White Box Testing (Unit Testing): [2 + 2 = 4 Hours]

- Write the Unit Test cases by using NUnit for your Back-End [CSharp File].

LoginWebapi NUnitTest
Test Case

- ApplicationDbContext_should_connect_to_mysql
- ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql
- Check_Register_with_old_user_should_return_Fail_status
- Check_Register_with_new_user_should_return_Ok_status
- Check_Login_with_correct_user_should_return_Ok_status
- Check_Login_with_wrong_password_should_return_Fail_status

Code

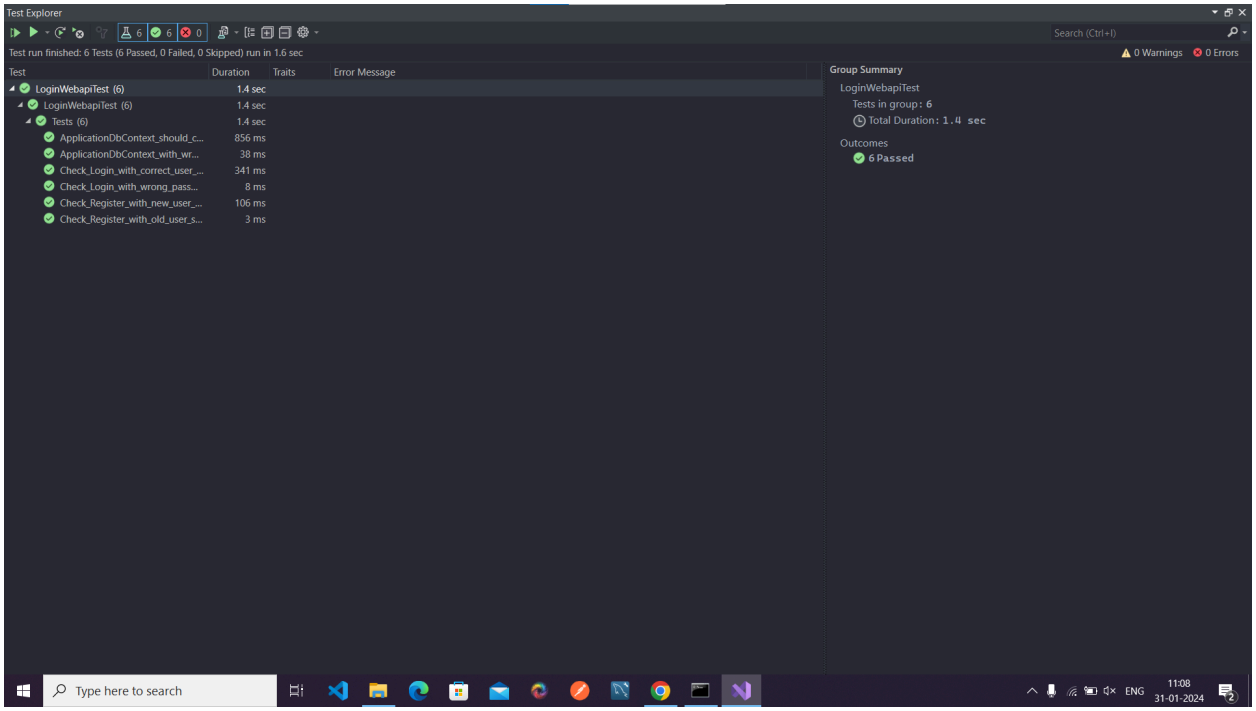
```
using LoginWebapi.Controller;
using LoginWebapi.Data;

using Microsoft.EntityFrameworkCore;
using LoginWebapi.Model;
using System;
using Microsoft.AspNetCore.Mvc;
using Google.Protobuf.WellKnownTypes;
using Org.BouncyCastle.Crypto.Macs;
class RegisterResult
{
   public bool Exists { get; set; }
}
namespace LoginWebapiTest
{
   public class Tests
   {
      dynamic optionsbuilder;
      AppDbContext applicationdbContext;

      [SetUp]
      public void Setup()
      {
         optionsbuilder = new
DbContextOptionsBuilder().UseMySql("server=localhost;user=root;password=admin123;databa
se=RelevantzMovieLogin", new MySqlServerVersion(new Version()));

         applicationdbContext = new AppDbContext(optionsbuilder.Options);
      }

      [Test]
```

```
        public void ApplicationDbContext_should_connect_to_mysql()
        {

            bool expected = true;

            // act
            bool result = applicationdbContext.Database.CanConnect();

            // assert
            Assert.AreEqual(expected, result);
        }
        [Test]
        public void ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql()
        {
            optionsbuilder = new
DbContextOptionsBuilder().UseMySql("server=localhost;user=root;password=wrongpassword;d
atabase=RelevantzMovieLogin", new MySqlServerVersion(new Version()));

            applicationdbContext = new AppDbContext(optionsbuilder.Options);

            bool expected = false;

            // act
            bool result = applicationdbContext.Database.CanConnect();

            // assert
            Assert.AreEqual(expected, result);
