Nice — below is a complete, ready-to-drop-in implementation for the **Artworks** REST API you described (ASP.NET Core 6 / EF Core 6). It includes the Artwork model, ApplicationDbContext, ArtworksController, Program.cs (configured to run on port **8080**) and an appsettings.json snippet for SQL Server. It also enables Swagger so you can view /swagger/index.html.

Copy each file into your project (matching folders) and run the migration/update steps you already listed.

**Files**

**1) Models/Artwork.cs**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace YourNamespace.Models

{

public class Artwork

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int ArtworkId { get; set; }

[Required]

public string Title { get; set; }

[Required]

public string Artist { get; set; }

public int Year { get; set; }

public string Medium { get; set; }

public string Description { get; set; }

}

}

**2) Data/ApplicationDbContext.cs**

using Microsoft.EntityFrameworkCore;

using YourNamespace.Models;

namespace YourNamespace.Data

{

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Artwork> Artworks { get; set; }

}

}

**3) Controllers/ArtworksController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using dotnetapp.Models;

namespace dotnetapp.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class ArtworksController : ControllerBase

    {

        private readonly ApplicationDbContext \_context;

        public ArtworksController(ApplicationDbContext context)

        {

            \_context = context;

        }

        // GET /api/artworks

        [HttpGet]

        public ActionResult<IEnumerable<Artwork>> GetArtworks()

        {

            var artworks = \_context.Artworks.ToList();

            return Ok(artworks);

        }

        // GET /api/artworks/{id}

        [HttpGet("{id}")]

        public ActionResult<Artwork> GetArtworkById(int id)

        {

            var artwork = \_context.Artworks.FirstOrDefault(a => a.ArtworkId == id);

            if (artwork == null)

            {

                return NotFound();

            }

            return Ok(artwork);

        }

        // GET /api/artworks/filter?artist={artistName}

        [HttpGet("filter")]

        public ActionResult<IEnumerable<Artwork>> GetArtworksByArtist([FromQuery] string artist)

        {

            if (string.IsNullOrEmpty(artist))

            {

                return Ok(\_context.Artworks.ToList());

            }

            var artworks = \_context.Artworks

                .Where(a => a.Artist.ToLower().Contains(artist.ToLower()))

                .ToList();

            return Ok(artworks);

        }

        // POST /api/artworks

        [HttpPost]

        public ActionResult<Artwork> CreateArtwork([FromBody] Artwork artwork)

        {

            if (artwork == null)

            {

                return BadRequest();

            }

            \_context.Artworks.Add(artwork);

            \_context.SaveChanges();

            return CreatedAtAction(nameof(GetArtworkById), new { id = artwork.ArtworkId }, artwork);

        }

    }

}

**4) Program.cs (ASP.NET Core 6 minimal-hosting)**

using Microsoft.EntityFrameworkCore;

using YourNamespace.Data;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// Configure EF Core (SQL Server) - connection string from appsettings.json

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"))

);

// Optional: allow DB seeding or other services here

var app = builder.Build();

// Use port 8080 for Kestrel hosting

// If you want to force URLs regardless of environment, uncomment the following:

// app.Urls.Add("http://0.0.0.0:8080");

// Configure middleware

if (app.Environment.IsDevelopment() || true)

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

// ensure app listens on port 8080 when running with `dotnet run`

app.Urls.Add("http://0.0.0.0:8080");

app.Run();

app.Urls.Add("http://0.0.0.0:8080"); forces the server to bind to port 8080 on all interfaces — useful for the exam environment/container. You can also use builder.WebHost.UseUrls("http://0.0.0.0:8080") before builder.Build() if you prefer.

**5) appsettings.json (snippet)**

{

"ConnectionStrings": {

"DefaultConnection": "User ID=sa;password=examlyMssql@123;server=localhost;Database=appdb;Trusted\_Connection=false;Persist Security Info=False;Encrypt=False"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

}

}

**6) Optional Properties/launchSettings.json (if using Visual Studio) — set the application URL to port 8080**

{

"profiles": {

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"dotnetapp": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"applicationUrl": "http://localhost:8080",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

**Migration & Run commands (summary)**

You already listed most of these — repeat here to be complete:

1. restore packages

cd dotnetapp

dotnet restore

1. create EF tool manifest (only first time)

dotnet new tool-manifest

dotnet tool install --local dotnet-ef --version 6.0.6

1. create migration

dotnet dotnet-ef migrations add InitialSetup

1. update database

dotnet dotnet-ef database update

1. run the app (ensures port 8080)

dotnet run

Because Program.cs added the URL http://0.0.0.0:8080, the application will be reachable on port **8080**. If needed, run:

dotnet run --urls "http://0.0.0.0:8080"

Open Swagger UI (after app starts): http://localhost:8080/swagger/index.html

**Example requests (curl)**

* Get all artworks:

curl http://localhost:8080/api/artworks

* Get artwork by id:

curl http://localhost:8080/api/artworks/1

* Filter by artist:

curl "http://localhost:8080/api/artworks/filter?artist=VanGogh"

* Create artwork:

curl -X POST http://localhost:8080/api/artworks \

-H "Content-Type: application/json" \

-d '{"Title":"Starry Night","Artist":"Vincent van Gogh","Year":1889,"Medium":"Oil on canvas","Description":"A famous painting"}'

**Notes, pitfalls & tips**

* Ensure your namespace values (replace YourNamespace) match your project. If you use default project namespace, update the using lines accordingly.
* If you run in a container/exam sandbox, binding to 0.0.0.0:8080 is required — that's why app.Urls.Add("http://0.0.0.0:8080"); is in Program.cs.
* If EF migration fails, enable sensitive logging temporarily or inspect the migration for required fields.
* Swagger URL will be http://<host>:8080/swagger/index.html as required by your submission instructions.

If you’d like, I can:

* add initial seed data for testing,
* provide a Dockerfile that runs on port 8080,
* or produce unit/integration tests for the controller endpoints.

Tell me which of those you'd like and I’ll include the code directly.

Top of Form