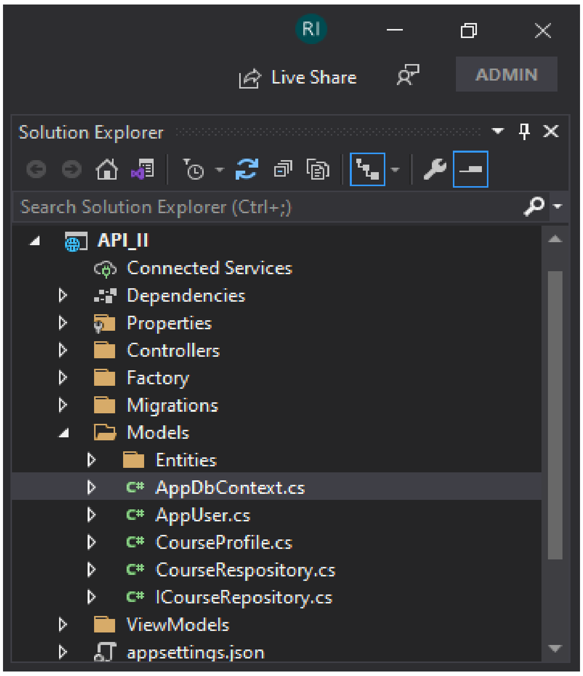
Application Security

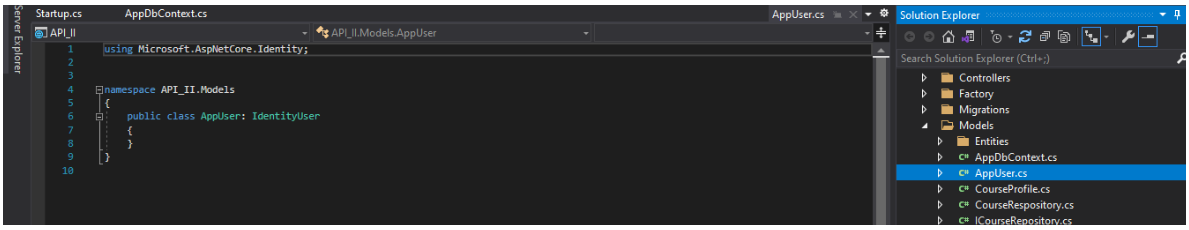


Install:

- Microsoft.AspNetCore.Identity.EntityFrameworkCore
- Microsoft.Extensions.Identity.Core
- Microsoft.Extensions.Identity.Store

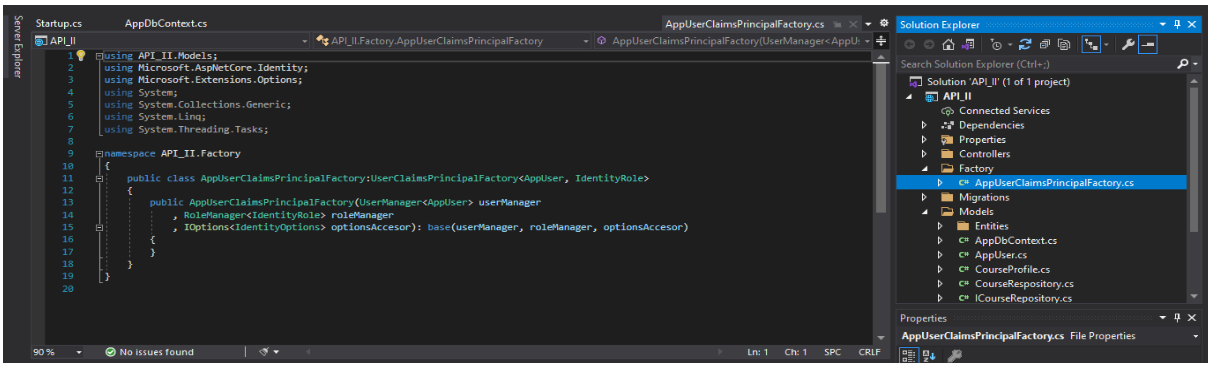
Steps:

1) Go to your AppdbContext, change public class AppDbContext:DbContext to public class AppDbContext:IdentityDbContext
2) Go to package manager control and write the following: add-migration ImplementIdentity
3) And then enter: update-database
4) Configure the middleware in the startup.cs add app.UseAuthentication();
5) Create AppUser Class



```
using Microsoft.AspNetCore.Identity;


namespace API_II.Models
{
    public class AppUser: IdentityUser
    {
    }
}

```

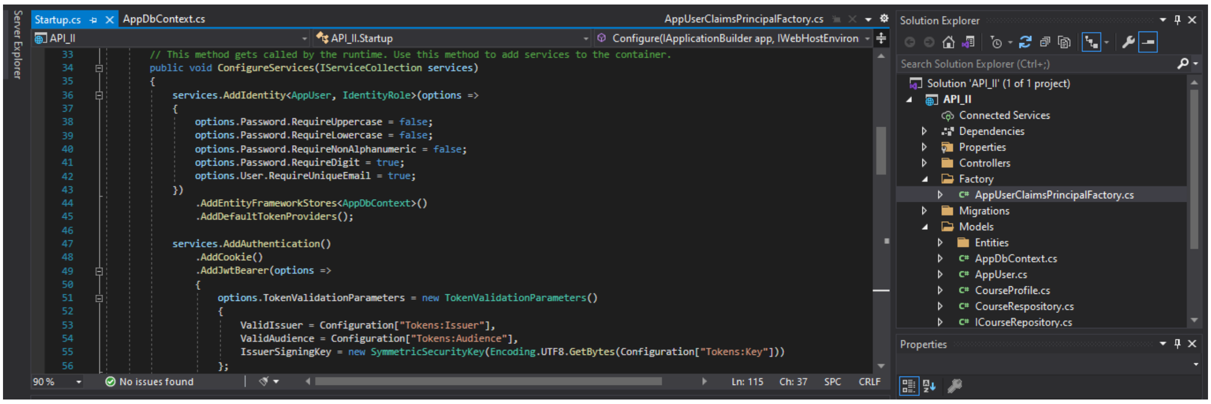
6) Create a class that will be aware of AppUser Class called AppUserClaimsPrincipal



