The following are the code snippets for creating the Web API spanning over their individual files as explained in the video tutorials. “//” Denotes comments

```
public class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
    public string Telephone { get; set; }
    public string Email { get; set; }
}// End Customer
//Start
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System.Linq;
using WebAPI.Models;

namespace WebAPI.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class CustomerController : ControllerBase
  {
    private ICustomerRepository customerRepository;
    public CustomerController(ICustomerRepository repo)
    {
      customerRepository = repo;
    }
    [HttpGet]
    public IEnumerable<Customer> GetCustomers()
    {
      return customerRepository.GetAllCustomers().ToList();
```

```
        }
        [HttpGet("{id}")]
        public Customer GetCustomerById(int id)
        {
            return customerRepository.GetCustomerById(id);
        }

        [HttpPost]
        public Customer Create([FromBody] Customer customer)
        {
            return customerRepository.AddCustomer(customer);
        }

        [HttpPut]
        public Customer Update([FromForm] Customer customer)
        {
            return customerRepository.UpdateCustomer(customer);
        }

        [HttpDelete("{id}")]
        public void Delete(int id)
        {
            customerRepository.DeleteCustomer(id);
        }

    }
}
//End CustomerController
```

```
//Start
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;


namespace WebAPI.Models
{
   public class CustomerRepository : ICustomerRepository
   {
      public IConfiguration Configuration { get; set; }
      public string connectionString;
      private readonly ILogger<CustomerRepository> _logger;
      public CustomerRepository(IConfiguration configuration, ILogger<CustomerRepository> logger )
      {
         Configuration = configuration;
         connectionString = Configuration["ConnectionStrings:DefaultConnection"];
      }


      public Customer AddCustomer(Customer customer)
      {
         using (SqlConnection connection = new SqlConnection(connectionString))
         {
            try
            {
               SqlCommand cmd = new SqlCommand("[dbo].[spInsertIntoCustomer]", connection);
               cmd.CommandType = CommandType.StoredProcedure;
               connection.Open();
               cmd.Parameters.AddWithValue("@Name", customer.Name);
               cmd.Parameters.AddWithValue("@Address", customer.Address);
               cmd.Parameters.AddWithValue("@Telephone", customer.Telephone);
               cmd.Parameters.AddWithValue("@Email", customer.Email);
               // cmd.Parameters.AddWithValue("@ret", ParameterDirection.Output);
               cmd.ExecuteNonQuery();

            }
            catch (Exception ex)
            {
               //ex.Message.ToString();
               _logger.LogError(ex, "Error at AddCustomer() :(");
               customer = null;
            }

         }
         return customer;

      }

      public void DeleteCustomer(int id)
      {
         using (SqlConnection connection = new SqlConnection(connectionString))
         {
            try
            {
               SqlCommand cmd = new SqlCommand("[dbo].[spDeleteCustomer]", connection);
               cmd.CommandType = CommandType.StoredProcedure;
               connection.Open();
               cmd.Parameters.AddWithValue("@Id", id);
               cmd.ExecuteNonQuery();

            }
            catch (Exception ex)
            {
               //ex.Message.ToString();
               _logger.LogError(ex, "Error at DeleteCustomer() :(");

            }

         }
      }

      public IEnumerable<Customer> GetAllCustomers()
      {
         List<Customer> customers = new List<Customer>();
         using(SqlConnection con = new SqlConnection(connectionString))
         {
            try
            {
               SqlCommand cmd = new SqlCommand("[dbo].[spSelectCustomer]", con);
               cmd.CommandType = System.Data.CommandType.StoredProcedure;
               con.Open();
               SqlDataReader rdr = cmd.ExecuteReader();
               while (rdr.Read())
               {
                   Customer customer = new Customer();
                   customer.Id = Convert.ToInt32(rdr["Id"]);
                   customer.Name = rdr["Name"].ToString();
                   customer.Address = rdr["Address"].ToString();
                   customer.Telephone = rdr["Telephone"].ToString();
                   customer.Email = rdr["Email"].ToString();
                   customers.Add(customer);
               }
               rdr.Close();
            }
            catch (Exception ex)
            {
               _logger.LogError(ex, "Error at GetAllCustomers() : (");
               customers = null;
            }
         }
         return customers;
      }

      public Customer GetCustomerById(int id)
      {

         Customer customer = new Customer();
         using (SqlConnection con = new SqlConnection(connectionString))
         {
            try
            {
               SqlCommand cmd = new SqlCommand("[dbo].[spSelectCustomerById]", con);
               cmd.CommandType = System.Data.CommandType.StoredProcedure;
               con.Open();
               cmd.Parameters.AddWithValue("@Id", id);
               SqlDataReader rdr = cmd.ExecuteReader();
