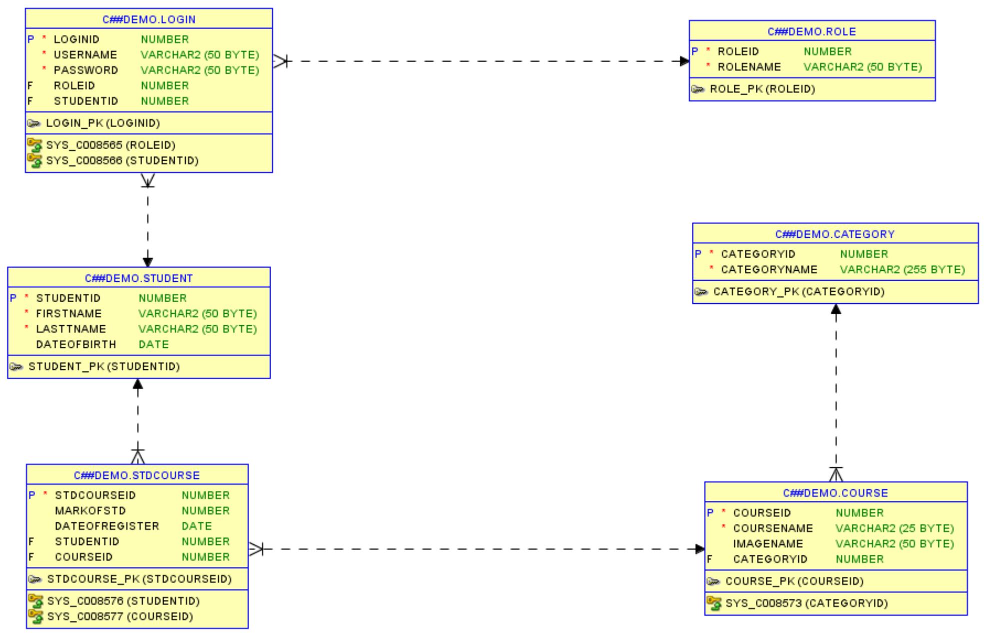
Thank you so much for your continuous effort and I hope you have a nice day.

Task 1:

This is the class diagram:



Task 2:

And here all the packages implementation

****

```
create or replace PACKAGE course_package
as
PROCEDURE GetAllCourses;
PROCEDURE createcourse(course_name in course.coursename%TYPE ,image_name in
course.imagename%TYPE , category_id in course.categoryid%TYPE );
PROCEDURE updatecourse(course_id in course.courseid%TYPE, course_name in
course.coursename%TYPE ,image_name in course.imagename%TYPE , category_id in
course.categoryid%TYPE );
PROCEDURE deletecourse(course_id in course.courseid%TYPE);
PROCEDURE  getcoursebyid (course_id in NUMBER);
end;

create or replace PACKAGE body course_package
as
PROCEDURE GetAllCourses
as
cur_all SYS_REFCURSOR ;

BEGIN
open cur_all for
SELECT * FROM course;
dbms_sql.return_result(cur_all);

end GetAllCourses;
PROCEDURE createcourse(course_name in course.coursename%TYPE ,image_name in
course.imagename%TYPE , category_id in course.categoryid%TYPE )
as
begin
INSERT INTO course VALUES (default , course_name ,image_name ,category_id);
commit;
end createcourse;
PROCEDURE updatecourse(course_id in course.courseid%TYPE, course_name in
course.coursename%TYPE ,image_name in course.imagename%TYPE , category_id in
course.categoryid%TYPE )
as
begin
UPDATE course
  SET
      coursename = course_name,
      imagename = image_name,
      categoryid = category_id
  where
     courseid = course_id ;
     COMMIT;

end updatecourse;
PROCEDURE deletecourse(course_id in course.courseid%TYPE)
```

```
as
BEGIN
DELETE FROM course
WHERE courseid = course_id;

end deletecourse;
PROCEDURE  getcoursebyid (course_id in NUMBER)
as
cur_item SYS_REFCURSOR;
begin
open cur_item for SELECT
   * FROM course
   WHERE courseid = course_id;
   dbms_sql.return_result(cur_item);
end getcoursebyid;
end course_package;



/*****************************************/


create or replace PACKAGE category_package
as
PROCEDURE GetAllcategory;
PROCEDURE makecategory(category_name in category.categoryname%TYPE );
PROCEDURE updatecategory(category_id in category.categoryid%TYPE, category_name in
category.categoryname%TYPE );
PROCEDURE deletecategory(category_id in category.categoryid%TYPE);
PROCEDURE  getcategorybyid (category_id in NUMBER);
end;

create or replace PACKAGE body category_package
as
PROCEDURE GetAllcategory

as
cur_all SYS_REFCURSOR ;

BEGIN
open cur_all for
SELECT * FROM category;
dbms_sql.return_result(cur_all);
```

```
end GetAllcategory;
PROCEDURE makecategory(category_name in category.categoryname%TYPE )
as
begin
INSERT INTO category VALUES (default , category_name );
commit;
end makecategory;
PROCEDURE updatecategory(category_id in category.categoryid%TYPE, category_name in
category.categoryname%TYPE )
as
begin
UPDATE category
  SET
      categoryname =category_name
  where
     categoryid = category_id ;
     COMMIT;
end updatecategory;
PROCEDURE deletecategory(category_id in category.categoryid%TYPE)

as
begin
DELETE FROM category
WHERE categoryid = category_id;
end deletecategory;
PROCEDURE  getcategorybyid (category_id in NUMBER)
as
cur_item SYS_REFCURSOR;
begin
open cur_item for SELECT
    * FROM category
    WHERE categoryid =category_id;
    dbms_sql.return_result(cur_item);
end getcategorybyid;
end category_package;


/***********************************************/

begin
student_package.getstudentbyfirstname('saif');
end;

create or replace PACKAGE login_package
```

```
as
PROCEDURE GetAlllogin;
PROCEDURE makelogin(user_name in login.username%TYPE ,log_password in
login.password%TYPE ,role_id in login.roleid%TYPE , student_id in login.studentid%TYPE );
PROCEDURE updatelogin(login_id in login.loginid%TYPE, user_name in
login.username%TYPE ,log_password in login.password%TYPE, role_id in login.roleid%TYPE ,
student_id in login.studentid%TYPE );
PROCEDURE deletelogin(login_id in login.loginid%TYPE);
PROCEDURE  getloginbyid (login_id in NUMBER);
end;

create or replace PACKAGE body login_package
as
PROCEDURE GetAlllogin

as
cur_all SYS_REFCURSOR ;

BEGIN
open cur_all for
SELECT * FROM login;
dbms_sql.return_result(cur_all);

end GetAlllogin;
PROCEDURE makelogin(user_name in login.username%TYPE ,log_password in
login.password%TYPE ,role_id in login.roleid%TYPE , student_id in login.studentid%TYPE )
as
begin
INSERT INTO login VALUES (default , user_name ,log_password ,role_id, student_id  );
commit;
end makelogin;
PROCEDURE updatelogin(login_id in login.loginid%TYPE, user_name in
login.username%TYPE ,log_password in login.password%TYPE, role_id in login.roleid%TYPE ,
student_id in login.studentid%TYPE )
as
begin
UPDATE login
  SET
      username = user_name,
     password = log_password,
      roleid = role_id,
      studentid = student_id
  where
    loginid = login_id ;
    COMMIT;

end updatelogin;
PROCEDURE deletelogin(login_id in login.loginid%TYPE)
as
begin
DELETE FROM login
WHERE loginid = login_id;
end deletelogin;
PROCEDURE  getloginbyid (login_id in NUMBER)
as
cur_item SYS_REFCURSOR;
begin
open cur_item for SELECT
   * FROM login
   WHERE loginid =login_id;
   dbms_sql.return_result(cur_item);
end getloginbyid;
end login_package;


/******************************************/


create or replace PACKAGE student_package
as
PROCEDURE GetAllstudent;
PROCEDURE makestudent(first_name in student.firstname%TYPE ,last_name in
student.lasttname%TYPE ,datebirth in student.dateofbirth%TYPE);
PROCEDURE updatestudent(student_id in student.studentid%TYPE ,first_name in
student.firstname%TYPE ,last_name in student.lasttname%TYPE ,datebirth in
student.dateofbirth%TYPE);
PROCEDURE deletestudent(student_id in student.studentid%TYPE);
PROCEDURE  getstudentbyid (student_id in NUMBER);
 PROCEDURE GetStudentByFirstName(first_name in student.firstname%TYPE);
 PROCEDURE GetStudentByBirthDate(datebirth in student.dateofbirth%TYPE);
 PROCEDURE GetStudentFNameAndLName;
 PROCEDURE GetStudentBetweenInterval(datefrom in student.dateofbirth%TYPE , DateTo in
student.dateofbirth%TYPE);
end;

create or replace PACKAGE body student_package
as
PROCEDURE GetAllstudent
```

```
as
cur_all SYS_REFCURSOR ;

BEGIN
open cur_all for
SELECT * FROM student;
dbms_sql.return_result(cur_all);

end GetAllstudent;
PROCEDURE makestudent(first_name in student.firstname%TYPE ,last_name in
student.lasttname%TYPE,datebirth in student.dateofbirth%TYPE)
as
begin
INSERT INTO student VALUES (default , first_name ,last_name ,datebirth);
commit;
end makestudent;
PROCEDURE updatestudent(student_id in student.studentid%TYPE ,first_name in
student.firstname%TYPE ,last_name in student.lasttname%TYPE ,datebirth in
student.dateofbirth%TYPE)
as
begin
UPDATE student
  SET
      firstname =first_name,
      lasttname = last_name,
     dateofbirth = datebirth
  where
     studentid = student_id ;
     COMMIT;
end updatestudent;
PROCEDURE deletestudent(student_id in student.studentid%TYPE)
as
begin
DELETE FROM student
WHERE studentid = student_id;
end deletestudent;
PROCEDURE  getstudentbyid (student_id in NUMBER)
as
cur_item SYS_REFCURSOR;
begin
open cur_item for SELECT
    * FROM student
    WHERE studentid = student_id;
    dbms_sql.return_result(cur_item);
end getstudentbyid;
 PROCEDURE GetStudentByFirstName(first_name in student.firstname%TYPE)
 as
 cur_all SYS_REFCURSOR ;
    BEGIN
    open cur_all for
        SELECT * FROM student WHERE firstname = first_name;
        dbms_sql.return_result(cur_all);
    END GetStudentByFirstName;
 PROCEDURE GetStudentByBirthDate(datebirth in student.dateofbirth%TYPE)
 as
    cur_all SYS_REFCURSOR ;
    BEGIN
     open cur_all for
        SELECT * FROM student WHERE dateofbirth = datebirth;
        dbms_sql.return_result(cur_all);
    END GetStudentByBirthDate;
PROCEDURE GetStudentFNameAndLName

AS

c_all sys_refcursor;

BEGIN

OPEN c_all FOR

SELECT FirstName,LasttName FROM Student;

DBMS_SQL.RETURN_RESULT(c_all);

END GetStudentFNameAndLName;
PROCEDURE GetStudentBetweenInterval(datefrom in student.dateofbirth%TYPE , DateTo in
student.dateofbirth%TYPE)
as
   c_all SYS_REFCURSOR ;
Begin
   open c_all for
     select * from student
            where dateofbirth >= datefrom and dateofbirth <= dateto;
   dbms_sql.return_result(c_all);

End GetStudentBetweenInterval ;
```

```
procedure GetStudentsWithHighestMarks(NumOfStudent in number)
as
   c_all SYS_REFCURSOR;
Begin
  open c_all for

select * from (select s.* from student s

inner join stdcourse sc

on s.studentid = sc.studentid

order by sc.markofstd desc)

where Rownum <= NumOfStudent;

Dbms_sql.return_result(c_all);

End GetStudentsWithHighestMarks;

end student_package;


/******************************/

create or replace PACKAGE stdcourse_package
as
PROCEDURE GetAllstdcourse;
PROCEDURE makestdcourse(mark in stdcourse.markofstd %TYPE ,dateofreg in
stdcourse.dateofregister%TYPE ,student_id in stdcourse.studentid%TYPE ,course_id in
stdcourse.courseid%TYPE);
PROCEDURE updatestdcourse(stdcourse_id in stdcourse.stdcourseid%TYPE ,mark in
stdcourse.markofstd %TYPE ,dateofreg in stdcourse.dateofregister%TYPE ,student_id in
stdcourse.studentid%TYPE ,course_id in stdcourse.courseid%TYPE);
PROCEDURE deletestddcourse(stdcourse_id in stdcourse.stdcourseid%TYPE);
PROCEDURE  getstdcousebyid (stdcourse_id in NUMBER);

end;
create or replace PACKAGE body stdcourse_package
as
PROCEDURE GetAllstdcourse

as
cur_all SYS_REFCURSOR ;
```

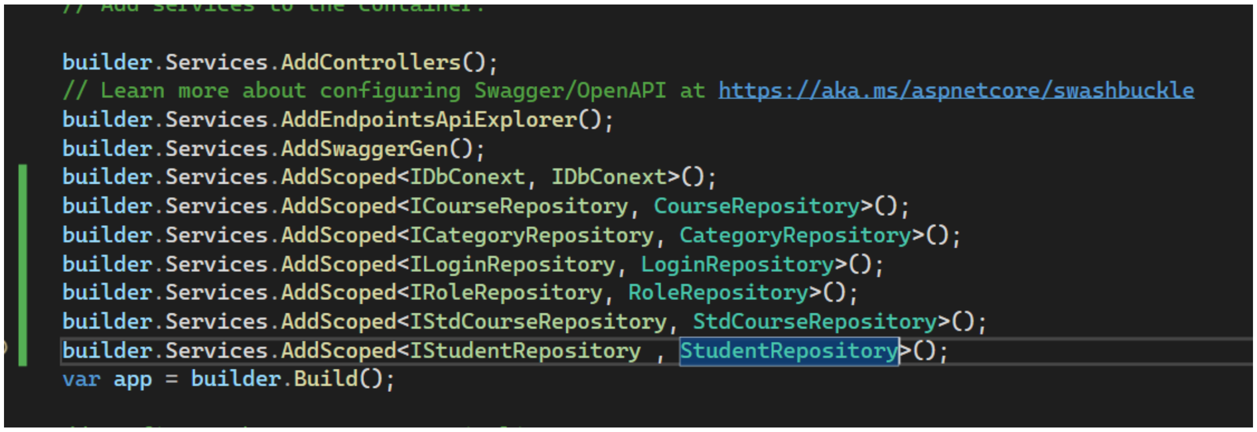
```
BEGIN
open cur_all for
SELECT * FROM stdcourse;
dbms_sql.return_result(cur_all);

end GetAllstdcourse;
PROCEDURE makestdcourse(mark in stdcourse.markofstd %TYPE ,dateofreg in
stdcourse.dateofregister%TYPE ,student_id in stdcourse.studentid%TYPE ,course_id in
stdcourse.courseid%TYPE)
as
begin
INSERT INTO stdcourse VALUES (default , mark ,dateofreg ,student_id ,course_id);
commit;
end makestdcourse;
PROCEDURE updatestdcourse(stdcourse_id in stdcourse.stdcourseid%TYPE ,mark in
stdcourse.markofstd %TYPE ,dateofreg in stdcourse.dateofregister%TYPE ,student_id in
stdcourse.studentid%TYPE ,course_id in stdcourse.courseid%TYPE)
as
begin
UPDATE stdcourse
  SET
      markofstd =mark,
      dateofregister = dateofreg,
     studentid = student_id,
     courseid=course_id
  where
     stdcourseid = stdcourse_id ;
     COMMIT;
end updatestdcourse;
PROCEDURE deletestddcourse(stdcourse_id in stdcourse.stdcourseid%TYPE)
as
begin
DELETE FROM stdcourse
WHERE stdcourseid = stdcourse_id;
end deletestddcourse;
PROCEDURE  getstdcousebyid (stdcourse_id in NUMBER)
as
cur_item SYS_REFCURSOR;
begin
open cur_item for SELECT
    * FROM stdcourse
    WHERE stdcourseid = stdcourse_id;
    dbms_sql.return_result(cur_item);
```

```
end getstdcousebyid;

end stdcourse_package;
/******************************/

create or replace PACKAGE role_package
as
PROCEDURE GetAllrole;
PROCEDURE makerole(role_name in role.rolename%TYPE );
PROCEDURE updaterole(role_id in role.roleid%TYPE, role_name in role.rolename%TYPE );
PROCEDURE deleterole(role_id in role.roleid%TYPE);
PROCEDURE  getrolebyid (role_id in NUMBER);

end;

create or replace PACKAGE body role_package
as
PROCEDURE GetAllrole

as
cur_all SYS_REFCURSOR ;

BEGIN
open cur_all for
SELECT * FROM role;
dbms_sql.return_result(cur_all);

end GetAllrole;
PROCEDURE makerole(role_name in role.rolename%TYPE )
as
begin
INSERT INTO role VALUES (default , role_name );
commit;
end makerole;
PROCEDURE updaterole(role_id in role.roleid%TYPE, role_name in role.rolename%TYPE )
as
begin
UPDATE role
  SET
      rolename =role_name
  where
     roleid = role_id ;
     COMMIT;
end updaterole;
```

```
PROCEDURE deleterole(role_id in role.roleid%TYPE)
as
begin
DELETE FROM role
WHERE roleid = role_id;
end deleterole;
PROCEDURE  getrolebyid (role_id in NUMBER)
as
cur_item SYS_REFCURSOR;
begin
open cur_item for SELECT
   * FROM role
   WHERE roleid =role_id;
   dbms_sql.return_result(cur_item);
end getrolebyid;
end role_package;
```

