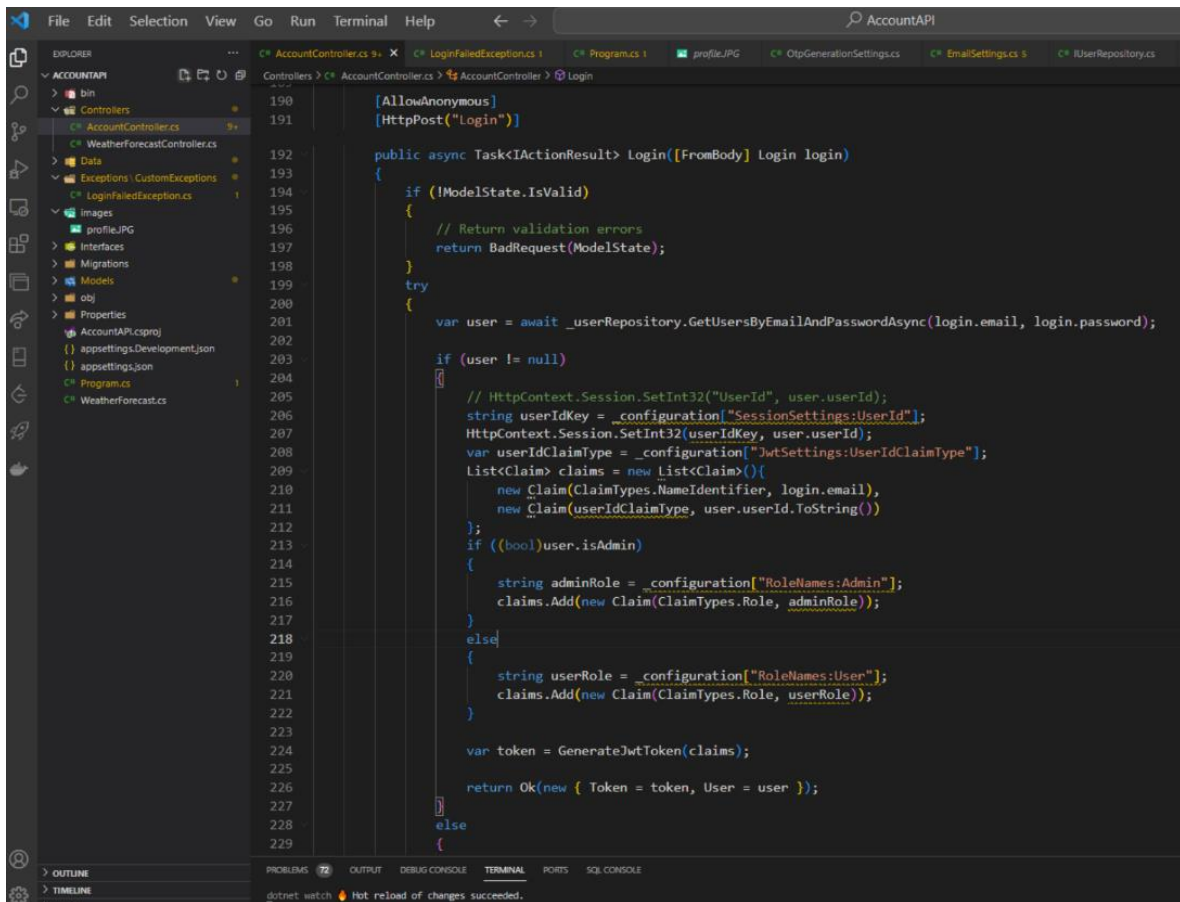


# Dotnet Task 1 – WebAPI with JWT Authentication

## Jwt Authentication:



The screenshot shows the Visual Studio IDE with the 'AccountAPI' project open. The 'Controllers' folder is expanded, and 'AccountController.cs' is selected. The 'Login' method is being edited. The code implements a login endpoint that checks if the user exists, generates a JWT token, and returns it along with the user information. It also includes error handling for login failures.

