using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using WebBaoDienTu.Models;

namespace WebBaoDienTu.Controllers

{

public class NewsController : Controller

{

private readonly BaoDienTuContext \_context;

public NewsController(BaoDienTuContext context)

{

\_context = context;

}

// GET: News

// GET: News

public async Task<IActionResult> Index()

{

var userRole = User.FindFirstValue(ClaimTypes.Role);

ViewBag.Categories = await \_context.Categories.ToListAsync();

ViewBag.HideNavElements = true; // Hide categories and search bar

if (userRole == "Admin")

{

var news = \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.Where(n => !n.IsDeleted && (!n.IsArchived || n.IsApproved))

.OrderByDescending(n => !n.IsApproved)

.ThenByDescending(n => n.CreatedAt);

return View("Index", await news.ToListAsync());

}

else if (userRole == "User")

{

return RedirectToAction("Create");

}

return RedirectToAction("Login", "Account");

}

// GET: News/Details/5

public async Task<IActionResult> Details(int? id)

{

if (!User.IsInRole("Admin"))

{

return Forbid();

}

ViewBag.Categories = await \_context.Categories.ToListAsync();

if (id == null)

{

return NotFound();

}

var news = await \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.FirstOrDefaultAsync(m => m.NewsId == id);

if (news == null)

{

return NotFound();

}

return View(news);

}

public async Task<IActionResult> GetNewsDetailsForDelete(int id)

{

var news = await \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.FirstOrDefaultAsync(m => m.NewsId == id);

if (news == null)

{

return NotFound();

}

return Json(new

{

title = news.Title,

createdAt = news.CreatedAt.ToString("dd/MM/yyyy HH:mm"),

authorFullName = news.Author.FullName,

imageUrl = news.ImageUrl

});

}

[Authorize(Roles = "User,Admin")]

public async Task<IActionResult> Create()

{

ViewBag.Categories = await \_context.Categories.ToListAsync();

ViewData["CategoryId"] = new SelectList(\_context.Categories, "CategoryId", "CategoryName");

ViewBag.HideNavElements = true; // Hide categories and search bar

return View();

}

[Authorize(Roles = "User,Admin")]

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Title,Content,ImageUrl,CategoryId")] News news, IFormFile ImageFile)

{

ViewData["CategoryId"] = new SelectList(\_context.Categories, "CategoryId", "CategoryName");

try

{

// Lấy AuthorId từ người dùng hiện tại

var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

if (string.IsNullOrEmpty(userId) || !int.TryParse(userId, out int authorId))

{

return Json(new { success = false, message = "Không thể xác định người dùng hiện tại. Vui lòng đăng nhập lại." });

}

// Gán AuthorId

news.AuthorId = authorId;

// Xóa lỗi validation cho AuthorId nếu có (vì nó không được gửi từ form)

ModelState.Remove("AuthorId");

// Kiểm tra dữ liệu

if (string.IsNullOrEmpty(news.Title))

{

ModelState.AddModelError("Title", "Tiêu đề không được để trống.");

}

if (string.IsNullOrEmpty(news.Content))

{

ModelState.AddModelError("Content", "Nội dung không được để trống.");

}

if (news.CategoryId <= 0)

{

ModelState.AddModelError("CategoryId", "Vui lòng chọn danh mục.");

}

// Debug dữ liệu gửi lên

Console.WriteLine($"Title: {news.Title}");

Console.WriteLine($"Content: {news.Content}");

Console.WriteLine($"CategoryId: {news.CategoryId}");

Console.WriteLine($"ImageUrl: {news.ImageUrl}");

if (!ModelState.IsValid)

{

var errors = ModelState.Values.SelectMany(v => v.Errors).Select(e => e.ErrorMessage).ToList();

return Json(new { success = false, message = string.Join("; ", errors) });

}

// Xử lý file ảnh nếu có

if (ImageFile != null && ImageFile.Length > 0)

{

string fileExtension = Path.GetExtension(ImageFile.FileName).ToLower();

string[] allowedExtensions = { ".jpg", ".jpeg", ".png", ".gif" };

if (!allowedExtensions.Contains(fileExtension))

{

return Json(new { success = false, message = "Chỉ được tải lên ảnh JPG, JPEG, PNG hoặc GIF." });

}

string uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/images");

if (!Directory.Exists(uploadsFolder)) Directory.CreateDirectory(uploadsFolder);

string uniqueFileName = Guid.NewGuid().ToString() + fileExtension;

string filePath = Path.Combine(uploadsFolder, uniqueFileName);

using (var fileStream = new FileStream(filePath, FileMode.Create))

{

await ImageFile.CopyToAsync(fileStream);

}

news.ImageUrl = "/images/" + uniqueFileName;

}

news.CreatedAt = DateTime.Now;

news.IsApproved = User.IsInRole("Admin");

news.IsDeleted = false;

\_context.Add(news);

await \_context.SaveChangesAsync();

return Json(new { success = true, message = "Tin tức đã được đăng thành công!", redirectTo = Url.Action("Index") });

}

catch (Exception ex)

{

Console.WriteLine($"Exception in Create: {ex.Message}");

Console.WriteLine($"Stack Trace: {ex.StackTrace}");

if (ex.InnerException != null)

