27      }
28
29      // POST: Blogs/Create
30      [Authorize]
31      [HttpPost]
32      [ValidateAntiForgeryToken]
33      public async Task<IActionResult> Create([Bind
          ("Id,Name,Description,Theme,AuthorId")] Blog blog,int userId)
34      {
35          if (_context.Blogs.Where(blog => blog.AuthorId == userId).Count
              () >= 5)
36          {
37              return RedirectToAction("Index", "Profile");
38          }
39          if (blog.Name==""|| blog.Name == null||
40              blog.Description == "" || blog.Description == null||
41              blog.Theme == "" || blog.Theme == null)
42          {
43              return View(blog);
44          }
45
46          if (ModelState.IsValid)
47          {
48              blog.AuthorId = userId;
49              _context.Add(blog);
50              await _context.SaveChangesAsync();
```

```
51              return RedirectToAction("Index", "Profile");
52          }
53          return View(blog);
54      }
55
56      // GET: Blogs/Edit/5
57      [Authorize]
58      public async Task<IActionResult> Edit(int? id)
59      {
60          if (id == null || _context.Blogs == null)
61          {
62              return NotFound();
63          }
64
65          var blog = await _context.Blogs.FindAsync(id);
66          if (blog.AuthorId != (int)_userService.GetUserId())
67          {
68              return RedirectToAction("Index", "Profile");
69          }
70          if (blog == null)
71          {
72              return NotFound();
73          }
74          return View(blog);
75      }
76
77      // POST: Blogs/Edit/5
78      [Authorize]
79      [HttpPost]
80      [ValidateAntiForgeryToken]
81      public async Task<IActionResult> Edit(int id, [Bind
          ("Id,Name,Description,Theme")] Blog blog)
82      {
83          if (id != blog.Id)
84          {
85              return NotFound();
86          }
87
88          if (blog.Name == "" || blog.Name == null ||
89            blog.Description == "" || blog.Description == null ||
90            blog.Theme == "" || blog.Theme == null)
91          {
92              return View(blog);
93          }
94
95          if (ModelState.IsValid)
96          {
97              try
98              {
99                  var existingBlog = await _context.Blogs.FindAsync(id);
100                 if (existingBlog == null)
101                 {
102                     return NotFound();
```

```
103                 }
104
105                 existingBlog.Name = blog.Name;
106                 existingBlog.Description = blog.Description;
107                 existingBlog.Theme = blog.Theme;
108
109                 _context.Update(existingBlog);
110                 await _context.SaveChangesAsync();
111             }
112             catch (DbUpdateConcurrencyException)
113             {
114                 if (!BlogExists(blog.Id))
115                 {
116                     return NotFound();
117                 }
118                 else
119                 {
120                     throw;
121                 }
122             }
123             return RedirectToAction("Index", "Profile");
124         }
125         return View(blog);
126     }
127
128     // GET: Blogs/Delete/5
129     [Authorize]
130     public async Task<IActionResult> Delete(int? id)
131     {
132         if (id == null)
133         {
134             return NotFound();
135         }
136
137         var blog = await _context.Blogs.FirstOrDefaultAsync(m => m.Id
              == id);
138         if (blog == null)
139         {
140             return NotFound();
141         }
142
143         var postsToDelete = await _context.Posts.Where(m => m.BlogId ==
               id).ToListAsync();
144         foreach (var post in postsToDelete)
145         {
146             var contentsToDelete = _context.Post_Contents.Where(m =>
                  m.PostId == post.Id);
147             _context.Post_Contents.RemoveRange(contentsToDelete);
148             var reactionsToDelete = _context.Reactions.Where(m =>
                  m.PostId == post.Id);
149             _context.Reactions.RemoveRange(reactionsToDelete);
150             var comentsToDelete = _context.Coments.Where(m => m.PostId
                  == post.Id);
```

```
151             _context.Coments.RemoveRange(comentsToDelete);
152             _context.Posts.Remove(post);
153             await _context.SaveChangesAsync();
154         }
155
156         _context.Blogs.Remove(blog);
157
158         await _context.SaveChangesAsync();
159
160         return RedirectToAction("Index", "Profile");
161     }
162
163     [HttpGet]
164     public async Task<IActionResult> GetBlogs()
165     {
166         var blogs = await _context.Blogs.ToListAsync();
167         var users = await _context.Users.ToListAsync();
168
169         var blogsWithAuthors = blogs.Select(blog => new BlogViewModel
170         {
171             Id = blog.Id,
172             Name = blog.Name,
173             Description = blog.Description,
174             Theme = blog.Theme,
175             AuthorName = users.FirstOrDefault(user => user.Id ==
                  blog.AuthorId)?.Name ?? "Unknown"
176         }).ToList();
177
178         return Json(blogsWithAuthors);
179     }
180
181     private bool BlogExists(int id)
182     {
183         return _context.Blogs.Any(e => e.Id == id);
184     }
185
186     [HttpGet]
187     public async Task<IActionResult> GetThemes()
188     {
189         var themes = await _context.Themes.ToListAsync();
190         return Json(themes);
191     }
192
193
194
195     public async Task<IActionResult> GetBlogsForRecomendation(int
          currentPage, int blogsPerPage,string? blogName)
196     {
197         int skip = (currentPage - 1) * blogsPerPage;
198
199         var query = _context.Blogs.AsQueryable(); ;
200
201         if (blogName != null)
```

```
202         {
203             query = query.Where(blog => blog.Name.Contains(blogName));
204         }
205 
206         var blogs = query
207             .OrderBy(blog => blog.Id)
208             .Skip(skip)
209             .Take(blogsPerPage+1)
210             .Select(blog => new BlogViewModel
211             {
212                 Id = blog.Id,
213                 Name = blog.Name,
214                 Description = blog.Description,
215                 Theme = blog.Theme,
216                 AuthorName = _context.Users
217                     .Where(user => user.Id == blog.AuthorId)
218                     .Select(user => user.Name)
219                     .FirstOrDefault() ?? "Unknown"
220             })
221             .AsNoTracking();
222         return Json(blogs);
223     }
224 }
225 
```