BASIC AUTHENTICATION

ASP.NET Core API

SETTING UP THE PROJECT

CREATE

- Create a new project – WEB API (UNDER WEB APPLICATIONS)

REMOVE

- REMOVE FILES UNDER CONTROLLERS FOLDER

REMOVE

- REMOVE FILE NAMED WEATHERFORECAST.cs

ADD

- ADD THE FOLLOWING FOLDERS
  - ENTITIES
  - MODELS
  - HELPERS
  - SERVICES

Controllers - define the end points / routes for the web api, controllers are the entry point into the web api from client applications via http requests.

Models - represent request and response models for controller methods, request models define the parameters for incoming requests, and response models can be used to define what data is returned.

ENTITIES

Represent the application data

SERVICES

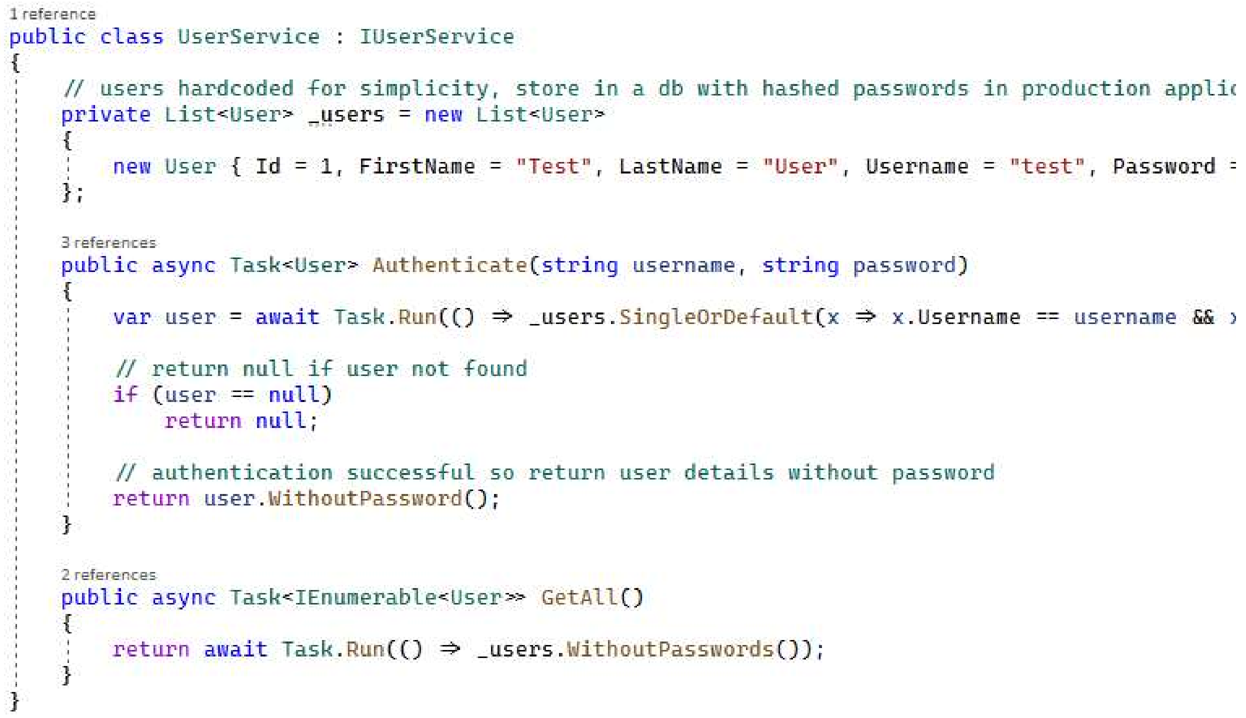
HELPERS

Anything that doesn’t fit into the previous folder scaffolds

LET’S GET STARTED!