{

Console.WriteLine($"Inner Exception: {ex.InnerException.Message}");

}

return Json(new { success = false, message = $"Có lỗi xảy ra: {ex.Message}" });

}

}

// GET: News/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (!User.IsInRole("Admin"))

{

return Forbid();

}

ViewBag.Categories = await \_context.Categories.ToListAsync();

if (id == null)

{

return NotFound();

}

var news = await \_context.News.FindAsync(id);

if (news == null)

{

return NotFound();

}

ViewData["AuthorId"] = new SelectList(\_context.Users, "UserId", "UserId", news.AuthorId);

ViewData["CategoryId"] = new SelectList(\_context.Categories, "CategoryId", "CategoryName", news.CategoryId);

return View(news);

}

// POST: News/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("NewsId,Title,Content,ImageUrl,AuthorId,CategoryId,IsApproved,CreatedAt")] News news, IFormFile ImageFile)

{

if (!User.IsInRole("Admin"))

{

return Forbid();

}

ViewBag.Categories = await \_context.Categories.ToListAsync();

ModelState.Clear();

if (id != news.NewsId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

if (ImageFile != null && ImageFile.Length > 0)

{

string fileExtension = Path.GetExtension(ImageFile.FileName).ToLower();

string[] allowedExtensions = { ".jpg", ".jpeg", ".png", ".gif" };

if (!allowedExtensions.Contains(fileExtension))

{

ModelState.AddModelError("ImageFile", "Chỉ được tải lên ảnh JPG, JPEG, PNG hoặc GIF.");

ViewBag.Categories = new SelectList(\_context.Categories, "Id", "Name");

return View(news);

}

string uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/images");

if (!Directory.Exists(uploadsFolder)) Directory.CreateDirectory(uploadsFolder);

string uniqueFileName = Guid.NewGuid().ToString() + fileExtension;

string filePath = Path.Combine(uploadsFolder, uniqueFileName);

using (var fileStream = new FileStream(filePath, FileMode.Create))

{

await ImageFile.CopyToAsync(fileStream);

}

news.ImageUrl = "/images/" + uniqueFileName;

}

\_context.Update(news);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!NewsExists(news.NewsId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["AuthorId"] = new SelectList(\_context.Users, "UserId", "UserId", news.AuthorId);

ViewData["CategoryId"] = new SelectList(\_context.Categories, "CategoryId", "CategoryName", news.CategoryId);

return View(news);

}

[HttpPost]

[Route("News/Delete/{id}")]

public IActionResult Delete(int id)

{

var newsItem = \_context.News.FirstOrDefault(n => n.NewsId == id);

if (newsItem != null)

{

newsItem.IsDeleted = true;

\_context.Update(newsItem);

\_context.SaveChanges();

return Json(new { success = true, message = "Xóa thành công" });

}

return Json(new { success = false, message = "Xóa thất bại: Tin tức không tồn tại" });

}

[HttpPost]

[Route("News/DeleteConfirmed/{id}")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var news = await \_context.News.FindAsync(id);

if (news != null)

{

if (User.IsInRole("Admin"))

{

\_context.News.Remove(news);

}

else

{

news.IsDeleted = true;

\_context.Update(news);

}

await \_context.SaveChangesAsync();

return Json(new { success = true, message = "Xóa tin tức thành công" });

}

return Json(new { success = false, message = "Không tìm thấy tin tức" });

}

private bool NewsExists(int id)

{

return \_context.News.Any(e => e.NewsId == id);

}

public async Task<IActionResult> Category(int id)

{

ViewBag.Categories = await \_context.Categories.ToListAsync();

var news = await \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.Where(n => n.CategoryId == id && n.IsApproved)

.ToListAsync();

return View("Index", news);

}

public async Task<IActionResult> Read(int? id)

{

ViewBag.Categories = await \_context.Categories.ToListAsync();

if (id == null)

{

return NotFound();

}

var news = await \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.FirstOrDefaultAsync(m => m.NewsId == id);

if (news == null)

{

return NotFound();

}

return View(news);

}

[HttpPost]

public async Task<IActionResult> Approve(int id)

{

var news = await \_context.News.FindAsync(id);

if (news == null)

{

return NotFound();

}

news.IsApproved = true;

\_context.Update(news);

await \_context.SaveChangesAsync();

TempData["Success"] = "News article approved successfully.";

return RedirectToAction(nameof(Details), new { id = news.NewsId });

}

[Authorize(Roles = "User")]

public async Task<IActionResult> MyNews()

{

var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

if (string.IsNullOrEmpty(userId) || !int.TryParse(userId, out int authorId))

{

return RedirectToAction("Login", "Account");

}

var myNews = await \_context.News

.Where(n => n.AuthorId == authorId && !n.IsDeleted)

.Include(n => n.Category)

.ToListAsync();

ViewBag.HideNavElements = true; // Hide categories and search bar

return View(myNews);

}

[HttpPost]

public async Task<IActionResult> Archive(int id)

{

var newsItem = await \_context.News.FirstOrDefaultAsync(n => n.NewsId == id);

if (newsItem != null && !newsItem.IsApproved)

{

newsItem.IsArchived = true; // Mark as archived

\_context.Update(newsItem);

await \_context.SaveChangesAsync();

}

return RedirectToAction("MyNews");

