**1. Simple HTTP Request and Response Logger**

**Goal**: Create middleware to log incoming HTTP requests and responses.

**Steps:**

1. Create a new ASP.NET Core Web App.
2. In the Startup.cs or Program.cs file, add custom middleware to log HTTP requests and responses.

### ****2. Basic CRUD Application Using MVC****

**Goal**: Build an ASP.NET Core MVC app to manage a "Product Inventory."

#### ****Steps****:

1. **Model**: Create a Product class.
2. **Controller**: Create a ProductController to handle CRUD operations.
3. **Views**: Create views for listing, adding, editing, and deleting products.

### ****3. RESTful API for Task Management****

**Goal**: Build a REST API for managing tasks using ASP.NET Core.

#### ****Steps****:

1. Create a new ASP.NET Core Web API project.
2. Add a Task model and a TaskController.

### ****4. Blog Application with ASP.NET Core MVC****

**Goal**: Build an MVC-based blog application with basic CRUD operations for posts.

#### ****Steps****:

1. **Model**: Create a BlogPost class.
2. **Controller**: Create a BlogController to handle CRUD operations.
3. **Views**: Create views for listing, creating, editing, and viewing blog posts.

#### ****Features****:

* List all blog posts.
* Add a new post.
* Edit or delete a post.
* View post details.

### ****5. Weather Forecast Application****

**Goal**: Build a small web application to display current weather using a public API (e.g., OpenWeatherMap).

#### ****Steps****:

1. Create an ASP.NET Core MVC app.
2. Create a controller to fetch data from the weather API.
3. Display the weather data in a view.

### ****6. E-Commerce Mini Application****

**Goal**: Create a mini e-commerce app with product listings, a shopping cart, and checkout functionality.

#### ****Features****:

* List available products.
* Add products to the shopping cart.
* View and update the cart.
* Checkout to place an order.

#### ****Implementation****:

* Use the MVC pattern.
* Integrate Razor Pages for checkout.