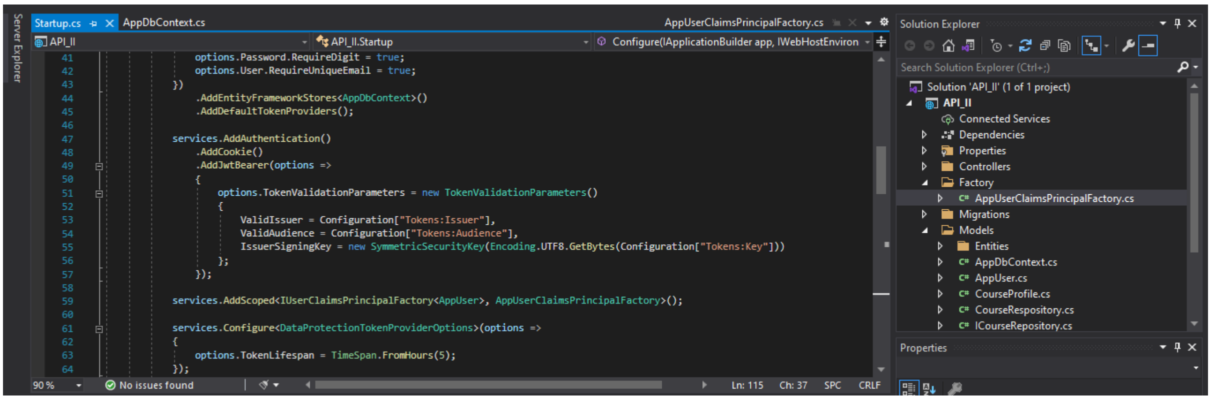
Startup.cs
AppDbContext.cs
AppUserClaimsPrincipalFactory.cs
API_II
API_II.Factory.AppUserClaimsPrincipalFactory
AppUserClaimsPrincipalFactory(UserManager<AppU:
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'API_II' (1 of 1 project)
API_II
Connected Services
Dependencies
Properties
Controllers
Factory
AppUserClaimsPrincipalFactory.cs
Migrations
Models
Entities
AppDbContext.cs
AppUser.cs
CourseProfile.cs
CourseRespository.cs
ICourseRepository.cs
Properties
AppUserClaimsPrincipalFactory.cs File Properties
90 %
No issues found
Ln: 1
Ch: 1
SPC
CRLF

```
using API_II.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.Options;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace API_II.Factory
{
    public class AppUserClaimsPrincipalFactory:UserClaimsPrincipalFactory<AppUser, IdentityRole>
    {
        public AppUserClaimsPrincipalFactory(UserManager<AppUser> userManager
            , RoleManager<IdentityRole> roleManager
            , IOptions<IdentityOptions> optionsAccesor): base(userManager, roleManager, optionsAccesor)
        {
        }
    }
}
```

7) Add configurations/ security stuff in startup.cs

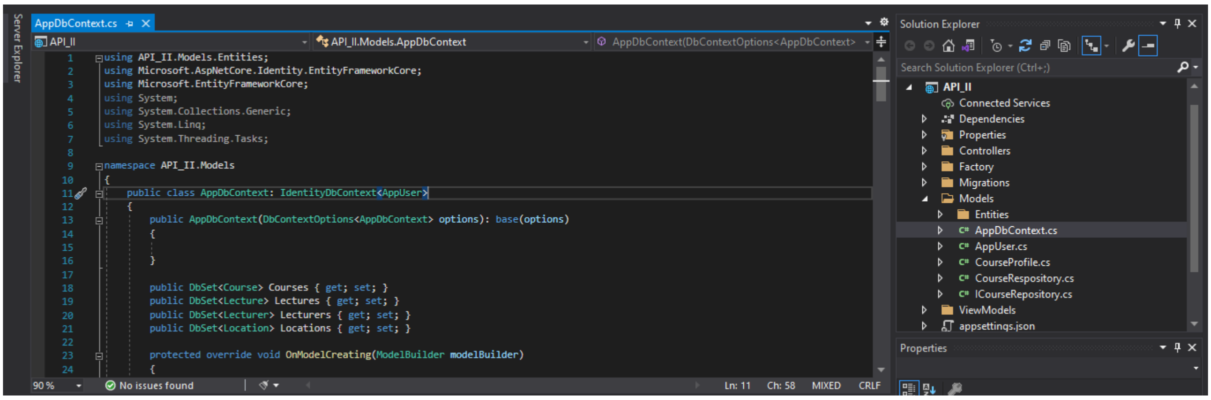


```
        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddIdentity<AppUser, IdentityRole>(options =>
            {
                options.Password.RequireUppercase = false;
                options.Password.RequireLowercase = false;
                options.Password.RequireNonAlphanumeric = false;
                options.Password.RequireDigit = true;
                options.User.RequireUniqueEmail = true;
            })
                .AddEntityFrameworkStores<AppDbContext>()
                .AddDefaultTokenProviders();

            services.AddAuthentication()
                .AddCookie()
                .AddJwtBearer(options =>
                {
                    options.TokenValidationParameters = new TokenValidationParameters()
                    {
                        ValidIssuer = Configuration["Tokens:Issuer"],
                        ValidAudience = Configuration["Tokens:Audience"],
                        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["Tokens:Key"]))
                    };
```

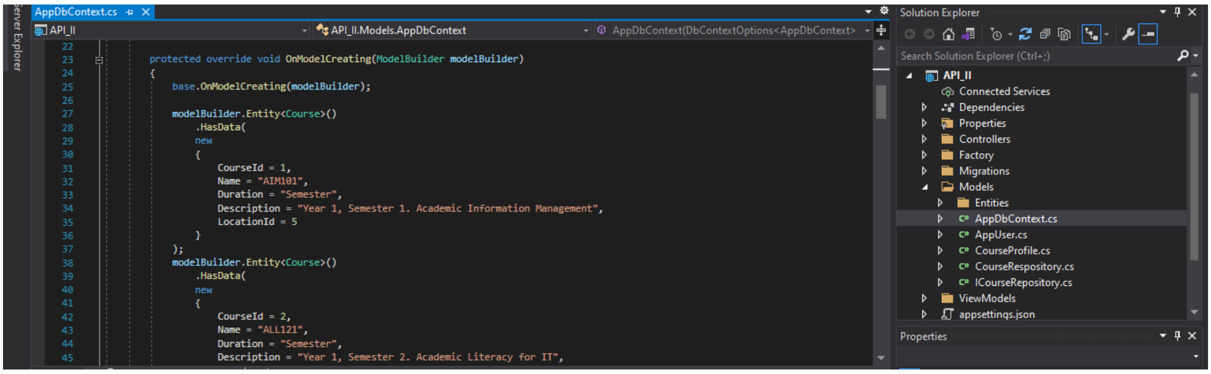


```
                options.Password.RequireDigit = true;
                options.User.RequireUniqueEmail = true;
            })
                .AddEntityFrameworkStores<AppDbContext>()
                .AddDefaultTokenProviders();

            services.AddAuthentication()
                .AddCookie()
                .AddJwtBearer(options =>
                {
                    options.TokenValidationParameters = new TokenValidationParameters()
                    {
                        ValidIssuer = Configuration["Tokens:Issuer"],
                        ValidAudience = Configuration["Tokens:Audience"],
                        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["Tokens:Key"]))
                    };
                });

            services.AddScoped<IUserClaimsPrincipalFactory<AppUser>, AppUserClaimsPrincipalFactory>();

            services.Configure<DataProtectionTokenProviderOptions>(options =>
            {
                options.TokenLifespan = TimeSpan.FromHours(5);
            });
```