}

[HttpPost]

public async Task<IActionResult> Repost(int id)

{

var newsItem = await \_context.News.FindAsync(id);

if (newsItem != null)

{

newsItem.IsApproved = false;

newsItem.IsDeleted = false; // Ensure it is not marked as deleted

newsItem.IsArchived = false; // Ensure it is not marked as archived

newsItem.CreatedAt = DateTime.Now; // Update the created date to the current date

\_context.Update(newsItem);

await \_context.SaveChangesAsync();

return Json(new { success = true, message = "Đăng lại thành công" });

}

return Json(new { success = false, message = "Đăng lại thất bại: Tin tức không tồn tại" });

}

}

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - WebBaoDienTu</title>

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css" integrity="sha512-DTOQO9RWCH3ppGqcWaEA1BIZOC6xxalwEsw9c2QQeAIftl+Vegovlnee1c9QX4TctnWMn13TZye+giMm8e2LwA==" crossorigin="anonymous" referrerpolicy="no-referrer" />

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap" rel="stylesheet">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;700&display=swap" rel="stylesheet">

<link rel="stylesheet" href="~/css/layout/title.css" asp-append-version="true" />

<link rel="stylesheet" href="~/css/layout/auth.css" asp-append-version="true" />

<link rel="stylesheet" href="~/css/layout/notification.css" asp-append-version="true" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-lg navbar-light">

<div class="container">

<!-- Logo và tiêu đề bên trái -->

<div class="d-flex align-items-center me-auto">

<a class="navbar-brand me-2" asp-controller="Home" asp-action="Index">

<img src="~/images/logo.png" alt="Logo" />

</a>

@if (ViewBag.HideNavElements != true)

{

<ul class="navbar-nav flex-row">

@if (ViewBag.Categories is IEnumerable<Category> categories)

{

foreach (var category in categories)

{

<li class="nav-item mx-1">

<a class="nav-link" asp-area="" asp-controller="Home" asp-action="Category" asp-route-id="@category.CategoryId">

@category.CategoryName

</a>

</li>

}

}

</ul>

}

</div>

<!-- Thanh tìm kiếm ở giữa -->

@if (ViewBag.HideNavElements != true)

{

<div class="input-group">

<form class="d-flex" asp-controller="Home" asp-action="Search" method="get">

<input class="form-control" type="search" name="query" placeholder="Nhập vào đây để tìm kiếm..." aria-label="Search">

<button class="btn btn-search" type="submit">

<i class="fas fa-magnifying-glass"></i>

</button>

</form>

</div>

}

<!-- Menu phải -->

<div class="d-flex align-items-center">

@if (User.Identity != null && User.Identity.IsAuthenticated)

{

<!-- Chuông thông báo -->

<div class="nav-item dropdown me-5 position-relative notification-wrapper">

<a class="nav-link position-relative" href="#" id="notificationDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

<i class="fas fa-bell"></i>

<span class="badge bg-danger notification-count" id="notification-badge" style="display: none;">0</span>

</a>

<div class="dropdown-menu dropdown-menu-end notification-dropdown" aria-labelledby="notificationDropdown">

<div class="dropdown-header d-flex justify-content-between align-items-center">

<span>Thông báo</span>

<button class="btn btn-sm btn-link" id="markAllAsRead">Đánh dấu đã đọc</button>

</div>

<div class="dropdown-divider"></div>

<div id="notification-container">

<div class="dropdown-item text-center text-muted" id="no-notifications">Không có thông báo</div>

</div>

</div>

</div>

<!-- Dropdown tài khoản căn phải -->

<div class="nav-item dropdown ms-auto">

<a class="nav-link dropdown-toggle" href="#" id="userDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

<span class="username">@(User.Claims.FirstOrDefault(c => c.Type == "DisplayName")?.Value ?? User.Claims.FirstOrDefault(c => c.Type == "FullName")?.Value ?? User.Identity.Name)</span>

@{

var random = new Random();

var icons = new[] { "fa-user-gear", "fa-user-circle", "fa-user-astronaut", "fa-user-tie", "fa-user-secret" };

var selectedIcon = icons[random.Next(icons.Length)];

}

<i class="fas @selectedIcon ms-2"></i>

</a>

<ul class="dropdown-menu dropdown-menu-end" aria-labelledby="userDropdown">

@if (User.IsInRole("Admin"))

{

<li><a class="dropdown-item" asp-controller="News" asp-action="Index"><i class="fas fa-list"></i> Danh sách tin tức</a></li>

}

@if (User.IsInRole("User"))

{

<li><a class="dropdown-item" href="/News/Create"><i class="fas fa-plus"></i> Đăng tin mới</a></li>

<li><a class="dropdown-item" asp-controller="News" asp-action="MyNews"><i class="fas fa-history"></i> Lịch sử tin tức</a></li>

}

<li><a class="dropdown-item" asp-controller="Sharing" asp-action="MyShares"><i class="fas fa-share-alt"></i> Tin đã chia sẻ</a></li>

<li><hr class="dropdown-divider"></li>

<li><a class="dropdown-item" href="javascript:void(0);" onclick="logoutUser()"><i class="fas fa-sign-out-alt"></i> Đăng xuất</a></li>

