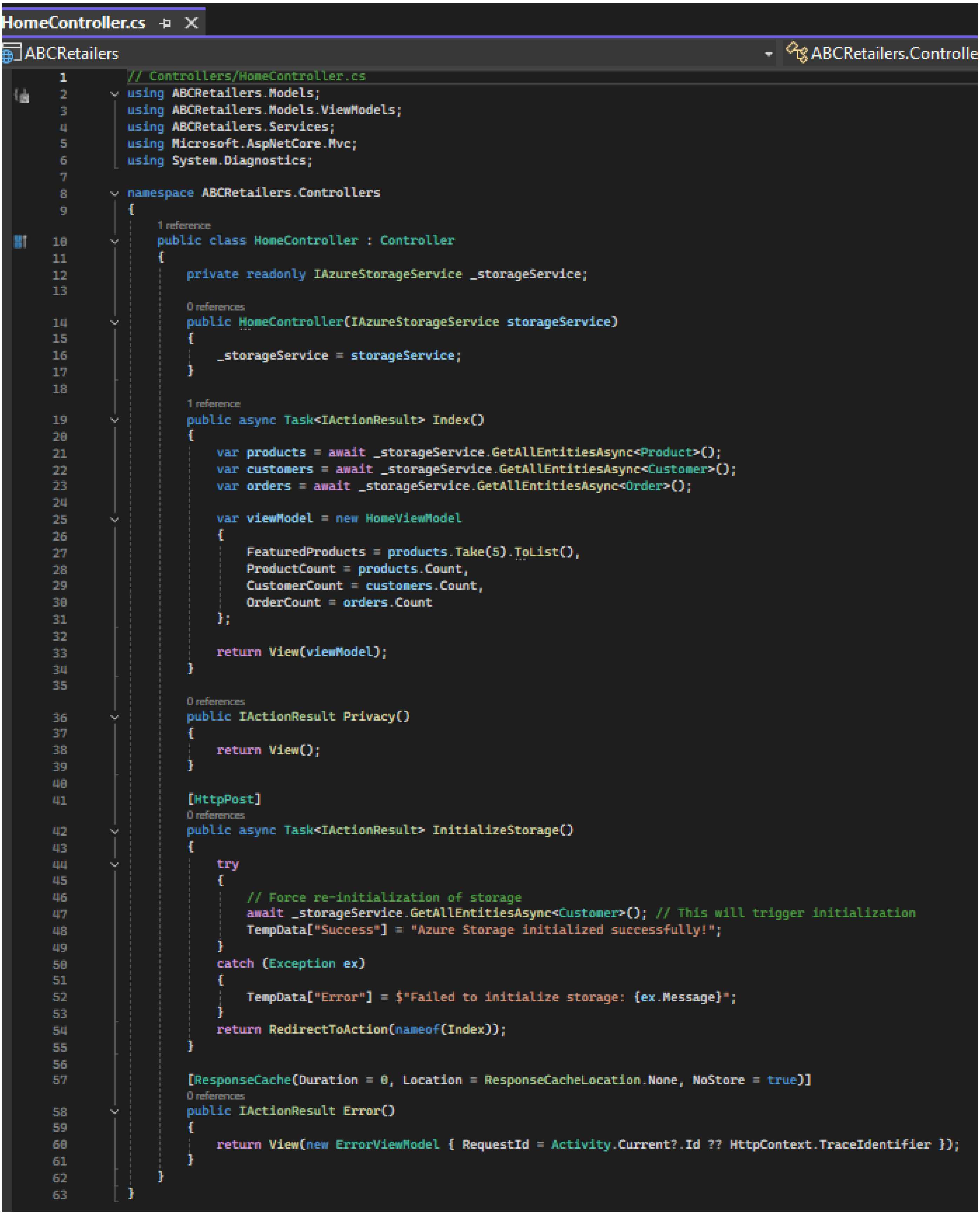
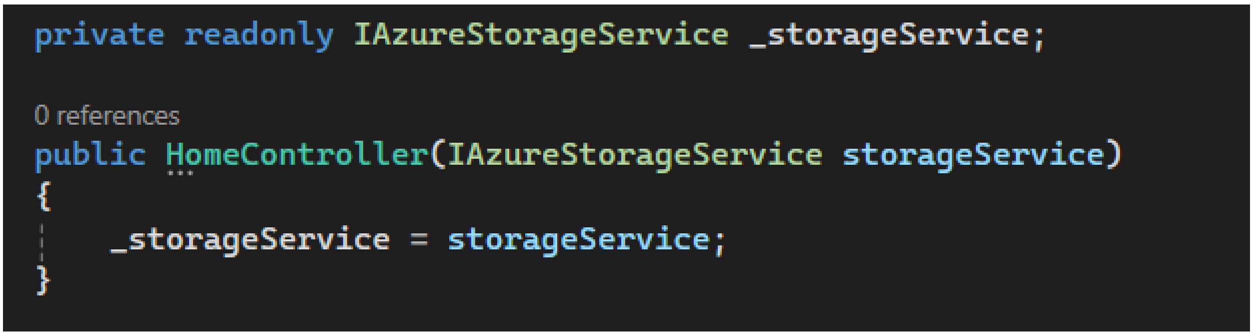
ABCRetailers Web Application - HomeController Code Snapshots

