**WEB API CRUD OPERATION USING DATABASE**

**SAI KUMARAVELU A.T**

**EMPCONTROLLER.CS**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using webapi;

namespace webapi.Controllers

{

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

[ApiController]

public class empController : ControllerBase

{

private readonly empdb \_context;

public empController(empdb context)

{

\_context = context;

}

// GET: api/emp

[HttpGet]

public async Task<ActionResult<IEnumerable<employee>>> Getemployee()

{

return await \_context.employee.ToListAsync();

}

// GET: api/emp/5

[HttpGet("{id}")]

public async Task<ActionResult<employee>> Getemployee(int id)

{

var employee = await \_context.employee.FindAsync(id);

if (employee == null)

{

return NotFound();

}

return employee;

}

// PUT: api/emp/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> Putemployee(int id, employee employee)

{

if (id != employee.id)

{

return BadRequest();

}

\_context.Entry(employee).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!employeeExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/emp

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<employee>> Postemployee(employee employee)

{

\_context.employee.Add(employee);

await \_context.SaveChangesAsync();

return CreatedAtAction("Getemployee", new { id = employee.id }, employee);

}

// DELETE: api/emp/5

[HttpDelete("{id}")]

public async Task<IActionResult> Deleteemployee(int id)

{

var employee = await \_context.employee.FindAsync(id);

if (employee == null)

{

return NotFound();

}

\_context.employee.Remove(employee);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool employeeExists(int id)

{

return \_context.employee.Any(e => e.id == id);

}

}

}

**EMPOLYEE.CS**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Threading.Tasks;

namespace webapi

{

public class employee

{

[Key]

public int id { get; set; }

public string name { get; set; }

public int age { get; set; }

public string des { get; set; }

}

}

**APPSETTING.JSON**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"MyConn": "Data Source=RAVI\\SQLEXPRESS;Initial Catalog=kumara;Integrated Security=True"

},

}

**EMPDB.CS**

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace webapi

{

public class empdb : DbContext

{

public empdb(DbContextOptions<empdb> options) : base(options)

{

}

public DbSet<employee> employee { get; set; }

}

}

**STARTUP.CS**

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using Microsoft.OpenApi.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace webapi

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

string conStr = this.Configuration.GetConnectionString("MyConn");

services.AddDbContext<empdb>(options => options.UseSqlServer(conStr));

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "webapi", Version = "v1" });

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseSwagger();

app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json", "webapi v1"));