</ul>

</div>

}

else

{

<!-- Đăng ký/Đăng nhập bên phải -->

<ul class="navbar-nav auth-links flex-row ms-auto">

<li class="nav-item mx-2">

<a class="nav-link" href="#" data-bs-toggle="modal" data-bs-target="#registerModal">

<i class="fas fa-user-plus me-1"></i> Đăng ký

</a>

</li>

<li class="nav-item mx-2">

<a class="nav-link" href="#" data-bs-toggle="modal" data-bs-target="#loginModal">

<i class="fas fa-right-to-bracket me-1"></i> Đăng nhập

</a>

</li>

</ul>

}

</div>

</div>

</nav>

<!-- Các modal giữ nguyên -->

<!-- Login Modal -->

<div class="modal fade" id="loginModal" tabindex="-1" aria-labelledby="loginModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered">

<div class="modal-content login-modal">

<div class="modal-header">

<h5 class="modal-title" id="loginModalLabel">

<i class="fas fa-door-open me-2"></i> Đăng nhập

</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<p class="text-muted mb-3">Vui lòng nhập thông tin để truy cập tài khoản của bạn.</p>

<form id="loginForm">

<div class="alert alert-danger" id="loginErrorMessage" style="display: none;"></div>

<div class="mb-3 form-group">

<label for="loginEmail" class="form-label"><i class="fas fa-envelope me-1"></i> Email</label>

<input type="email" class="form-control" id="loginEmail" placeholder="Nhập email của bạn" required>

</div>

<div class="mb-3 form-group">

<label for="loginPassword" class="form-label"><i class="fas fa-lock me-1"></i> Mật khẩu</label>

<input type="password" class="form-control" id="loginPassword" placeholder="Nhập mật khẩu" required>

</div>

<div class="mb-3 form-check">

<input type="checkbox" class="form-check-input" id="rememberMe">

<label class="form-check-label" for="rememberMe">Ghi nhớ đăng nhập</label>

</div>

<button type="submit" class="btn btn-primary w-100 btn-login">

<i class="fas fa-arrow-right me-2"></i> Đăng nhập

</button>

</form>

</div>

<div class="modal-footer">

<small class="text-muted">Chưa có tài khoản? <a href="#" data-bs-toggle="modal" data-bs-target="#registerModal" data-bs-dismiss="modal">Đăng ký ngay</a></small>

</div>

</div>

</div>

</div>

<!-- Register Modal -->

<div class="modal fade" id="registerModal" tabindex="-1" aria-labelledby="registerModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered">

<div class="modal-content register-modal">

<div class="modal-header">

<h5 class="modal-title" id="registerModalLabel">

<i class="fas fa-user-check me-2"></i> Đăng ký

</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<p class="text-muted mb-3">Tạo tài khoản để tham gia ngay hôm nay!</p>

<form id="registerForm">

<div class="alert alert-danger" id="registerErrorMessage" style="display: none;"></div>

<div class="mb-3 form-group">

<label for="registerFullName" class="form-label"><i class="fas fa-user me-1"></i> Họ và tên</label>

<input type="text" class="form-control" id="registerFullName" placeholder="Nhập họ và tên" required>

</div>

<div class="mb-3 form-group">

<label for="registerEmail" class="form-label"><i class="fas fa-envelope me-1"></i> Email</label>

<input type="email" class="form-control" id="registerEmail" placeholder="Nhập email của bạn" required>

</div>

<div class="mb-3 form-group">

<label for="registerPassword" class="form-label"><i class="fas fa-lock me-1"></i> Mật khẩu</label>

<input type="password" class="form-control" id="registerPassword" placeholder="Tạo mật khẩu" required>

</div>

<div class="mb-3 form-group">

<label for="confirmPassword" class="form-label"><i class="fas fa-lock me-1"></i> Xác nhận mật khẩu</label>

<input type="password" class="form-control" id="confirmPassword" placeholder="Nhập lại mật khẩu" required>

</div>

<button type="submit" class="btn btn-primary w-100 btn-register">

<i class="fas fa-user-check me-2"></i> Đăng ký

</button>

</form>

</div>

<div class="modal-footer">

<small class="text-muted">Đã có tài khoản? <a href="#" data-bs-toggle="modal" data-bs-target="#loginModal" data-bs-dismiss="modal">Đăng nhập ngay</a></small>

</div>

</div>

</div>

</div>

<!-- Success Modal -->

<div class="modal fade" id="successModal" tabindex="-1" aria-labelledby="successModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered">

<div class="modal-content success-modal">

<div class="modal-header">

<h5 class="modal-title" id="successModalLabel">

<i class="fas fa-check-circle me-2"></i> Thành công

</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body text-center">

<i class="fas fa-check-circle fa-4x text-success mb-3 animate\_\_animated animate\_\_bounceIn"></i>

<h4 id="successMessage" class="fw-bold text-success mb-3" style="font-family: 'Poppins', sans-serif;"></h4>

<p class="text-muted">Cảm ơn bạn đã sử dụng dịch vụ của chúng tôi!</p>

<button type="button" class="btn btn-success btn-lg" data-bs-dismiss="modal">

<i class="fas fa-thumbs-up me-2"></i> Tuyệt vời!