HomeController.cs (Full Code Snapshot)



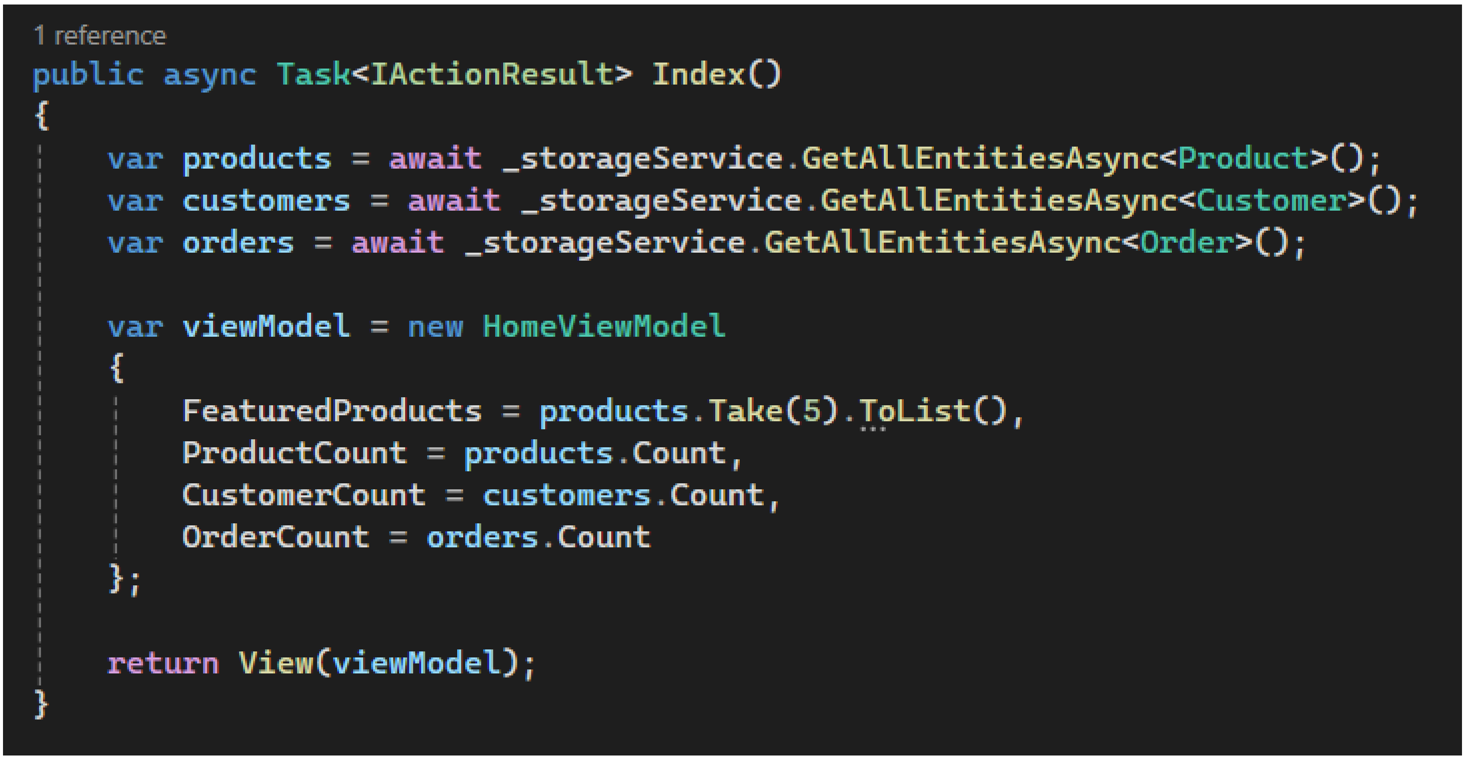
```csharp
// Controllers/HomeController.cs
using ABCRetailers.Models;
using ABCRetailers.Models.ViewModels;
using ABCRetailers.Services;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;

namespace ABCRetailers.Controllers
{
    1 reference
    public class HomeController : Controller
    {
        private readonly IAzureStorageService _storageService;

        0 references
        public HomeController(IAzureStorageService storageService)
        {
            _storageService = storageService;
        }

        1 reference
        public async Task<IActionResult> Index()
        {
            var products = await _storageService.GetAllEntitiesAsync<Product>();
            var customers = await _storageService.GetAllEntitiesAsync<Customer>();
            var orders = await _storageService.GetAllEntitiesAsync<Order>();

            var viewModel = new HomeViewModel
            {
                FeaturedProducts = products.Take(5).ToList(),
                ProductCount = products.Count,
                CustomerCount = customers.Count,
                OrderCount = orders.Count
            };

            return View(viewModel);
        }

        0 references
        public IActionResult Privacy()
        {
            return View();
        }

        [HttpPost]
        0 references
        public async Task<IActionResult> InitializeStorage()
        {
            try
            {
                // Force re-initialization of storage
                await _storageService.GetAllEntitiesAsync<Customer>(); // This will trigger initialization
                TempData["Success"] = "Azure Storage initialized successfully!";
            }
            catch (Exception ex)
            {
                TempData["Error"] = $"Failed to initialize storage: {ex.Message}";
            }
            return RedirectToAction(nameof(Index));
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        0 references
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}
```

