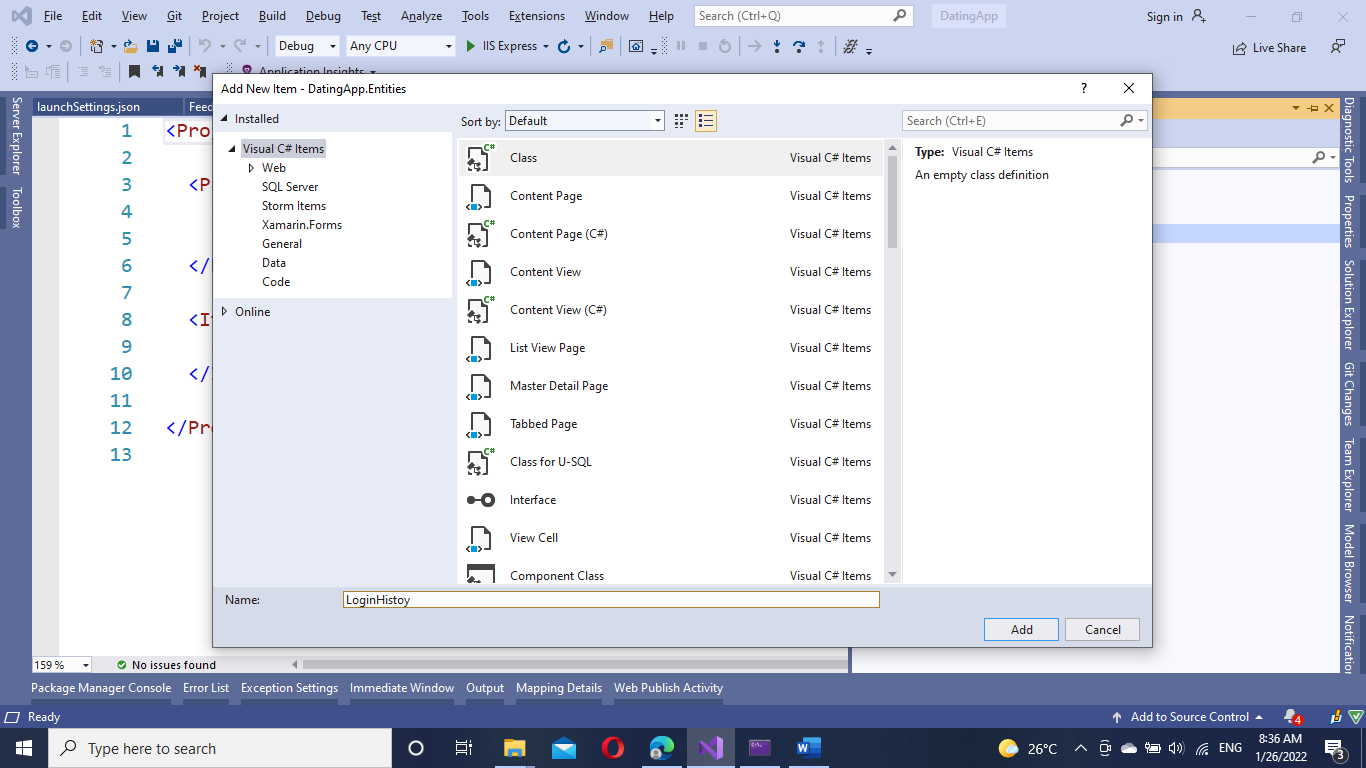
Step 1:

DatingApp.Entities

Create new class,



Create Your Class,

namespace DatingApp.Entities

{

public class LoginHistoy

{

[Key]