        }

        [Test]
        public void Check_Register_with_new_user_should_return_Ok_status()
        {
            UserLogin user1 = new UserLogin()
            {
                UserEmail="stdao3@gmail.com",
                UserName="Tessaaero3",
                UserPassword="test1234"
            };

            UserController userController = new UserController(applicationdbContext);
            var result = (OkObjectResult)userController.Post(user1).Result;
            var value=result.Value;

            Assert.AreEqual("{\"Exists\":false}", value);
```

```
}
[Test]
public void Check_Register_with_old_user_should_return_Fail_status()
{
    UserLogin user1 = new UserLogin()
    {
        UserEmail = "Karthick@gmail.com",
        UserName = "Karthick Subburaj",
        UserPassword = "Karthick123"
    };

    UserController userController = new UserController(applicationdbContext);
    var result = (OkObjectResult)userController.Post(user1).Result;
    var value = result.Value;

    Assert.AreEqual("{\"Exists\":true}", value);
}

[Test]
public void Check_Login_with_wrong_password_should_return_Fail_status()
{
    LoginDetails user1 = new LoginDetails()
    {
        UserEmail = "Karthick@gmail.com",
        UserPassword = "Karthick1234n"
    };

    UserController userController = new UserController(applicationdbContext);
    var result = (OkObjectResult)userController.Login(user1).Result;
    var value = result.Value;

    Assert.AreEqual("{\"account\":true,\"emailstatus\":true,\"passwordstatus\":false}", value);

}
[Test]
public void Check_Login_with_correct_user_should_return_Ok_status()
{
    LoginDetails user1 = new LoginDetails()
    {
        UserEmail = "Karthick@gmail.com",
        UserPassword = "Karthick123"
    };

    UserController userController = new UserController(applicationdbContext);
```

```
        var result = (OkObjectResult)userController.Login(user1).Result;
        var value = result.Value;


Assert.AreEqual("{\"account\":true,\"emailstatus\":true,\"passwordstatus\":true,\"usernamer\":\"Ka
rthick Subburaj\"}", value);

      }
   }
}
```

