**Part 11: API Versioning and Documentation**

**Step 1: Install API Versioning Package**

* **Action: Install the required NuGet package for API versioning.**
* **Command:**

**dotnet add package Microsoft.AspNetCore.Mvc.Versioning**

**Step 2: Configure API Versioning**

* **Action: Configure API versioning in your Program.cs.**
* **Code:**

**builder.Services.AddApiVersioning(options =>**

**{**

**options.DefaultApiVersion = new ApiVersion(1, 0); // Default to version 1.0**

**options.AssumeDefaultVersionWhenUnspecified = true;**

**options.ReportApiVersions = true; // Include version information in the response headers**

**options.ApiVersionReader = new UrlSegmentApiVersionReader(); // Read version from URL segment**

**});**

**Step 3: Version the Controllers**

* **Action: Update existing controllers to support versioning and introduce a new versioned controller.**
  + **For ProductsController:**

**using Microsoft.AspNetCore.Mvc;**

**[ApiVersion("1.0")]**

**[Route("api/v{version:apiVersion}/[controller]")]**

**[ApiController]**

**public class ProductsController : ControllerBase**

**{**

**// Existing code**

**}**

* **Create ProductsV2Controller in the Controllers folder:**

**using Microsoft.AspNetCore.Mvc;**

**[ApiVersion("2.0")]**

**[Route("api/v{version:apiVersion}/[controller]")]**

**[ApiController]**

**public class ProductsV2Controller : ControllerBase**

**{**

**/// <summary>**

**/// Gets products for API version 2.0.**

**/// </summary>**

**/// <returns>A message indicating this is version 2.0 of the Products API.</returns>**

**[HttpGet]**

**public ActionResult<string> GetProducts()**

**{**

**return "This is version 2.0 of the Products API.";**

**}**

**}**

**Step 4: Test API Versioning**

* **Action:**
  + **Access different versions:**
    - **Version 1.0: GET /api/v1/Products**
    - **Version 2.0: GET /api/v2/Products**
  + **Expected Result:**
    - **Check that each version returns the appropriate response.**
    - **Look for headers like api-supported-versions and api-deprecated-versions in responses.**

**Step 5: Install Swagger/OpenAPI for Documentation**

* **Action: Install the Swagger package for documentation.**
* **Command:**

**dotnet add package Swashbuckle.AspNetCore**

**Step 6: Configure Swagger**

* **Action: Add Swagger services and UI configuration in Program.cs. Include security definition for JWT.**
* **Code:**

**builder.Services.AddSwaggerGen(options =>**

**{**

**options.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo**

**{**

**Version = "1.0",**

**Title = "E-Commerce API",**

**Description = "API for managing an e-commerce platform (v1.0)"**

**});**

**options.SwaggerDoc("v2", new Microsoft.OpenApi.Models.OpenApiInfo**

**{**

**Version = "2.0",**

**Title = "E-Commerce API",**

**Description = "API for managing an e-commerce platform (v2.0)"**

**});**

**// JWT Security Definition**

**options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme**

**{**

**In = ParameterLocation.Header,**

**Description = "Please insert JWT with Bearer into field",**

**Name = "Authorization",**

**Type = SecuritySchemeType.ApiKey**

**});**

**// Apply the security globally if all endpoints require authentication**

**options.AddSecurityRequirement(new OpenApiSecurityRequirement**

**{**

**{**

**new OpenApiSecurityScheme**

**{**

**Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "Bearer" }**

**},**

**new string[] {}**

**}**

**});**

**});**

**var app = builder.Build();**

**if (app.Environment.IsDevelopment())**

**{**

**app.UseSwagger();**

**app.UseSwaggerUI(options =>**

**{**

**options.SwaggerEndpoint("/swagger/v1/swagger.json", "E-Commerce API v1.0");**

**options.SwaggerEndpoint("/swagger/v2/swagger.json", "E-Commerce API v2.0");**

**});**

**}**

**Step 7: Add Annotations to Enhance Documentation**

* **Action: Use XML comments to enhance Swagger documentation.**
* **Example:**

**[ApiVersion("1.0")]**

**[Route("api/v{version:apiVersion}/[controller]")]**

**[ApiController]**

**public class ProductsController : ControllerBase**

**{**

**/// <summary>**

**/// Retrieves all products.**

**/// </summary>**

**/// <returns>A list of products.</returns>**

**[HttpGet]**

**public ActionResult<IEnumerable<string>> GetProducts()**

**{**

**return new[] { "Product1", "Product2" };**

**}**

**/// <summary>**

**/// Retrieves a specific product by ID.**

**/// </summary>**

**/// <param name="id">The product ID.</param>**

**/// <returns>The requested product.</returns>**

**[HttpGet("{id}")]**

**public ActionResult<string> GetProduct(int id)**

**{**

**return $"Product {id}";**

**}**

**}**

**Step 8: Test the Documentation**

* **Action:**
  + **Run your application and navigate to the Swagger UI (e.g., http://localhost:5000/swagger).**
  + **Test different endpoints directly from the UI, switching between versions. Check if the JWT authentication button appears for secured endpoints.**

**Step 9: Future Enhancements**

* **Considerations:**
  + **Deprecation Notices: Use [ApiVersion("1.0")][ApiVersion("1.0", Deprecated = true)] for old versions.**
  + **Authentication Support: Now set up, you can further refine JWT handling or add other authentication schemes.**
  + **Custom Styling: Customize the Swagger UI with CSS or by adjusting the SwaggerUI options for branding.**