public int LoginTransId { get; set; }

public string LoginUser { get; set; }

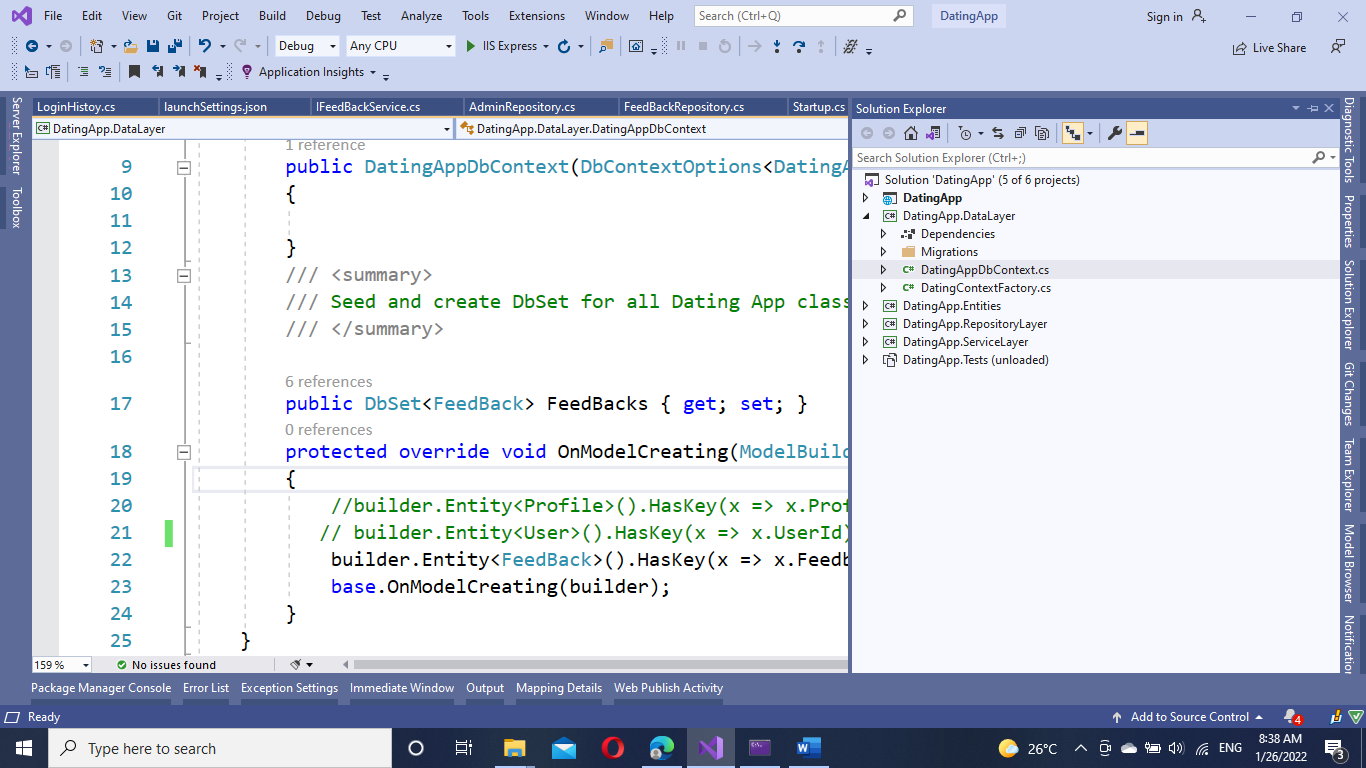
public DateTime Logintime { get; set; }

}

}

Step 2 :

Add newly created class into the DatingApp.DataLayer,



public class DatingAppDbContext : IdentityDbContext<UserMaster>

{

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

{

}

/// <summary>

/// Seed and create DbSet for all Dating App classes

/// </summary>

public DbSet<FeedBack> FeedBacks { get; set; }

public DbSet<LoginHistoy> LoginHistoys { get; set; }

protected override void OnModelCreating(ModelBuilder builder)

{

//builder.Entity<Profile>().HasKey(x => x.ProfileId);

builder.Entity<LoginHistoy>().HasKey(x => x.LoginTransId);

builder.Entity<FeedBack>().HasKey(x => x.FeedbackId);

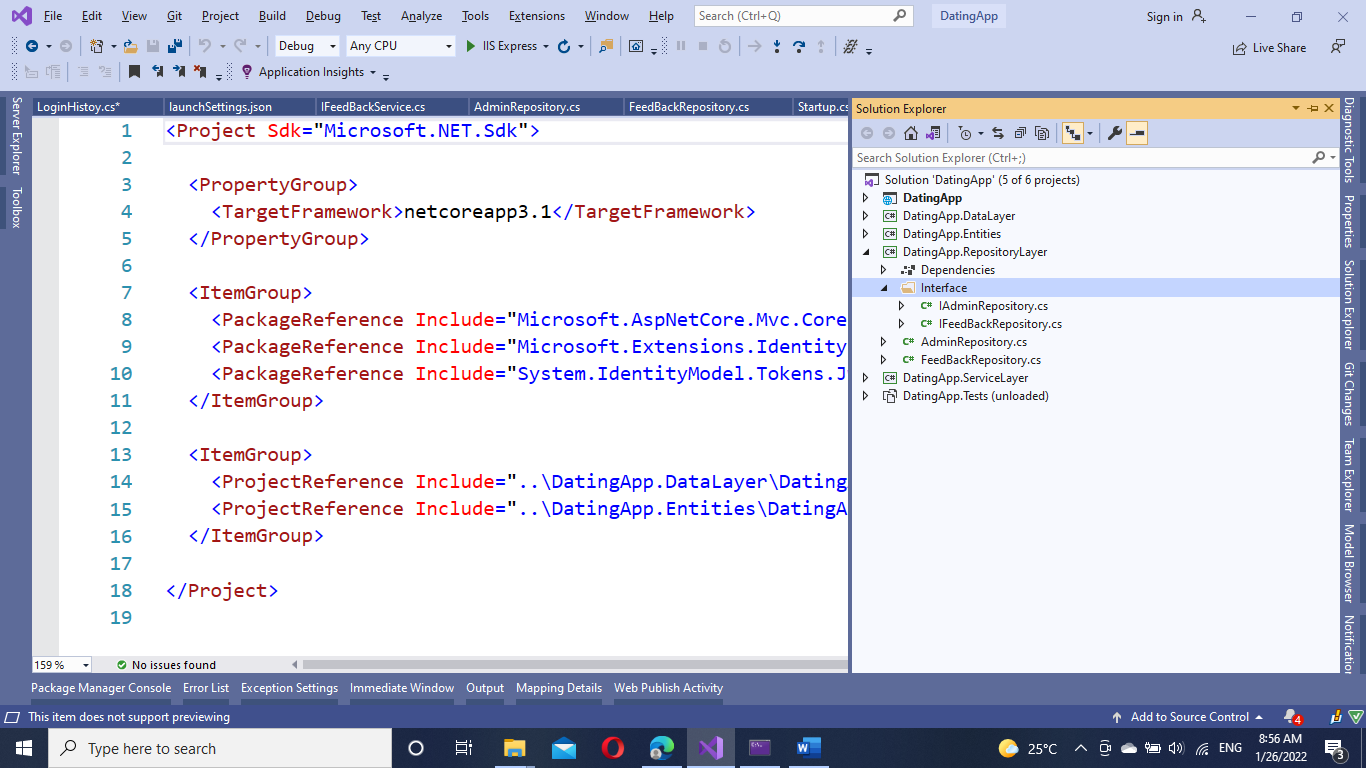
base.OnModelCreating(builder);