Test Case Passed



CreateWebapi Nunit Test
Test Case

- ApplicationDbContext_should_connect_to_mysql
- ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql
- Post_with_same_id_returns_wrongResult
- Post_with_correct_type_returns_correctResult

Code

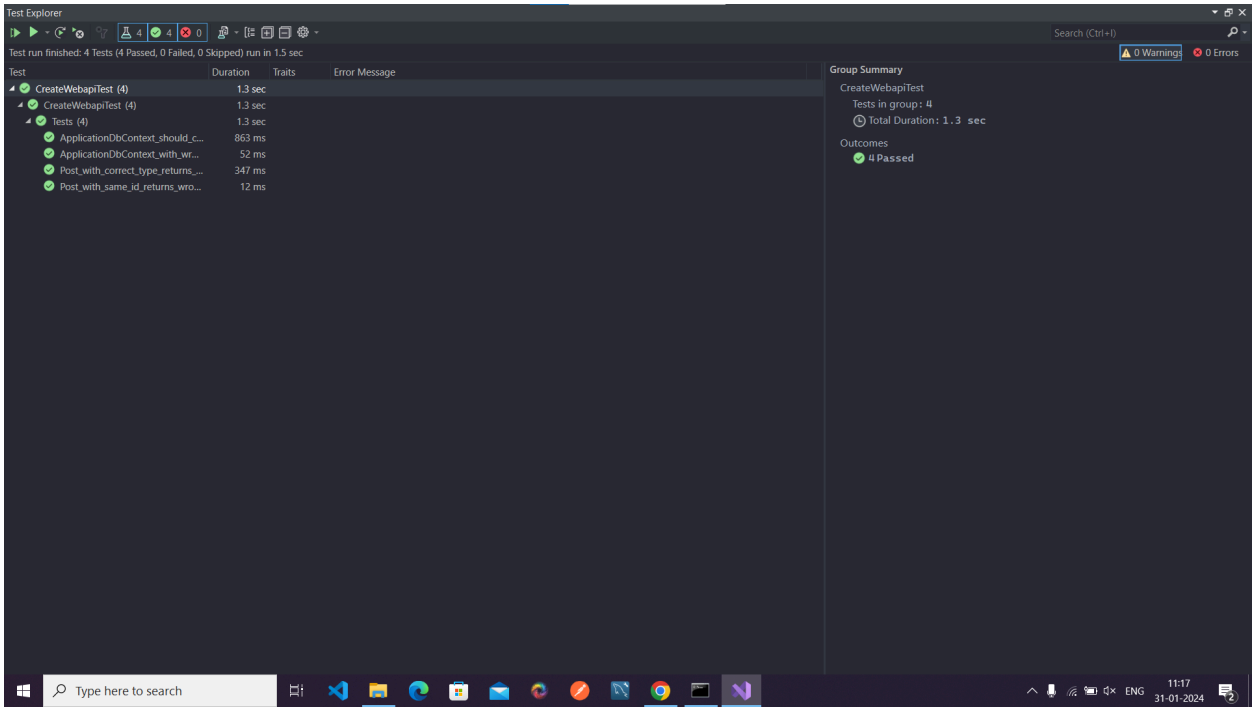
```
using CreateWebapi.Model;
using CreateWebapi.Controllers;
using CreateWebapi.Data;
```

```
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Options;
using Microsoft.AspNetCore.Mvc;
using System.ComponentModel.DataAnnotations;

namespace CreateWebapiTest
{
    public class Tests
    {
        dynamic optionsbuilder;
        AppDbContext applicationdbContext;

        [SetUp]
        public void Setup()
        {
            optionsbuilder = new
DbContextOptionsBuilder().UseMySql("server=localhost;user=root;password=admin123;databa
se=RelevantzMovieCenter", new MySqlServerVersion(new Version())); ;
            applicationdbContext = new AppDbContext(optionsbuilder.Options);
        }


        [Test]
        public void ApplicationDbContext_should_connect_to_mysql()
        {

            bool expected = true;

            // act
            bool result = applicationdbContext.Database.CanConnect();

            // assert
            Assert.AreEqual(expected, result);

        }
        [Test]
        public void ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql()
        {
            optionsbuilder = new
DbContextOptionsBuilder().UseMySql("server=localhost;user=root;password=wrongpassword;d
atabase=RelevantzMovieCenter", new MySqlServerVersion(new Version()));

            applicationdbContext = new AppDbContext(optionsbuilder.Options);
```

```
            bool expected = false;

            // act
            bool result = applicationdbContext.Database.CanConnect();

            // assert
            Assert.AreEqual(expected, result);
        }
        [Test]
        public void Post_with_correct_type_returns_correctResult()
        {
            AddMovieController addMovieController = new
AddMovieController(applicationdbContext);

            MovieDetails movieDetails = new MovieDetails()
            {
              MovieId =0,
              MovieName="Kiren1",
              MovieType="Romance",
              MovieLanguage="Tamil",
              MovieDurations="2.40 hour"
            };
            //var oldmovie = applicationdbContext.MovieDetails.Count();

            var result = addMovieController.Post(movieDetails).Result;

            //var newmovie = applicationdbContext.MovieDetails.Count();
            Assert.IsInstanceOf<OkResult>(result);
        }
        [Test]
        public void Post_with_same_id_returns_wrongResult()
        {
            AddMovieController addMovieController = new
AddMovieController(applicationdbContext);

            MovieDetails movieDetails = new MovieDetails()
            {
               MovieId = 25,
               MovieName = "Kiren1",
               MovieType = "Action",
               MovieLanguage = "Tamil",
               MovieDurations = "2.40 hour"
            };
            //var oldmovie = applicationdbContext.MovieDetails.Count();
```

```
            var result = addMovieController.Post(movieDetails).Result;

            //var newmovie = applicationdbContext.MovieDetails.Count();
            Assert.IsInstanceOf<BadRequestObjectResult>(result);
        }

    }
}
```