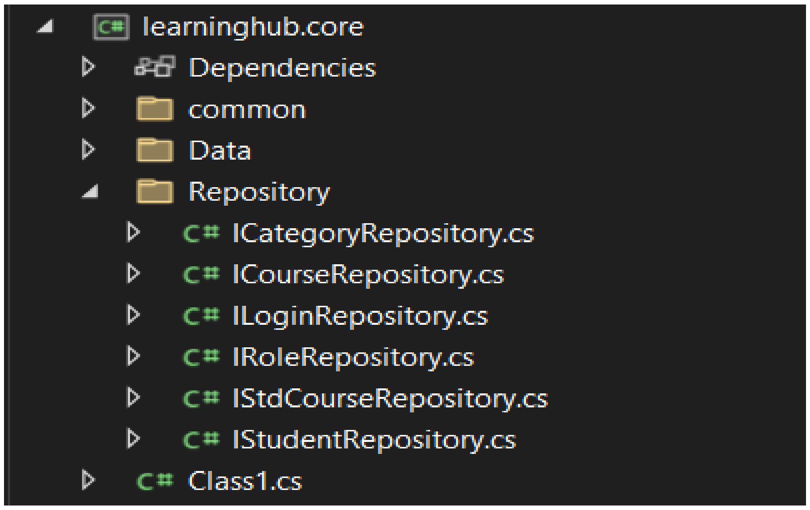
task3

This is in program.cs



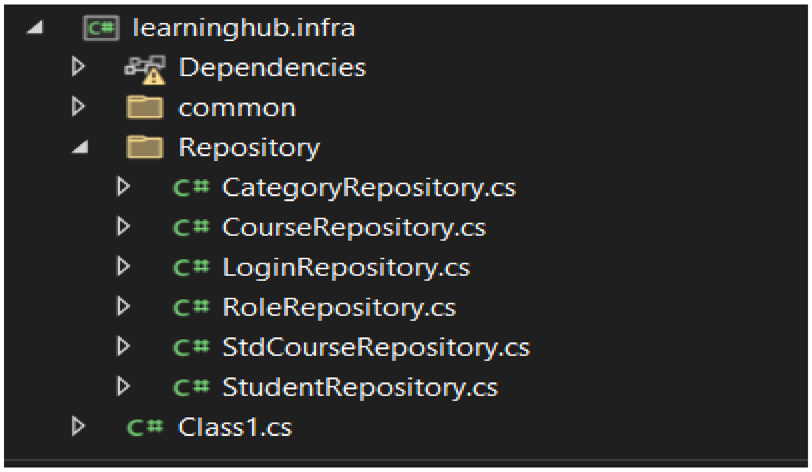
```
// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddScoped<IDbConext, IDbConext>();
builder.Services.AddScoped<ICourseRepository, CourseRepository>();
builder.Services.AddScoped<ICategoryRepository, CategoryRepository>();
builder.Services.AddScoped<ILoginRepository, LoginRepository>();
builder.Services.AddScoped<IRoleRepository, RoleRepository>();
builder.Services.AddScoped<IStdCourseRepository, StdCourseRepository>();
builder.Services.AddScoped<IStudentRepository , StudentRepository>();
var app = builder.Build();
```

this is interface repository



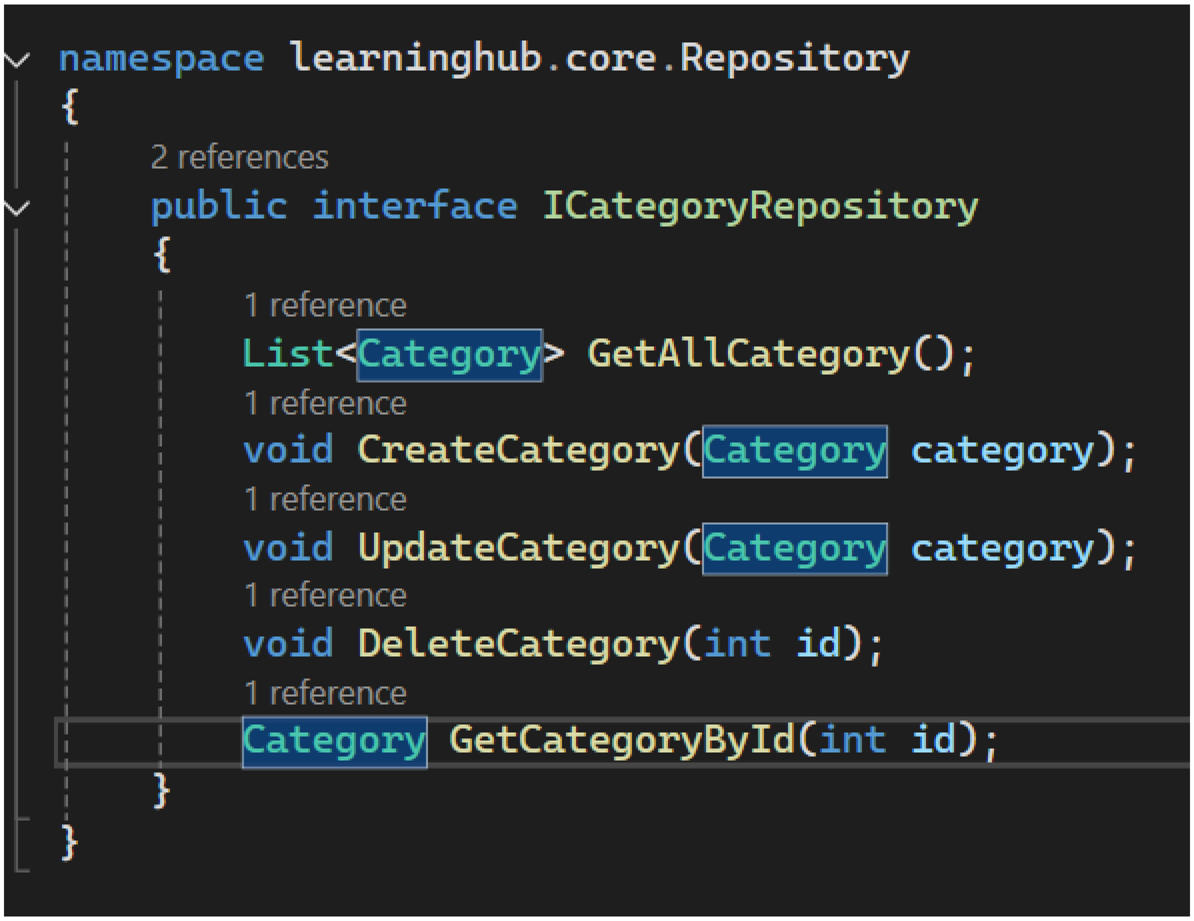
```
◢ learninghub.core
    ▷ Dependencies
    ▷ common
    ▷ Data
    ◢ Repository
        ▷ C# ICategoryRepository.cs
        ▷ C# ICourseRepository.cs
        ▷ C# ILoginRepository.cs
        ▷ C# IRoleRepository.cs
        ▷ C# IStdCourseRepository.cs
        ▷ C# IStudentRepository.cs
    ▷ C# Class1.cs
```

this is class repository:



```
◢ learninghub.infra
    ▷ Dependencies
    ▷ common
    ◢ Repository
        ▷ C# CategoryRepository.cs
        ▷ C# CourseRepository.cs
        ▷ C# LoginRepository.cs
        ▷ C# RoleRepository.cs
        ▷ C# StdCourseRepository.cs
        ▷ C# StudentRepository.cs
    ▷ C# Class1.cs
```

ICategoryRepository interface

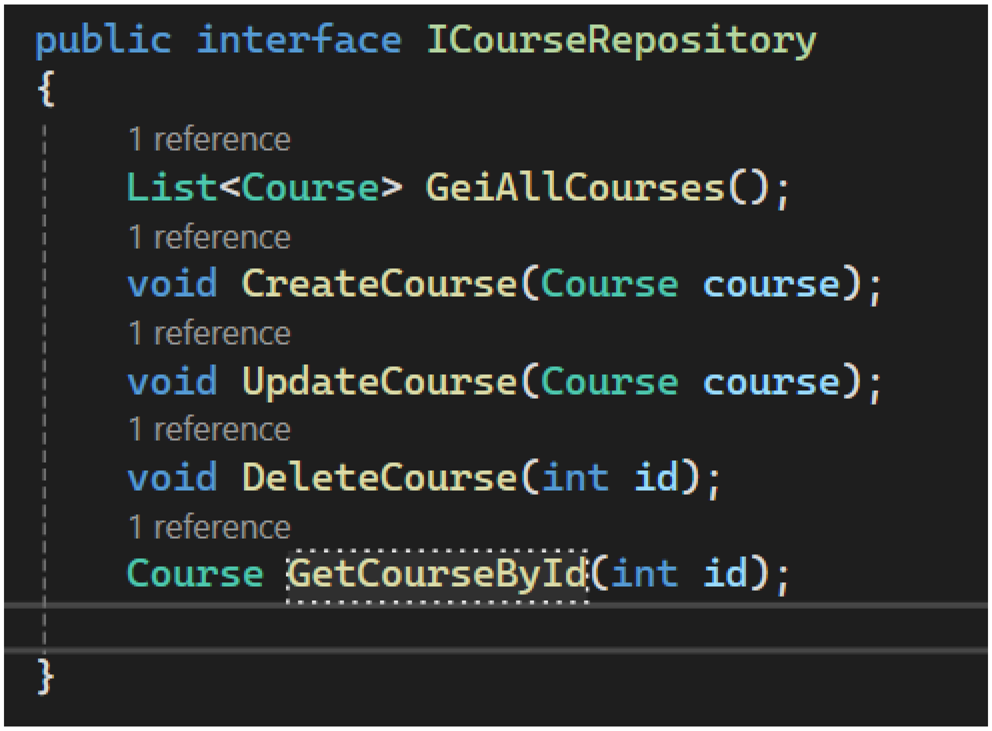


```
namespace learninghub.core.Repository
{
    2 references
    public interface ICategoryRepository
    {
        1 reference
        List<Category> GetAllCategory();
        1 reference
        void CreateCategory(Category category);
        1 reference
        void UpdateCategory(Category category);
        1 reference
        void DeleteCategory(int id);
        1 reference
        Category GetCategoryById(int id);
    }
}
```

This is implementation for this interface:

```
public class CategoryRepository : ICategoryRepository
{
    private readonly IDbConext _dbConext;

    public CategoryRepository(IDbConext dbConext)
    {

        _dbConext = dbConext;

    }
    public List<Category> GetAllCategory()
    {
        IEnumerable<Category> result = _dbConext.connection.Query<Category>
          ("category_package.GetAllcategory", commandType: CommandType.StoredProcedure);

        return result.ToList();
    }
    public void CreateCategory(Category category)
    {
        var p = new DynamicParameters();
        p.Add("category_name", category.Categoryname, dbType: DbType.String, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("category_package.makecategory", p,
commandType: CommandType.StoredProcedure);
    }

    public void DeleteCategory(int id)
    {
        var p = new DynamicParameters();
        p.Add("category_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);
        var result = _dbConext.connection.Execute("category_package.deletecategory", p,
commandType: CommandType.StoredProcedure);

    }

    public void UpdateCategory(Category category)
    {
        var p = new DynamicParameters();
        p.Add("category_id", category.Categoryid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        p.Add("category_name", category.Categoryname, dbType: DbType.String, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("category_package.updatecategory", p,
commandType: CommandType.StoredProcedure);
    }

    public Category GetCategoryById(int id)
    {
        var p = new DynamicParameters();
        p.Add("category_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);

        IEnumerable<Category> result = _dbConext.connection.Query<Category>
            ("category_package.getcategorybyid", p, commandType:
CommandType.StoredProcedure);

        return result.FirstOrDefault();
    }


 }
```

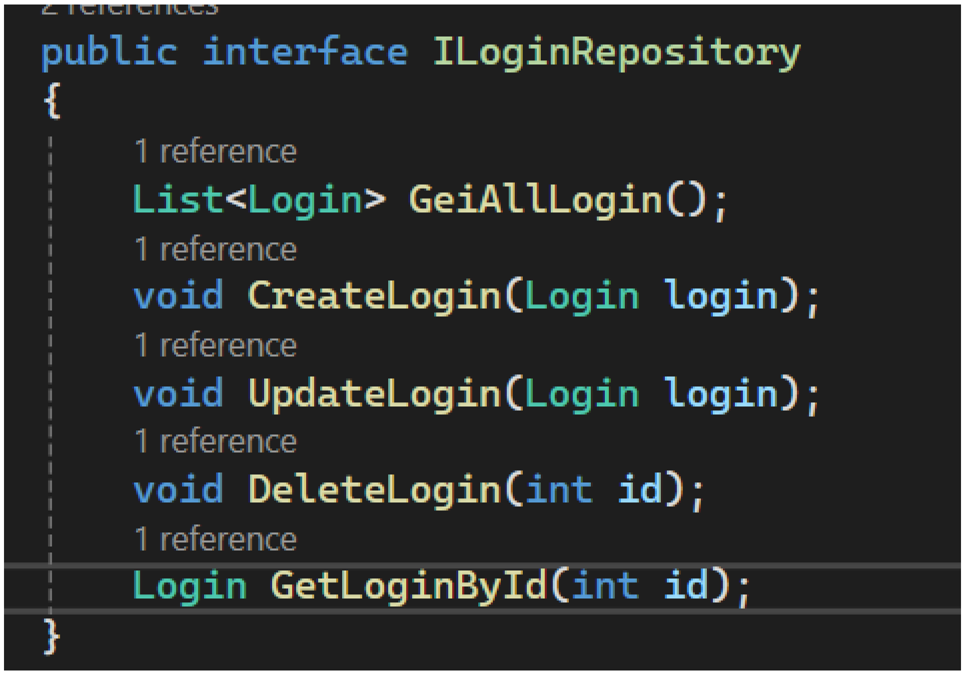
ICourseRepository interface



```
public interface ICourseRepository
{
    1 reference
    List<Course> GeiAllCourses();
    1 reference
    void CreateCourse(Course course);
    1 reference
    void UpdateCourse(Course course);
    1 reference
    void DeleteCourse(int id);
    1 reference
    Course GetCourseById(int id);

}
```