}

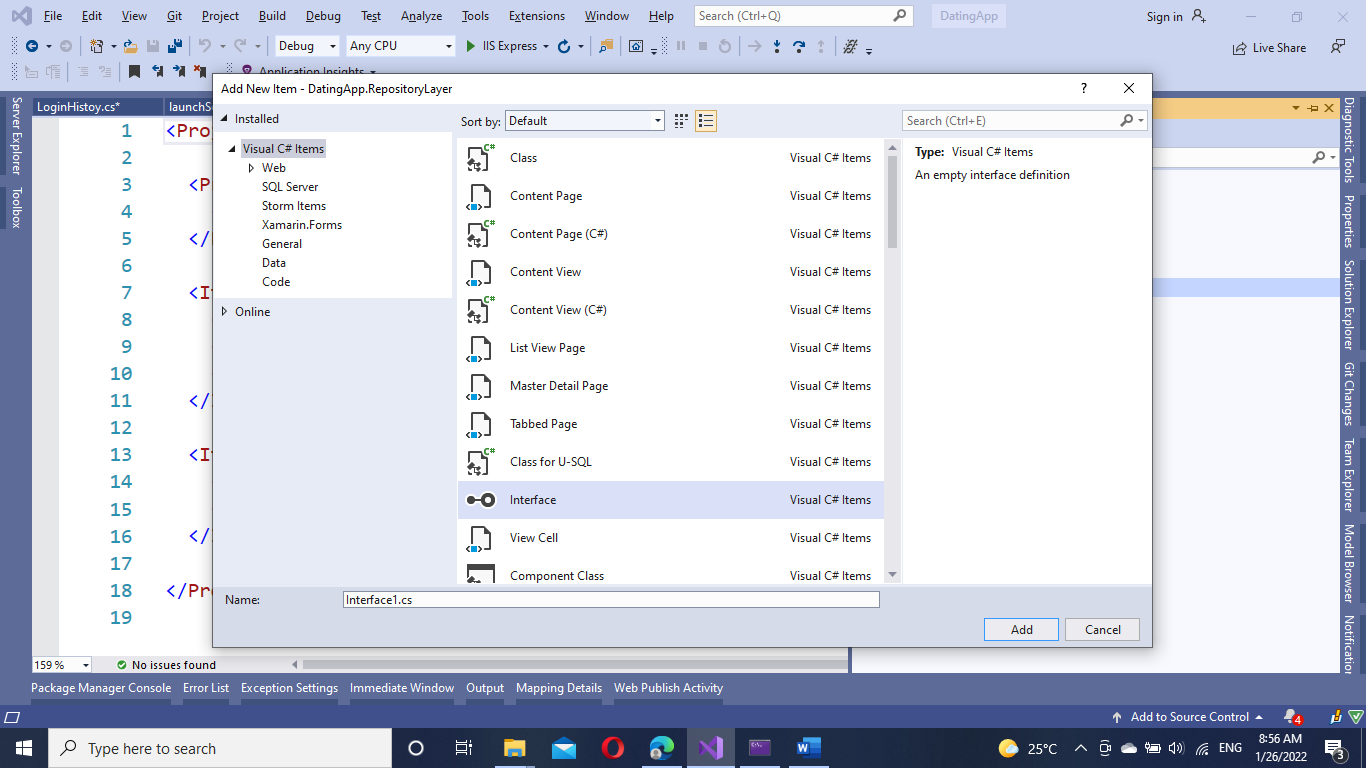
}

Step 3:

Add the class functionality into DatingApp.RepositoryLayer,



Add new interface,



namespace DatingApp.RepositoryLayer.Interface

{

public interface ILoginHistory

{

Task<LoginHistoy> GetLoginHistoryById(long dateId);

Task<LoginHistoy> GetLoginHistoryByUser(string RequestSenderName);

Task<LoginHistoy> GetLoginHistoryByDate(DateTime DateOfRequest);

Task<bool> AddLoginHistory(LoginHistoy appointment);

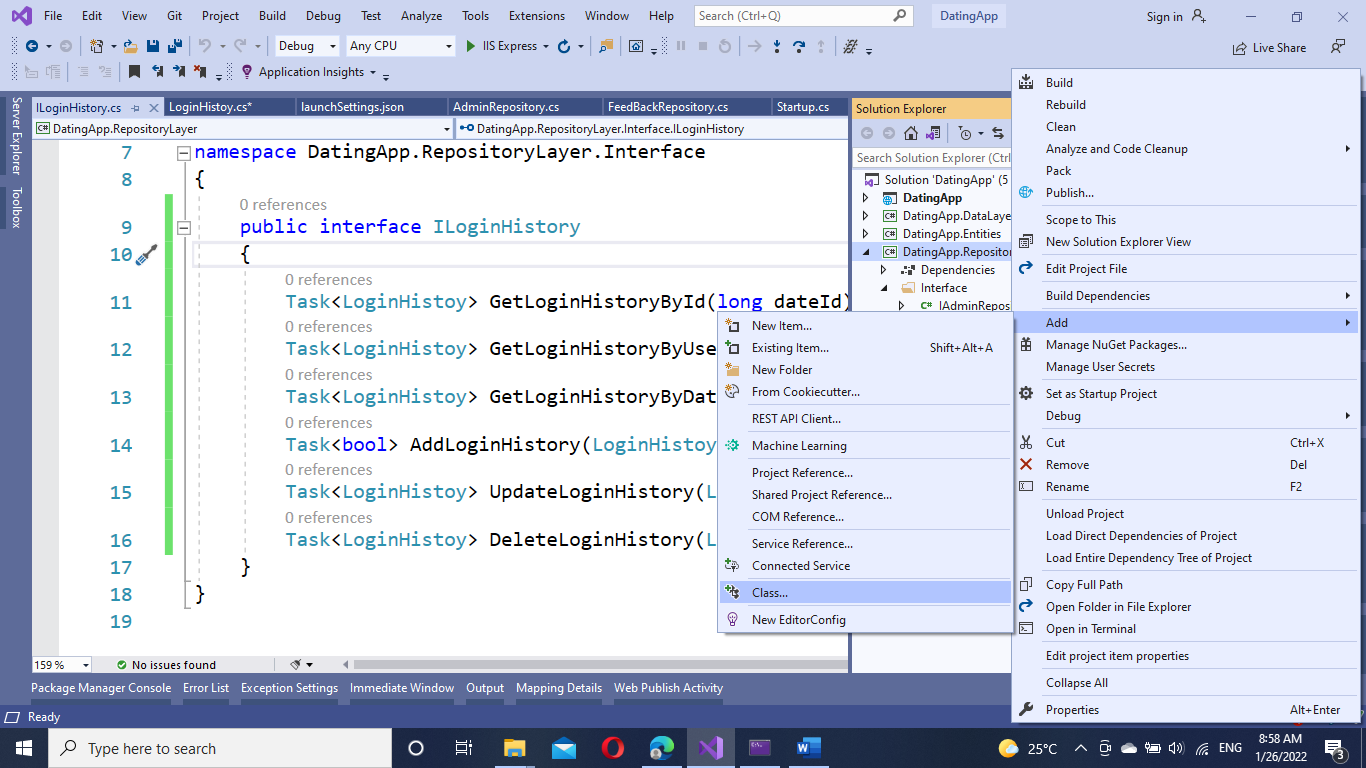
Task<LoginHistoy> UpdateLoginHistory(LoginHistoy appointment);