</button>

</div>

</div>

</div>

</div>

<!-- Debug Modal -->

<div class="modal fade" id="debugModal" tabindex="-1" aria-labelledby="debugModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered modal-lg">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="debugModalLabel">

<i class="fas fa-bug me-2"></i> Debug Information

</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<div id="debugContent" class="p-3 bg-light">

<div>Đang tải thông tin debug...</div>

</div>

</div>

</div>

</div>

</div>

</header>

@Html.AntiForgeryToken()

<div class="container">

<main role="main" class="pb-3">

<div class="container">

@if (TempData["SuccessMessage"] != null)

{

<div class="alert alert-success alert-dismissible fade show" role="alert">

<i class="fas fa-check-circle me-2"></i> @TempData["SuccessMessage"]

<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>

</div>

}

@if (TempData["ErrorMessage"] != null)

{

<div class="alert alert-danger alert-dismissible fade show" role="alert">

<i class="fas fa-exclamation-triangle me-2"></i> @TempData["ErrorMessage"]

<button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>

</div>

}

@RenderBody()

</div>

</main>

</div>

<footer class="border-top footer text-muted py-3">

<div class="container">

© 2025 - WebBaoDienTu - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</div>

</footer>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

<script>

function initializeDropdown() {

console.log("Dropdown initialized");

$('.dropdown-toggle').dropdown();

}

function logoutUser() {

$.ajax({

type: "POST",

url: "/api/Auth/logout",

headers: {

"RequestVerificationToken": $('input[name="\_\_RequestVerificationToken"]').val()

},

success: function(response) {

if (response.success) {

window.location.href = "/";

} else {

alert("Đăng xuất thất bại: " + (response.message || "Lỗi không xác định"));

}

},

error: function(xhr) {

alert("Có lỗi xảy ra khi đăng xuất: " + (xhr.responseJSON?.message || xhr.statusText));

}

});

}

$(document).ready(function() {

initializeDropdown();

$("#loginForm").submit(function(e) {

e.preventDefault();

$("#loginErrorMessage").hide();

$(this).find('button[type="submit"]').html('<span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span> Đang xử lý...');

$(this).find('button[type="submit"]').prop('disabled', true);

var formData = new FormData();

formData.append('Email', $("#loginEmail").val());

formData.append('Password', $("#loginPassword").val());

formData.append('RememberMe', $("#rememberMe").is(":checked"));

$.ajax({

type: "POST",

url: "/api/Auth/login",

data: formData,

processData: false,

contentType: false,

cache: false,

success: function(response) {

if (response.success) {

window.location.reload();

} else {

$("#loginErrorMessage").text(response.message).show();

$("#loginForm").find('button[type="submit"]').html('<i class="fas fa-arrow-right me-2"></i> Đăng nhập');

$("#loginForm").find('button[type="submit"]').prop('disabled', false);

}

},

error: function(xhr) {

var errorMsg = "Có lỗi xảy ra. Vui lòng thử lại sau.";

if (xhr.responseJSON && xhr.responseJSON.message) {

errorMsg = xhr.responseJSON.message;

}

$("#loginErrorMessage").text(errorMsg).show();

$("#loginForm").find('button[type="submit"]').html('<i class="fas fa-arrow-right me-2"></i> Đăng nhập');

$("#loginForm").find('button[type="submit"]').prop('disabled', false);

}

});

});

$("#registerForm").submit(function(e) {

e.preventDefault();

$("#registerErrorMessage").hide();

$(this).find('button[type="submit"]').html('<span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span> Đang xử lý...');

$(this).find('button[type="submit"]').prop('disabled', true);

var password = $("#registerPassword").val();

var passwordRegex = /^(?=.\*[A-Z])(?=.\*\d).{6,8}$/;

if (!passwordRegex.test(password)) {

$("#registerErrorMessage").text("Mật khẩu phải từ 6-8 ký tự, chứa ít nhất một chữ cái in hoa và một số.").show();

$(this).find('button[type="submit"]').html('<i class="fas fa-user-check me-2"></i> Đăng ký');

$(this).find('button[type="submit"]').prop('disabled', false);

return;

}

if ($("#registerPassword").val() !== $("#confirmPassword").val()) {

$("#registerErrorMessage").text("Mật khẩu xác nhận không khớp.").show();

$(this).find('button[type="submit"]').html('<i class="fas fa-user-check me-2"></i> Đăng ký');

$(this).find('button[type="submit"]').prop('disabled', false);

return;

}

var formData = new FormData();

formData.append('FullName', $("#registerFullName").val());

formData.append('Email', $("#registerEmail").val());

formData.append('Password', $("#registerPassword").val());

formData.append('ConfirmPassword', $("#confirmPassword").val());

$.ajax({

type: "POST",

url: "/api/Auth/register",

data: formData,

processData: false,

contentType: false,

cache: false,

success: function(response) {

if (response.success) {

$('#successMessage').text(response.message);

$('#successModal').modal('show');

setTimeout(function() {

$('#successModal').modal('hide'); // Ẩn modal thành công sau 2 giây

$('#registerModal').modal('hide'); // Ẩn modal đăng ký

$('#loginModal').modal('show'); // Hiển thị modal đăng nhập

}, 2000);

$("#registerForm")[0].reset(); // Reset form đăng ký

} else {

$("#registerErrorMessage").text(response.message).show();