This is implementation for this interface:

```
public class CourseRepository : ICourseRepository
{
    private readonly IDbConext _dbConext;

    public CourseRepository(IDbConext dbConext)
    {

        _dbConext = dbConext;

    }
    public List<Course> GeiAllCourses()
    {
        IEnumerable<Course> result = _dbConext.connection.Query<Course>
            ("course_package.GetAllCourses", commandType: CommandType.StoredProcedure);

        return result.ToList();
    }


   public void CreateCourse(Course course)
    {
        var p = new DynamicParameters();
        p.Add("course_name",course.Coursename , dbType: DbType.String ,
direction:ParameterDirection.Input);
        p.Add("image_name", course.Imagename, dbType: DbType.String, direction:
ParameterDirection.Input);
        p.Add("category_id", course.Categoryid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("course_package.createcourse", p,
commandType: CommandType.StoredProcedure);
    }

    public void DeleteCourse(int id)
    {
        var p = new DynamicParameters();
        p.Add("course_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);
        var result = _dbConext.connection.Execute("course_package.deletecourse", p,
commandType: CommandType.StoredProcedure);

    }

    public Course GetCourseById(int id)
    {
        var p = new DynamicParameters();
        p.Add("course_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);

        IEnumerable<Course> result = _dbConext.connection.Query<Course>
            ("course_package.getcoursebyid",p, commandType: CommandType.StoredProcedure);

        return result.FirstOrDefault();
    }

    public void UpdateCourse(Course course)
    {
        var p = new DynamicParameters();
        p.Add("course_id", course.Courseid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        p.Add("course_name", course.Coursename, dbType: DbType.String, direction:
ParameterDirection.Input);
        p.Add("image_name", course.Imagename, dbType: DbType.String, direction:
ParameterDirection.Input);
        p.Add("category_id", course.Categoryid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("course_package.updatecourse", p,
commandType: CommandType.StoredProcedure);     } }
```
ILoginRepository interface



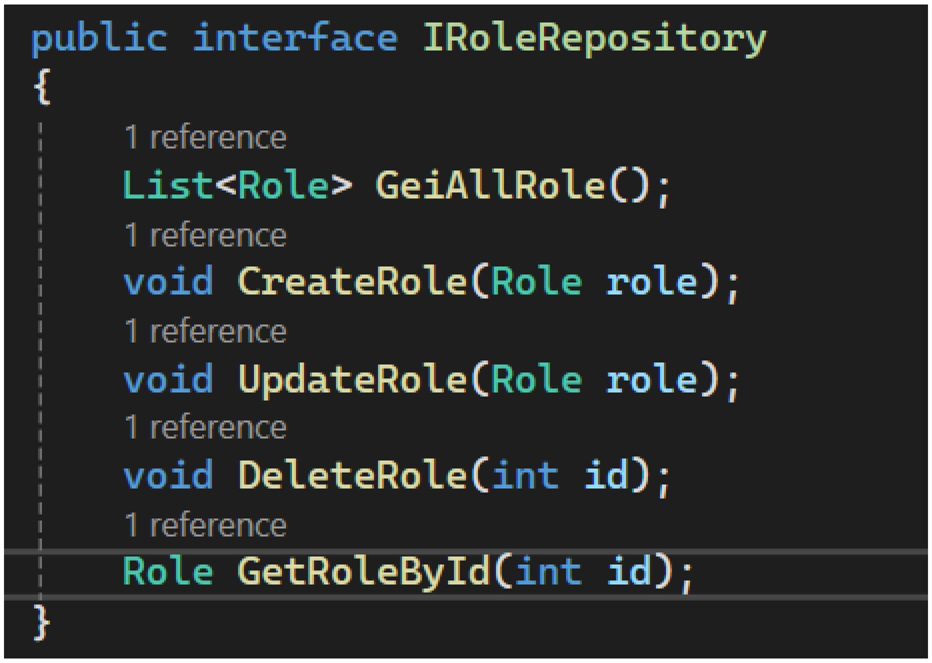
```
2 references
public interface ILoginRepository
{
    1 reference
    List<Login> GeiAllLogin();
    1 reference
    void CreateLogin(Login login);
    1 reference
    void UpdateLogin(Login login);
    1 reference
    void DeleteLogin(int id);
    1 reference
    Login GetLoginById(int id);
}
```

This is implementation for this interface:

```
public class LoginRepository : ILoginRepository
{
   private readonly IDbConext _dbConext;

   public LoginRepository(IDbConext dbConext)
   {

      _dbConext = dbConext;

   }
   public List<Login> GeiAllLogin()
   {
      IEnumerable<Login> result = _dbConext.connection.Query<Login>
         ("login_package.GetAlllogin", commandType: CommandType.StoredProcedure);

      return result.ToList();
   }

   public void CreateLogin(Login login)
   {
      var p = new DynamicParameters();
      p.Add("user_name", login.Username, dbType: DbType.String, direction:
ParameterDirection.Input);
      p.Add("log_password", login.Password, dbType: DbType.String, direction:
ParameterDirection.Input);
```

```
        p.Add("role_id", login.Roleid, dbType: DbType.Int32, direction: ParameterDirection.Input);
        p.Add("student_id", login.Studentid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("login_package.makelogin", p, commandType:
CommandType.StoredProcedure);
    }

    public void DeleteLogin(int id)
    {
        var p = new DynamicParameters();
        p.Add("login_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);
        var result = _dbConext.connection.Execute("login_package.deletelogin", p, commandType:
CommandType.StoredProcedure);
    }
    public void UpdateLogin(Login login)
    {
        var p = new DynamicParameters();
        p.Add("login_id", login.Loginid, dbType: DbType.Int32, direction: ParameterDirection.Input);
        p.Add("user_name", login.Username, dbType: DbType.String, direction:
ParameterDirection.Input);
        p.Add("log_password", login.Password, dbType: DbType.String, direction:
ParameterDirection.Input);
        p.Add("role_id", login.Roleid, dbType: DbType.Int32, direction: ParameterDirection.Input);
        p.Add("student_id", login.Studentid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("login_package.updatelogin", p,
commandType: CommandType.StoredProcedure);
    }


    public Login GetLoginById(int id)
    {
        var p = new DynamicParameters();
        p.Add("login_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);

        IEnumerable<Login> result = _dbConext.connection.Query<Login>
            ("login_package.getloginbyid", p, commandType: CommandType.StoredProcedure);

        return result.FirstOrDefault();
    }
```

IRoleRepository interface



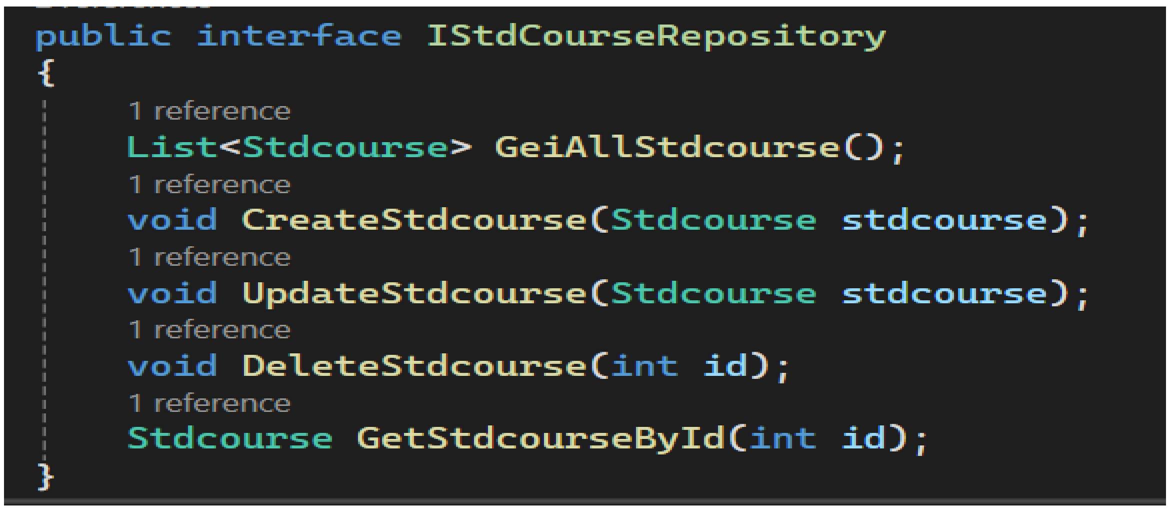
```
public interface IRoleRepository
{
    1 reference
    List<Role> GeiAllRole();
    1 reference
    void CreateRole(Role role);
    1 reference
    void UpdateRole(Role role);
    1 reference
    void DeleteRole(int id);
    1 reference
    Role GetRoleById(int id);
}
```

This is implementation for this interface:

```
 public class RoleRepository : IRoleRepository
 {
     private readonly IDbConext _dbConext;

     public RoleRepository(IDbConext dbConext)
     {

         _dbConext = dbConext;

     }

     public List<Role> GeiAllRole()
     {
         IEnumerable<Role> result = _dbConext.connection.Query<Role>
             ("role_package.GetAllrole", commandType: CommandType.StoredProcedure);

         return result.ToList();
     }
     public void CreateRole(Role role)
     {
         var p = new DynamicParameters();
         p.Add("role_name", role.Rolename, dbType: DbType.String, direction:
ParameterDirection.Input);
         var result = _dbConext.connection.Execute("role_package.makerole", p, commandType:
CommandType.StoredProcedure);
     }
```

```
    public void DeleteRole(int id)
    {
        var p = new DynamicParameters();
        p.Add("role_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);
        var result = _dbConext.connection.Execute("role_package.deleterole", p, commandType:
CommandType.StoredProcedure);
    }

    public void UpdateRole(Role role)
    {
        var p = new DynamicParameters();
        p.Add("role_id", role.Roleid, dbType: DbType.Int32, direction: ParameterDirection.Input);
        p.Add("role_name", role.Rolename, dbType: DbType.String, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("role_package.updaterole", p, commandType:
CommandType.StoredProcedure);
    }

    public Role GetRoleById(int id)
    {
       var p = new DynamicParameters(); p.Add("role_id", id, dbType: DbType.Int32, direction:
ParameterDirection.Input);

        IEnumerable<Role> result = _dbConext.connection.Query<Role>
            ("role_package.getrolebyid", p, commandType: CommandType.StoredProcedure);

        return result.FirstOrDefault();
    }
```

interface IStdCourseRepository

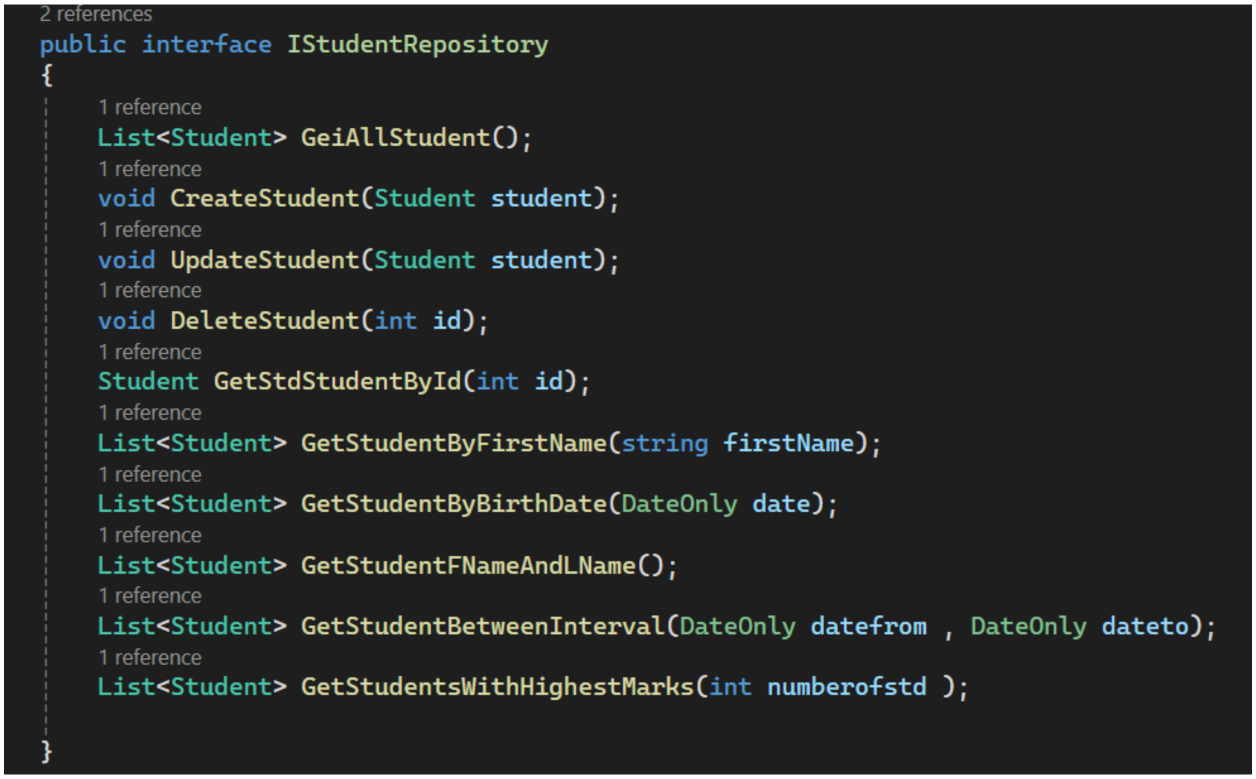


```
public interface IStdCourseRepository
{
        1 reference
        List<Stdcourse> GeiAllStdcourse();
        1 reference
        void CreateStdcourse(Stdcourse stdcourse);
        1 reference
        void UpdateStdcourse(Stdcourse stdcourse);
        1 reference
        void DeleteStdcourse(int id);
        1 reference
        Stdcourse GetStdcourseById(int id);
}
```

This is implementation for this interface:

```
 public class StdCourseRepository : IStdCourseRepository
 {
     private readonly IDbConext _dbConext;

     public StdCourseRepository(IDbConext dbConext)
     {
         _dbConext = dbConext;
     }

     public List<Stdcourse> GeiAllStdcourse()
     {
         IEnumerable<Stdcourse> result = _dbConext.connection.Query<Stdcourse>
           ("stdcourse_package.GetAllstdcourse", commandType:
CommandType.StoredProcedure);

         return result.ToList();
     }
     public void CreateStdcourse(Stdcourse stdcourse)
     {
         var p = new DynamicParameters();
         p.Add("mark", stdcourse.Markofstd, dbType: DbType.Int32, direction:
ParameterDirection.Input);
         p.Add("dateofreg" , stdcourse.Dateofregister , dbType: DbType.Date , direction:
ParameterDirection.Input);
         p.Add("student_id", stdcourse.Studentid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
         p.Add("course_id", stdcourse.Courseid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
         var result = _dbConext.connection.Execute("stdcourse_package.makestdcourse", p,
commandType: CommandType.StoredProcedure);
     }

     public void DeleteStdcourse(int id)
     {
         var p = new DynamicParameters();
         p.Add("stdcourse_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);
         var result = _dbConext.connection.Execute("stdcourse_package.deletestddcourse", p,
commandType: CommandType.StoredProcedure);
     }

     public void UpdateStdcourse(Stdcourse stdcourse)
     {
         var p = new DynamicParameters();
```

```
        p.Add("stdcourse_id", stdcourse.Stdcourseid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        p.Add("mark", stdcourse.Markofstd, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        p.Add("dateofreg", stdcourse.Dateofregister, dbType: DbType.Date, direction:
ParameterDirection.Input);
        p.Add("student_id", stdcourse.Studentid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        p.Add("course_id", stdcourse.Courseid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("stdcourse_package.updatestdcourse", p,
commandType: CommandType.StoredProcedure);
    }

    public Stdcourse GetStdcourseById(int id)
    {
        var p = new DynamicParameters();
        p.Add("stdcourse_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);

        IEnumerable<Stdcourse> result = _dbConext.connection.Query<Stdcourse>
            ("stdcourse_package.getstdcousebyid", p, commandType:
CommandType.StoredProcedure);

        return result.FirstOrDefault();
    } }
```
IStudentRepository interface


```
2 references
public interface IStudentRepository
{
    1 reference
    List<Student> GeiAllStudent();
    1 reference
    void CreateStudent(Student student);
    1 reference
    void UpdateStudent(Student student);
    1 reference
    void DeleteStudent(int id);
    1 reference
    Student GetStdStudentById(int id);
    1 reference
    List<Student> GetStudentByFirstName(string firstName);
    1 reference
    List<Student> GetStudentByBirthDate(DateOnly date);
    1 reference
    List<Student> GetStudentFNameAndLName();
    1 reference
    List<Student> GetStudentBetweenInterval(DateOnly datefrom , DateOnly dateto);
    1 reference
    List<Student> GetStudentsWithHighestMarks(int numberofstd );

}
```