               while (rdr.Read())
               {

                   customer.Id = id;
               customer.Name = rdr["Name"].ToString();
               customer.Address = rdr["Address"].ToString();
               customer.Telephone = rdr["Telephone"].ToString();
               customer.Email = rdr["Email"].ToString();

             }
             rdr.Close();
            }
            catch (Exception ex)
            {
             _logger.LogError(ex, "Error at GetCustomerById() : (");
             customer= null;
            }
          }
          return customer;
         }

         public Customer UpdateCustomer(Customer customer)
         {
          //throw new NotImplementedException();
          using (SqlConnection connection = new SqlConnection(connectionString))
          {
            try
            {
             SqlCommand cmd = new SqlCommand("[dbo].[spUpdateCustomer]", connection);
             cmd.CommandType = CommandType.StoredProcedure;
             connection.Open();
             cmd.Parameters.AddWithValue("@Id", customer.Id);
             cmd.Parameters.AddWithValue("@Name", customer.Name);
             cmd.Parameters.AddWithValue("@Address", customer.Address);
             cmd.Parameters.AddWithValue("@Telephone", customer.Telephone);
```

```
                cmd.Parameters.AddWithValue("@Email", customer.Email);
                cmd.ExecuteNonQuery();


              }
              catch (Exception ex)
              {
                //ex.Message.ToString();
                _logger.LogError(ex, "Error at UpdateCustomer() :(");
                customer = null;
              }
            }


            return customer;
          }
        }
      }          //End CustomerRepository
```

```
//Start
using Microsoft.AspNetCore.Mvc;

namespace WebAPI.Controllers
{

   public class HomeController : ControllerBase
   {
      public  string Index()
      {
         return "API Running...";
      }
   }
}
//End Home Controller

//Start
public interface ICustomerRepository
   {
      IEnumerable<Customer> GetAllCustomers();
      Customer GetCustomerById(int id);
      Customer AddCustomer(Customer customer);
      Customer UpdateCustomer(Customer customer);
      void DeleteCustomer(int id);
   }
//End ICustomerRepository

//Start
  {
      IEnumerable<Order> GetAllOrders();
      Order GetOrderById(int id);
```

```
        Order AddOrder(Order order);
        Order UpdateOrder(Order order);
        void DeleteOrder(int id);
    }
//End IOrderRepository

//Start
public class Order
    {
        public int Id { get; set; }
        public int CustomerId { get; set; }
        public string Description { get; set; }
        public decimal OrderCost { get; set; }
    }
//End Order

//Start
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using WebAPI.Models;

namespace WebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrderController : ControllerBase
    {
        private IOrderRepository orderRepository;
        public OrderController(IOrderRepository repo)
        {
            orderRepository = repo;
        }

        [HttpGet]

        public IEnumerable<Order> GetOrders()

        {

            return orderRepository.GetAllOrders();

        }

        [HttpGet("{id}")]

        public Order GetOrderById(int id)

        {

            return orderRepository.GetOrderById(id);

        }

        [HttpPost]

        public Order Create([FromBody] Order order)

        {

            return orderRepository.AddOrder(order);

        }

        [HttpPut]

        public Order Update([FromForm] Order order)

        {

            return orderRepository.UpdateOrder(order);

        }

        [HttpDelete("{id}")]

        public void Delete(int id)

        {

            orderRepository.DeleteOrder(id);

        }

    }

} //End OrderController
```

```
//Start
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;

namespace WebAPI.Models
{
    public class OrderRepository : IOrderRepository
    {
        public IConfiguration Configuration { get; set; }
        public string connectionString;
        private readonly ILogger<OrderRepository> _logger;
        public OrderRepository(IConfiguration configuration, ILogger<OrderRepository> logger)
        {
            Configuration = configuration;
            connectionString = Configuration["ConnectionStrings:DefaultConnection"];
            _logger = logger;
        }
        public Order AddOrder(Order order)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                try
                {
                    _logger.LogInformation("Could break here!!");
                    SqlCommand cmd = new SqlCommand("[dbo].[spInsertIntoOrder]", connection);
                    cmd.CommandType = CommandType.StoredProcedure;
                    connection.Open();
                    cmd.Parameters.AddWithValue("@CustomerId", order.CustomerId);
                    cmd.Parameters.AddWithValue("@Description", order.Description);
                    cmd.Parameters.AddWithValue("@OrderCost", order.OrderCost);

                    // cmd.Parameters.AddWithValue("@ret", ParameterDirection.Output);
                    cmd.ExecuteNonQuery();

                }
                catch (Exception ex)
                {
                    //ex.Message.ToString();
                    _logger.LogError(ex, "Error at AddOrder() :(");
                    order = null;