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

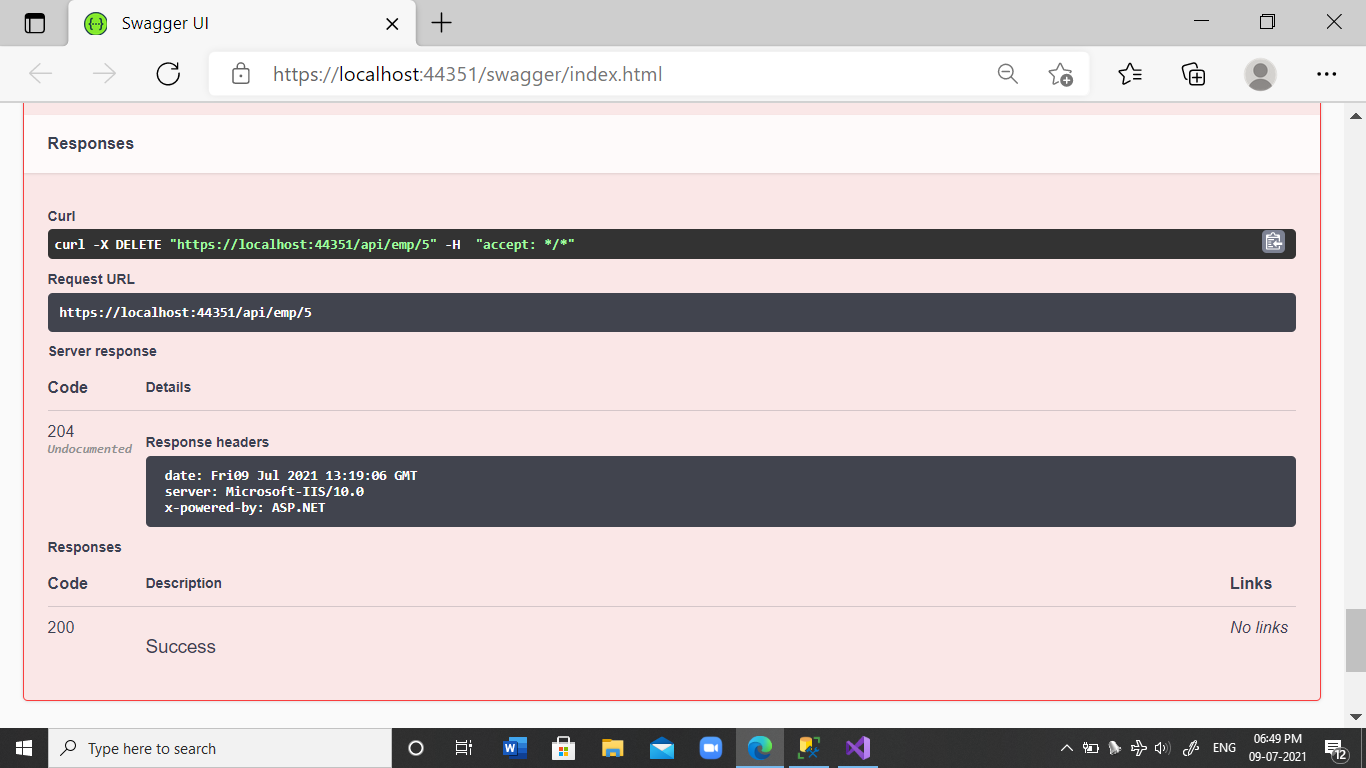
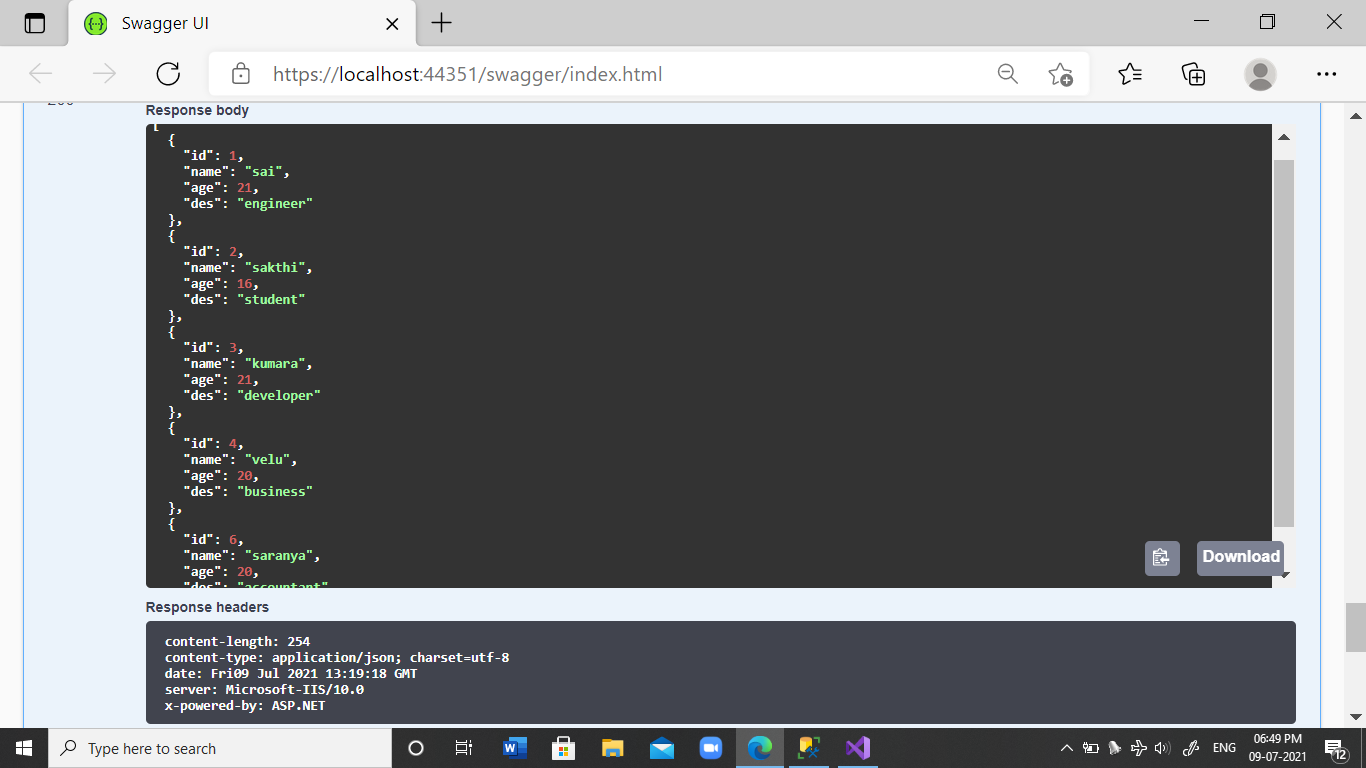