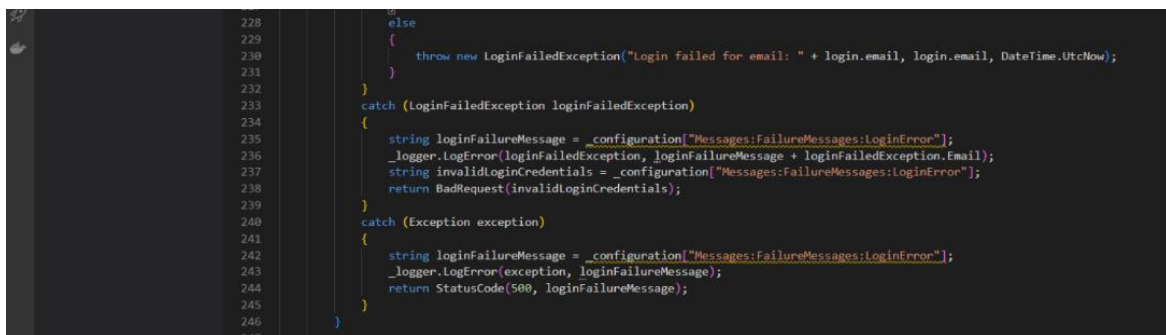
```
[AllowAnonymous]
[HttpPost("Login")]

public async Task<IActionResult> Login([FromBody] Login login)
{
    if (!ModelState.IsValid)
    {
        // Return validation errors
        return BadRequest(ModelState);
    }
    try
    {
        var user = await _userRepository.GetUsersByEmailAndPasswordAsync(login.email, login.password);

        if (user != null)
        {
            // HttpContext.Session.SetInt32("UserId", user.userId);
            string userIdKey = _configuration["SessionSettings:UserId"];
            HttpContext.Session.SetInt32(userIdKey, user.userId);
            var userIdClaimType = _configuration["JwtSettings:UserIdClaimType"];
            List<Claim> claims = new List<Claim>()
            {
                new Claim(ClaimTypes.NameIdentifier, login.email),
                new Claim(userIdClaimType, user.userId.ToString())
            };
            if ((bool)user.isAdmin)
            {
                string adminRole = _configuration["RoleNames:Admin"];
                claims.Add(new Claim(ClaimTypes.Role, adminRole));
            }
            else
            {
                string userRole = _configuration["RoleNames:User"];
                claims.Add(new Claim(ClaimTypes.Role, userRole));
            }

            var token = GenerateJwtToken(claims);

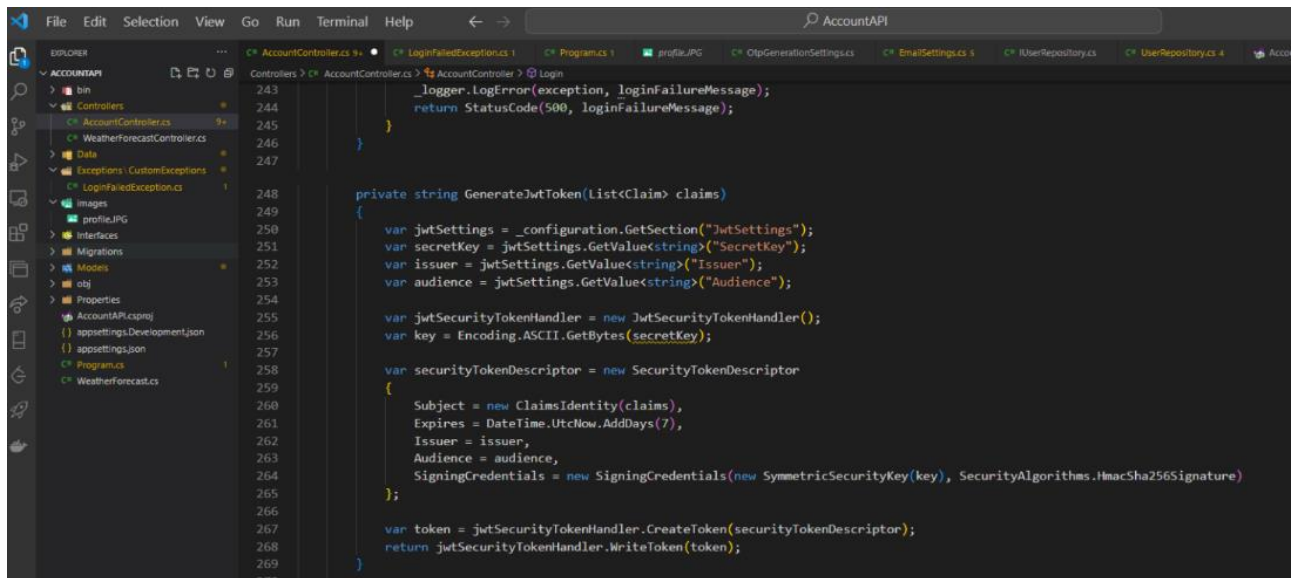
            return Ok(new { Token = token, User = user });
        }
        else
        {
            throw new LoginFailedException("Login failed for email: " + login.email, login.email, DateTime.UtcNow);
        }
    }
    catch (LoginFailedException loginFailedException)
    {
        string loginFailureMessage = _configuration["Messages:FailureMessages:LoginError"];
        _logger.LogError(loginFailedException, loginFailureMessage + loginFailedException.Email);
        string invalidLoginCredentials = _configuration["Messages:FailureMessages:LoginError"];
        return BadRequest(invalidLoginCredentials);
    }
    catch (Exception exception)
    {
        string loginFailureMessage = _configuration["Messages:FailureMessages:LoginError"];
        _logger.LogError(exception, loginFailureMessage);
        return StatusCode(500, loginFailureMessage);
    }
}
```