Test Case Passed



FetchWebapi Nunit Test (read data)

Test Case

- ApplicationDbContext_should_connect_to_mysql
- ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql
- GetAll_returnsTypeCheck
- Get_returnsAllResults
- GetById_returns_correctResultType
- GetById_returns_correctResult
- Get_by_id_returns_NotFound_for_invalid_Id

Code

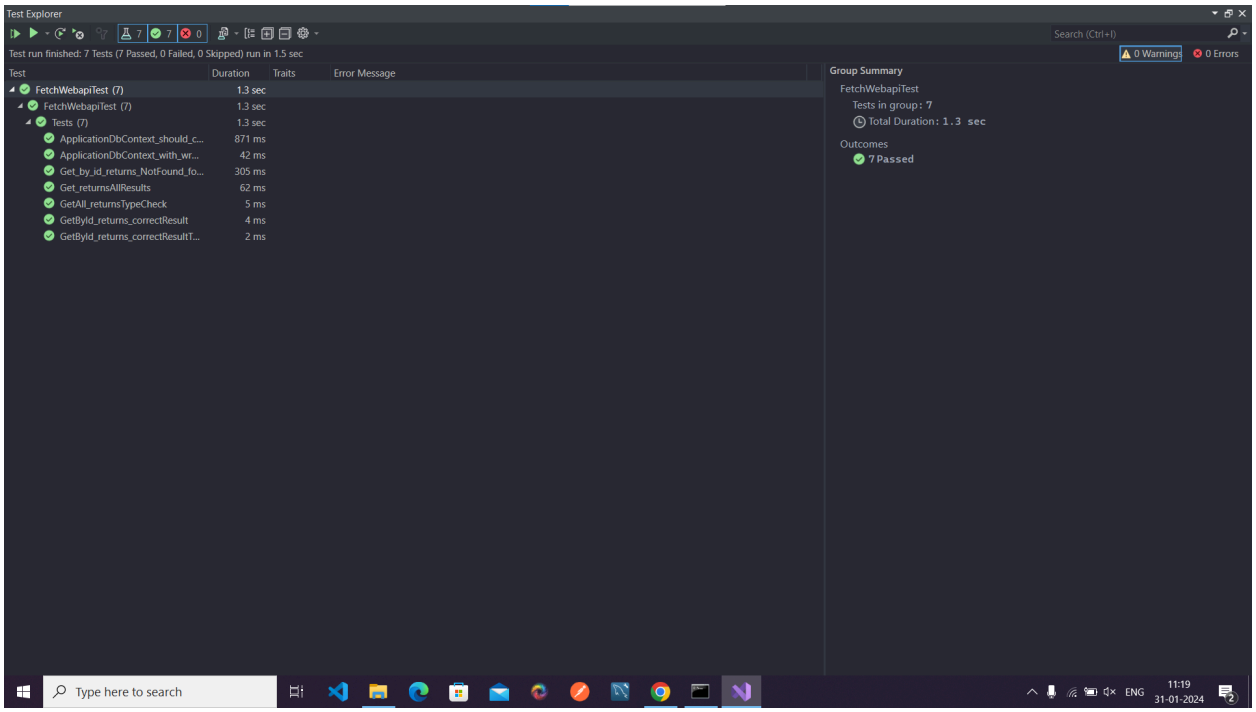
```
using FetchWebapi.Controllers;
using FetchWebapi.Data;
```

```
using Microsoft.EntityFrameworkCore;
using FetchWebapi.Model;
using System;
using Microsoft.AspNetCore.Mvc;
namespace FetchWebapiTest
{
   public class Tests

   {
      dynamic optionsbuilder;
      Appdbcontext applicationdbContext;

       [SetUp]
      public void Setup()
      {
         optionsbuilder = new
DbContextOptionsBuilder().UseMySql("server=localhost;user=root;password=admin123;databa
se=RelevantzMovieCenter", new MySqlServerVersion(new Version()));

         applicationdbContext = new Appdbcontext(optionsbuilder.Options);
      }

      [Test]
      public void ApplicationDbContext_should_connect_to_mysql()
      {

         bool expected = true;

         // act
         bool result = applicationdbContext.Database.CanConnect();

         // assert
         Assert.AreEqual(expected, result);
      }
      [Test]
      public void ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql()
      {
         optionsbuilder = new
DbContextOptionsBuilder().UseMySql("server=localhost;user=root;password=wrongpassword;d
atabase=RelevantzMovieCenter", new MySqlServerVersion(new Version()));

         applicationdbContext = new Appdbcontext(optionsbuilder.Options);

         bool expected = false;
```

```
            // act
            bool result = applicationdbContext.Database.CanConnect();

            // assert
            Assert.AreEqual(expected, result);
        }
        [Test]
        public void Get_by_id_returns_NotFound_for_invalid_Id()
        {
            FetchMovieController fetchMovieController = new
FetchMovieController(applicationdbContext);
            var result = fetchMovieController.GetIndividual(1).Result;