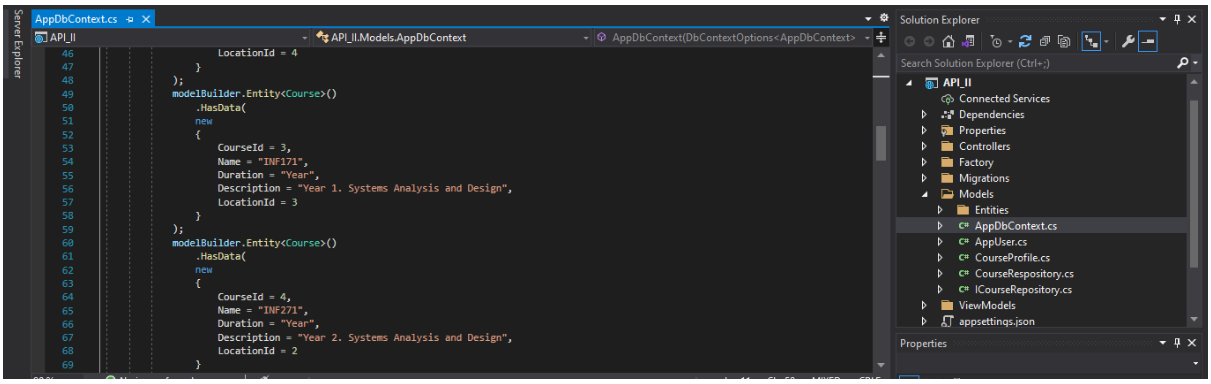
AppDbContext.cs



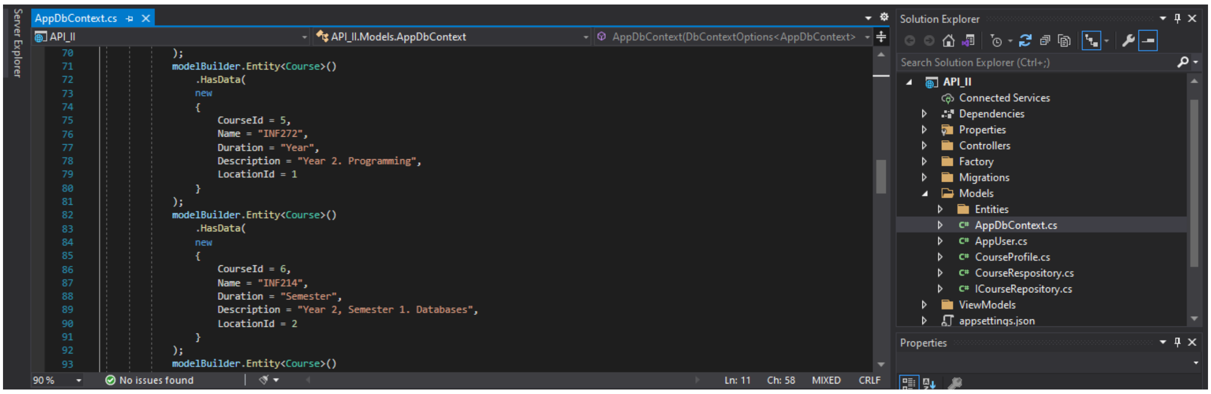
```
using API_II.Models.Entities;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace API_II.Models
{
    public class AppDbContext: IdentityDbContext<AppUser>
    {
        public AppDbContext(DbContextOptions<AppDbContext> options): base(options)
        {

        }

        public DbSet<Course> Courses { get; set; }
        public DbSet<Lecture> Lectures { get; set; }
        public DbSet<Lecturer> Lecturers { get; set; }
        public DbSet<Location> Locations { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
```



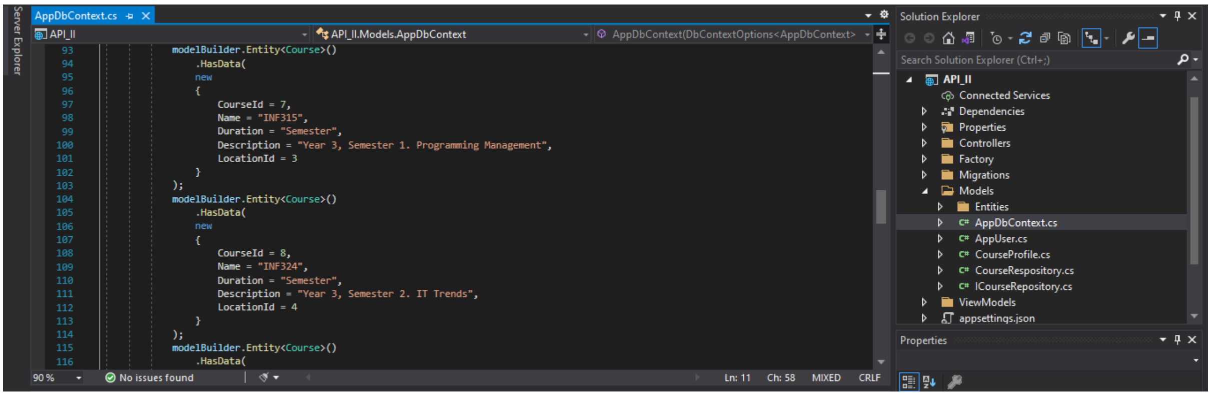
```
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 1,
                    Name = "AIM101",
                    Duration = "Semester",
                    Description = "Year 1, Semester 1. Academic Information Management",
                    LocationId = 5
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 2,
                    Name = "ALL121",
                    Duration = "Semester",
                    Description = "Year 1, Semester 2. Academic Literacy for IT",
```



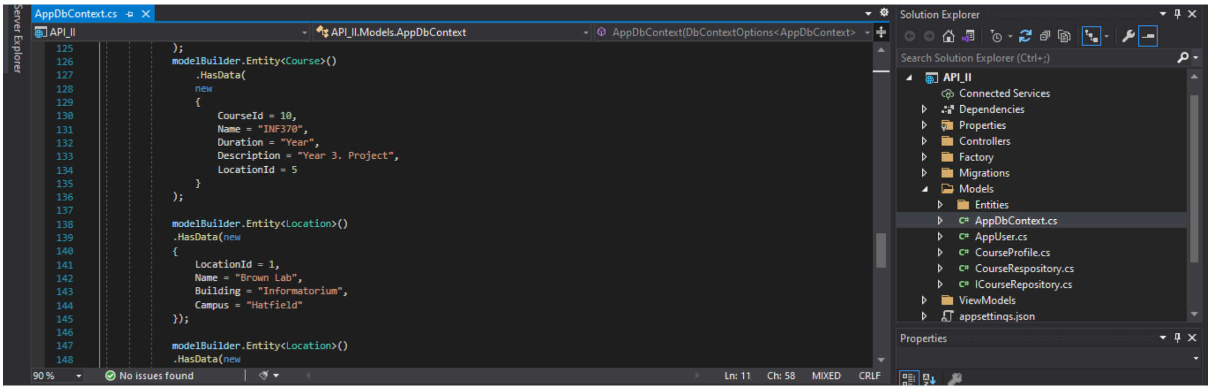
```
                    LocationId = 4
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 3,
                    Name = "INF171",
                    Duration = "Year",
                    Description = "Year 1. Systems Analysis and Design",
                    LocationId = 3
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 4,
                    Name = "INF271",
                    Duration = "Year",
                    Description = "Year 2. Systems Analysis and Design",
                    LocationId = 2
                }
```