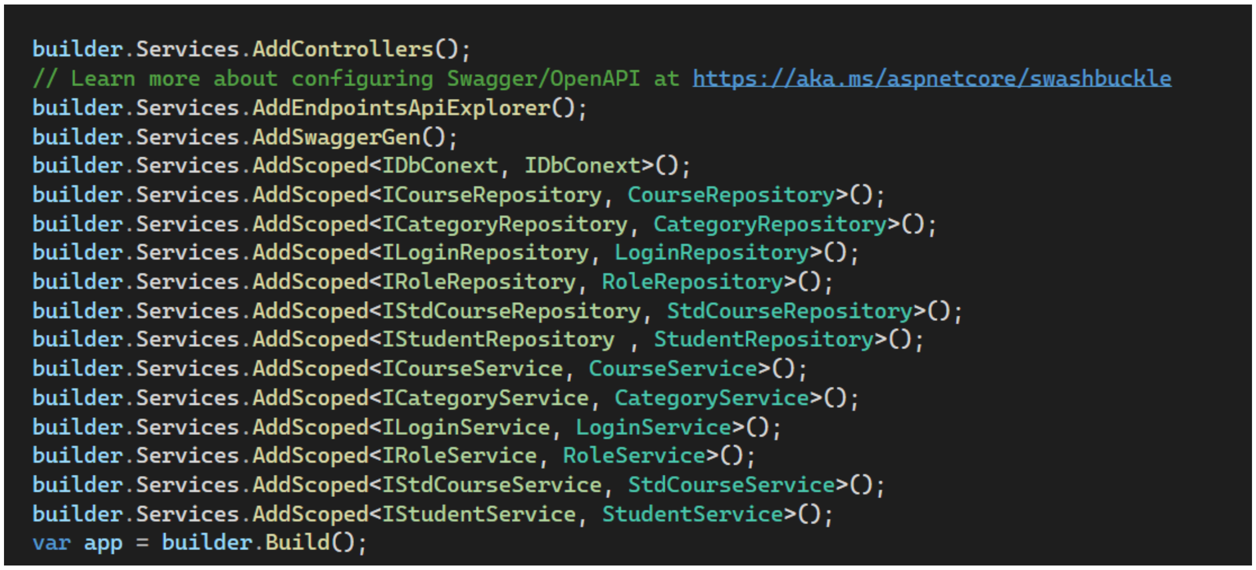
This is implementation for this interface:

```
public class StudentRepository : IStudentRepository
{

    private readonly IDbConext _dbConext;

    public StudentRepository(IDbConext dbConext)
    {

        _dbConext = dbConext;

    }
    public List<Student> GeiAllStudent()
    {
        IEnumerable<Student> result = _dbConext.connection.Query<Student>
          ("student_package.GetAllstudent", commandType: CommandType.StoredProcedure);

        return result.ToList();
    }

    public void CreateStudent(Student student)
    {
        var p = new DynamicParameters();
        p.Add("first_name", student.Firstname, dbType: DbType.String, direction:
ParameterDirection.Input);
        p.Add("last_name", student.Lasttname, dbType: DbType.String, direction:
ParameterDirection.Input);
        p.Add("datebirth", student.Dateofbirth, dbType: DbType.Date, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("student_package.makestudent", p,
commandType: CommandType.StoredProcedure);

    }

    public void DeleteStudent(int id)
    {
        var p = new DynamicParameters();
        p.Add("student_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);
        var result = _dbConext.connection.Execute("student_package.deletestudent", p,
commandType: CommandType.StoredProcedure);

    }
    public void UpdateStudent(Student student)
    {
        var p = new DynamicParameters();
        p.Add("student_id", student.Studentid, dbType: DbType.Int32, direction:
ParameterDirection.Input);
        p.Add("first_name", student.Firstname, dbType: DbType.String, direction:
ParameterDirection.Input);
        p.Add("last_name", student.Lasttname, dbType: DbType.String, direction:
ParameterDirection.Input);
        p.Add("datebirth", student.Dateofbirth, dbType: DbType.Date, direction:
ParameterDirection.Input);
        var result = _dbConext.connection.Execute("student_package.updatestudent", p,
commandType: CommandType.StoredProcedure);

    }

    public Student GetStdStudentById(int id)
    {
        var p = new DynamicParameters();
        p.Add("student_id", id, dbType: DbType.Int32, direction: ParameterDirection.Input);

        IEnumerable<Student> result = _dbConext.connection.Query<Student>
```

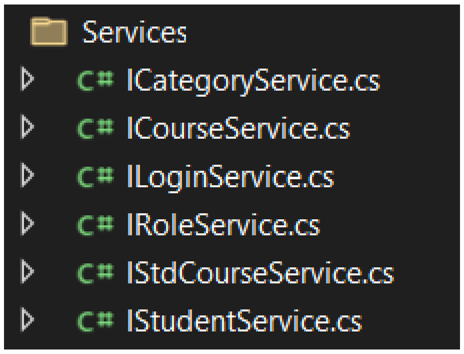
```
          ("student_package.getstudentbyid", p, commandType:
CommandType.StoredProcedure);

        return result.FirstOrDefault();
    }

  public  List<Student> GetStudentByFirstName(string firstName)
    {
        var p = new DynamicParameters();
        p.Add("first_name", firstName, dbType: DbType.String, direction:
ParameterDirection.Input);

        IEnumerable<Student> result = _dbConext.connection.Query<Student>
          ("student_package.GetStudentByFirstName", p, commandType:
CommandType.StoredProcedure);
        return result.ToList();
    }

  public  List<Student> GetStudentByBirthDate(DateOnly date)
  {
        var p = new DynamicParameters();
        p.Add("datebirth",date, dbType: DbType.Date, direction: ParameterDirection.Input);

        IEnumerable<Student> result = _dbConext.connection.Query<Student>
          ("student_package.GetStudentByBirthDate", p, commandType:
CommandType.StoredProcedure);
        return result.ToList();
    }
  public  List<Student> GetStudentFNameAndLName()
  {
        IEnumerable<Student> result = _dbConext.connection.Query<Student>
          ("student_package.GetStudentFNameAndLName",commandType:
CommandType.StoredProcedure);

        return result.ToList();
  }

    public List<Student> GetStudentBetweenInterval(DateOnly datefrom, DateOnly dateto)
    {
        var p = new DynamicParameters();
        p.Add("datebirth",datefrom, dbType: DbType.Date, direction: ParameterDirection.Input);
        p.Add("datebirth", dateto , dbType: DbType.Date, direction: ParameterDirection.Input);

        IEnumerable<Student> result = _dbConext.connection.Query<Student>
```

```
                ("student_package.GetStudentBetweenInterval",p, commandType:
CommandType.StoredProcedure);

        return result.ToList();
    }
    public List<Student> GetStudentsWithHighestMarks(int numberofstd)
    {
        var p = new DynamicParameters();
        p.Add("NumOfStudent", numberofstd, dbType: DbType.Int32, direction:
ParameterDirection.Input);

        IEnumerable<Student> result = _dbConext.connection.Query<Student>
                ("student_package.GetStudentsWithHighestMarks", p, commandType:
CommandType.StoredProcedure);

        return result.ToList();
    }
}
```

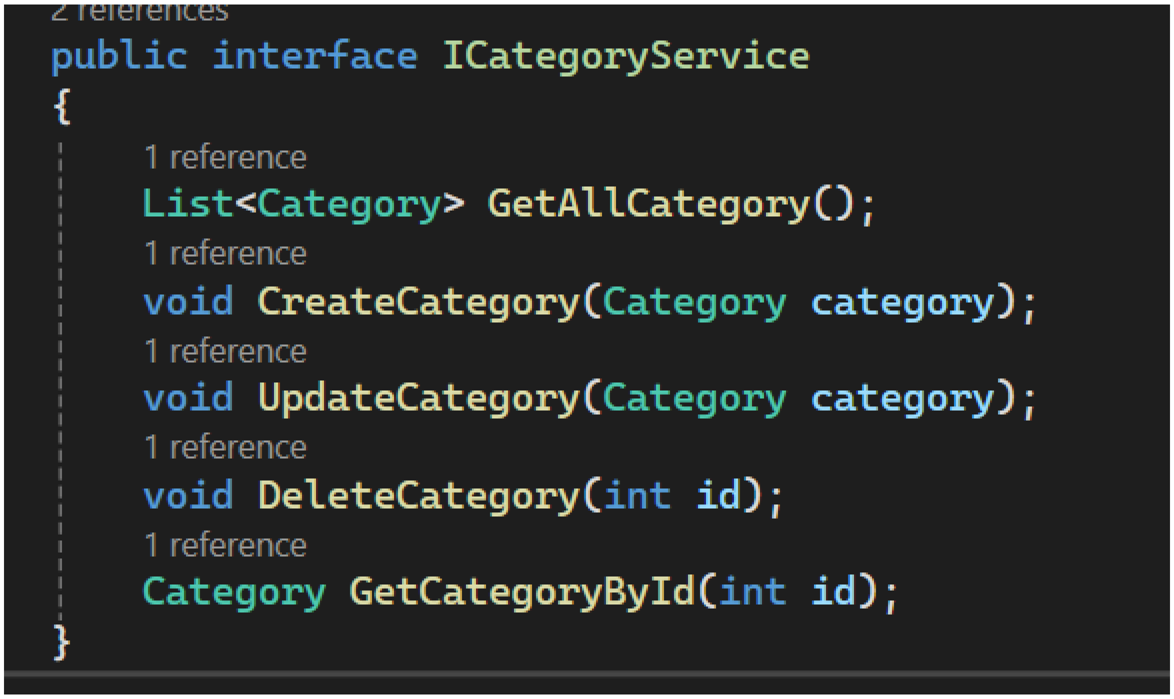
Task 4:



```
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddScoped<IDbConext, IDbConext>();
builder.Services.AddScoped<ICourseRepository, CourseRepository>();
builder.Services.AddScoped<ICategoryRepository, CategoryRepository>();
builder.Services.AddScoped<ILoginRepository, LoginRepository>();
builder.Services.AddScoped<IRoleRepository, RoleRepository>();
builder.Services.AddScoped<IStdCourseRepository, StdCourseRepository>();
builder.Services.AddScoped<IStudentRepository , StudentRepository>();
builder.Services.AddScoped<ICourseService, CourseService>();
builder.Services.AddScoped<ICategoryService, CategoryService>();
builder.Services.AddScoped<ILoginService, LoginService>();
builder.Services.AddScoped<IRoleService, RoleService>();
builder.Services.AddScoped<IStdCourseService, StdCourseService>();
builder.Services.AddScoped<IStudentService, StudentService>();
var app = builder.Build();
```

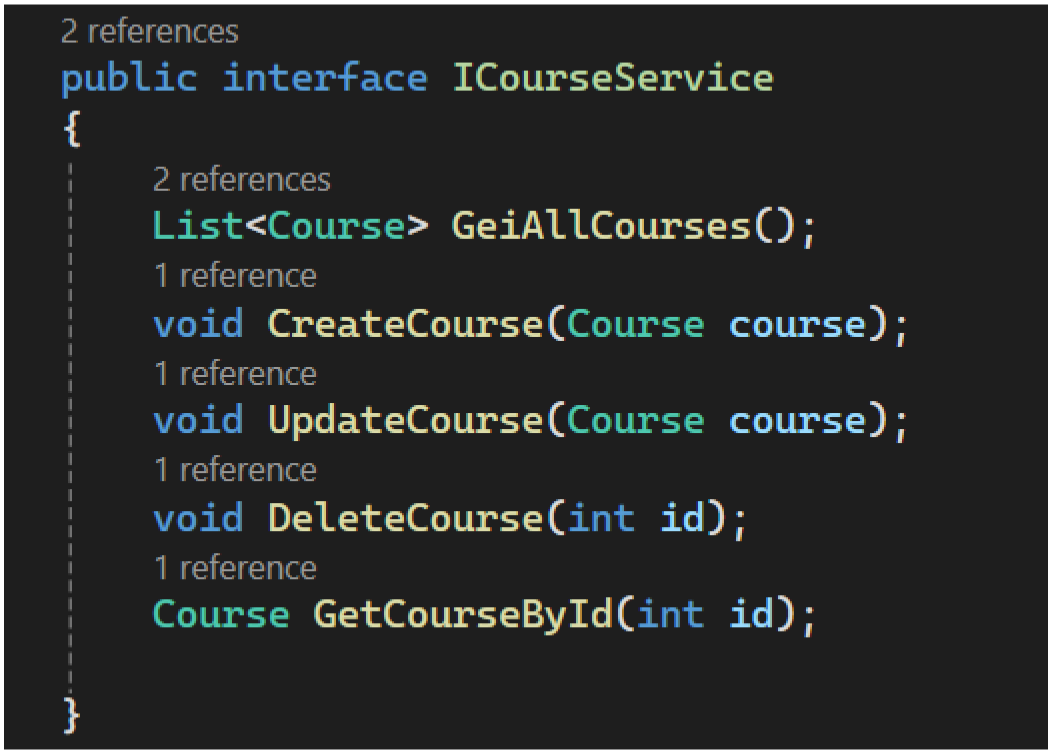


```
Services
  ICategoryService.cs
  ICourseService.cs
  ILoginService.cs
  IRoleService.cs
  IStdCourseService.cs
  IStudentService.cs
```