HomeController.cs (IAzureStorageService Action Code Snapshot)



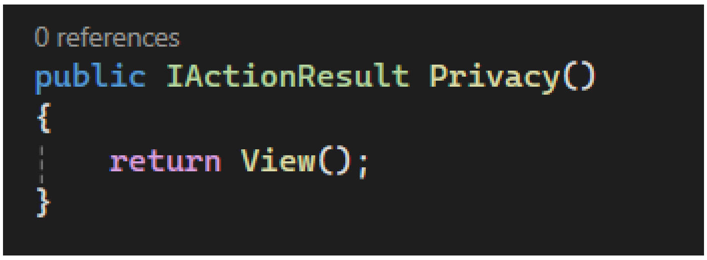
```
private readonly IAzureStorageService _storageService;

0 references
public HomeController(IAzureStorageService storageService)
{
    _storageService = storageService;
}
```

HomeController.cs (Index Action Code Snapshot)



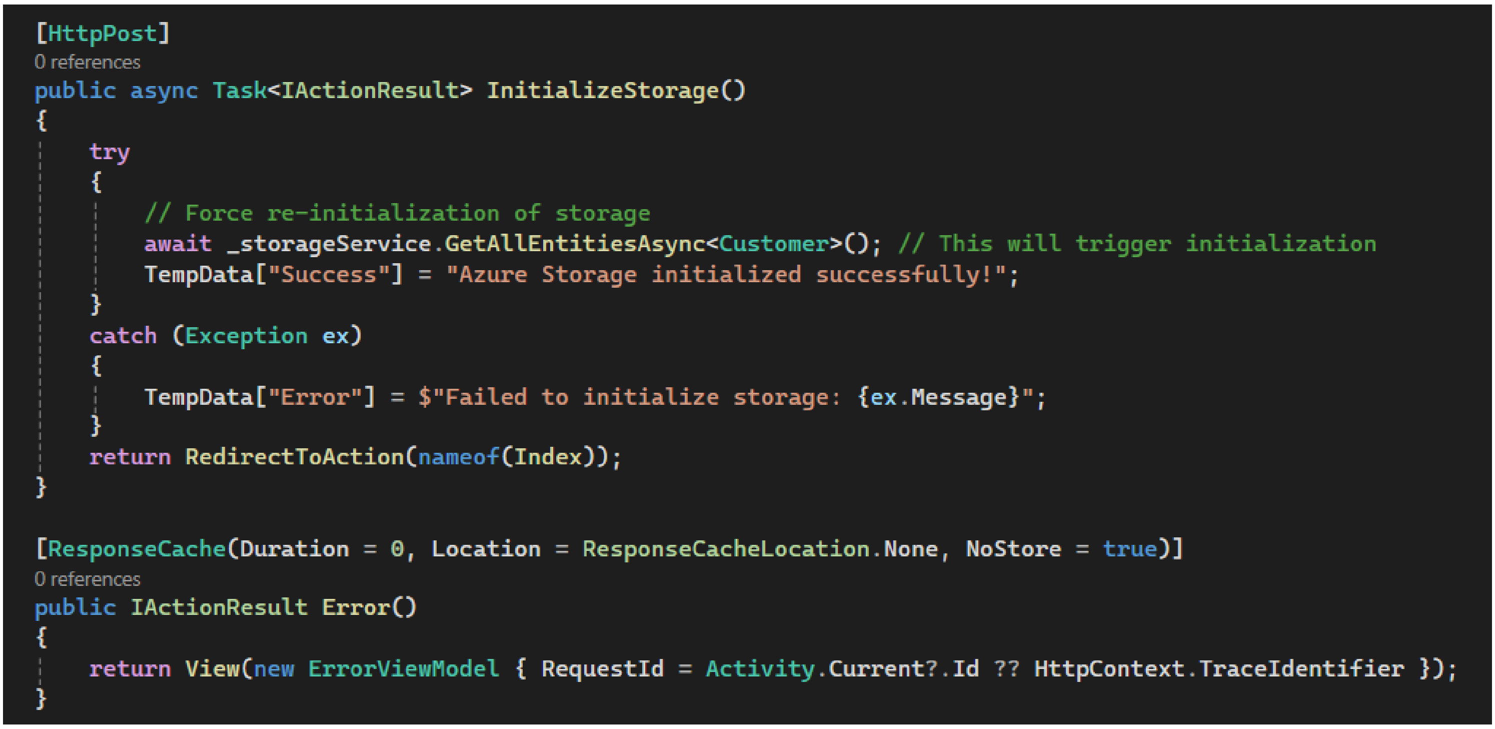
```
1 reference
public async Task<IActionResult> Index()
{
    var products = await _storageService.GetAllEntitiesAsync<Product>();