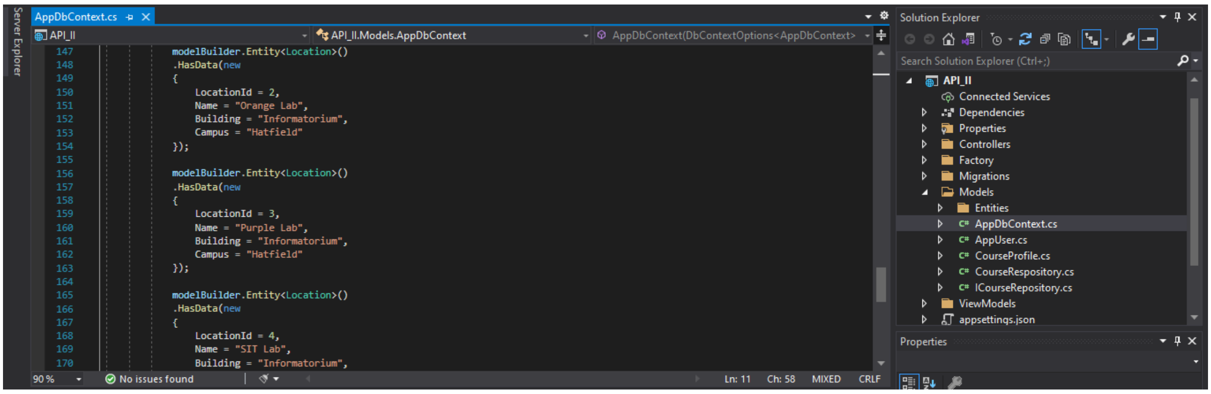
```
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 5,
                    Name = "INF272",
                    Duration = "Year",
                    Description = "Year 2. Programming",
                    LocationId = 1
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 6,
                    Name = "INF214",
                    Duration = "Semester",
                    Description = "Year 2, Semester 1. Databases",
                    LocationId = 2
                }
            );
            modelBuilder.Entity<Course>()
```



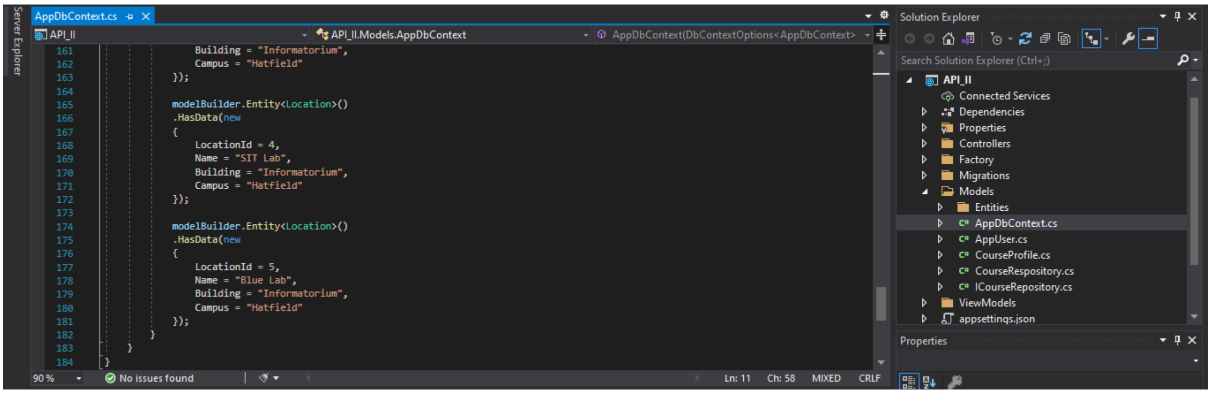
```
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 7,
                    Name = "INF315",
                    Duration = "Semester",
                    Description = "Year 3, Semester 1. Programming Management",
                    LocationId = 3
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 8,
                    Name = "INF324",
                    Duration = "Semester",
                    Description = "Year 3, Semester 2. IT Trends",
                    LocationId = 4
                }
            );
            modelBuilder.Entity<Course>()
                .HasData(
```



```
            );
            modelBuilder.Entity<Course>()
                .HasData(
                new
                {
                    CourseId = 10,
                    Name = "INF370",
                    Duration = "Year",
                    Description = "Year 3. Project",
                    LocationId = 5
                }
            );

            modelBuilder.Entity<Location>()
            .HasData(new
            {
                LocationId = 1,
                Name = "Brown Lab",
                Building = "Informatorium",
                Campus = "Hatfield"
            });

            modelBuilder.Entity<Location>()
            .HasData(new
```



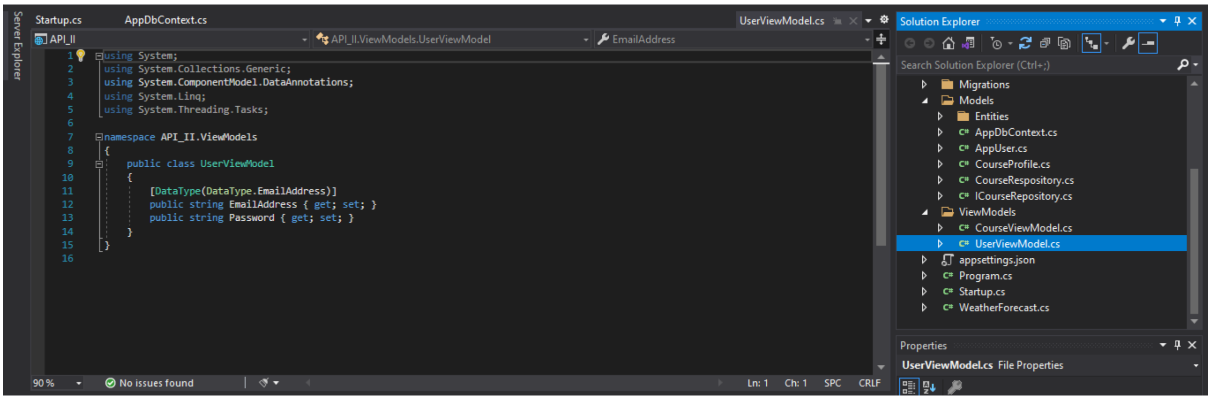
```
            modelBuilder.Entity<Location>()
            .HasData(new
            {
                LocationId = 2,
                Name = "Orange Lab",
                Building = "Informatorium",
                Campus = "Hatfield"
            });

            modelBuilder.Entity<Location>()
            .HasData(new
            {
                LocationId = 3,
                Name = "Purple Lab",
                Building = "Informatorium",
                Campus = "Hatfield"
            });

            modelBuilder.Entity<Location>()
            .HasData(new
            {
                LocationId = 4,
                Name = "SIT Lab",
                Building = "Informatorium",
```



```
                Building = "Informatorium",
                Campus = "Hatfield"
            });

            modelBuilder.Entity<Location>()
            .HasData(new
            {
                LocationId = 4,
                Name = "SIT Lab",
                Building = "Informatorium",
                Campus = "Hatfield"
            });

            modelBuilder.Entity<Location>()
            .HasData(new
            {
                LocationId = 5,
                Name = "Blue Lab",
                Building = "Informatorium",
                Campus = "Hatfield"
            });
        }
    }
}
```