Task<LoginHistoy> DeleteLoginHistory(LoginHistoy appointment);

}

}

Implement the interface into a new class, add new class on the repository project

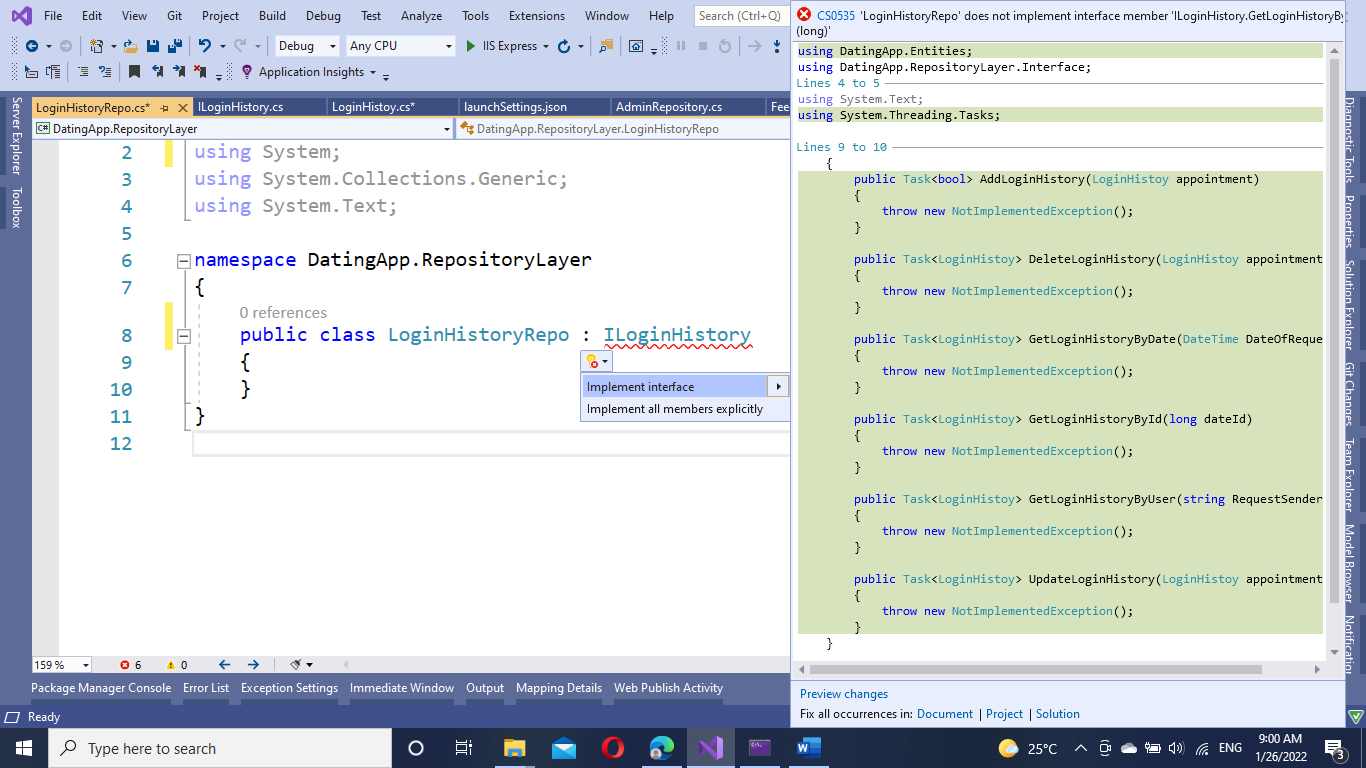


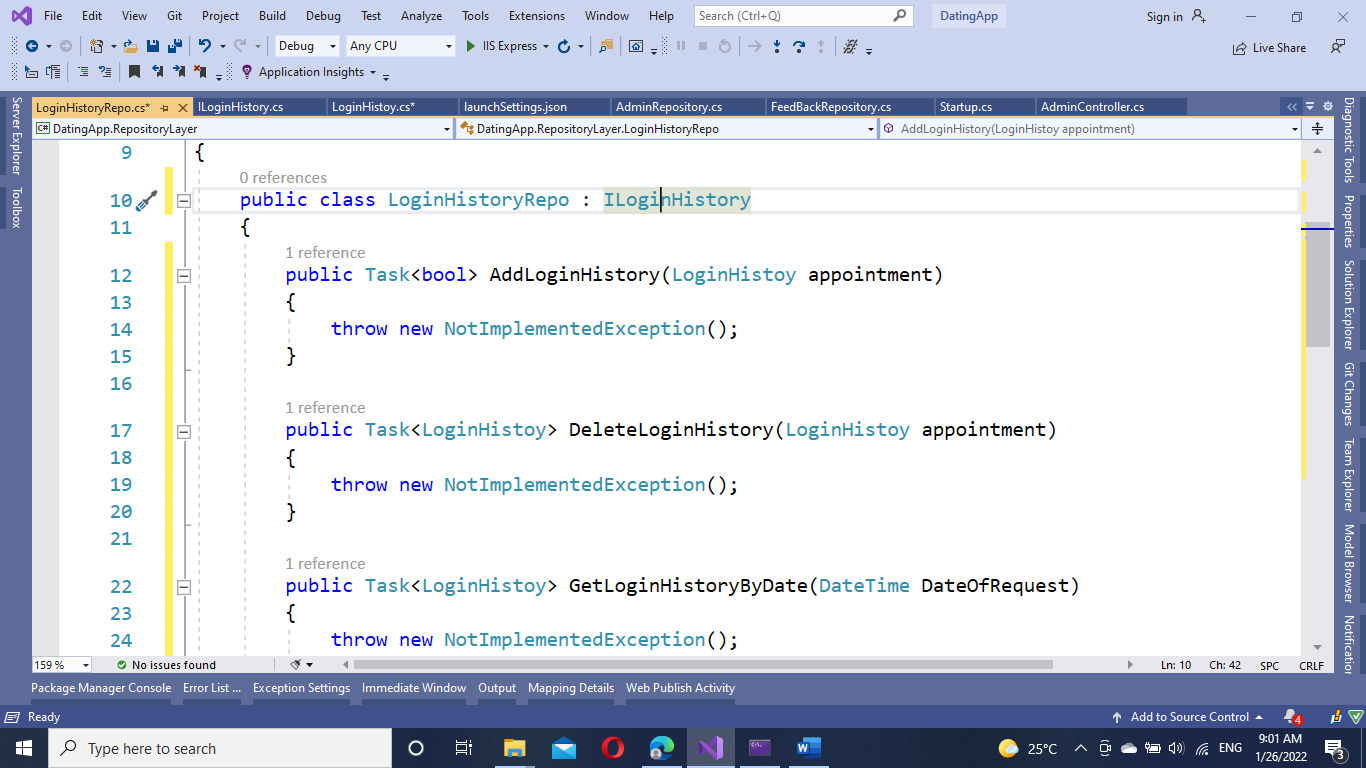
public class LoginHistoryRepo : ILoginHistory

{

}

Implement the Interface,





Create dependency injection into the class,

public class LoginHistoryRepo : ILoginHistory

{

/// <summary>

/// Creating and injecting DbContext in DateRepository constructor

/// </summary>

private readonly DatingAppDbContext \_datingContext;

public LoginHistoryRepo(DatingAppDbContext datingDbContext)

{

\_datingContext = datingDbContext;

}

public Task<bool> AddLoginHistory(LoginHistoy appointment)

{

throw new NotImplementedException();

}

Implement you code login for all the methods,

Add Method,

public async Task<bool> AddLoginHistory(LoginHistoy appointment)

{

try

{

await \_datingContext.LoginHistoys.AddAsync(appointment);

await \_datingContext.SaveChangesAsync();

return true;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

Delete Method,

public async Task<LoginHistoy> DeleteLoginHistory(LoginHistoy loginHistory)

{

try

{

\_datingContext.LoginHistoys.Remove(loginHistory);

await \_datingContext.SaveChangesAsync();

return await Task.FromResult(loginHistory);