            Assert.IsInstanceOf<NotFoundResult>(result);
        }
        [Test]
        public void GetById_returns_correctResult(){
            FetchMovieController fetchMovieController = new
FetchMovieController(applicationdbContext);
            var result = (OkObjectResult)fetchMovieController.GetIndividual(25).Result;
            var value = (MovieDetails)result.Value;
            Assert.AreEqual(25, value.MovieId);

        }
        [Test]
        public void GetById_returns_correctResultType()
        {
            FetchMovieController fetchMovieController = new
FetchMovieController(applicationdbContext);
            var result = (OkObjectResult)fetchMovieController.GetIndividual(25).Result;
            var value = (MovieDetails)result.Value;
            Assert.IsInstanceOf<MovieDetails>(value);

        }
        [Test]
        public void Get_returnsAllResults()
        {
            FetchMovieController fetchMovieController = new
FetchMovieController(applicationdbContext);
            var result = fetchMovieController.Get().Result as List<MovieDetails>;
            List<MovieDetails> value = (List<MovieDetails>)result ;
            var countresult = (OkObjectResult)fetchMovieController.TotalCount().Result;
            int countvalue = (int)countresult.Value;
```

```
            //assert
            Assert.AreEqual(countvalue, value.Count);

        }
        [Test]
        public void GetAll_returnsTypeCheck()
        {
            FetchMovieController fetchMovieController = new
FetchMovieController(applicationdbContext);
            var result = fetchMovieController.Get().Result as List<MovieDetails>;
            List<MovieDetails> value = (List<MovieDetails>)result;
            Assert.IsInstanceOf<MovieDetails>(value.First());

        }
    }
}
```

Test Case Passed



UpdateWebapi NUnit Test

Test Case

- ApplicationDbContext_should_connect_to_mysql
- ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql
- Update_with_0_id_returns_wrongResult
- Update_with_correct_type_returns_correctResult(