UserService
ExtensionMethods
BasicAuthentcationHandler
Startup

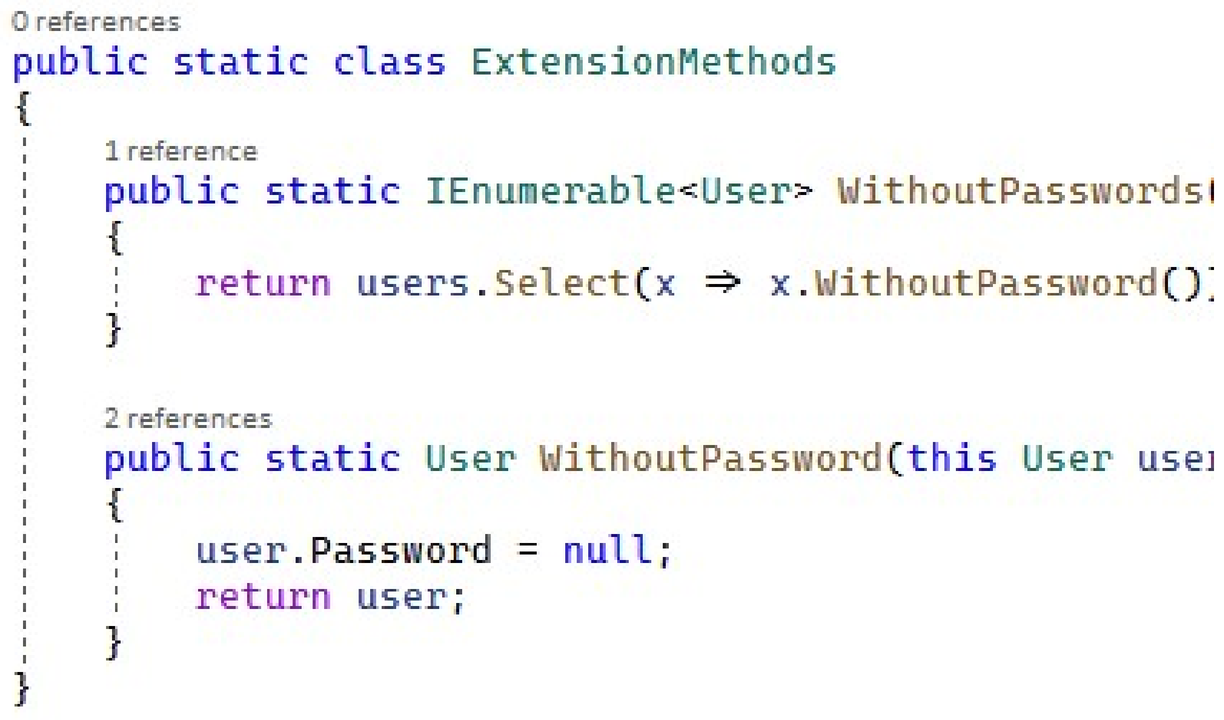
CODE REVIEW



```
1 reference
public class UserService : IUserService
{
    // users hardcoded for simplicity, store in a db with hashed passwords in production applic
    private List<User> _users = new List<User>
    {
        new User { Id = 1, FirstName = "Test", LastName = "User", Username = "test", Password =
    };

    3 references
    public async Task<User> Authenticate(string username, string password)
    {
        var user = await Task.Run(() => _users.SingleOrDefault(x => x.Username == username && x

        // return null if user not found
        if (user == null)
            return null;

        // authentication successful so return user details without password
        return user.WithoutPassword();
    }

    2 references
    public async Task<IEnumerable<User>> GetAll()
    {
        return await Task.Run(() => _users.WithoutPasswords());
    }
}
```