    var customers = await _storageService.GetAllEntitiesAsync<Customer>();
    var orders = await _storageService.GetAllEntitiesAsync<Order>();

    var viewModel = new HomeViewModel
    {
        FeaturedProducts = products.Take(5).ToList(),
        ProductCount = products.Count,
        CustomerCount = customers.Count,
        OrderCount = orders.Count
    };

    return View(viewModel);
}
```

HomeController.cs (Privacy Action Code Snapshot)



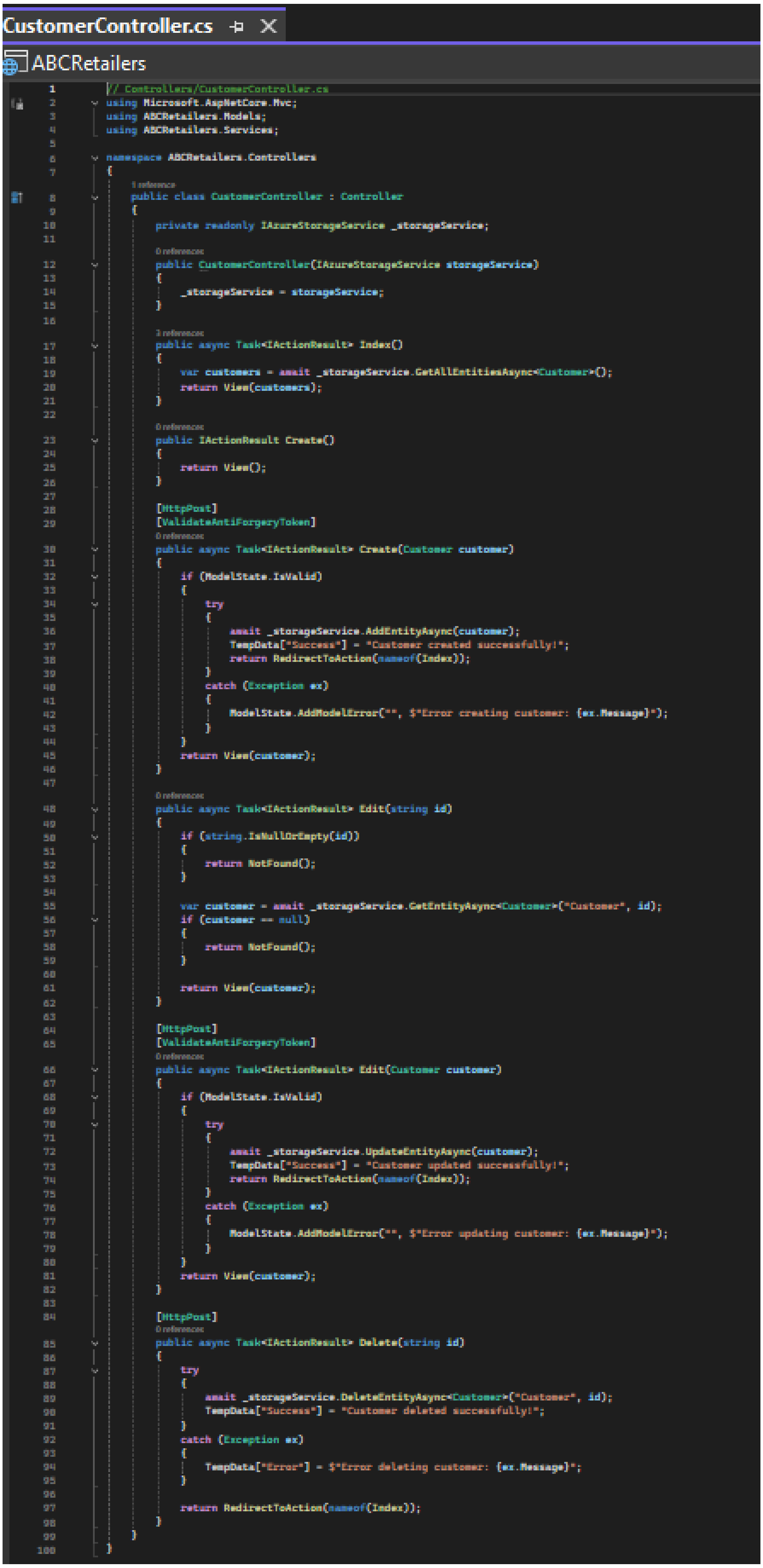
```
0 references
public IActionResult Privacy()
{
    return View();
}
```