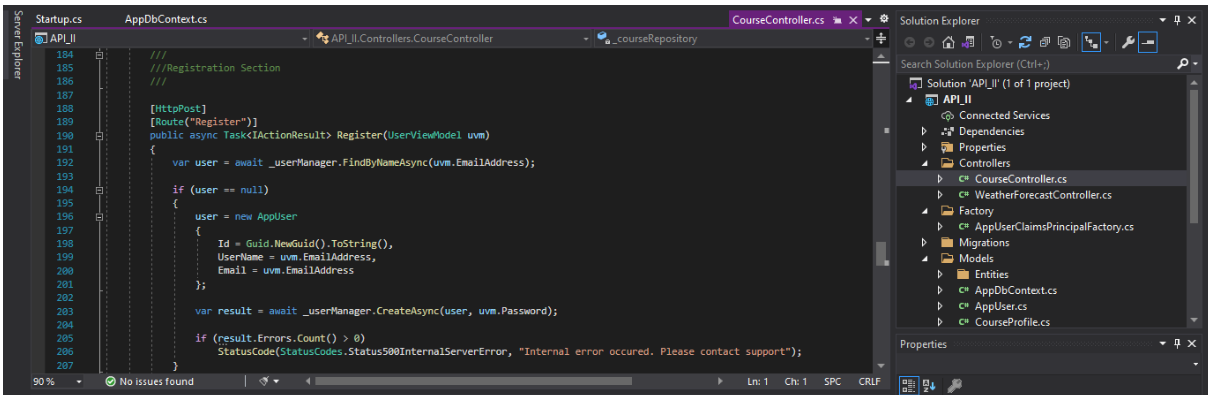
Register User

UserViewModel.cs

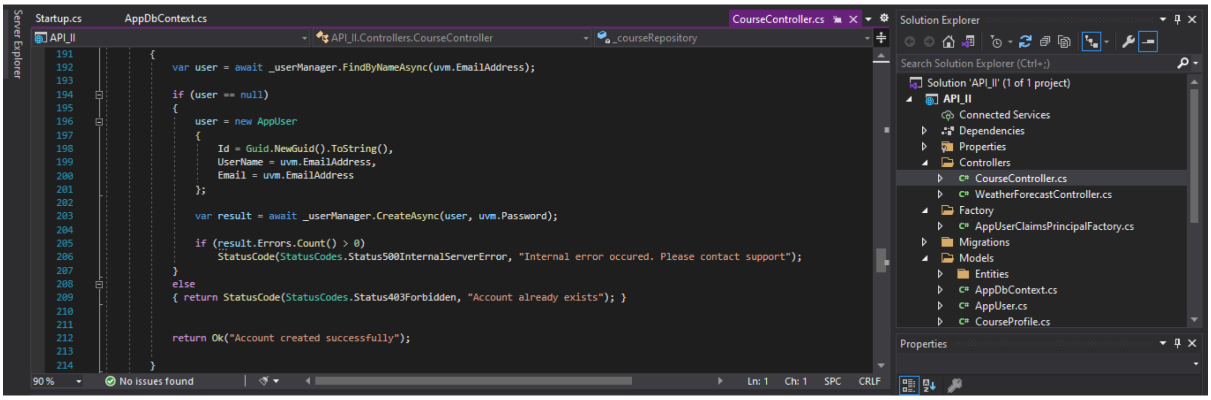


```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace API_II.ViewModels
{
    public class UserViewModel
    {
        [DataType(DataType.EmailAddress)]
        public string EmailAddress { get; set; }
        public string Password { get; set; }
    }
}
```

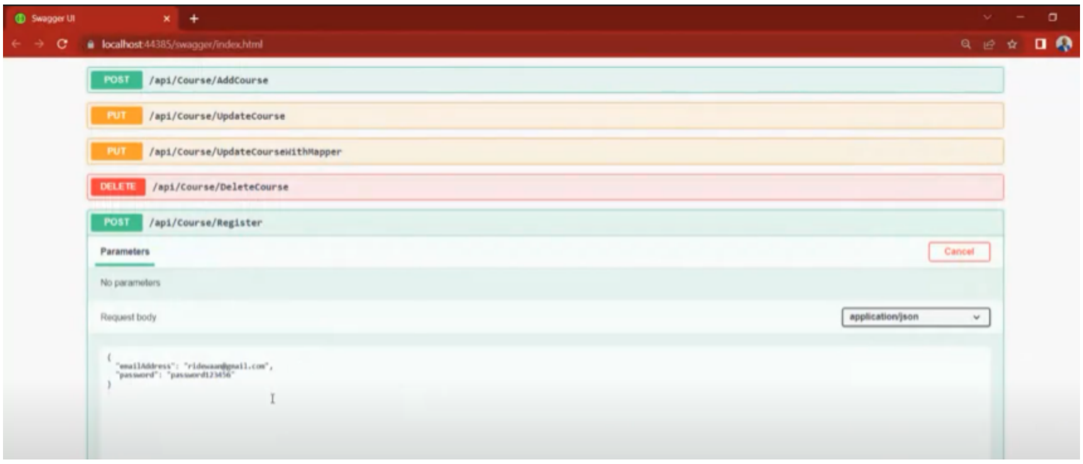
CourseController.cs



```
        ///
        ///Registration Section
        ///

        [HttpPost]
        [Route("Register")]
        public async Task<IActionResult> Register(UserViewModel uvm)
        {
            var user = await _userManager.FindByNameAsync(uvm.EmailAddress);

            if (user == null)
            {
                user = new AppUser
                {
                    Id = Guid.NewGuid().ToString(),
                    UserName = uvm.EmailAddress,
                    Email = uvm.EmailAddress
                };

                var result = await _userManager.CreateAsync(user, uvm.Password);

                if (result.Errors.Count() > 0)
                    StatusCode(StatusCodes.Status500InternalServerError, "Internal error occured. Please contact support");
            }
```