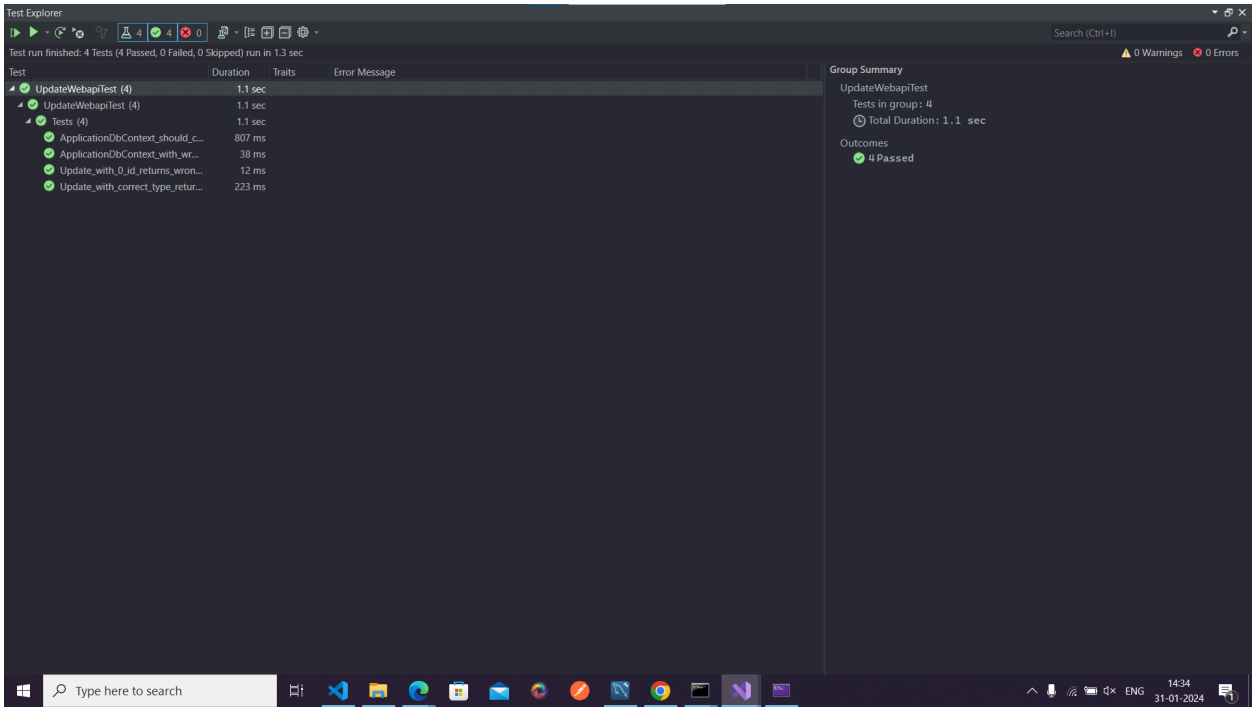
Code

```
using UpdateWebapi.Model;
using UpdateWebapi.Controllers;
using UpdateWebapi.Data;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Options;
using Microsoft.AspNetCore.Mvc;

namespace UpdateWebapiTest
{
   public class Tests
   {
      dynamic optionsbuilder;
      AppDbContext appdbContext;

      [SetUp]
      public void Setup()
      {
         optionsbuilder = new
DbContextOptionsBuilder().UseMySQL("server=localhost;user=root;password=admin123;datab
ase=RelevantzMovieCenter");
         appdbContext = new AppDbContext(optionsbuilder.Options);

      }


      [Test]
      public void ApplicationDbContext_should_connect_to_mysql()
      {
         var expected = true;

         // act
         var result = appdbContext.Database.CanConnect();
         Assert.That(result, Is.EqualTo(expected));
         // assert

      }
      [Test]
      public void ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql()
      {
         optionsbuilder = new
DbContextOptionsBuilder().UseMySQL("server=localhost;user=root;password=wrongpassword;
database=RelevantzMovieCenter");
```

```
            appdbContext = new AppDbContext(optionsbuilder.Options);

            bool expected = false;

            // act
            bool result = appdbContext.Database.CanConnect();

            // assert
            Assert.That(result, Is.EqualTo(expected));

        }
        [Test]
        public void Update_with_correct_type_returns_correctResult()
        {

            UpdateMovieController updateMovieController = new
UpdateMovieController(appdbContext);

            MovieDetails movieDetails = new MovieDetails()
            {
                MovieId =13,
                MovieName = "Kiren1",
                MovieType = "Romance",
                MovieLanguage = "Tamil",
                MovieDurations = "2.40 hour"
            };
            //var oldmovie = applicationdbContext.MovieDetails.Count();

            var result = updateMovieController.UpdateMovie(movieDetails).Result;

            //var newmovie = applicationdbContext.MovieDetails.Count();
            Assert.IsInstanceOf<OkResult>(result);

        }
        [Test]
        public void Update_with_0_id_returns_wrongResult()
        {
            UpdateMovieController updateMovieController = new
UpdateMovieController(appdbContext);

            MovieDetails movieDetails = new MovieDetails()
            {
                MovieId = 0,
                MovieName = "Kiren1",
```

```
            MovieType = "Action",
            MovieLanguage = "Tamil",
            MovieDurations = "2.40 hour"
        };
        //var oldmovie = applicationdbContext.MovieDetails.Count();

        var result = updateMovieController.UpdateMovie(movieDetails).Result;

        //var newmovie = applicationdbContext.MovieDetails.Count();
        Assert.IsInstanceOf<BadRequestResult>(result);
    }

  }
}
```