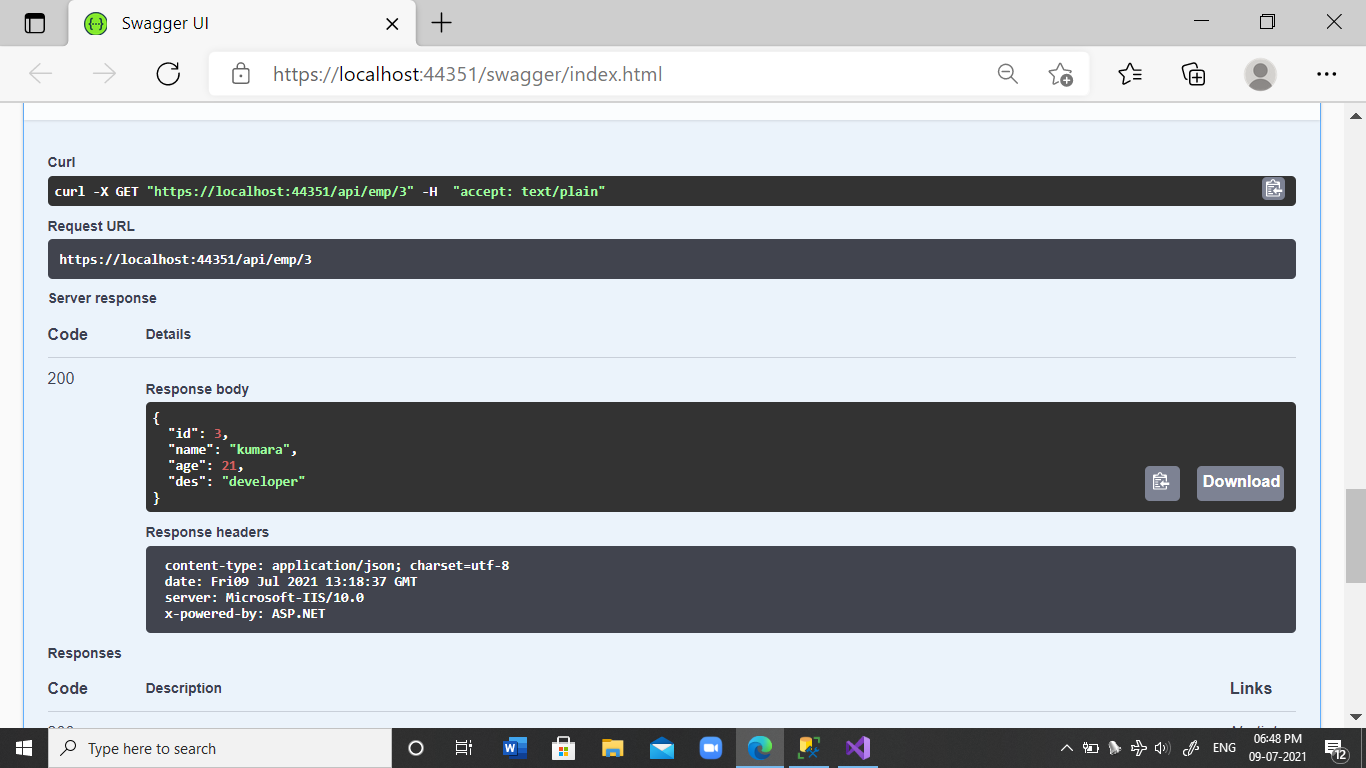
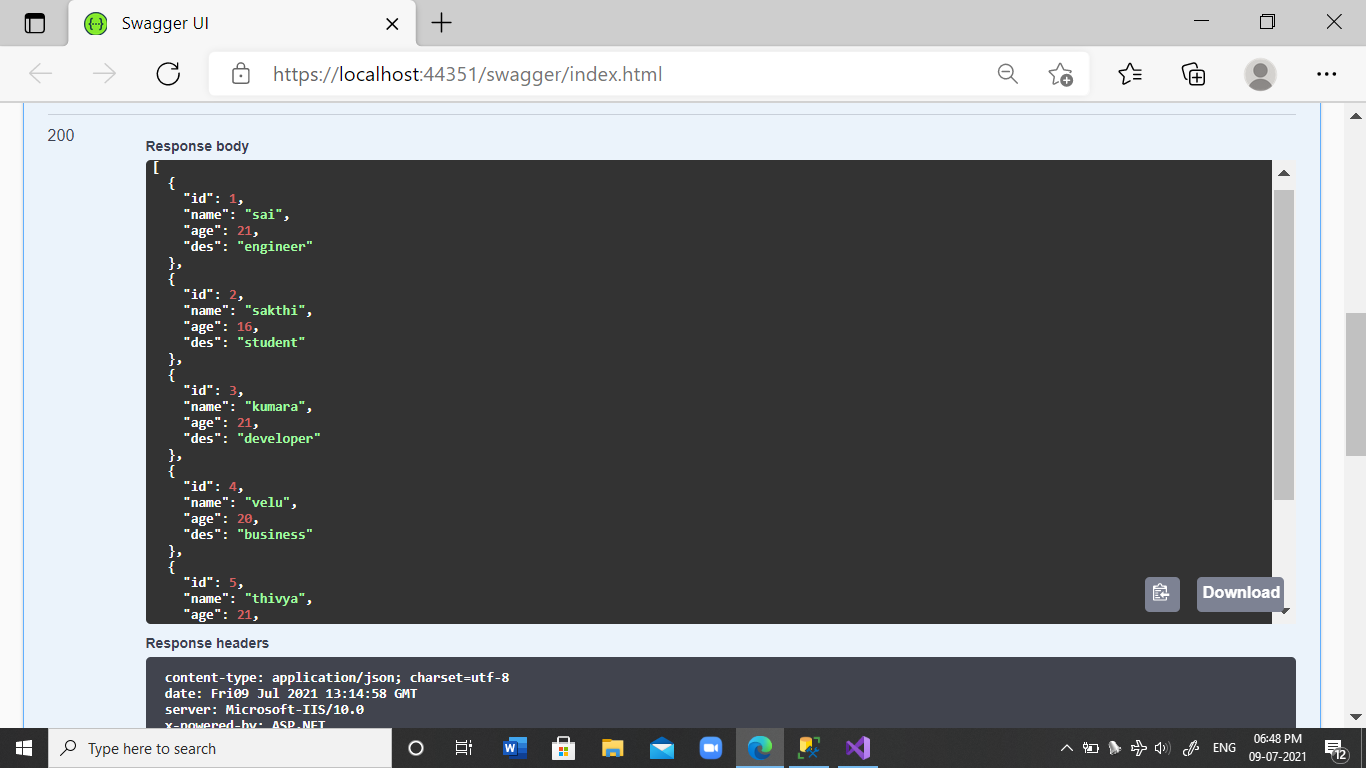