$(this).find('button[type="submit"]').html('<i class="fas fa-user-check me-2"></i> Đăng ký');

$(this).find('button[type="submit"]').prop('disabled', false);

}

},

error: function(xhr) {

var errorMsg = xhr.responseJSON?.message || "Có lỗi xảy ra. Vui lòng thử lại sau.";

$("#registerErrorMessage").text(errorMsg).show();

$("#registerForm").find('button[type="submit"]').html('<i class="fas fa-user-check me-2"></i> Đăng ký');

$("#registerForm").find('button[type="submit"]').prop('disabled', false);

}

});

});

// Add this to your existing JavaScript, inside the document ready function:

var lastNotificationCount = 0;

function loadNotifications() {

console.log("Bắt đầu tải thông báo...");

$.ajax({

type: "GET",

url: "/Sharing/GetNotifications",

success: function(response) {

console.log("Kết quả API notifications:", response);

if (response.success) {

var notifications = response.notifications || [];

// Sort notifications: unread first, then by date

notifications.sort(function(a, b) {

if (a.isRead !== b.isRead) {

return a.isRead ? 1 : -1;

}

return new Date(b.shareDate) - new Date(a.shareDate);

});

var unreadCount = notifications.filter(n => !n.isRead).length;

var $notificationBadge = $('#notification-badge');

var $notificationContainer = $('#notification-container');

var $noNotifications = $('#no-notifications');

var $bellIcon = $('.fas.fa-bell');

// Trigger bell animation if unread count increases

if (unreadCount > lastNotificationCount && lastNotificationCount !== 0) {

$bellIcon.removeClass('bell-animate'); // Reset animation

void $bellIcon[0].offsetWidth; // Force reflow to restart animation

$bellIcon.addClass('bell-animate'); // Apply animation

}

lastNotificationCount = unreadCount;

// Update badge visibility

if (unreadCount > 0) {

$notificationBadge.text(unreadCount).show();

$noNotifications.hide();

} else {

$notificationBadge.hide();

if (notifications.length === 0) {

$noNotifications.show();

} else {

$noNotifications.hide();

}

}

$notificationContainer.find('.notification-item').remove();

if (notifications.length > 0) {

notifications.forEach(function(notification) {

var date = new Date(notification.shareDate);

var formattedDate = new Intl.DateTimeFormat('vi-VN', {

day: '2-digit',

month: '2-digit',

year: 'numeric',

hour: '2-digit',

minute: '2-digit'

}).format(date);

var itemClass = notification.isRead ? '' : 'unread';

var messageHtml = notification.message ?

`<small class="text-muted">"${notification.message}"</small><br>` : '';

var notificationHtml = `

<a href="/News/Read/${notification.newsId}" class="dropdown-item notification-item ${itemClass}" data-id="${notification.id}">

<div class="d-flex w-100 justify-content-between">

<h6 class="mb-1 text-truncate">${notification.title}</h6>

<small>${formattedDate}</small>

</div>

${messageHtml}

<small class="text-primary">Nhấn để xem bài viết</small>

</a>

`;

$notificationContainer.append(notificationHtml);

});

$noNotifications.hide();

} else {

$noNotifications.show();

}

$('.notification-item').on('click', function() {

var notificationId = $(this).data('id');

markNotificationAsRead(notificationId);

});

} else {

console.error("Lỗi khi tải thông báo:", response.message);

}

},

error: function(xhr, status, error) {

console.error("AJAX error:", status, error);

console.error("Response:", xhr.responseText);

}

});

}

function markNotificationAsRead(notificationId) {

$.ajax({

type: "POST",

url: "/Sharing/MarkAsRead",

data: { notificationId: notificationId },

headers: {

"RequestVerificationToken": $('input[name="\_\_RequestVerificationToken"]').val()

},

success: function(response) {

if (response.success) {

console.log("Đã đánh dấu thông báo đã đọc:", notificationId);

}

},

error: function(xhr) {

console.error("Lỗi khi đánh dấu thông báo là đã đọc:", xhr.responseText);

}

});

}

$("#markAllAsRead").click(function(e) {

e.preventDefault();

e.stopPropagation();

$.ajax({

type: "POST",

url: "/Sharing/MarkAllAsRead",

headers: {

"RequestVerificationToken": $('input[name="\_\_RequestVerificationToken"]').val()

},

success: function(response) {

if (response.success) {

loadNotifications();

}

}

});

});

if ($('#notificationDropdown').length) {

loadNotifications();

setInterval(loadNotifications, 30000);

}

});

</script>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using WebBaoDienTu.Models;

namespace WebBaoDienTu.Controllers

{

public class NewsController : Controller

{

private readonly BaoDienTuContext \_context;

public NewsController(BaoDienTuContext context)

{

\_context = context;

}

// GET: News

// GET: News

public async Task<IActionResult> Index()

{

var userRole = User.FindFirstValue(ClaimTypes.Role);

ViewBag.Categories = await \_context.Categories.ToListAsync();

ViewBag.HideNavElements = true; // Hide categories and search bar

if (userRole == "Admin")

{

var news = \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.Where(n => !n.IsDeleted && (!n.IsArchived || n.IsApproved))

.OrderByDescending(n => !n.IsApproved)