HomeController.cs (InitializeStorage Action Code Snapshot)



```
[HttpPost]
0 references
public async Task<IActionResult> InitializeStorage()
{
    try
    {
        // Force re-initialization of storage
        await _storageService.GetAllEntitiesAsync<Customer>(); // This will trigger initialization
        TempData["Success"] = "Azure Storage initialized successfully!";
    }
    catch (Exception ex)
    {
        TempData["Error"] = $"Failed to initialize storage: {ex.Message}";
    }
    return RedirectToAction(nameof(Index));
}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
0 references
public IActionResult Error()
{
    return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
}
```

CustomerController.cs (Full Code Snapshot)



```csharp
// Controllers/CustomerController.cs
using Microsoft.AspNetCore.Mvc;
using ABCRetailers.Models;
using ABCRetailers.Services;

namespace ABCRetailers.Controllers
{
    public class CustomerController : Controller
    {
        private readonly IAzureStorageService _storageService;

        public CustomerController(IAzureStorageService storageService)
        {
            _storageService = storageService;
        }

        public async Task<IActionResult> Index()
        {
            var customers = await _storageService.GetAllEntitiesAsync<Customer>();
            return View(customers);
        }

        public IActionResult Create()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(Customer customer)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    await _storageService.AddEntityAsync(customer);
                    TempData["Success"] = "Customer created successfully!";
                    return RedirectToAction(nameof(Index));
                }
                catch (Exception ex)
                {
                    ModelState.AddModelError("", $"Error creating customer: {ex.Message}");
                }
            }
            return View(customer);
        }

        public async Task<IActionResult> Edit(string id)
        {
            if (string.IsNullOrEmpty(id))
            {
                return NotFound();
            }

            var customer = await _storageService.GetEntityAsync<Customer>("Customer", id);
            if (customer == null)
            {
                return NotFound();
            }

            return View(customer);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(Customer customer)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    await _storageService.UpdateEntityAsync(customer);
                    TempData["Success"] = "Customer updated successfully!";
                    return RedirectToAction(nameof(Index));
                }
                catch (Exception ex)
                {
                    ModelState.AddModelError("", $"Error updating customer: {ex.Message}");
                }
            }
            return View(customer);
        }

        [HttpPost]
        public async Task<IActionResult> Delete(string id)
        {
            try
            {
                await _storageService.DeleteEntityAsync<Customer>("Customer", id);
                TempData["Success"] = "Customer deleted successfully!";
            }
            catch (Exception ex)
            {
                TempData["Error"] = $"Error deleting customer: {ex.Message}";
            }

            return RedirectToAction(nameof(Index));
        }
    }
}
```

ProductController.cs

OrderController.cs

UploadController.cs