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

Get method with Date param,

public async Task<LoginHistoy> GetLoginHistoryByDate(DateTime DateOfRequest)

{

try

{

var result = await \_datingContext.LoginHistoys.FirstOrDefaultAsync(h => h.Logintime.Equals(DateOfRequest));

return result;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

Get method with Id param,

public async Task<LoginHistoy> GetLoginHistoryById(long dateId)

{

try

{

var result = await \_datingContext.LoginHistoys

.FirstOrDefaultAsync(h => h.LoginTransId.Equals(dateId));

return result;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

Get method with string param,

public async Task<LoginHistoy> GetLoginHistoryByUser(string RequestSenderName)

{

try

{

var result = await \_datingContext.LoginHistoys

.FirstOrDefaultAsync(h => h.LoginUser.Equals(RequestSenderName));

return result;

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

}

Update Method,

public async Task<LoginHistoy> UpdateLoginHistory(LoginHistoy loginHistory)

{

try

{

\_datingContext.LoginHistoys.Update(loginHistory);

await \_datingContext.SaveChangesAsync();

return await Task.FromResult(loginHistory);

}

catch (Exception ex)

{

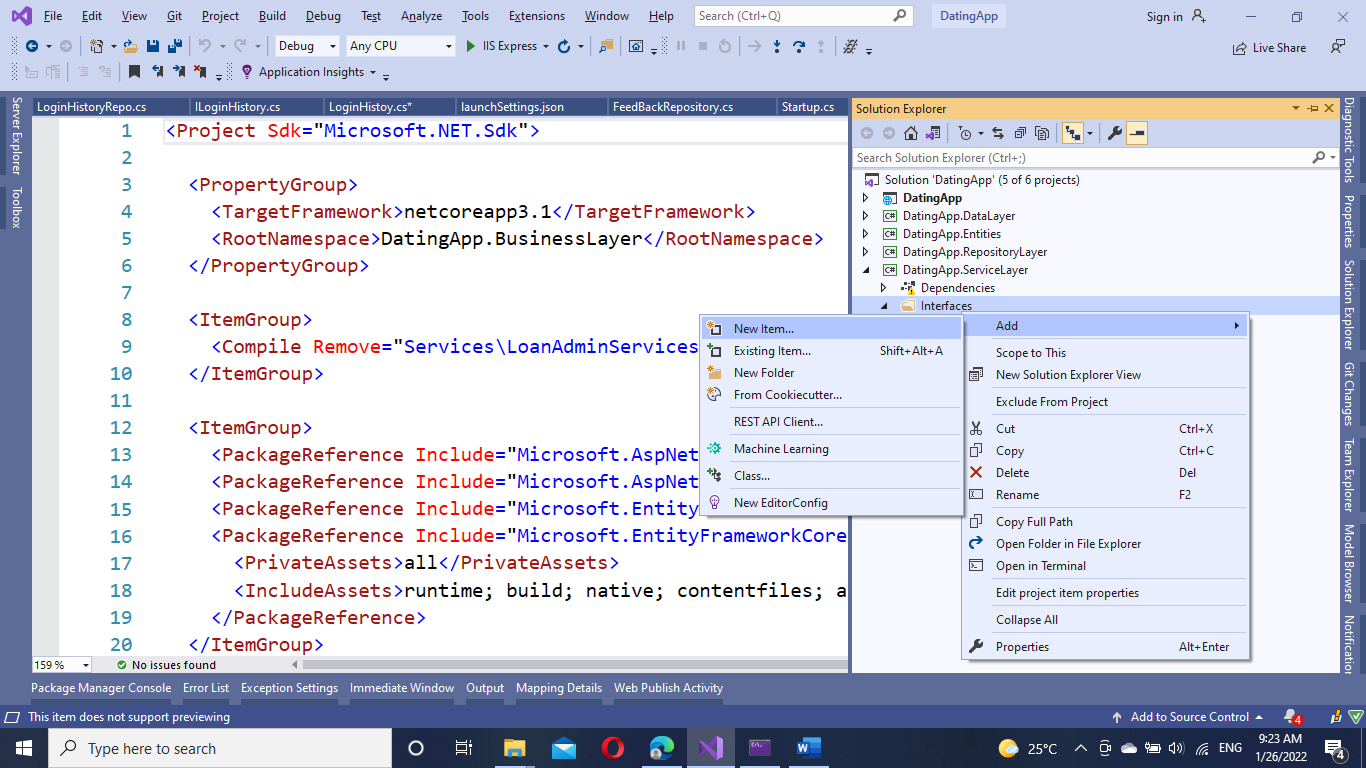
throw new Exception(ex.Message);