.ThenByDescending(n => n.CreatedAt);

return View("Index", await news.ToListAsync());

}

else if (userRole == "User")

{

return RedirectToAction("Create");

}

return RedirectToAction("Login", "Account");

}

// GET: News/Details/5

public async Task<IActionResult> Details(int? id)

{

if (!User.IsInRole("Admin"))

{

return Forbid();

}

ViewBag.Categories = await \_context.Categories.ToListAsync();

if (id == null)

{

return NotFound();

}

var news = await \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.FirstOrDefaultAsync(m => m.NewsId == id);

if (news == null)

{

return NotFound();

}

return View(news);

}

public async Task<IActionResult> GetNewsDetailsForDelete(int id)

{

var news = await \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.FirstOrDefaultAsync(m => m.NewsId == id);

if (news == null)

{

return NotFound();

}

return Json(new

{

title = news.Title,

createdAt = news.CreatedAt.ToString("dd/MM/yyyy HH:mm"),

authorFullName = news.Author.FullName,

imageUrl = news.ImageUrl

});

}

[Authorize(Roles = "User,Admin")]

public async Task<IActionResult> Create()

{

ViewBag.Categories = await \_context.Categories.ToListAsync();

ViewData["CategoryId"] = new SelectList(\_context.Categories, "CategoryId", "CategoryName");

ViewBag.HideNavElements = true; // Hide categories and search bar

return View();

}

[Authorize(Roles = "User,Admin")]

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Title,Content,ImageUrl,CategoryId")] News news, IFormFile ImageFile)

{

ViewData["CategoryId"] = new SelectList(\_context.Categories, "CategoryId", "CategoryName");

try

{

// Lấy AuthorId từ người dùng hiện tại

var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

if (string.IsNullOrEmpty(userId) || !int.TryParse(userId, out int authorId))

{

return Json(new { success = false, message = "Không thể xác định người dùng hiện tại. Vui lòng đăng nhập lại." });

}

// Gán AuthorId

news.AuthorId = authorId;

// Xóa lỗi validation cho AuthorId nếu có (vì nó không được gửi từ form)

ModelState.Remove("AuthorId");

// Kiểm tra dữ liệu

if (string.IsNullOrEmpty(news.Title))

{

ModelState.AddModelError("Title", "Tiêu đề không được để trống.");

}

if (string.IsNullOrEmpty(news.Content))

{

ModelState.AddModelError("Content", "Nội dung không được để trống.");

}

if (news.CategoryId <= 0)

{

ModelState.AddModelError("CategoryId", "Vui lòng chọn danh mục.");

}

// Debug dữ liệu gửi lên

Console.WriteLine($"Title: {news.Title}");

Console.WriteLine($"Content: {news.Content}");

Console.WriteLine($"CategoryId: {news.CategoryId}");

Console.WriteLine($"ImageUrl: {news.ImageUrl}");

if (!ModelState.IsValid)

{

var errors = ModelState.Values.SelectMany(v => v.Errors).Select(e => e.ErrorMessage).ToList();

return Json(new { success = false, message = string.Join("; ", errors) });

}

// Xử lý file ảnh nếu có

if (ImageFile != null && ImageFile.Length > 0)

{

string fileExtension = Path.GetExtension(ImageFile.FileName).ToLower();

string[] allowedExtensions = { ".jpg", ".jpeg", ".png", ".gif" };

if (!allowedExtensions.Contains(fileExtension))

{

return Json(new { success = false, message = "Chỉ được tải lên ảnh JPG, JPEG, PNG hoặc GIF." });

}

string uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/images");

if (!Directory.Exists(uploadsFolder)) Directory.CreateDirectory(uploadsFolder);

string uniqueFileName = Guid.NewGuid().ToString() + fileExtension;

string filePath = Path.Combine(uploadsFolder, uniqueFileName);

using (var fileStream = new FileStream(filePath, FileMode.Create))

{

await ImageFile.CopyToAsync(fileStream);

}

news.ImageUrl = "/images/" + uniqueFileName;

}

news.CreatedAt = DateTime.Now;

news.IsApproved = User.IsInRole("Admin");

news.IsDeleted = false;

\_context.Add(news);

await \_context.SaveChangesAsync();

return Json(new { success = true, message = "Tin tức đã được đăng thành công!", redirectTo = Url.Action("Index") });

}

catch (Exception ex)

{

Console.WriteLine($"Exception in Create: {ex.Message}");

Console.WriteLine($"Stack Trace: {ex.StackTrace}");

if (ex.InnerException != null)

{

Console.WriteLine($"Inner Exception: {ex.InnerException.Message}");

}

return Json(new { success = false, message = $"Có lỗi xảy ra: {ex.Message}" });

}

}

// GET: News/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (!User.IsInRole("Admin"))

{

return Forbid();

}

ViewBag.Categories = await \_context.Categories.ToListAsync();

if (id == null)

{

return NotFound();

}

var news = await \_context.News.FindAsync(id);

if (news == null)

{

return NotFound();

}

ViewData["AuthorId"] = new SelectList(\_context.Users, "UserId", "UserId", news.AuthorId);

ViewData["CategoryId"] = new SelectList(\_context.Categories, "CategoryId", "CategoryName", news.CategoryId);

return View(news);