Test Case Passed



DeleteWebapi NUnit Test

Test Case

- ApplicationDbContext_should_connect_to_mysql
- ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql
- Delete_returns_CorrectResult
- Delete_returns_NotFound_NotExistsId
- Delete_returns_BadRequest_InvalidId

Code

```
using System;

using DeleteWebapi.Controllers;
using DeleteWebapi.Data;

using Microsoft.EntityFrameworkCore;
using DeleteWebapi.Controllers;
using System;
using Microsoft.AspNetCore.Mvc;

namespace DeleteWebapiTest
{
   public class Tests

   {
      dynamic optionsbuilder;
      AppDbContext applicationdbContext;

      [SetUp]
      public void Setup()
      {
         optionsbuilder = new
DbContextOptionsBuilder().UseMySql("server=localhost;user=root;password=admin123;databa
se=RelevantzMovieCenter", new MySqlServerVersion(new Version()));

         applicationdbContext = new AppDbContext(optionsbuilder.Options);
      }

      [Test]
      public void ApplicationDbContext_should_connect_to_mysql()
      {

         bool expected = true;

         // act
         bool result = applicationdbContext.Database.CanConnect();

         // assert
         Assert.AreEqual(expected, result);
      }
      [Test]
      public void ApplicationDbContext_with_wrong_password_should_not_connect_to_mysql()
      {
```

```
            optionsbuilder = new
DbContextOptionsBuilder().UseMySql("server=localhost;user=root;password=wrongpassword;d
atabase=RelevantzMovieCenter", new MySqlServerVersion(new Version()));

            applicationdbContext = new AppDbContext(optionsbuilder.Options);

            bool expected = false;

            // act
            bool result = applicationdbContext.Database.CanConnect();

            // assert
            Assert.AreEqual(expected, result);
        }
        [Test]
        public void Delete_returns_BadRequest_InvalidId()
        {
            DeleteMovieController deleteMovieController = new
DeleteMovieController(applicationdbContext);
            var result=deleteMovieController.DeleteMovie(-1).Result;
            Assert.IsInstanceOf<BadRequestResult>(result);

        }
        [Test]
        public void Delete_returns_NotFound_NotExistsId()
        {
            DeleteMovieController deleteMovieController = new
DeleteMovieController(applicationdbContext);
            var result = deleteMovieController.DeleteMovie(1).Result;
            Assert.IsInstanceOf<NotFoundResult>(result);

        }
        [Test]
        public void Delete_returns_CorrectResult()
        {
            DeleteMovieController deleteMovieController = new
DeleteMovieController(applicationdbContext);
            var result = deleteMovieController.DeleteMovie(27).Result;
            Assert.IsInstanceOf<OkResult>(result);

        }
    }
}
```

Test Case passed