**1. Folder Structure**

dotnetapp/

│

├── Controllers/

│ ├── SongController.cs

│ └── SingerController.cs

│

├── Models/

│ ├── Song.cs

│ ├── Singer.cs

│ └── SongApplicationDbContext.cs

│

├── Program.cs

├── appsettings.json

└── HungamaGaana.csproj

**2. Models**

**Models/Song.cs**

using System.ComponentModel.DataAnnotations;

namespace dotnetapp.Models

{

public class Song

{

[Key]

public int SongID { get; set; }

[Required]

public string SongName { get; set; }

[Required]

public string ReleaseYear { get; set; }

[Required]

public string SingerName { get; set; }

}

}

**Models/Singer.cs**

using System.ComponentModel.DataAnnotations;

namespace dotnetapp.Models

{

public class Singer

{

[Key]

public int SingerID { get; set; }

[Required]

public string SingerName { get; set; }

[Required]

public string Country { get; set; }

}

}

**3. Database Context**

**Models/SongApplicationDbContext.cs**

using Microsoft.EntityFrameworkCore;

namespace dotnetapp.Models

{

public class SongApplicationDbContext : DbContext

{

public SongApplicationDbContext(DbContextOptions<SongApplicationDbContext> options)

: base(options)

{

}

public DbSet<Song> Songs { get; set; }

public DbSet<Singer> Singers { get; set; }

}

}

**4. Controllers**

**Controllers/SongController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using dotnetapp.Models;

namespace dotnetapp.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class SongController : ControllerBase

{

private readonly SongApplicationDbContext \_context;

public SongController(SongApplicationDbContext context)

{

\_context = context;

}

// 1. GET /api/Song

[HttpGet]

public async Task<ActionResult<IEnumerable<Song>>> GetAllSongs()

{

var songs = await \_context.Songs.ToListAsync();

return Ok(songs);

}

// 2. GET /api/Song/{id}

[HttpGet("{id}")]

public async Task<ActionResult<Song>> GetSongById(int id)

{

var song = await \_context.Songs.FindAsync(id);

if (song == null)

return NotFound();

return Ok(song);

}

// 3. POST /api/Songs

[HttpPost]

[Route("/api/Songs")]

public async Task<ActionResult> AddSong([FromBody] Song song)

{

if (song == null || string.IsNullOrEmpty(song.SongName)

|| string.IsNullOrEmpty(song.ReleaseYear)

|| string.IsNullOrEmpty(song.SingerName))

{

return BadRequest("Invalid song details.");

}

await \_context.Songs.AddAsync(song);

await \_context.SaveChangesAsync();

return Ok();

}

// 4. DELETE /api/Song/{id}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteSong(int id)

{

if (id <= 0)

return BadRequest("Not a valid song id");

var song = await \_context.Songs.FindAsync(id);

if (song == null)

return NotFound();

\_context.Songs.Remove(song);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

**Controllers/SingerController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using dotnetapp.Models;

namespace dotnetapp.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class SingerController : ControllerBase

{

private readonly SongApplicationDbContext \_context;

public SingerController(SongApplicationDbContext context)

{

\_context = context;

}

// 1. GET /api/Singer

[HttpGet]

public async Task<ActionResult<IEnumerable<Singer>>> GetAllSingers()

{

var singers = await \_context.Singers.ToListAsync();

return Ok(singers);

}

// 2. POST /api/Singer

[HttpPost]

public async Task<ActionResult> AddSinger([FromBody] Singer singer)

{

if (singer == null || string.IsNullOrEmpty(singer.SingerName)

|| string.IsNullOrEmpty(singer.Country))

{

return BadRequest("Invalid singer details.");

}

await \_context.Singers.AddAsync(singer);

await \_context.SaveChangesAsync();

return Ok();

}

// 3. DELETE /api/Singer/{id}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteSinger(int id)

{

if (id <= 0)

return BadRequest("Not a valid Singer id");

var singer = await \_context.Singers.FindAsync(id);

if (singer == null)

return NotFound();

\_context.Singers.Remove(singer);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

**5. Program.cs**

using dotnetapp.Models;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// SQL Server connection

var connectionString = "User ID=sa;password=examlyMssql@123;server=localhost;Database=appdb;trusted\_connection=false;Persist Security Info=False;Encrypt=False";

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

options.UseSqlServer(connectionString));

var app = builder.Build();

// Configure the HTTP pipeline

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**6. appsettings.json (Optional)**

If you want to move the connection string here:

{

"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": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

Then in Program.cs, replace connection string with:

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");

**7. Entity Framework Commands**

cd dotnetapp

dotnet new tool-manifest

dotnet tool install --local dotnet-ef --version 6.0

dotnet restore

dotnet build

dotnet dotnet-ef migrations add "InitialSetup"

dotnet dotnet-ef database update

This creates tables: **Songs** and **Singers** in appdb.

**8. Run & Test**

dotnet run

Open:

http://localhost:8080/swagger/index.html

You’ll see endpoints for:

* GET /api/Song
* GET /api/Song/{id}
* POST /api/Songs
* DELETE /api/Song/{id}
* GET /api/Singer
* POST /api/Singer
* DELETE /api/Singer/{id}

**9. Validation Summary**

| **Operation** | **Success** | **Validation Errors** | **Failure** |
| --- | --- | --- | --- |
| **AddSong** | 200 OK | "Invalid song details." | — |
| **DeleteSong** | `204 No Conte |  |  |

Top of Form

Bottom of Form