}

}

Add DatingApp.ServiceLayer,

Add new interface,



namespace DatingApp.BusinessLayer.Interfaces

{

public interface ILoginHistoryService

{

public Task<LoginHistoy> GetloginHistoryAsync(long id);

public List<LoginHistoy> AddloginHistory(LoginHistoy LoginHistoy);

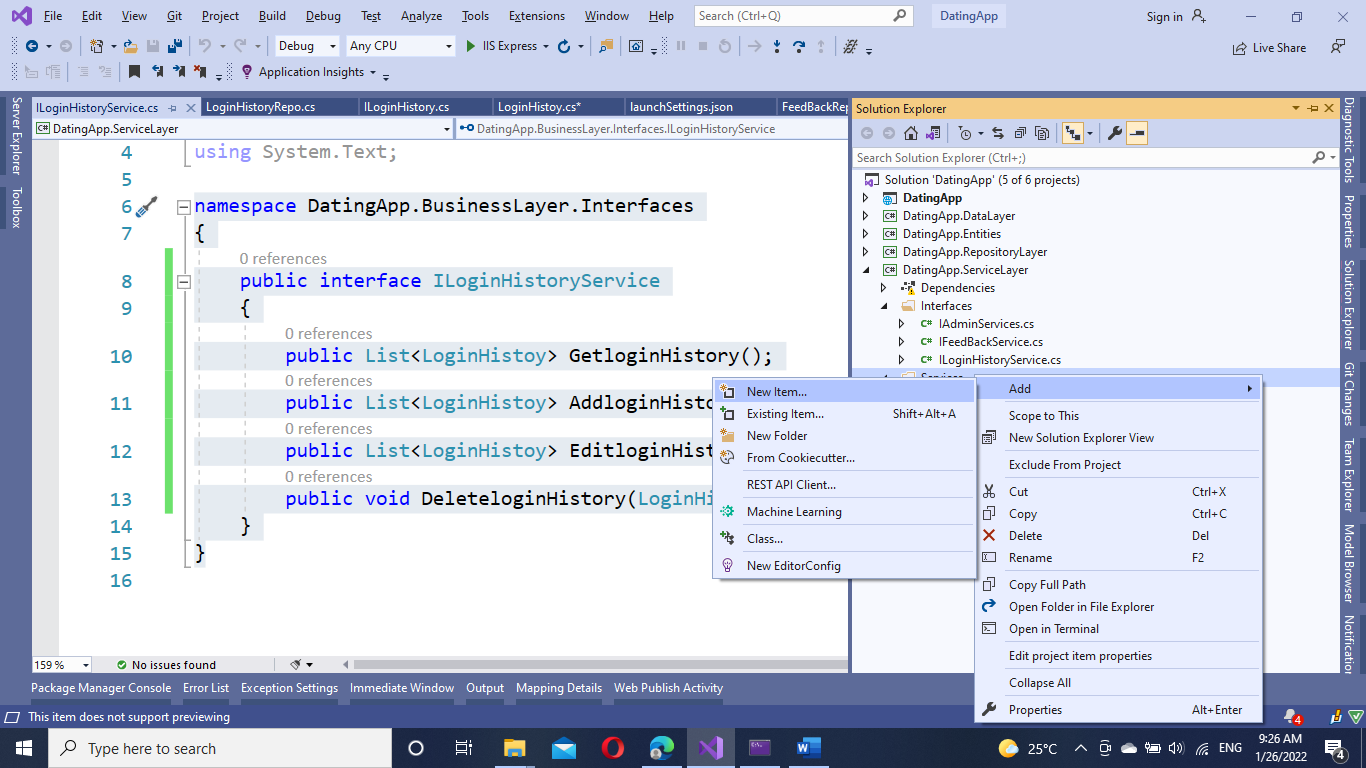
public LoginHistoy EditloginHistory(LoginHistoy LoginHistoy);

public void DeleteloginHistory(LoginHistoy LoginHistoy);