USERSERVICE.CS

Code Review



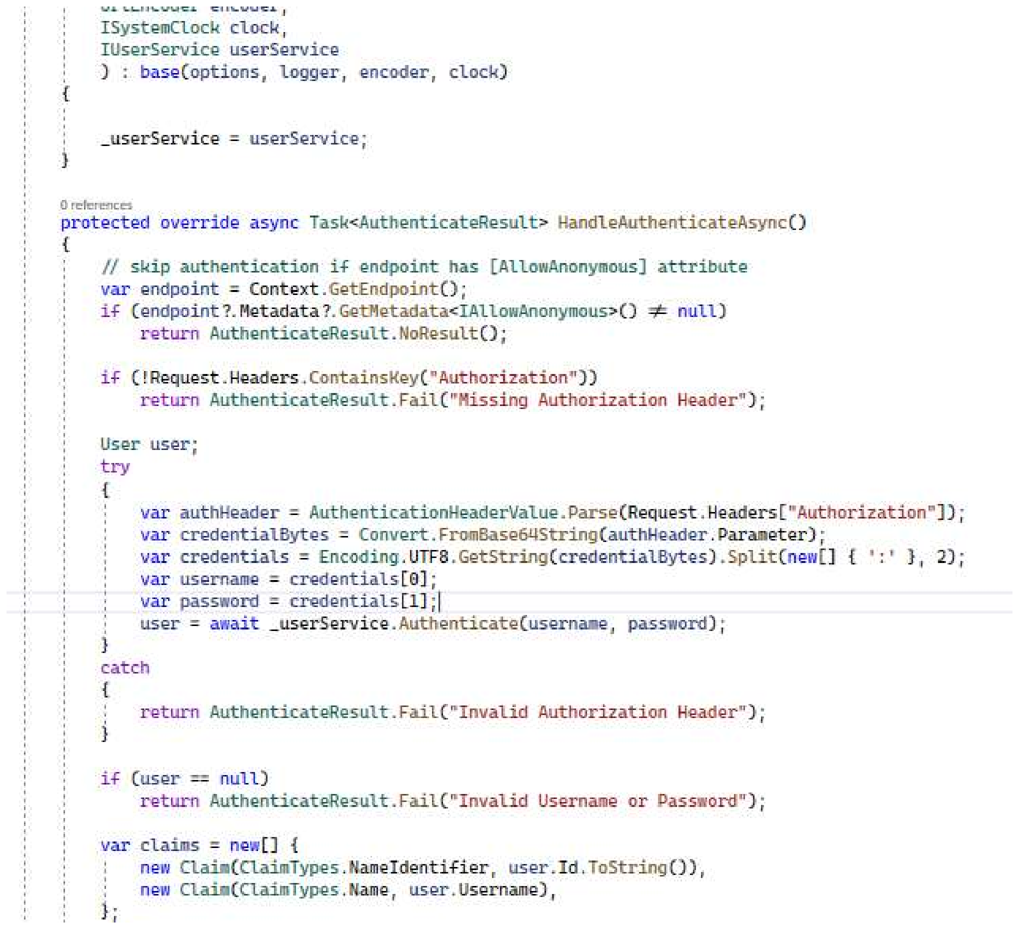
```
0 references
public static class ExtensionMethods
{
    1 reference
    public static IEnumerable<User> WithoutPasswords(
    {
        return users.Select(x => x.WithoutPassword()
    }

    2 references
    public static User WithoutPassword(this User use
    {
        user.Password = null;
        return user;
    }
}
```