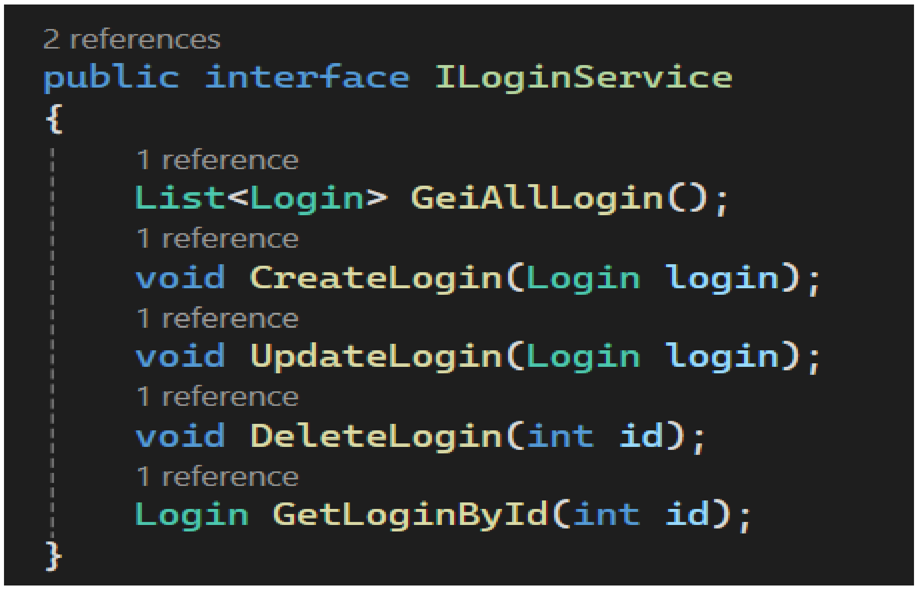
```
2 references
public interface ICategoryService
{
    1 reference
    List<Category> GetAllCategory();
    1 reference
    void CreateCategory(Category category);
    1 reference
    void UpdateCategory(Category category);
    1 reference
    void DeleteCategory(int id);
    1 reference
    Category GetCategoryById(int id);
}
```

```
public class CategoryService : ICategoryService
{
    private readonly ICategoryRepository _categoryRepository;

    public CategoryService(ICategoryRepository categoryRepository)
    {
        _categoryRepository = categoryRepository;
    }
    public void CreateCategory(Category category)
    {
        _categoryRepository.CreateCategory(category);
    }

    public void DeleteCategory(int id)
    {
        _categoryRepository.DeleteCategory(id);
    }

    public List<Category> GetAllCategory()
    {
       return _categoryRepository.GetAllCategory();
    }

    public Category GetCategoryById(int id)
    {
        return _categoryRepository.GetCategoryById(id);
    }

    public void UpdateCategory(Category category)
    {
      _categoryRepository.UpdateCategory(category);
    }
}
```



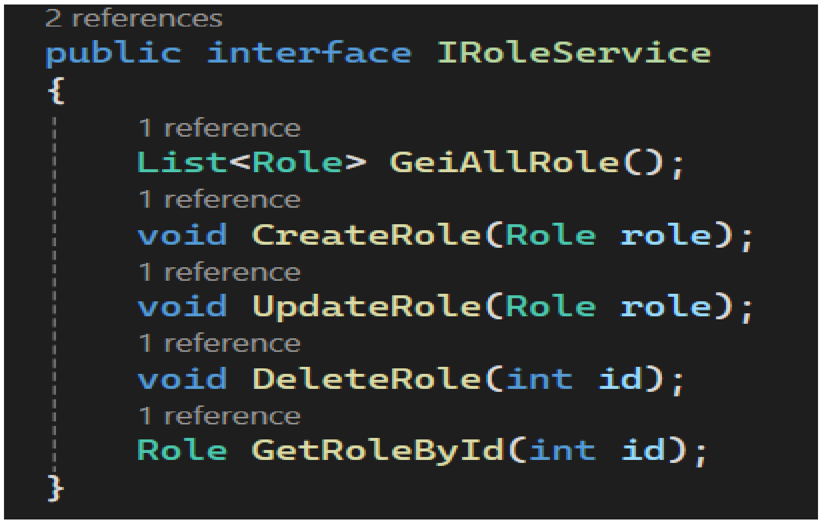
```
2 references
public interface ICourseService
{
    2 references
    List<Course> GeiAllCourses();
    1 reference
    void CreateCourse(Course course);
    1 reference
    void UpdateCourse(Course course);
    1 reference
    void DeleteCourse(int id);
    1 reference
    Course GetCourseById(int id);

}
```

```
public class CourseService: ICourseService
{
    private readonly  ICourseRepository _courseRepository;

    public CourseService(ICourseRepository courseRepository)
    {
       _courseRepository = courseRepository;
    }

    public void CreateCourse(Course course)
    {
      _courseRepository.CreateCourse(course);
    }

    public void DeleteCourse(int id)
    {
     _courseRepository.DeleteCourse(id);
    }

    public List<Course> GeiAllCourses()
    {
        return GeiAllCourses();
    }

    public Course GetCourseById(int id)
    {
      return _courseRepository.GetCourseById(id);
    }

    public void UpdateCourse(Course course)
    {
       _courseRepository.UpdateCourse(course);
    }
}
```



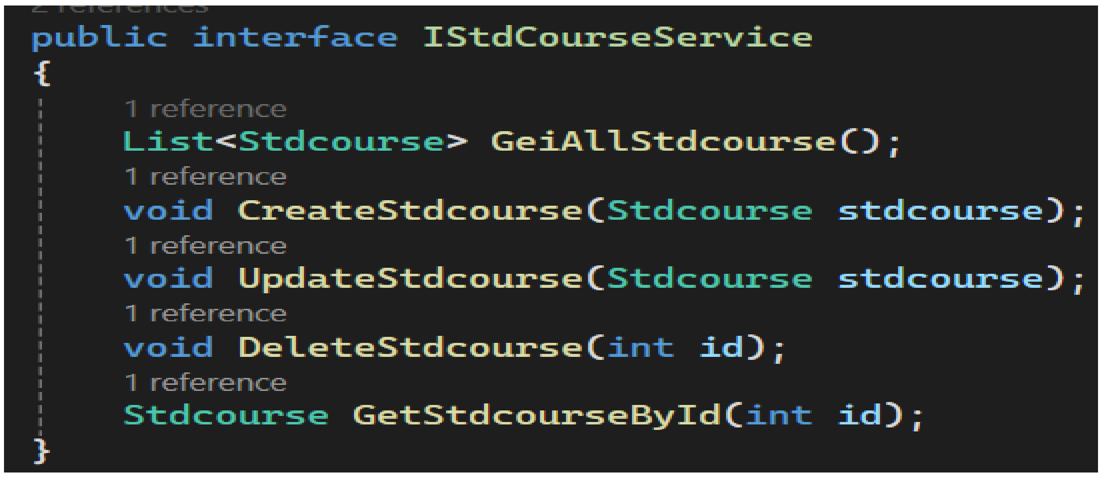
```
2 references
public interface ILoginService
{
    1 reference
    List<Login> GeiAllLogin();
    1 reference
    void CreateLogin(Login login);
    1 reference
    void UpdateLogin(Login login);
    1 reference
    void DeleteLogin(int id);
    1 reference
    Login GetLoginById(int id);
}
```

```
public class LoginService:ILoginService
{
    private readonly ILoginRepository _loginRepository;
    public LoginService(ILoginRepository loginRepository)
    {
        _loginRepository = loginRepository;
    }

    public void CreateLogin(Login login)
    {
        _loginRepository.CreateLogin(login);
    }

    public void DeleteLogin(int id)
    {
        _loginRepository.DeleteLogin(id);
    }

    public List<Login> GeiAllLogin()
    {
       return _loginRepository.GeiAllLogin();
    }

    public Login GetLoginById(int id)
    {
       return _loginRepository.GetLoginById(id);
    }
    public void UpdateLogin(Login login)
    {
       _loginRepository.UpdateLogin(login);}}
```



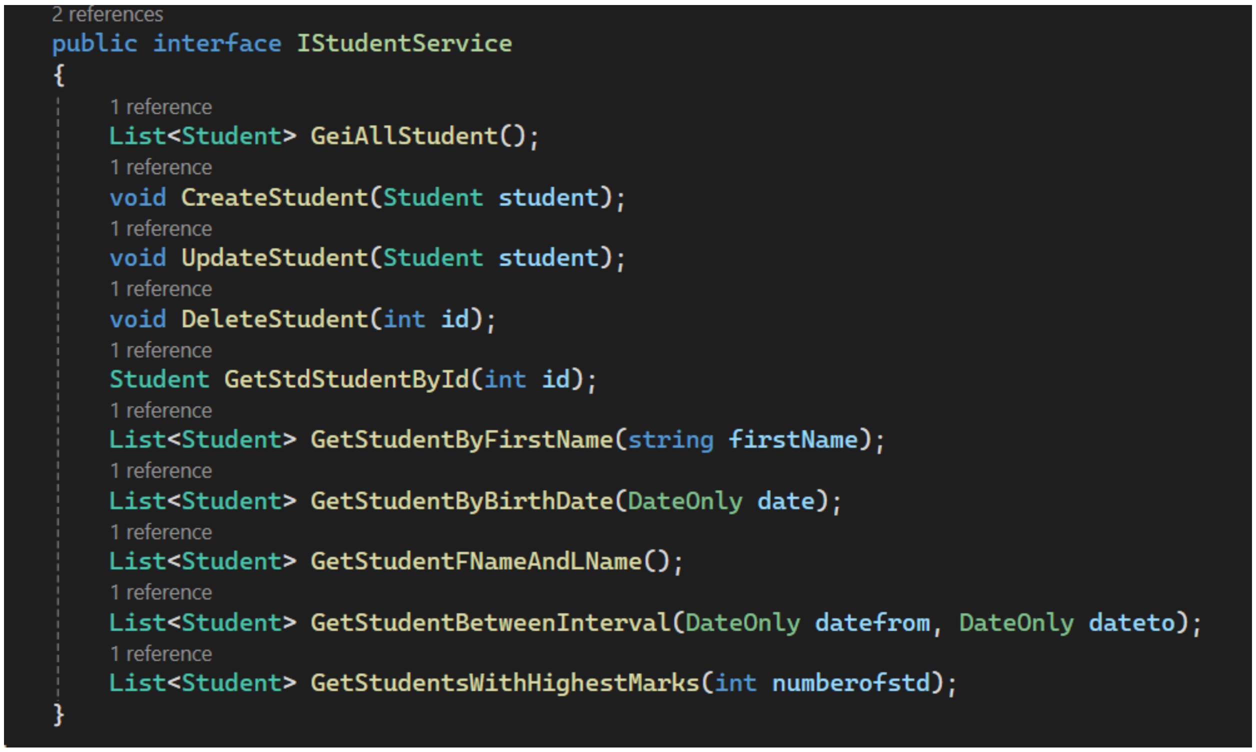
```
2 references
public interface IRoleService
{
    1 reference
    List<Role> GeiAllRole();
    1 reference
    void CreateRole(Role role);
    1 reference
    void UpdateRole(Role role);
    1 reference
    void DeleteRole(int id);
    1 reference
    Role GetRoleById(int id);
}
```

```
public class RoleService: IRoleService
{
    private readonly RoleRepository _roleRepository;
    public RoleService(RoleRepository roleRepository ) {
        _roleRepository = roleRepository;
    }

    public void CreateRole(Role role)
    {
       _roleRepository.CreateRole(role);
    }

    public void DeleteRole(int id)
    {
        _roleRepository.DeleteRole(id);
    }

    public List<Role> GeiAllRole()
    {
      return   _roleRepository.GeiAllRole();
    }

    public Role GetRoleById(int id)
    {
       return _roleRepository.GetRoleById(id);
    }

    public void UpdateRole(Role role)
    {
      _roleRepository.UpdateRole(role);
    }
}
```



```
2 references
public interface IStdCourseService
{
    1 reference
    List<Stdcourse> GeiAllStdcourse();
    1 reference
    void CreateStdcourse(Stdcourse stdcourse);
    1 reference
    void UpdateStdcourse(Stdcourse stdcourse);
    1 reference
    void DeleteStdcourse(int id);
    1 reference
    Stdcourse GetStdcourseById(int id);
}
```

```
public class StdCourseService : IStdCourseService
{
    private readonly StdCourseRepository _StdCourseRepository;
    public StdCourseService(StdCourseRepository stdCourseRepository)
    {
        _StdCourseRepository = stdCourseRepository;
    }
    public void CreateStdcourse(Stdcourse stdcourse)
    {
        _StdCourseRepository.CreateStdcourse(stdcourse);
    }

    public void DeleteStdcourse(int id)
    {
        _StdCourseRepository.DeleteStdcourse(id);
    }

    public List<Stdcourse> GeiAllStdcourse()
    {
      return _StdCourseRepository.GeiAllStdcourse();
    }

    public Stdcourse GetStdcourseById(int id)
    {
      return _StdCourseRepository.GetStdcourseById(id);

    }

    public void UpdateStdcourse(Stdcourse stdcourse)
    {
     _StdCourseRepository.UpdateStdcourse(stdcourse);
    }
}
```



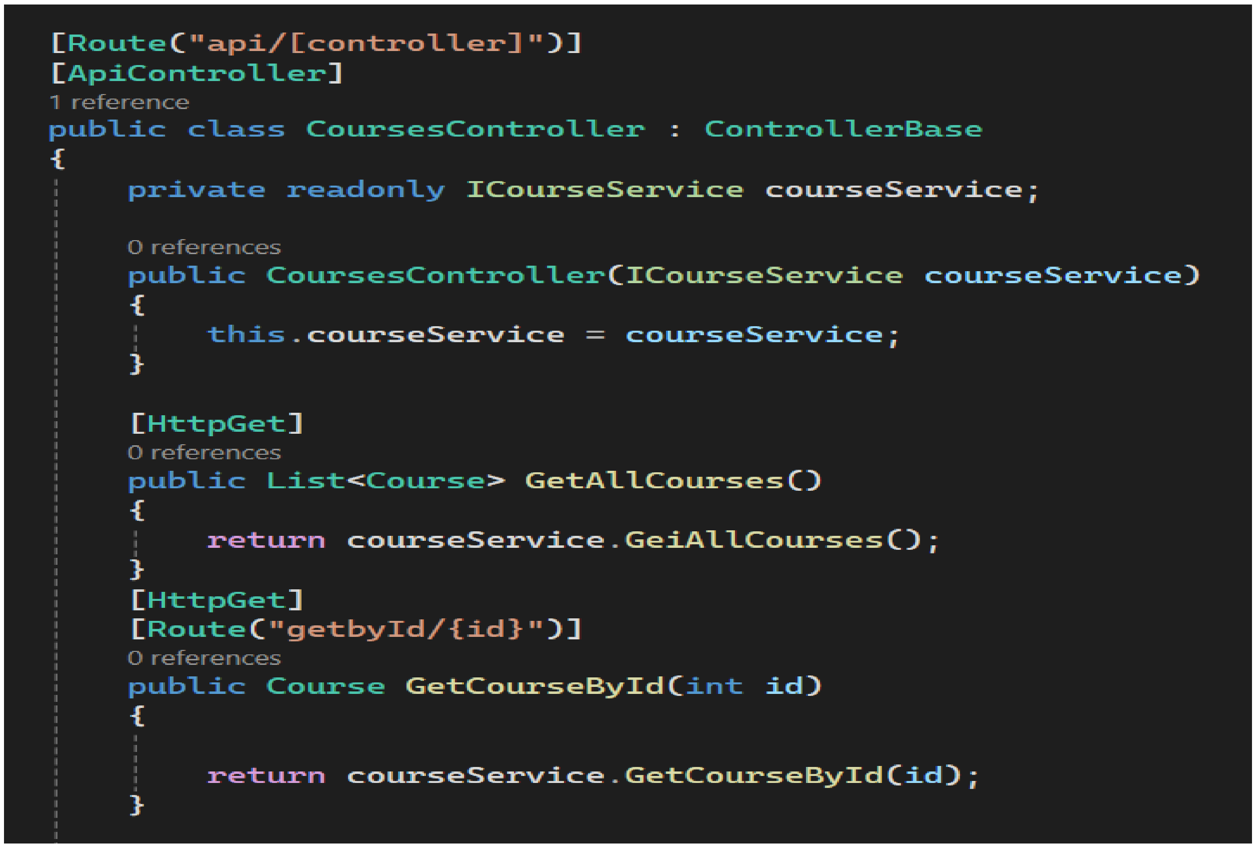
```
2 references
public interface IStudentService
{
    1 reference
    List<Student> GeiAllStudent();
    1 reference
    void CreateStudent(Student student);
    1 reference
    void UpdateStudent(Student student);
    1 reference
    void DeleteStudent(int id);
    1 reference
    Student GetStdStudentById(int id);
    1 reference
    List<Student> GetStudentByFirstName(string firstName);
    1 reference
    List<Student> GetStudentByBirthDate(DateOnly date);
    1 reference
    List<Student> GetStudentFNameAndLName();
    1 reference
    List<Student> GetStudentBetweenInterval(DateOnly datefrom, DateOnly dateto);
    1 reference
    List<Student> GetStudentsWithHighestMarks(int numberofstd);
}
```

```
public class StudentService : IStudentService
{
```

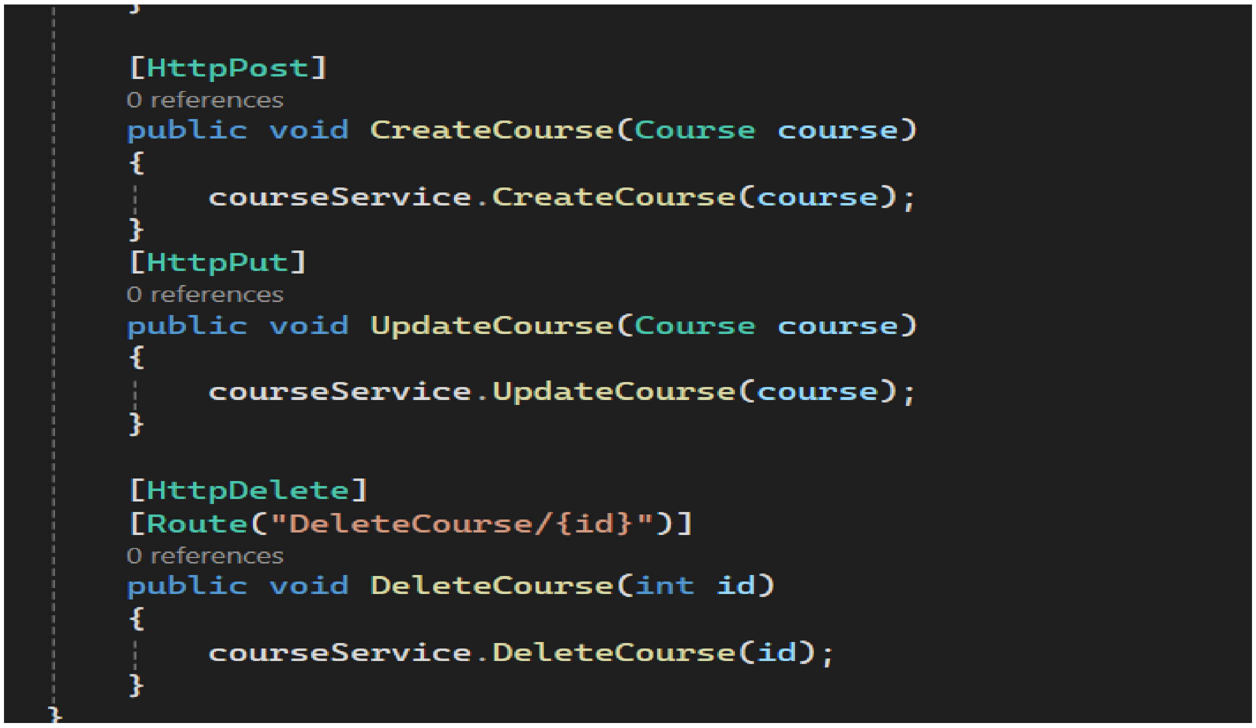
```
private readonly StudentRepository _studentRepository;

public StudentService (StudentRepository studentRepository)
{
   _studentRepository = studentRepository;
}
public void CreateStudent(Student student)
{
   _studentRepository.CreateStudent(student);
}

public void DeleteStudent(int id)
{
   _studentRepository.DeleteStudent(id);
}

public List<Student> GeiAllStudent()
{
   return _studentRepository.GeiAllStudent();
}

public Student GetStdStudentById(int id)
{
   return _studentRepository.GetStdStudentById(id);
}

public List<Student> GetStudentBetweenInterval(DateOnly datefrom, DateOnly dateto)
{
   return _studentRepository.GetStudentBetweenInterval(datefrom, dateto);
}

public List<Student> GetStudentByBirthDate(DateOnly date)
{
  return _studentRepository.GetStudentByBirthDate(date);
}

public List<Student> GetStudentByFirstName(string firstName)
{
   return _studentRepository.GetStudentByFirstName(firstName);
}

public List<Student> GetStudentFNameAndLName()
{
   return _studentRepository.GetStudentFNameAndLName();
```

```
    }

    public List<Student> GetStudentsWithHighestMarks(int numberofstd)
    {
        return _studentRepository.GetStudentsWithHighestMarks(numberofstd);
    }

    public void UpdateStudent(Student student)
    {
        _studentRepository.UpdateStudent(student);
    }
}
```

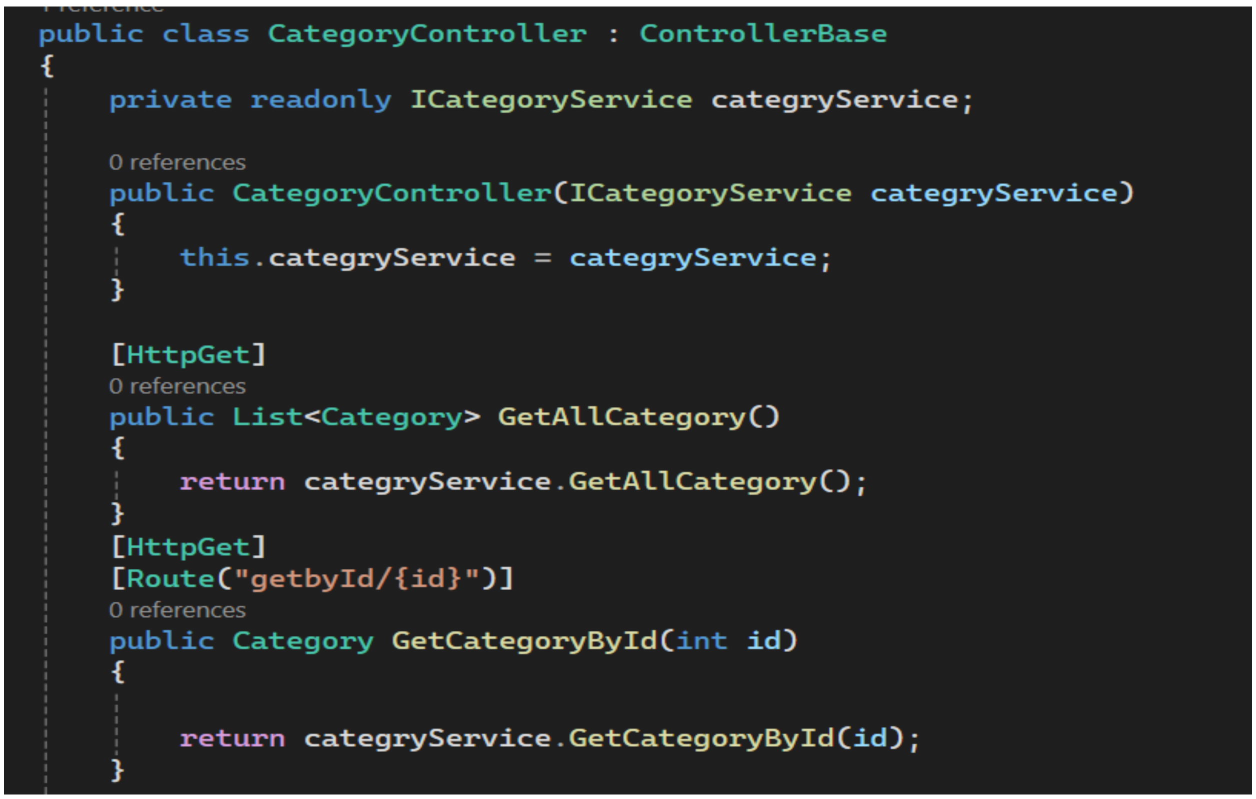
Task 5:



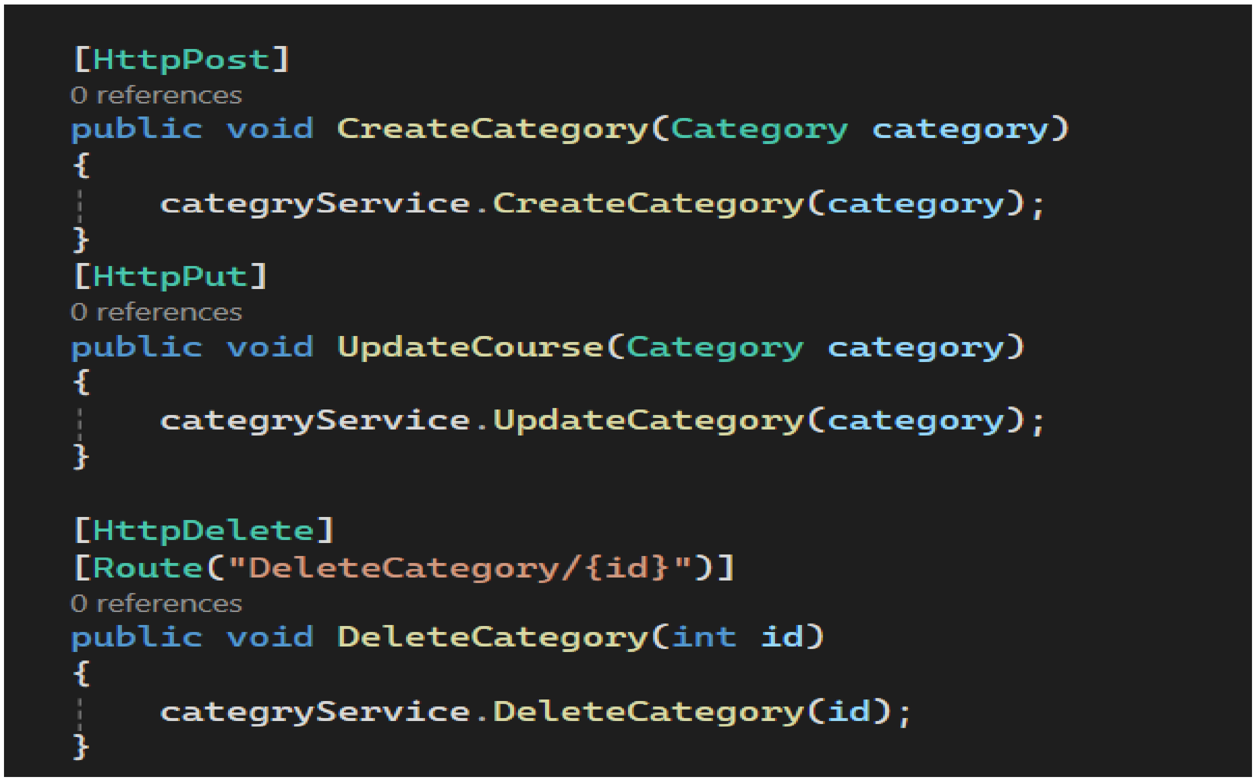
```
[Route("api/[controller]")]
[ApiController]
1 reference
public class CoursesController : ControllerBase
{
    private readonly ICourseService courseService;

    0 references
    public CoursesController(ICourseService courseService)
    {
        this.courseService = courseService;
    }

    [HttpGet]
    0 references
    public List<Course> GetAllCourses()
    {
        return courseService.GeiAllCourses();
    }
    [HttpGet]
    [Route("getbyId/{id}")]
    0 references
    public Course GetCourseById(int id)
    {

        return courseService.GetCourseById(id);
    }
```



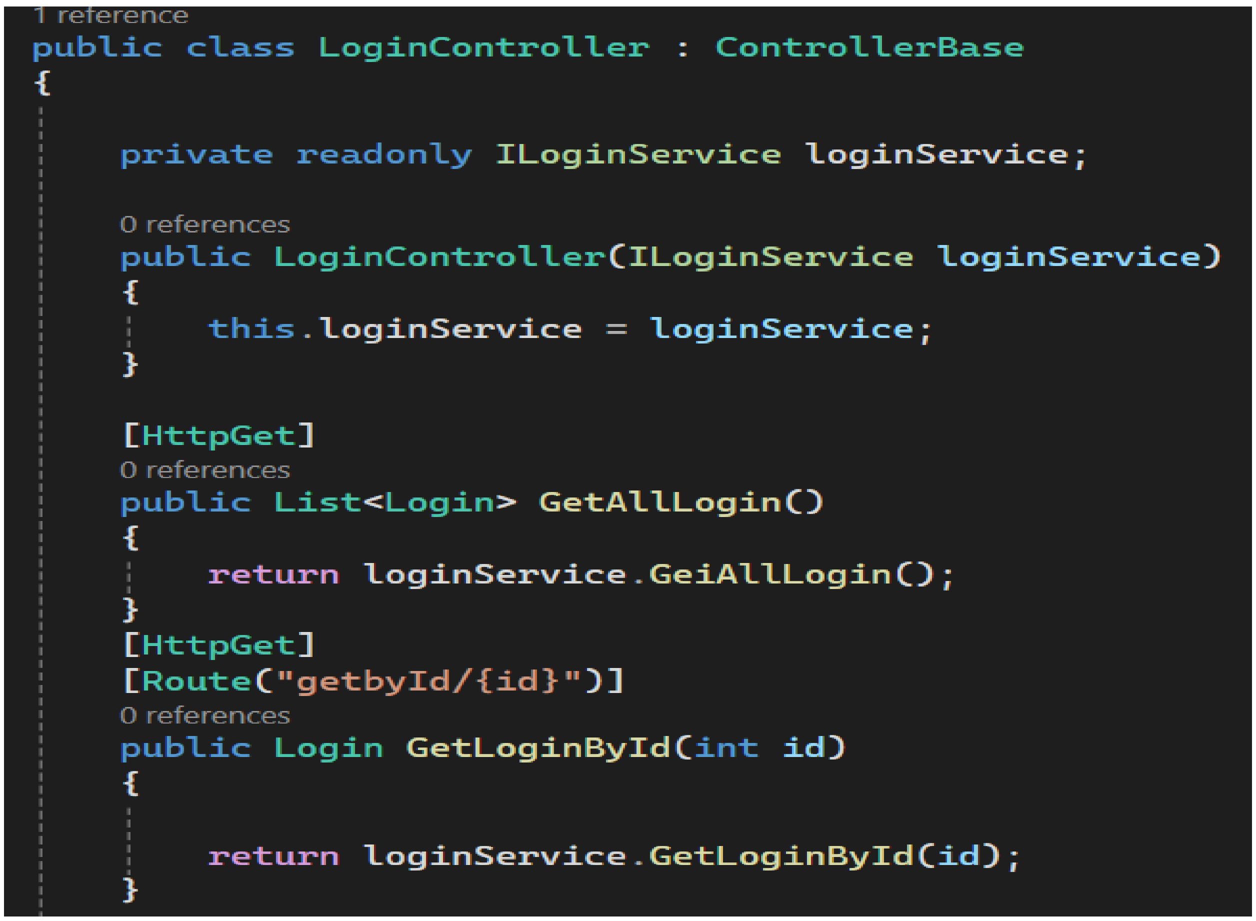
```

    [HttpPost]
    0 references
    public void CreateCourse(Course course)
    {
        courseService.CreateCourse(course);
    }
    [HttpPut]
    0 references
    public void UpdateCourse(Course course)
    {
        courseService.UpdateCourse(course);
    }

    [HttpDelete]
    [Route("DeleteCourse/{id}")]
    0 references
    public void DeleteCourse(int id)
    {
        courseService.DeleteCourse(id);
    }
}
```



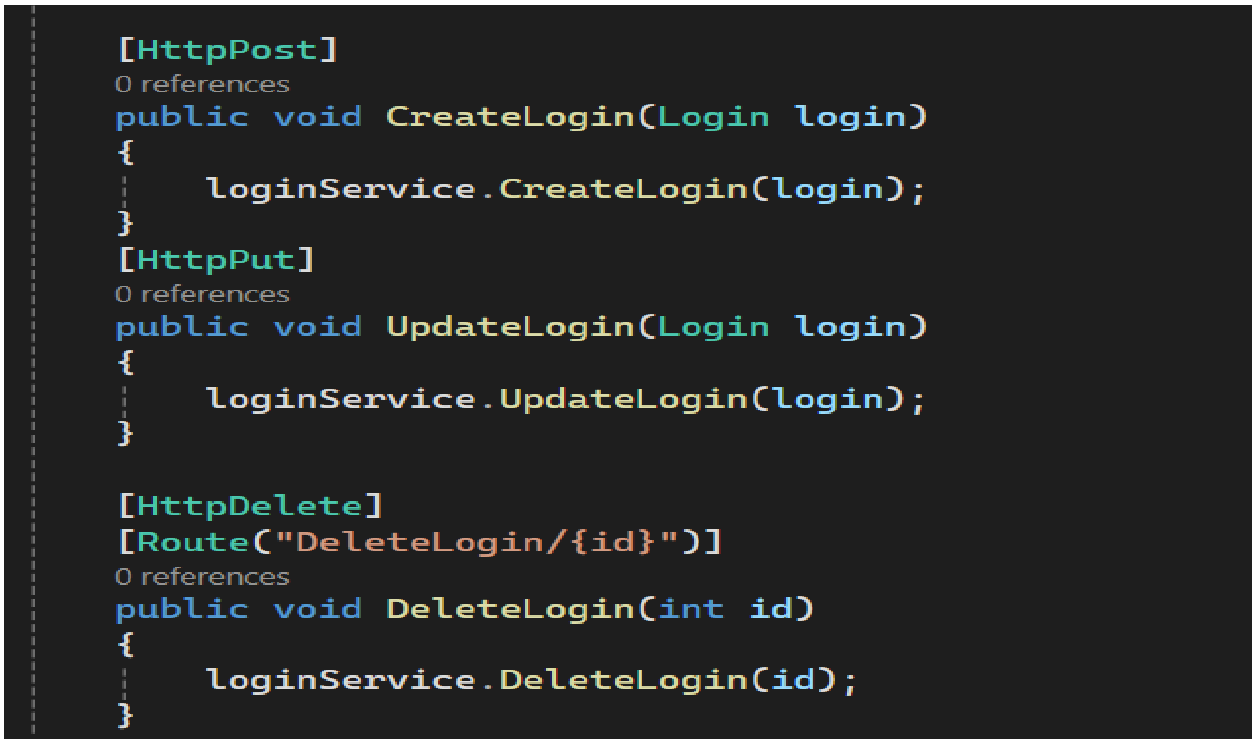
```
public class CategoryController : ControllerBase
{
    private readonly ICategoryService categryService;

    0 references
    public CategoryController(ICategoryService categryService)
    {
        this.categryService = categryService;
    }

    [HttpGet]
    0 references
    public List<Category> GetAllCategory()
    {
        return categryService.GetAllCategory();
    }
    [HttpGet]
    [Route("getbyId/{id}")]
    0 references
    public Category GetCategoryById(int id)
    {

        return categryService.GetCategoryById(id);
    }
```



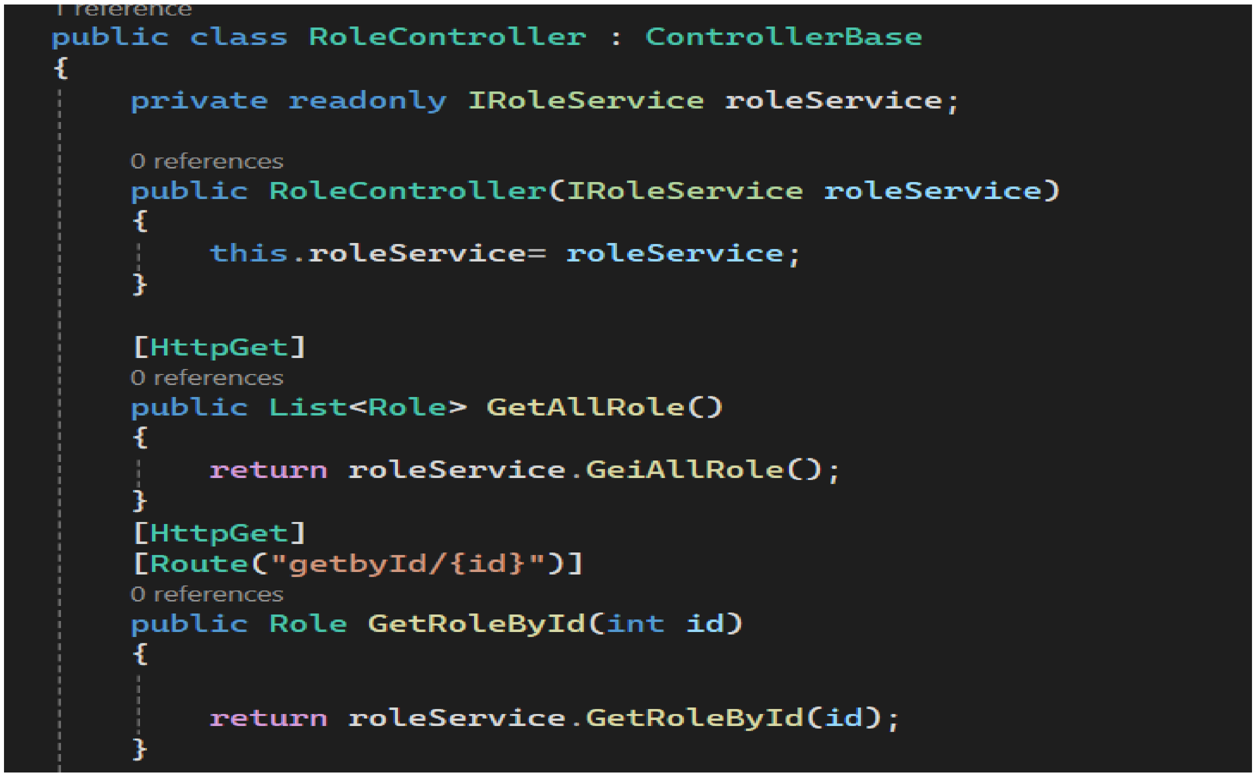
```
[HttpPost]
0 references
public void CreateCategory(Category category)
{
    categryService.CreateCategory(category);
}
[HttpPut]
0 references
public void UpdateCourse(Category category)
{
    categryService.UpdateCategory(category);
}

[HttpDelete]
[Route("DeleteCategory/{id}")]
0 references
public void DeleteCategory(int id)
{
    categryService.DeleteCategory(id);
}
```



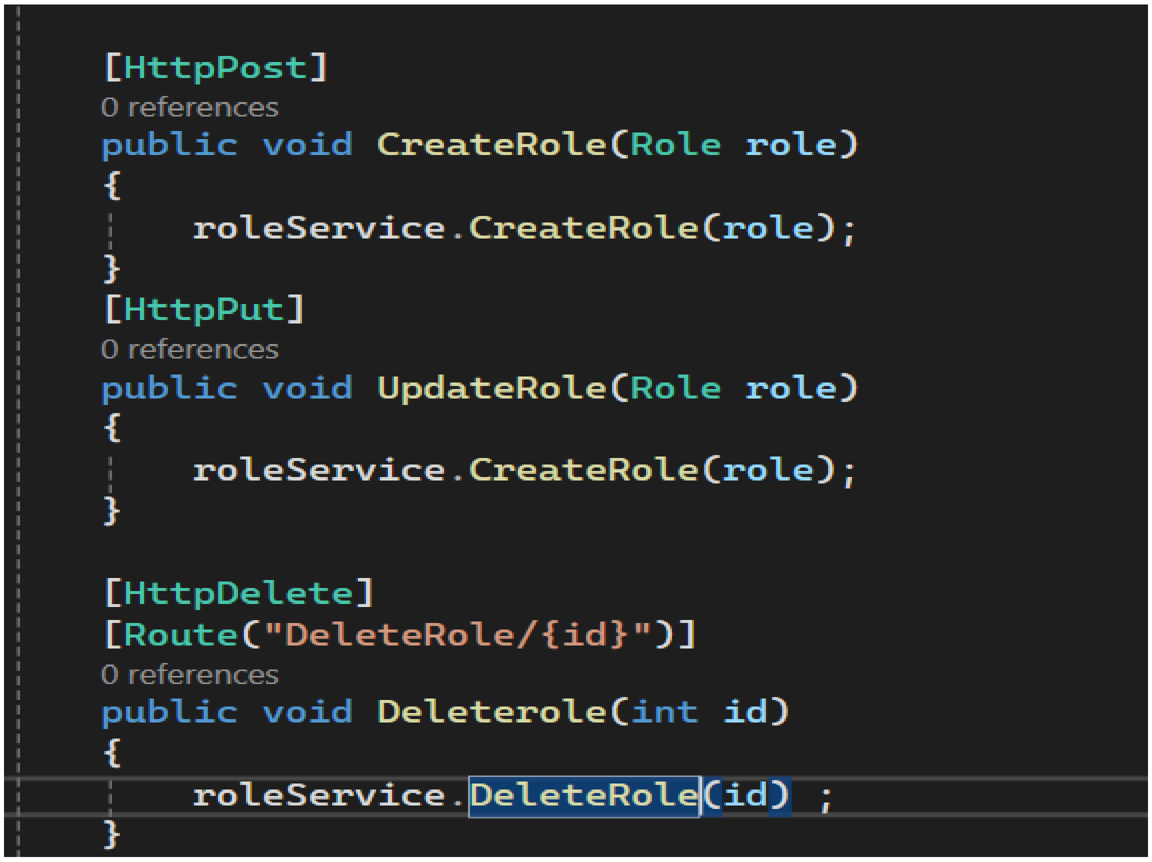
```
1 reference
public class LoginController : ControllerBase
{

    private readonly ILoginService loginService;

    0 references
    public LoginController(ILoginService loginService)
    {
        this.loginService = loginService;
    }

    [HttpGet]
    0 references
    public List<Login> GetAllLogin()
    {
        return loginService.GeiAllLogin();
    }
    [HttpGet]
    [Route("getbyId/{id}")]
    0 references
    public Login GetLoginById(int id)
    {

        return loginService.GetLoginById(id);
    }
```



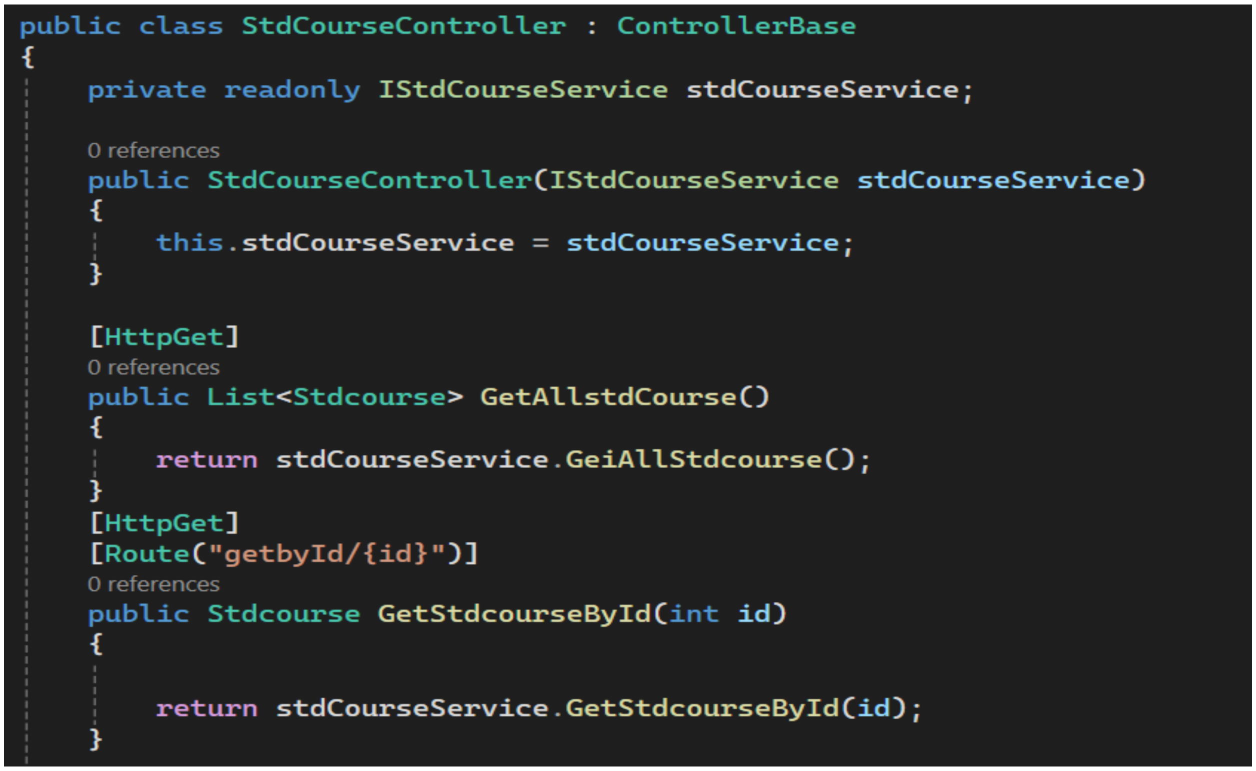
```
    [HttpPost]
    0 references
    public void CreateLogin(Login login)
    {
        loginService.CreateLogin(login);
    }
    [HttpPut]
    0 references
    public void UpdateLogin(Login login)
    {
        loginService.UpdateLogin(login);
    }

    [HttpDelete]
    [Route("DeleteLogin/{id}")]
    0 references
    public void DeleteLogin(int id)
    {
        loginService.DeleteLogin(id);
    }
```



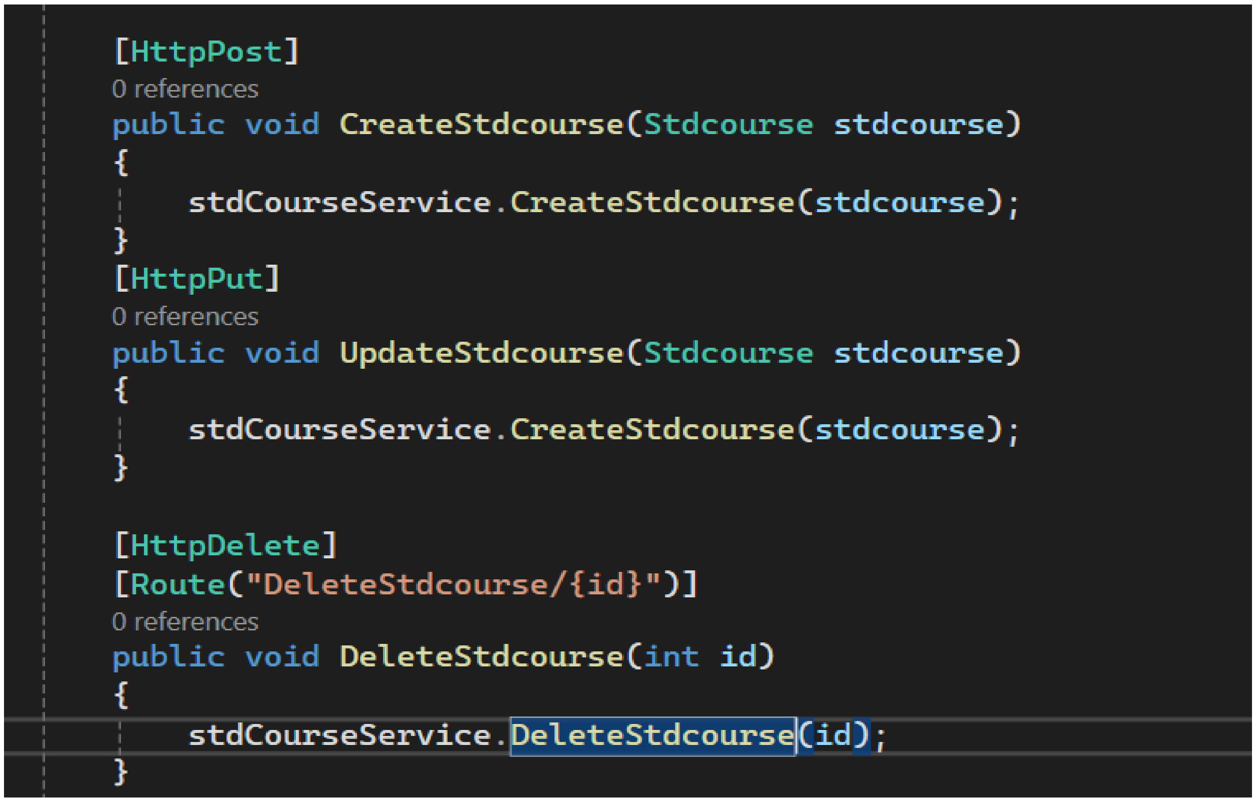
```
    1 reference
    public class RoleController : ControllerBase
    {
        private readonly IRoleService roleService;

        0 references
        public RoleController(IRoleService roleService)
        {
            this.roleService= roleService;
        }

        [HttpGet]
        0 references
        public List<Role> GetAllRole()
        {
            return roleService.GeiAllRole();
        }
        [HttpGet]
        [Route("getbyId/{id}")]
        0 references
        public Role GetRoleById(int id)
        {

            return roleService.GetRoleById(id);
        }
```