}

// POST: News/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("NewsId,Title,Content,ImageUrl,AuthorId,CategoryId,IsApproved,CreatedAt")] News news, IFormFile ImageFile)

{

if (!User.IsInRole("Admin"))

{

return Forbid();

}

ViewBag.Categories = await \_context.Categories.ToListAsync();

ModelState.Clear();

if (id != news.NewsId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

if (ImageFile != null && ImageFile.Length > 0)

{

string fileExtension = Path.GetExtension(ImageFile.FileName).ToLower();

string[] allowedExtensions = { ".jpg", ".jpeg", ".png", ".gif" };

if (!allowedExtensions.Contains(fileExtension))

{

ModelState.AddModelError("ImageFile", "Chỉ được tải lên ảnh JPG, JPEG, PNG hoặc GIF.");

ViewBag.Categories = new SelectList(\_context.Categories, "Id", "Name");

return View(news);

}

string uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot/images");

if (!Directory.Exists(uploadsFolder)) Directory.CreateDirectory(uploadsFolder);

string uniqueFileName = Guid.NewGuid().ToString() + fileExtension;

string filePath = Path.Combine(uploadsFolder, uniqueFileName);

using (var fileStream = new FileStream(filePath, FileMode.Create))

{

await ImageFile.CopyToAsync(fileStream);

}

news.ImageUrl = "/images/" + uniqueFileName;

}

\_context.Update(news);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!NewsExists(news.NewsId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["AuthorId"] = new SelectList(\_context.Users, "UserId", "UserId", news.AuthorId);

ViewData["CategoryId"] = new SelectList(\_context.Categories, "CategoryId", "CategoryName", news.CategoryId);

return View(news);

}

[HttpPost]

[Route("News/Delete/{id}")]

public IActionResult Delete(int id)

{

var newsItem = \_context.News.FirstOrDefault(n => n.NewsId == id);

if (newsItem != null)

{

newsItem.IsDeleted = true;

\_context.Update(newsItem);

\_context.SaveChanges();

return Json(new { success = true, message = "Xóa thành công" });

}

return Json(new { success = false, message = "Xóa thất bại: Tin tức không tồn tại" });

}

[HttpPost]

[Route("News/DeleteConfirmed/{id}")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var news = await \_context.News.FindAsync(id);

if (news != null)

{

if (User.IsInRole("Admin"))

{

\_context.News.Remove(news);

}

else

{

news.IsDeleted = true;

\_context.Update(news);

}

await \_context.SaveChangesAsync();

return Json(new { success = true, message = "Xóa tin tức thành công" });

}

return Json(new { success = false, message = "Không tìm thấy tin tức" });

}

private bool NewsExists(int id)

{

return \_context.News.Any(e => e.NewsId == id);

}

public async Task<IActionResult> Category(int id)

{

ViewBag.Categories = await \_context.Categories.ToListAsync();

var news = await \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.Where(n => n.CategoryId == id && n.IsApproved)

.ToListAsync();

return View("Index", news);

}

public async Task<IActionResult> Read(int? id)

{

ViewBag.Categories = await \_context.Categories.ToListAsync();

if (id == null)

{

return NotFound();

}

var news = await \_context.News

.Include(n => n.Author)

.Include(n => n.Category)

.FirstOrDefaultAsync(m => m.NewsId == id);

if (news == null)

{

return NotFound();

}

return View(news);

}

[HttpPost]

public async Task<IActionResult> Approve(int id)

{

var news = await \_context.News.FindAsync(id);

if (news == null)

{

return NotFound();

}

news.IsApproved = true;

\_context.Update(news);

await \_context.SaveChangesAsync();

TempData["Success"] = "News article approved successfully.";

return RedirectToAction(nameof(Details), new { id = news.NewsId });

}

[Authorize(Roles = "User")]

public async Task<IActionResult> MyNews()

{

var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

if (string.IsNullOrEmpty(userId) || !int.TryParse(userId, out int authorId))

{

return RedirectToAction("Login", "Account");

}

var myNews = await \_context.News

.Where(n => n.AuthorId == authorId && !n.IsDeleted)

.Include(n => n.Category)

.ToListAsync();

ViewBag.HideNavElements = true; // Hide categories and search bar

return View(myNews);

}

[HttpPost]

public async Task<IActionResult> Archive(int id)

{

var newsItem = await \_context.News.FirstOrDefaultAsync(n => n.NewsId == id);

if (newsItem != null && !newsItem.IsApproved)

{

newsItem.IsArchived = true; // Mark as archived

\_context.Update(newsItem);

await \_context.SaveChangesAsync();

}

return RedirectToAction("MyNews");

}

[HttpPost]

public async Task<IActionResult> Repost(int id)

{

var newsItem = await \_context.News.FindAsync(id);

if (newsItem != null)

{

newsItem.IsApproved = false;

newsItem.IsDeleted = false; // Ensure it is not marked as deleted

newsItem.IsArchived = false; // Ensure it is not marked as archived

newsItem.CreatedAt = DateTime.Now; // Update the created date to the current date

\_context.Update(newsItem);

await \_context.SaveChangesAsync();

return Json(new { success = true, message = "Đăng lại thành công" });

}

return Json(new { success = false, message = "Đăng lại thất bại: Tin tức không tồn tại" });

}

}

}