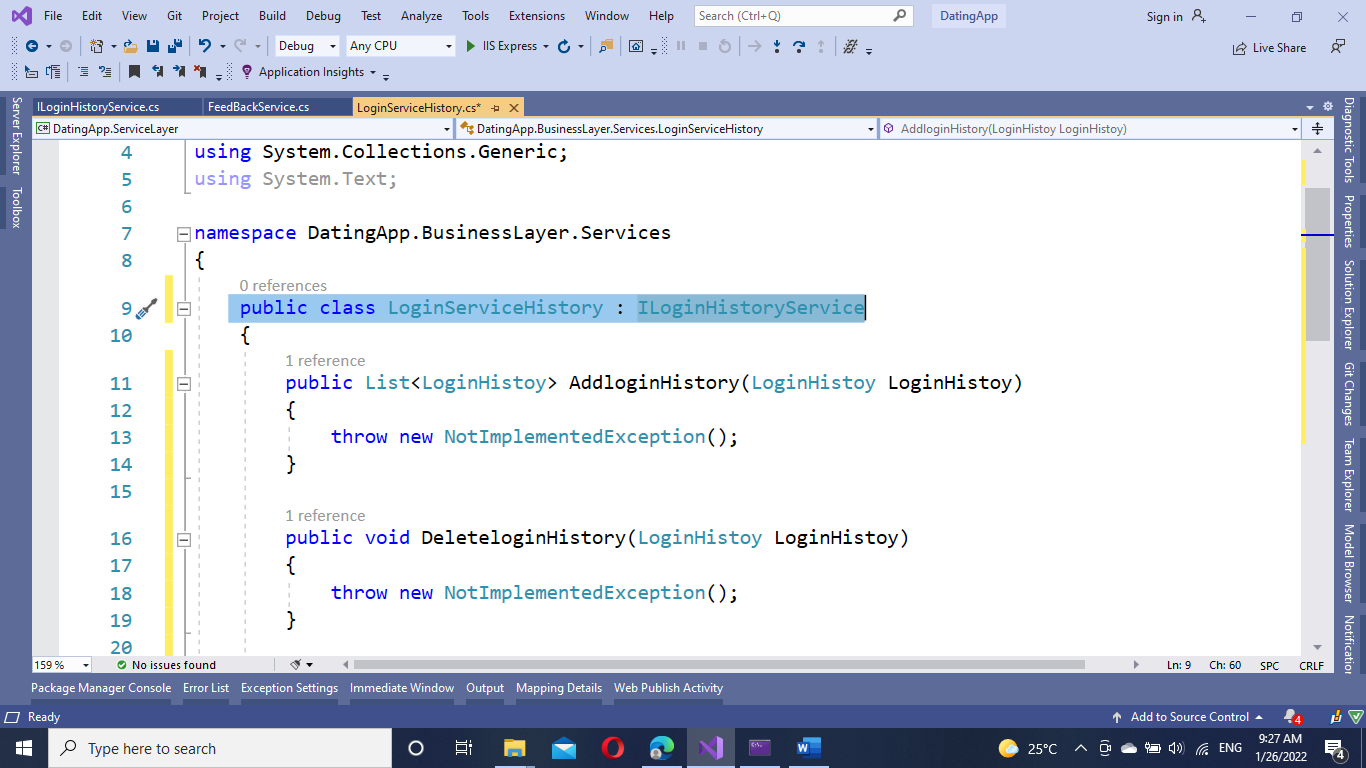
}

}

Add new class to implement the interface,



Implement interface,



Add dependency injection,

/// <summary>

/// Creating instance/field of DateRepository and injecting into DateService Constructor

/// </summary>

private readonly ILoginHistoryRepo \_loginHistoryRepository;

public LoginServiceHistory(ILoginHistoryRepo loginHistoryRepository)

{

\_loginHistoryRepository = loginHistoryRepository;

}

Implement the business logic code into the methods,

Add method,

public List<LoginHistoy> AddloginHistory(LoginHistoy LoginHistoy)

{

List<LoginHistoy> loginHistory = new List<LoginHistoy>();

\_loginHistoryRepository.AddLoginHistory(LoginHistoy);

return loginHistory;

}

Delete Method,

public void DeleteloginHistory(LoginHistoy LoginHistoy)

{

\_loginHistoryRepository.DeleteLoginHistory(LoginHistoy);

}

Edit method,

public async Task<LoginHistoy> EditloginHistory(LoginHistoy LoginHistoy)

{

var result = await \_loginHistoryRepository.UpdateLoginHistory(LoginHistoy);

return result;

}

Get Method,

public async Task<LoginHistoy> GetloginHistoryAsync(long id)