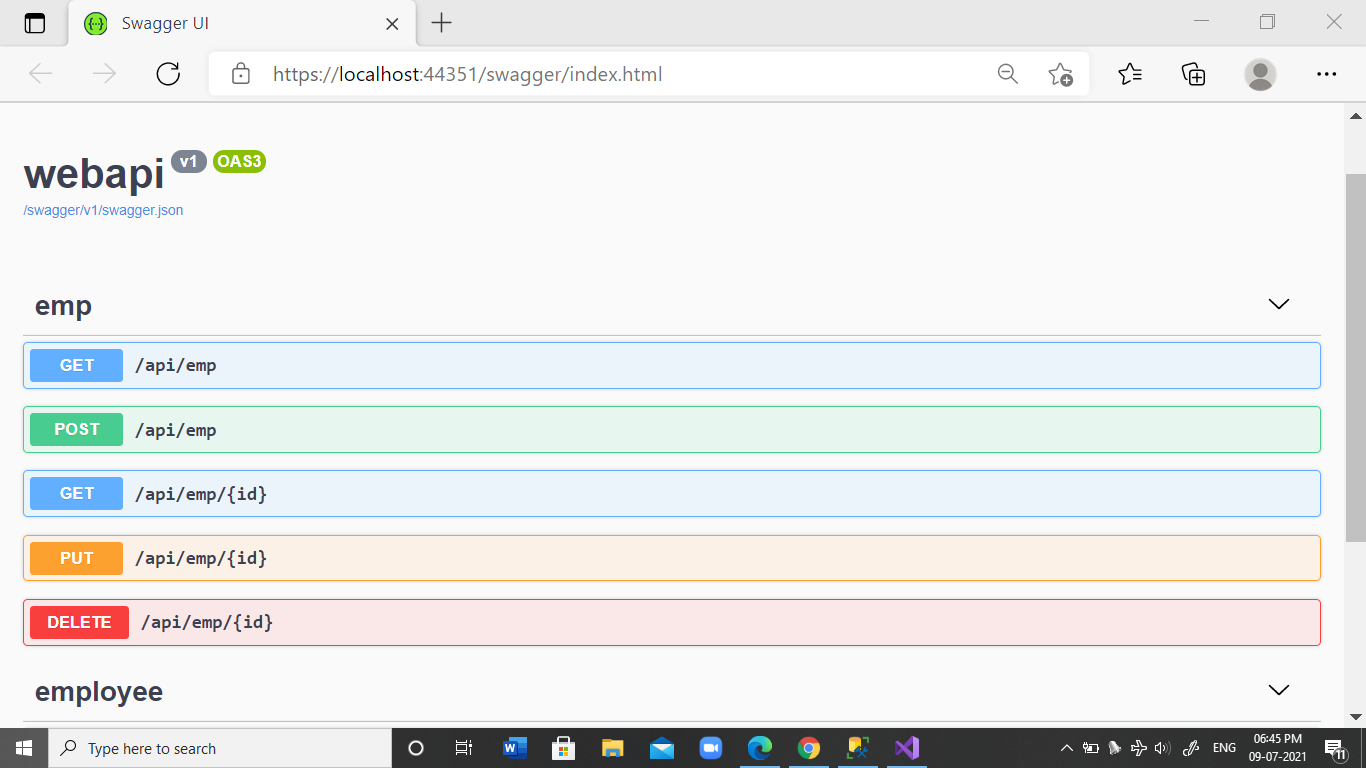
});

}

}

}

**OUTPUT**

****