This block shows the continuation of the 'Login' method, specifically the error handling section. It catches 'LoginFailedException' and 'Exception' and returns appropriate responses based on the configuration.

```

        {
            throw new LoginFailedException("Login failed for email: " + login.email, login.email, DateTime.UtcNow);
        }
    }
    catch (LoginFailedException loginFailedException)
    {
        string loginFailureMessage = _configuration["Messages:FailureMessages:LoginError"];
        _logger.LogError(loginFailedException, loginFailureMessage + loginFailedException.Email);
        string invalidLoginCredentials = _configuration["Messages:FailureMessages:LoginError"];
        return BadRequest(invalidLoginCredentials);
    }
    catch (Exception exception)
    {
        string loginFailureMessage = _configuration["Messages:FailureMessages:LoginError"];
        _logger.LogError(exception, loginFailureMessage);
        return StatusCode(500, loginFailureMessage);
    }
}
```



```
File Edit Selection View Go Run Terminal Help
AccountAPI

EXPLORER
  ACCOUNTAPI
    bin
    Controllers
      AccountController.cs
      WeatherForecastController.cs
    Data
    Exceptions: CustomExceptions
      LoginFailedException.cs
    Images
    profile.JPG
    Interfaces
    Migrations
    Models
    obj
    Properties
    AccountAPICsproj
    appsettings.Development.json
    appsettings.json
    Program.cs
    WeatherForecast.cs

243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270

_login.Logger.LogError(exception, loginFailureMessage);
return StatusCode(500, loginFailureMessage);
}

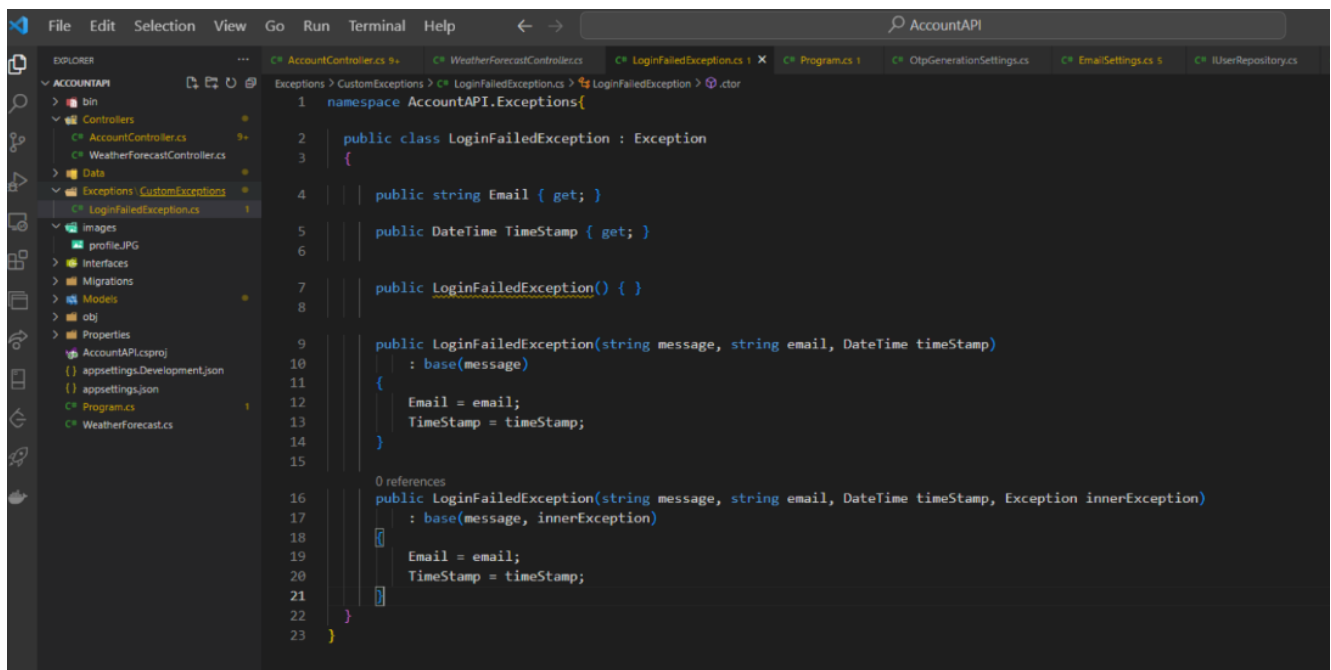
private string GenerateJwtToken(List<Claim> claims)
{
    var jwtSettings = _configuration.GetSection("JwtSettings");
    var secretKey = jwtSettings.GetValue<string>("SecretKey");
    var issuer = jwtSettings.GetValue<string>("Issuer");
    var audience = jwtSettings.GetValue<string>("Audience");

    var jwtSecurityTokenHandler = new JwtSecurityTokenHandler();
    var key = Encoding.ASCII.GetBytes(secretKey);

    var securityTokenDescriptor = new SecurityTokenDescriptor
    {
        Subject = new ClaimsIdentity(claims),
        Expires = DateTime.UtcNow.AddDays(7),
        Issuer = issuer,
        Audience = audience,
        SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
    };

    var token = jwtSecurityTokenHandler.CreateToken(securityTokenDescriptor);
    return jwtSecurityTokenHandler.WriteToken(token);
}
```