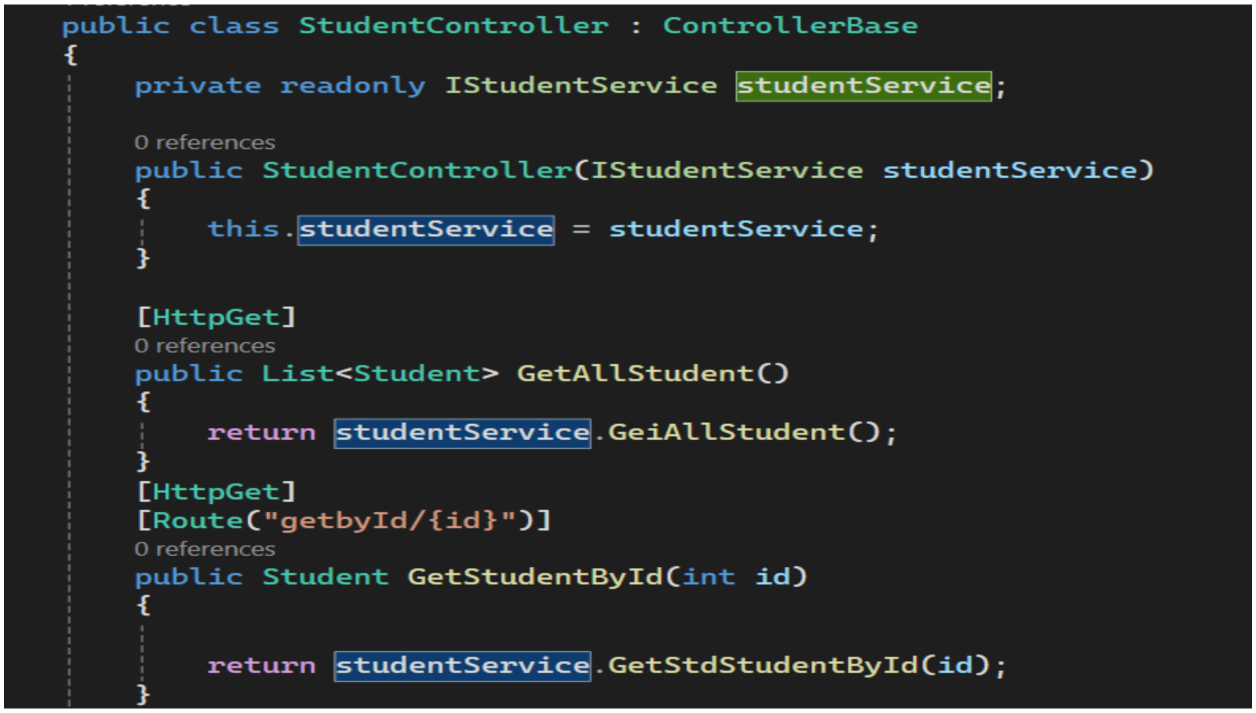
```
        [HttpPost]
        0 references
        public void CreateRole(Role role)
        {
            roleService.CreateRole(role);
        }
        [HttpPut]
        0 references
        public void UpdateRole(Role role)
        {
            roleService.CreateRole(role);
        }

        [HttpDelete]
        [Route("DeleteRole/{id}")]
        0 references
        public void Deleterole(int id)
        {
            roleService.DeleteRole(id) ;
        }
```



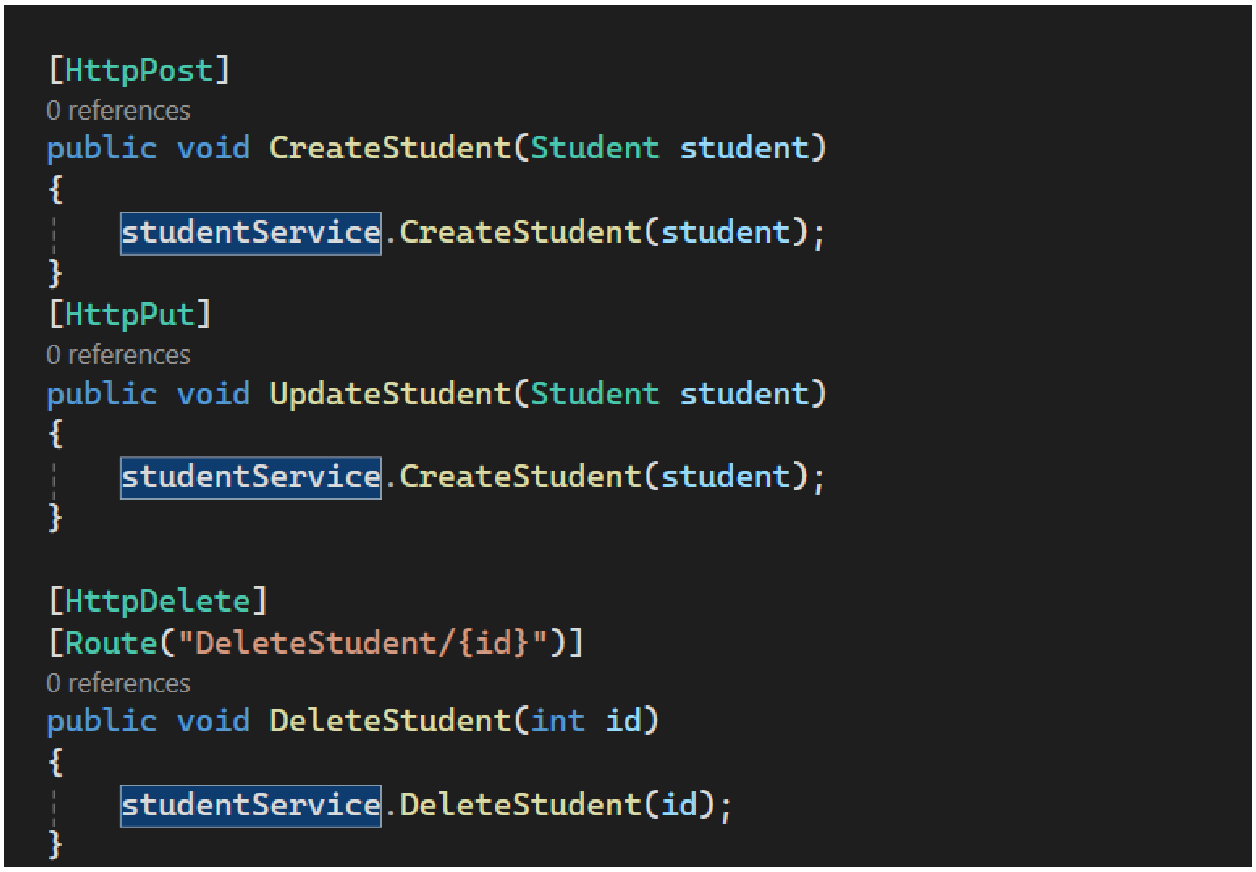
```
public class StdCourseController : ControllerBase
{
    private readonly IStdCourseService stdCourseService;

    0 references
    public StdCourseController(IStdCourseService stdCourseService)
    {
        this.stdCourseService = stdCourseService;
    }

    [HttpGet]
    0 references
    public List<Stdcourse> GetAllstdCourse()
    {
        return stdCourseService.GeiAllStdcourse();
    }
    [HttpGet]
    [Route("getbyId/{id}")]
    0 references
    public Stdcourse GetStdcourseById(int id)
    {

        return stdCourseService.GetStdcourseById(id);
    }
```



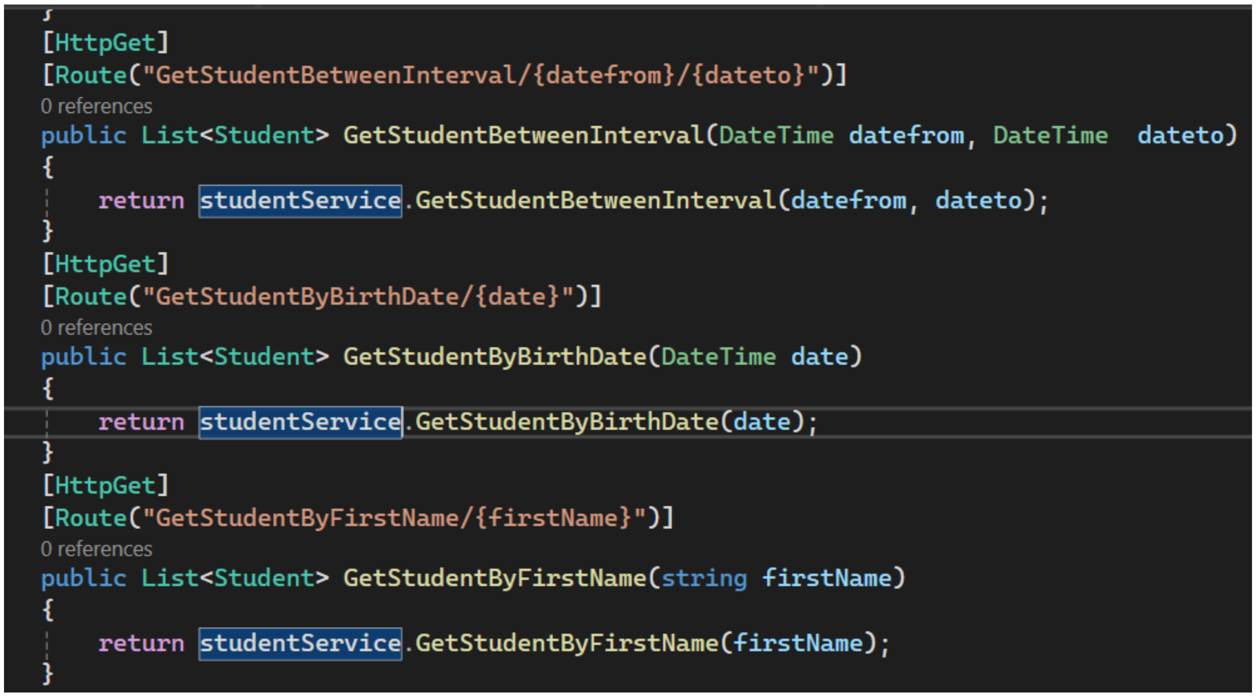
```
    [HttpPost]
    0 references
    public void CreateStdcourse(Stdcourse stdcourse)
    {
        stdCourseService.CreateStdcourse(stdcourse);
    }
    [HttpPut]
    0 references
    public void UpdateStdcourse(Stdcourse stdcourse)
    {
        stdCourseService.CreateStdcourse(stdcourse);
    }

    [HttpDelete]
    [Route("DeleteStdcourse/{id}")]
    0 references
    public void DeleteStdcourse(int id)
    {
        stdCourseService.DeleteStdcourse(id);
    }
```



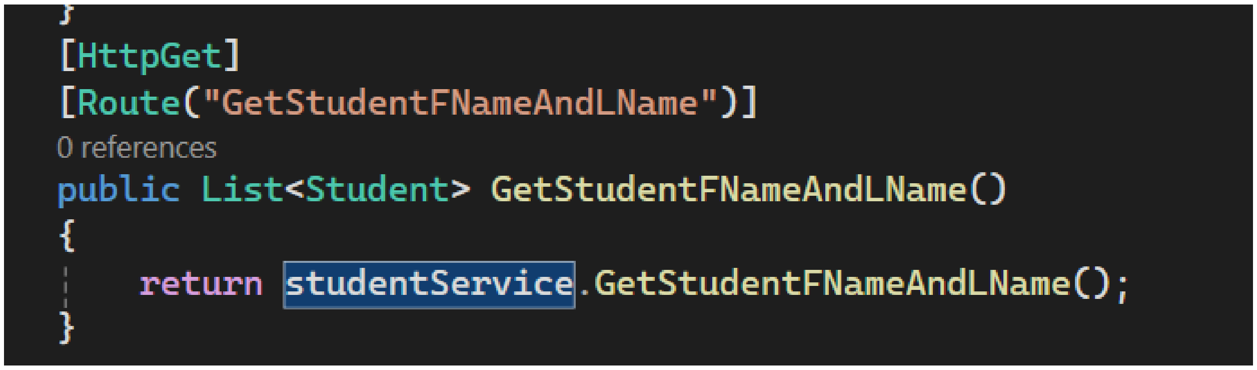
```
public class StudentController : ControllerBase
{
    private readonly IStudentService studentService;

    0 references
    public StudentController(IStudentService studentService)
    {
        this.studentService = studentService;
    }

    [HttpGet]
    0 references
    public List<Student> GetAllStudent()
    {
        return studentService.GeiAllStudent();
    }
    [HttpGet]
    [Route("getbyId/{id}")]
    0 references
    public Student GetStudentById(int id)
    {

        return studentService.GetStdStudentById(id);
    }
```



```
[HttpPost]
0 references
public void CreateStudent(Student student)
{
    studentService.CreateStudent(student);
}
[HttpPut]
0 references
public void UpdateStudent(Student student)
{
    studentService.CreateStudent(student);
}

[HttpDelete]
[Route("DeleteStudent/{id}")]
0 references
public void DeleteStudent(int id)
{
    studentService.DeleteStudent(id);
}
```



```
[HttpGet]
[Route("GetStudentBetweenInterval/{datefrom}/{dateto}")]
0 references
public List<Student> GetStudentBetweenInterval(DateTime datefrom, DateTime  dateto)
{
    return studentService.GetStudentBetweenInterval(datefrom, dateto);
}
[HttpGet]
[Route("GetStudentByBirthDate/{date}")]
0 references
public List<Student> GetStudentByBirthDate(DateTime date)
{
    return studentService.GetStudentByBirthDate(date);
}
[HttpGet]
[Route("GetStudentByFirstName/{firstName}")]
0 references
public List<Student> GetStudentByFirstName(string firstName)
{
    return studentService.GetStudentByFirstName(firstName);
}
```



```
[HttpGet]
[Route("GetStudentFNameAndLName")]
0 references
public List<Student> GetStudentFNameAndLName()
{
    return studentService.GetStudentFNameAndLName();
}
```