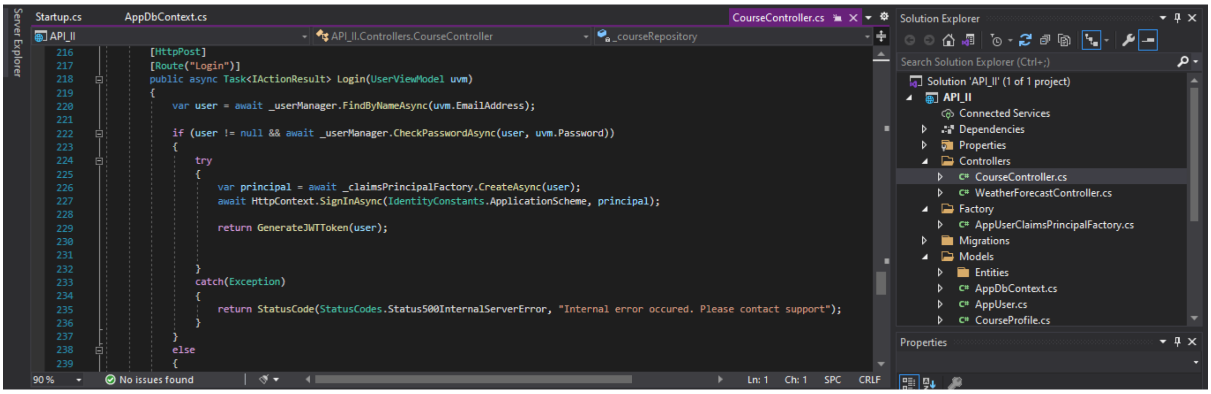
```
        {
            var user = await _userManager.FindByNameAsync(uvm.EmailAddress);

            if (user == null)
            {
                user = new AppUser
                {
                    Id = Guid.NewGuid().ToString(),
                    UserName = uvm.EmailAddress,
                    Email = uvm.EmailAddress
                };

                var result = await _userManager.CreateAsync(user, uvm.Password);

                if (result.Errors.Count() > 0)
                    StatusCode(StatusCodes.Status500InternalServerError, "Internal error occured. Please contact support");
            }
            else
            { return StatusCode(StatusCodes.Status403Forbidden, "Account already exists"); }


            return Ok("Account created successfully");

        }
```


Swagger UI
localhost:44385/swagger/index.html
POST /api/Course/AddCourse
PUT /api/Course/UpdateCourse
PUT /api/Course/UpdateCourseWithMapper
DELETE /api/Course/DeleteCourse
POST /api/Course/Register
Parameters
Cancel
No parameters
Request body
application/json

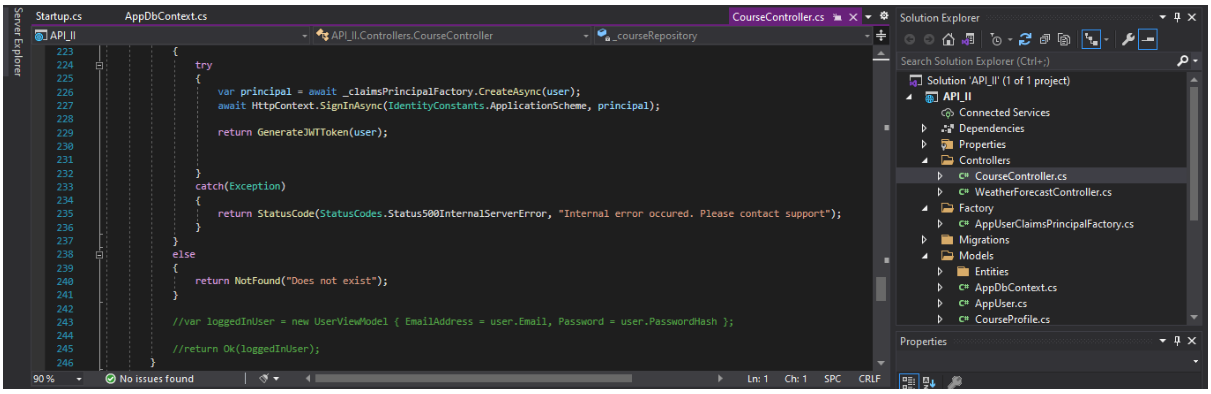
```
{
  "emailAddress": "ridewaan@gmail.com",
  "password": "password123456"
}
```

Login

CourseController.cs



```
        [HttpPost]
        [Route("Login")]
        public async Task<IActionResult> Login(UserViewModel uvm)
        {
            var user = await _userManager.FindByNameAsync(uvm.EmailAddress);

            if (user != null && await _userManager.CheckPasswordAsync(user, uvm.Password))
            {
                try
                {
                    var principal = await _claimsPrincipalFactory.CreateAsync(user);
                    await HttpContext.SignInAsync(IdentityConstants.ApplicationScheme, principal);

                    return GenerateJWTToken(user);


                }
                catch(Exception)
                {
                    return StatusCode(StatusCodes.Status500InternalServerError, "Internal error occured. Please contact support");
                }
            }
            else
            {
```



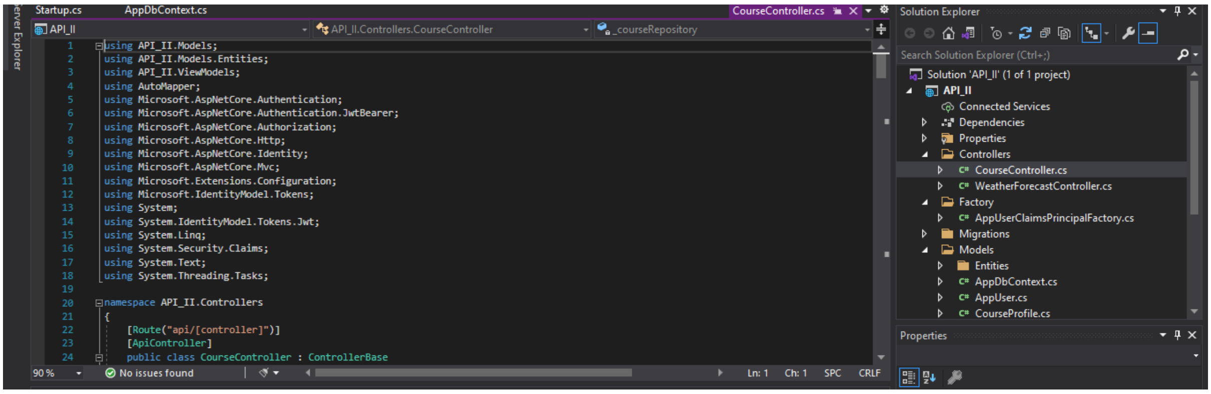
```
            {
                try
                {
                    var principal = await _claimsPrincipalFactory.CreateAsync(user);
                    await HttpContext.SignInAsync(IdentityConstants.ApplicationScheme, principal);

                    return GenerateJWTToken(user);


                }
                catch(Exception)
                {
                    return StatusCode(StatusCodes.Status500InternalServerError, "Internal error occured. Please contact support");
                }
            }
            else
            {
                return NotFound("Does not exist");
            }

            //var loggedInUser = new UserViewModel { EmailAddress = user.Email, Password = user.PasswordHash };

            //return Ok(loggedInUser);
        }
```