## Custom Exception:



```
File Edit Selection View Go Run Terminal Help
AccountAPI

EXPLORER
  ACCOUNTAPI
    bin
    Controllers
      AccountController.cs
      WeatherForecastController.cs
    Data
    Exceptions: CustomExceptions
      LoginFailedException.cs
    Images
    profile.JPG
    Interfaces
    Migrations
    Models
    obj
    Properties
    AccountAPICsproj
    appsettings.Development.json
    appsettings.json
    Program.cs
    WeatherForecast.cs

Exceptions > CustomExceptions > LoginFailedException.cs > LoginFailedException > .ctor
1 namespace AccountAPI.Exceptions{
2
3     public class LoginFailedException : Exception
4     {
5
6         public string Email { get; }
7
8         public DateTime TimeStamp { get; }
9
10        public LoginFailedException() { }
11
12        public LoginFailedException(string message, string email, DateTime timeStamp)
13        : base(message)
14        {
15            Email = email;
16            TimeStamp = timeStamp;
17        }
18
19        0 references
20        public LoginFailedException(string message, string email, DateTime timeStamp, Exception innerException)
21        : base(message, innerException)
22        {
23            Email = email;
24            TimeStamp = timeStamp;
25        }
26    }
27 }
```

## Dispose method in IDisposable Interface:

