Student.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Net.Mime;

namespace EFCodeFirstApproach.Models
{
    public class Student
    {
        [Key]
        public int Id { get; set; }

        [Required]
        [Column("StudentName", TypeName = "Varchar(100)")]


        public String Name { get; set; }

        [Column("StudentGender", TypeName = "Varchar(100)")]
        [Required]
        public String Gender { get; set; }


        [Required]
        public int? Age { get; set; }

        [Required]
        public int? Standard { get; set; }
    }
}
```

StudentDBContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace EFCodeFirstApproach.Models
{
    public class StudentDBContext : DbContext
    {
        public StudentDBContext(DbContextOptions options): base(options)
        {

        }

        public DbSet<Student> Students { get; set; }
    }
}
```

Appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },

  "ConnectionStrings": {
    "dbcs":
"Server=localhost\\SQLEXPRESS;Database=CodeFirstDB;Trusted_Connection=True;"
  },
  "AllowedHosts": "*"
}
```

Program.cs

```
using EFCodeFirstApproach.Models;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();


//step 4
var provider = builder.Services.BuildServiceProvider(); // making service
var config = provider.GetRequiredService<IConfiguration>(); //service for
configuration
builder.Services.AddDbContext<StudentDBContext>(item =>
item.UseSqlServer(config.GetConnectionString("dbcs")));//register context class,
connection string, and database provider

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production
scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

HomeController.cs

```
using System.Diagnostics;
using EFCodeFirstApproach.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EFCodeFirstApproach.Controllers
{
    public class HomeController : Controller
    {



        //private readonly ILogger<HomeController> _logger;

        //public HomeController(ILogger<HomeController> logger)
        //{
        //    _logger = logger;
        //}

        private readonly StudentDBContext studentDB;
        public HomeController(StudentDBContext studentDB)
        {
            this.studentDB = studentDB;
        }
        public async Task<IActionResult> Index()
        {
            var stdData = await studentDB.Students.ToListAsync();
            return View(stdData);
        }

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

        [HttpPost]
        public async Task<IActionResult> Create(Student std)
        {
            if(ModelState.IsValid)
            {
                await studentDB.Students.AddAsync(std);
                await studentDB.SaveChangesAsync();
                return RedirectToAction("Index","Home");
            }
            return View(std);
        }


        public async Task<IActionResult> Details(int id)
        {

            if(id==null || studentDB.Students == null)
            {
                return NotFound();
            }
            var stdData = await studentDB.Students.FirstOrDefaultAsync(x=>x.Id==id);

            if (stdData == null)
            {
                return NotFound();
            }
            return View(stdData);
        }

        //EDIT
        // GET
        public IActionResult Edit(int id)
        {
            var student = studentDB.Students.Find(id);
            return View(student);
        }

        // POST
        [HttpPost]
        public IActionResult Edit(Student student)
        {
            studentDB.Students.Update(student);
            studentDB.SaveChanges();
            return RedirectToAction("Index");
        }

        // DELETE
        // GET
        public IActionResult Delete(int id)
        {
            var student = studentDB.Students.Find(id);
            return View(student);
        }

        [HttpPost]
        [ActionName("Delete")]
        public IActionResult DeleteConfirmed(int id)
        {
            var student = studentDB.Students.Find(id);
            studentDB.Students.Remove(student);
            studentDB.SaveChanges();
            return RedirectToAction("Index");
        }
```

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


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