                }
            }
            return order;
        }

        public void DeleteOrder(int id)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                try
                {
                    SqlCommand cmd = new SqlCommand("[dbo].[spDeleteOrder]", connection);
                    cmd.CommandType = CommandType.StoredProcedure;
                    connection.Open();
                    cmd.Parameters.AddWithValue("@Id", id);
                    cmd.ExecuteNonQuery();

                }
                catch (Exception ex)
                {
                    //ex.Message.ToString();
                    _logger.LogError(ex, "Error at DeleteOrder() :(");

                }
            }
        }

        public IEnumerable<Order> GetAllOrders()
        {
            List<Order> orders = new List<Order>();
            using (SqlConnection con = new SqlConnection(connectionString))
            {
                try
                {
                    SqlCommand cmd = new SqlCommand("[dbo].[spSelectOrder]", con);
                    cmd.CommandType = CommandType.StoredProcedure;
                    con.Open();
                    SqlDataReader rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        Order order = new Order();
                        order.Id = Convert.ToInt32(rdr["Id"]);
                        order.CustomerId = Convert.ToInt32(rdr["CustomerId"]);
                        order.Description = rdr["Description"].ToString();
                        order.OrderCost = Convert.ToDecimal(rdr["OrderCost"]);
                        orders.Add(order);

                    }
                    rdr.Close();
                }
                catch (Exception ex)
                {
                    //ex.Message.ToString();
                    _logger.LogError(ex, "Error at GetAllOrders() :(");
                    orders = null;
                }
            }
            return orders;
        }

        public Order GetOrderById(int id)
        {
            Order order = new Order();
            using (SqlConnection con = new SqlConnection(connectionString))
            {
                try
                {
                    SqlCommand cmd = new SqlCommand("dbo.spSelectOrderById", con);
                    cmd.CommandType = CommandType.StoredProcedure;
                    con.Open();
                    cmd.Parameters.AddWithValue("@Id", id);
                    SqlDataReader rdr = cmd.ExecuteReader();
                    while (rdr.Read())
                    {
                        order.Id = Convert.ToInt32(rdr["Id"]);
                        order.CustomerId = Convert.ToInt32(rdr["CustomerId"]);
                        order.Description = rdr["Description"].ToString();
                        order.OrderCost = Convert.ToDecimal(rdr["OrderCost"]);
                    }
                }
                catch (Exception ex)
                {
                    //ex.Message.ToString();
                    _logger.LogError(ex, "Error at GetOrderById() :(");

                }
                return order;

            }

        }

        public Order UpdateOrder(Order order)
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                try
                {
                    SqlCommand cmd = new SqlCommand("[dbo].[spUpdateOrder]", connection);
                    cmd.CommandType = CommandType.StoredProcedure;
                    connection.Open();
                    cmd.Parameters.AddWithValue("@Id", order.Id);
                    cmd.Parameters.AddWithValue("@CustomerId", order.CustomerId);
                    cmd.Parameters.AddWithValue("@Description", order.Description);
                    cmd.Parameters.AddWithValue("@OrderCost", order.OrderCost);
                    cmd.ExecuteNonQuery();

                }
                catch (Exception ex)
```

```
            {
                //ex.Message.ToString();
                _logger.LogError(ex, "Error at UpdateOrder() :(");
                order = null;

            }
        }

        return order;

      }
   }
} //End OrderRepository

//Start
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Serilog;
using WebAPI.Models;

namespace WebAPI
{
    public class Startup
    {

        IConfiguration Configuration;
        public Startup(IConfiguration configuration)
        {
            Log.Logger = new
LoggerConfiguration().ReadFrom.Configuration(configuration).CreateLogger();
            Configuration = configuration;
        }
        // This method gets called by the runtime. Use this method to add services to the container.
        // For more information on how to configure your application, visit
https://go.microsoft.com/fwlink/?LinkID=398940
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddSingleton<ICustomerRepository, CustomerRepository>();
            services.AddSingleton<IOrderRepository, OrderRepository>();
            services.AddControllersWithViews().AddNewtonsoftJson();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request
pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env, ILoggerFactory
loggerFactory)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            loggerFactory.AddSerilog();

            app.UseRouting();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                  name: "default",
                  pattern: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
} //End Startup
```