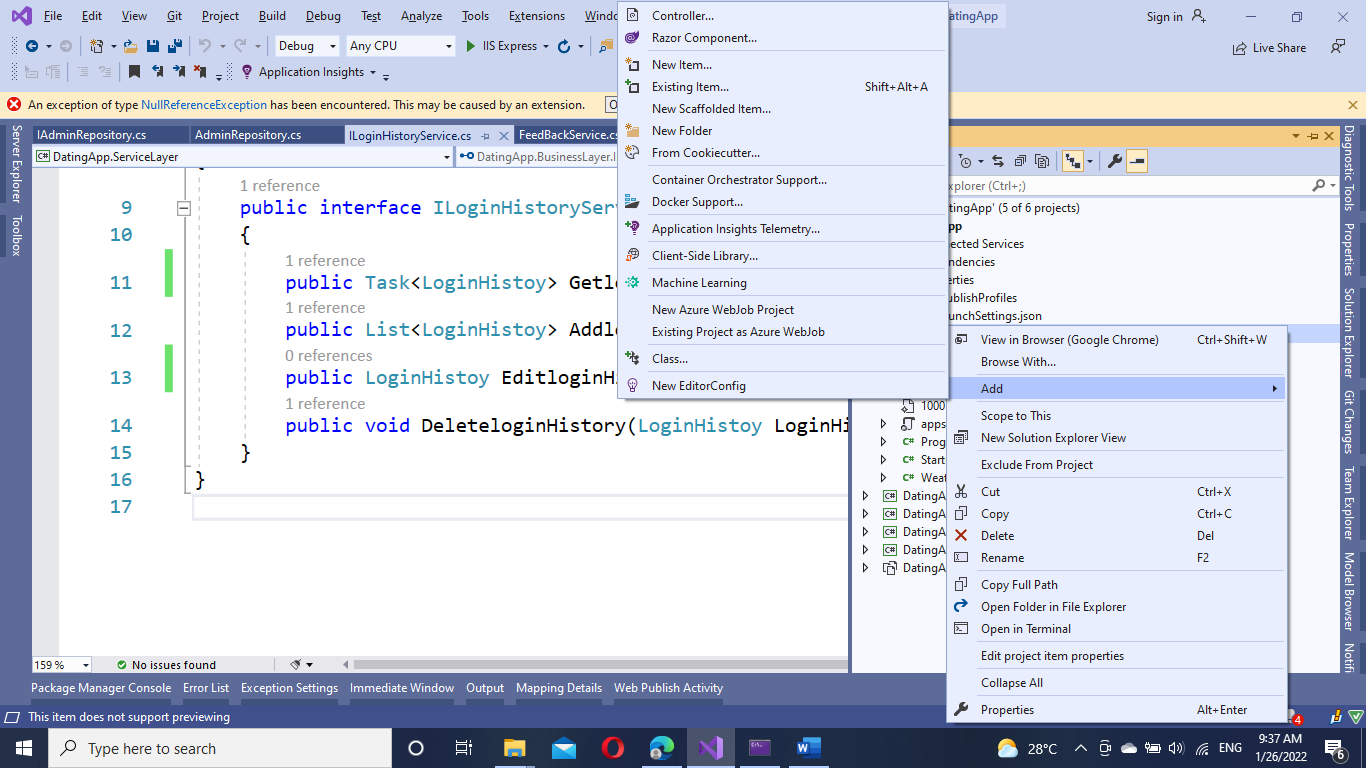
{

var result = await \_loginHistoryRepository.GetLoginHistoryById(id);

return result;

}

Add New Controller:



Add dependency injection,

public class LoginHistoryController : Controller

{

private readonly ILoginHistoryService \_loginHistoryServices;

public LoginHistoryController(ILoginHistoryService loginHistoryServices)

{

\_loginHistoryServices = loginHistoryServices;

}

public IActionResult Index()

{

return View();

}

}

Implement the methods,

[HttpGet]

[Route("api/GetFeedBacks")]

public async Task<LoginHistoy> GetFeedBacks(long id)

{

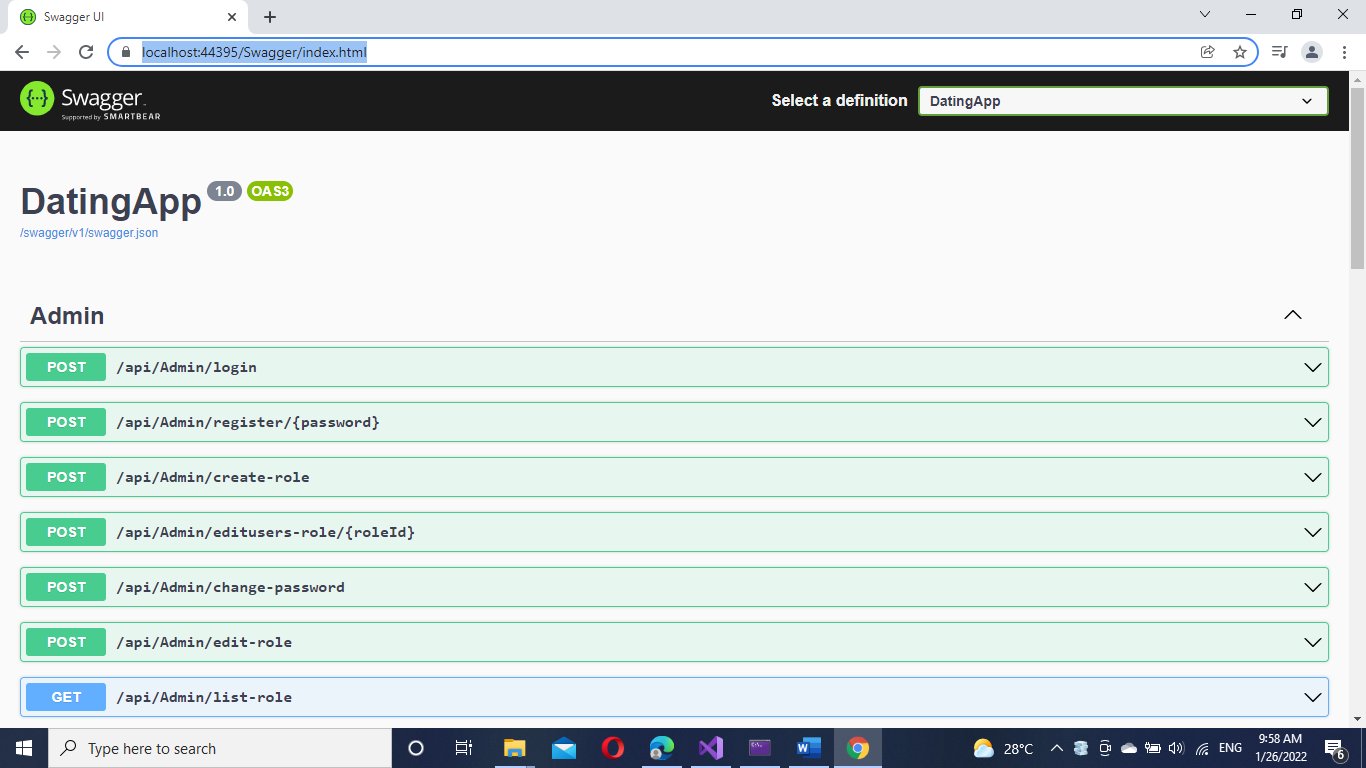
LoginHistoy feedBack = new LoginHistoy();

var feedbacks = await \_loginHistoryServices.GetloginHistoryAsync(id);

return feedbacks;

}

Add all the methods in the controller and run the application,



Check the newly created Api,