EXTENSIONMETHODS.CS

Code Review

BASICAUTHENTICATIONHANDLER.CS

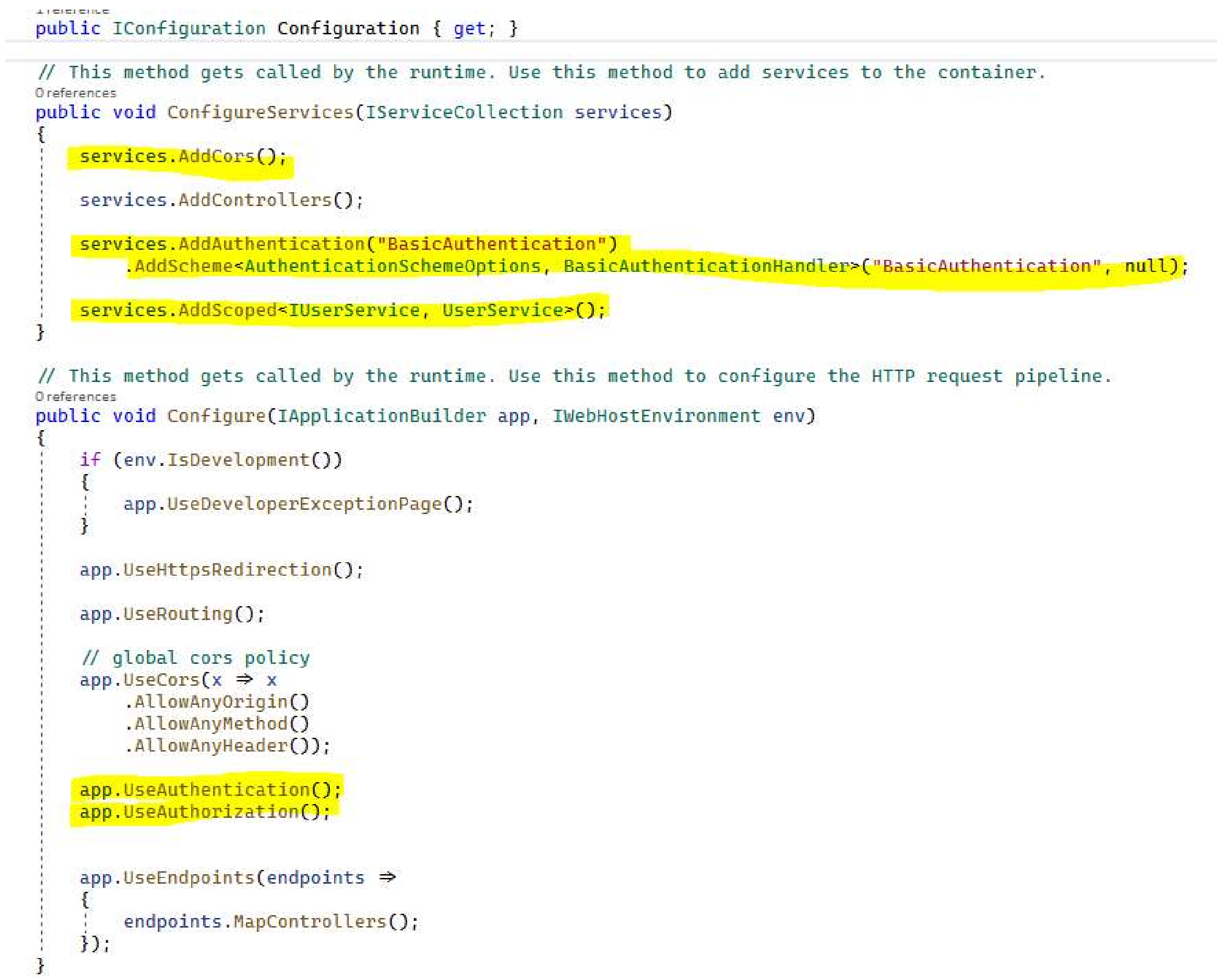
Code Review



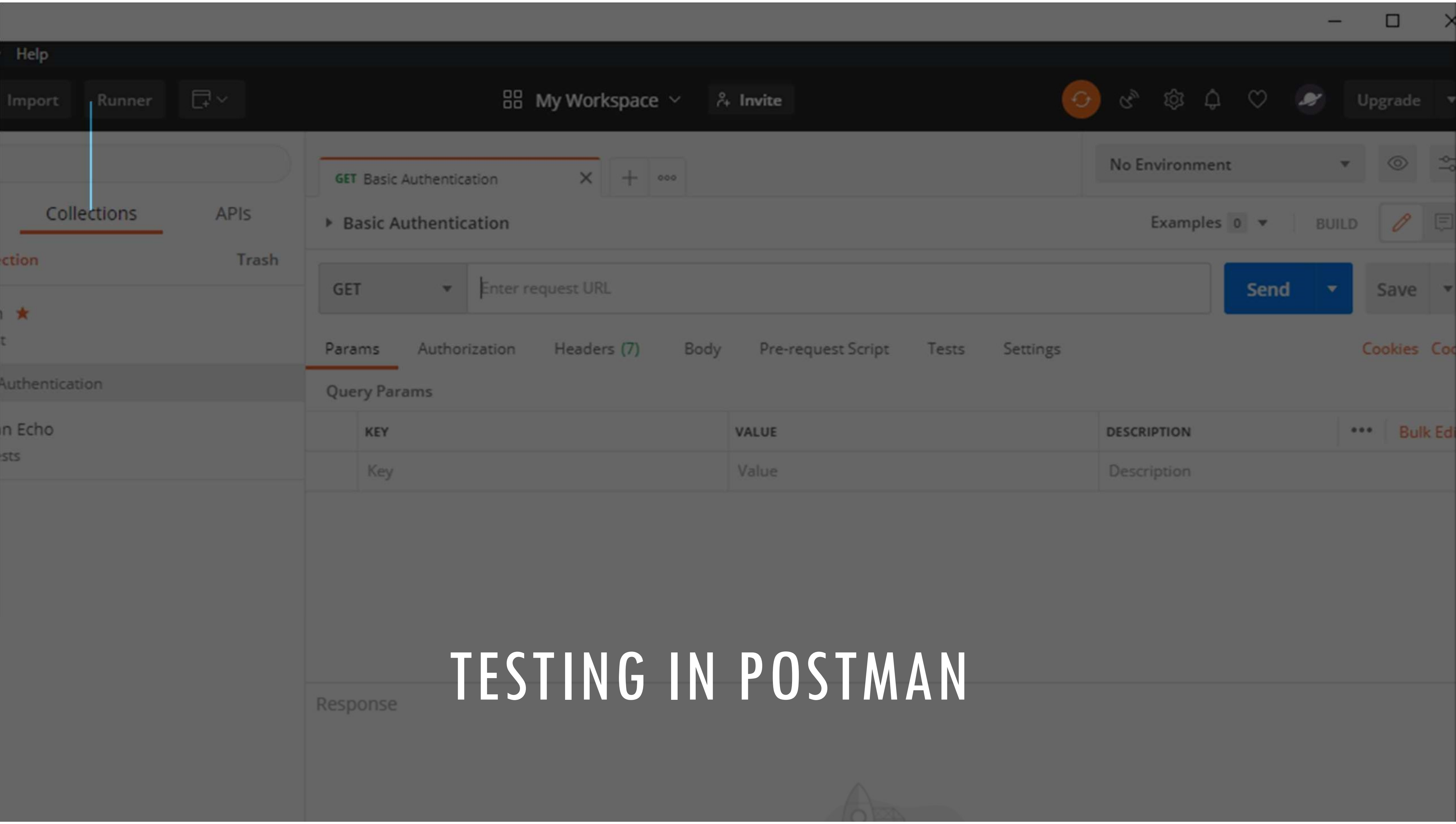
```
    ISystemClock clock,
    IUserService userService
    ) : base(options, logger, encoder, clock)
{
    _userService = userService;
}

0 references
protected override async Task<AuthenticateResult> HandleAuthenticateAsync()
{
    // skip authentication if endpoint has [AllowAnonymous] attribute
    var endpoint = Context.GetEndpoint();
    if (endpoint?.Metadata?.GetMetadata<IAllowAnonymous>() ≠ null)
        return AuthenticateResult.NoResult();

    if (!Request.Headers.ContainsKey("Authorization"))
        return AuthenticateResult.Fail("Missing Authorization Header");

    User user;
    try
    {
        var authHeader = AuthenticationHeaderValue.Parse(Request.Headers["Authorization"]);
        var credentialBytes = Convert.FromBase64String(authHeader.Parameter);
        var credentials = Encoding.UTF8.GetString(credentialBytes).Split(new[] { ':' }, 2);
        var username = credentials[0];
        var password = credentials[1];
        user = await _userService.Authenticate(username, password);
    }
    catch
    {
        return AuthenticateResult.Fail("Invalid Authorization Header");
    }

    if (user == null)
        return AuthenticateResult.Fail("Invalid Username or Password");

    var claims = new[] {
        new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
        new Claim(ClaimTypes.Name, user.Username),
    };
```

STARTUP.CS

Code Review



```
public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors();

    services.AddControllers();

    services.AddAuthentication("BasicAuthentication")
        .AddScheme<AuthenticationSchemeOptions, BasicAuthenticationHandler>("BasicAuthentication", null);

    services.AddScoped<IUserService, UserService>();
}

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

    app.UseHttpsRedirection();

    app.UseRouting();

    // global cors policy
    app.UseCors(x => x
        .AllowAnyOrigin()
        .AllowAnyMethod()
        .AllowAnyHeader());

    app.UseAuthentication();
    app.UseAuthorization();


    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```



TESTING IN POSTMAN

CONCLUSION