Create API Bearer JWT (To keep track of users)

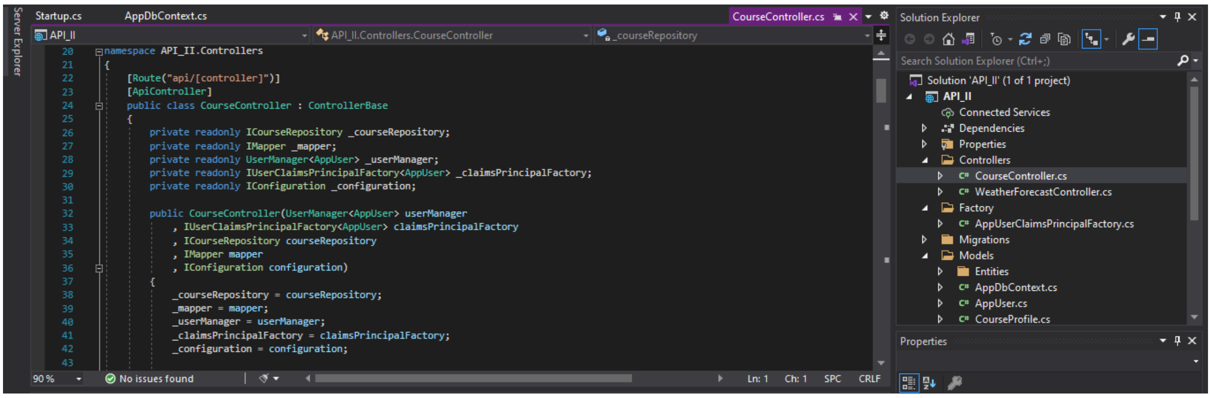
Install:

Microsoft.AspNetCore.Authentication.JwtBearer

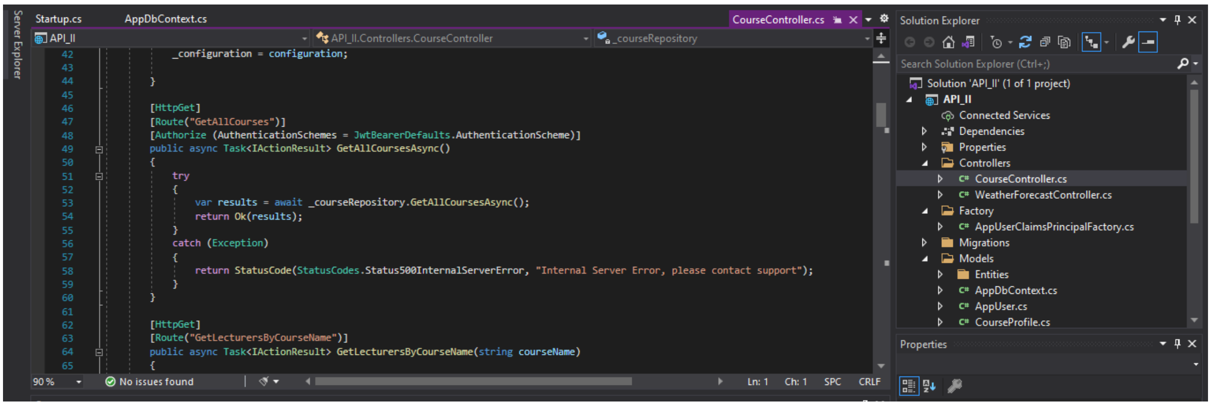
CourseController.cs



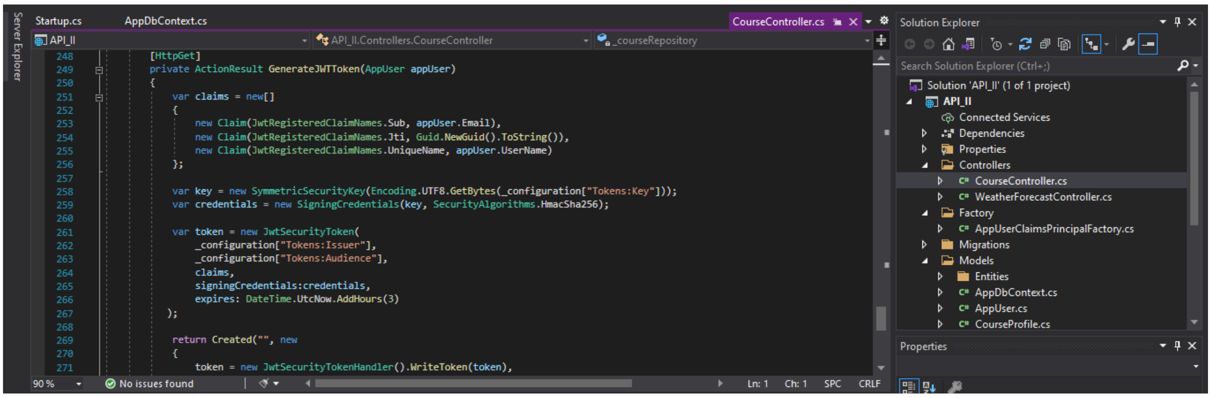
```csharp
using API_II.Models;
using API_II.Models.Entities;
using API_II.ViewModels;
using AutoMapper;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.IdentityModel.Tokens;
using System;
using System.IdentityModel.Tokens.Jwt;
using System.Linq;
using System.Security.Claims;
using System.Text;
using System.Threading.Tasks;

namespace API_II.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CourseController : ControllerBase
```



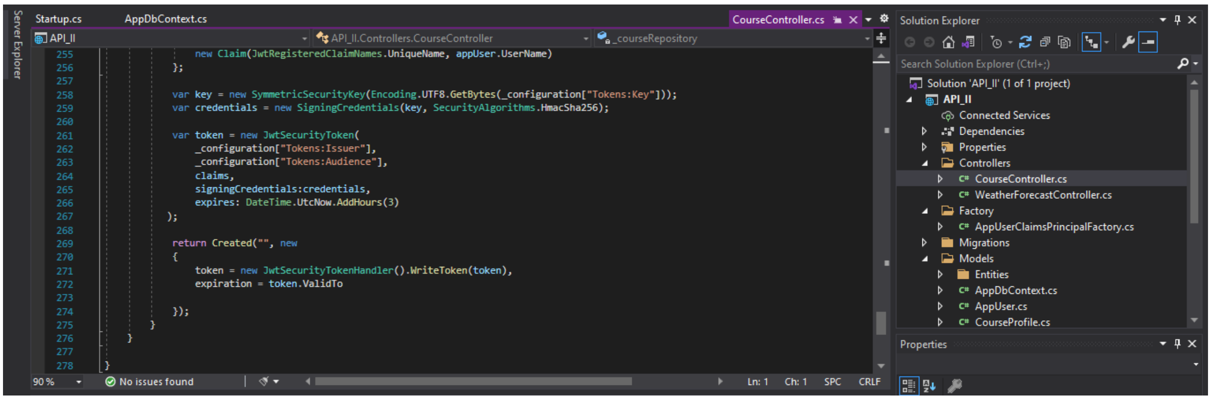
```csharp
namespace API_II.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CourseController : ControllerBase
    {
        private readonly ICourseRepository _courseRepository;
        private readonly IMapper _mapper;
        private readonly UserManager<AppUser> _userManager;
        private readonly IUserClaimsPrincipalFactory<AppUser> _claimsPrincipalFactory;
        private readonly IConfiguration _configuration;

        public CourseController(UserManager<AppUser> userManager
            , IUserClaimsPrincipalFactory<AppUser> claimsPrincipalFactory
            , ICourseRepository courseRepository
            , IMapper mapper
            , IConfiguration configuration)
        {
            _courseRepository = courseRepository;
            _mapper = mapper;
            _userManager = userManager;
            _claimsPrincipalFactory = claimsPrincipalFactory;
            _configuration = configuration;

```



```csharp
            _configuration = configuration;

        }

        [HttpGet]
        [Route("GetAllCourses")]
        [Authorize (AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
        public async Task<IActionResult> GetAllCoursesAsync()
        {
            try
            {
                var results = await _courseRepository.GetAllCoursesAsync();
                return Ok(results);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Internal Server Error, please contact support");
            }
        }

        [HttpGet]
        [Route("GetLecturersByCourseName")]
        public async Task<IActionResult> GetLecturersByCourseName(string courseName)
        {
```

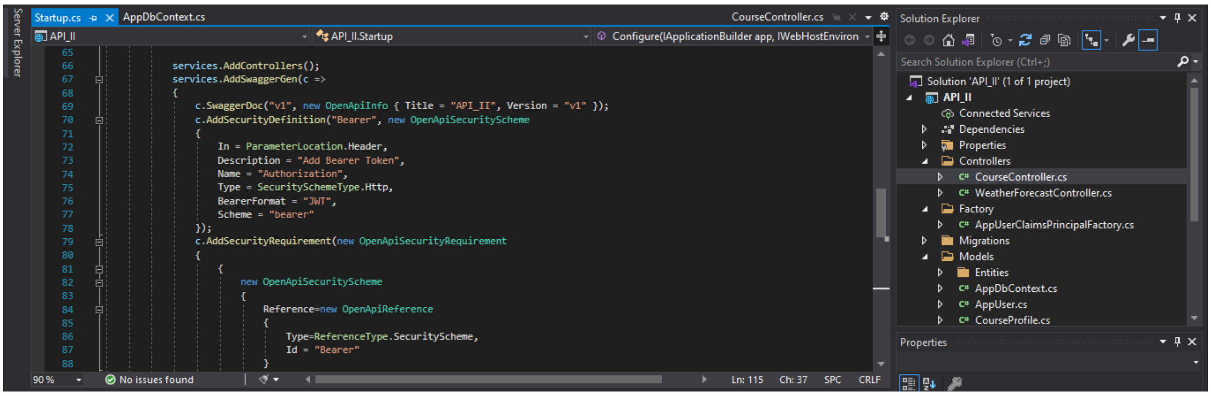


```
[HttpGet]
private ActionResult GenerateJWTToken(AppUser appUser)
{
    var claims = new[]
    {
        new Claim(JwtRegisteredClaimNames.Sub, appUser.Email),
        new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
        new Claim(JwtRegisteredClaimNames.UniqueName, appUser.UserName)
    };

    var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Tokens:Key"]));
    var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

    var token = new JwtSecurityToken(
        _configuration["Tokens:Issuer"],
        _configuration["Tokens:Audience"],
        claims,
        signingCredentials:credentials,
        expires: DateTime.UtcNow.AddHours(3)
   );

    return Created("", new
    {
        token = new JwtSecurityTokenHandler().WriteToken(token),
```

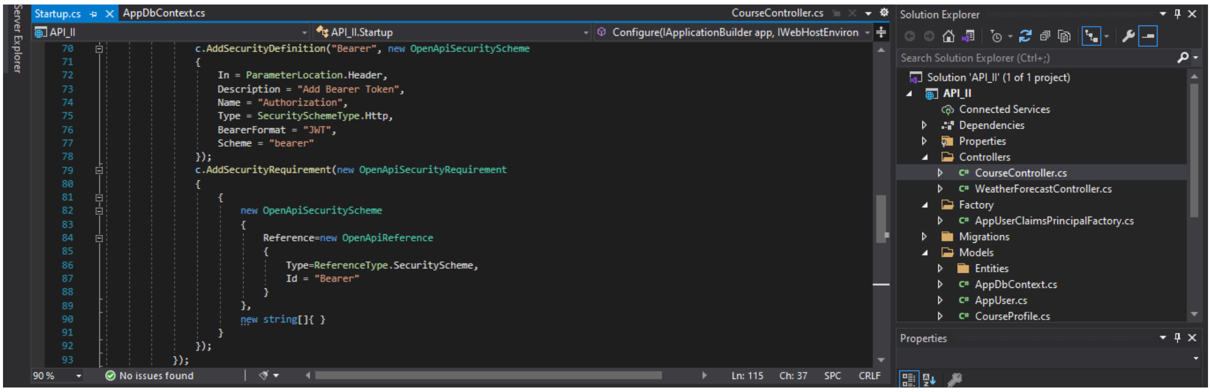


```
                new Claim(JwtRegisteredClaimNames.UniqueName, appUser.UserName)
            };

            var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Tokens:Key"]));
            var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(
                _configuration["Tokens:Issuer"],
                _configuration["Tokens:Audience"],
                claims,
                signingCredentials:credentials,
                expires: DateTime.UtcNow.AddHours(3)
           );

            return Created("", new
            {
                token = new JwtSecurityTokenHandler().WriteToken(token),
                expiration = token.ValidTo

            });
        }
    }

}
```

Configure Swagger & Enable Headers



```

services.AddControllers();
services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo { Title = "API_II", Version = "v1" });
    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        In = ParameterLocation.Header,
        Description = "Add Bearer Token",
        Name = "Authorization",
        Type = SecuritySchemeType.Http,
        BearerFormat = "JWT",
        Scheme = "bearer"
    });
    c.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
                Reference=new OpenApiReference
                {
                    Type=ReferenceType.SecurityScheme,
                    Id = "Bearer"
                }
```



```
    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        In = ParameterLocation.Header,
        Description = "Add Bearer Token",
        Name = "Authorization",
        Type = SecuritySchemeType.Http,
        BearerFormat = "JWT",
        Scheme = "bearer"
    });
    c.AddSecurityRequirement(new OpenApiSecurityRequirement
    {
        {
            new OpenApiSecurityScheme
            {
                Reference=new OpenApiReference
                {
                    Type=ReferenceType.SecurityScheme,
                    Id = "Bearer"
                }
            },
            new string[]{ }
        }
    });
});
```

Endpoint to Accept Jwt Token

```
[Authorize (AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
```