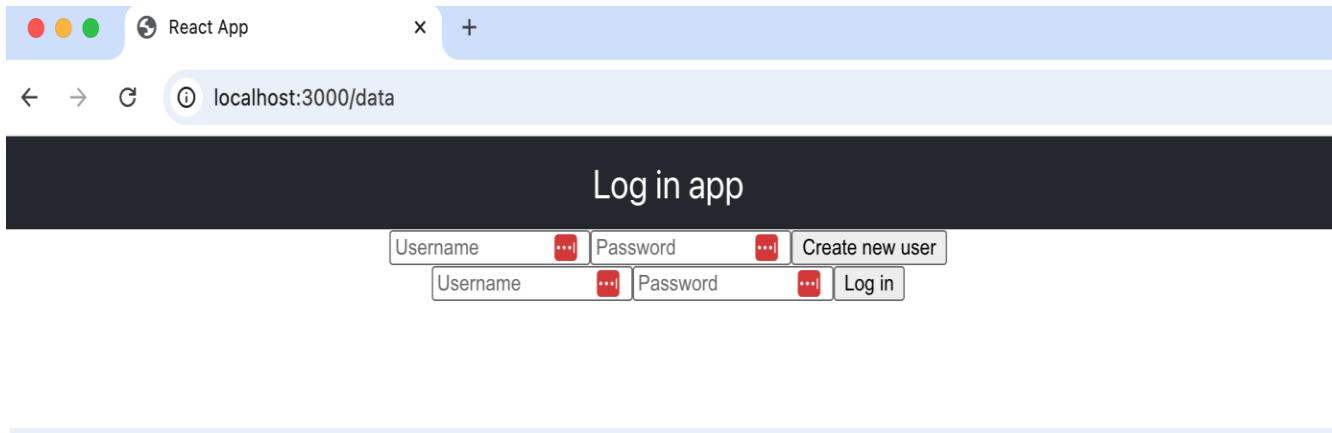
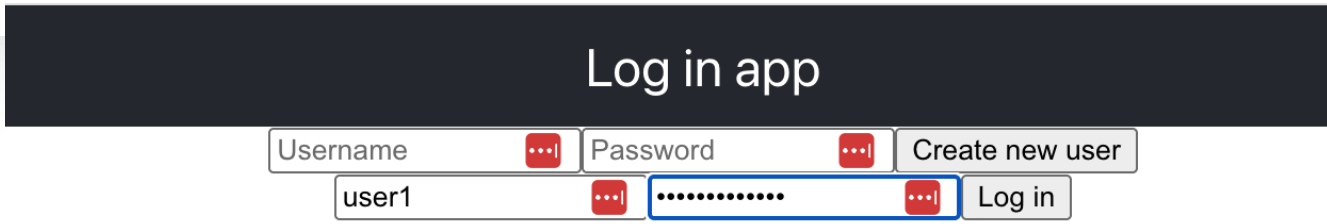
LOGIN-APP

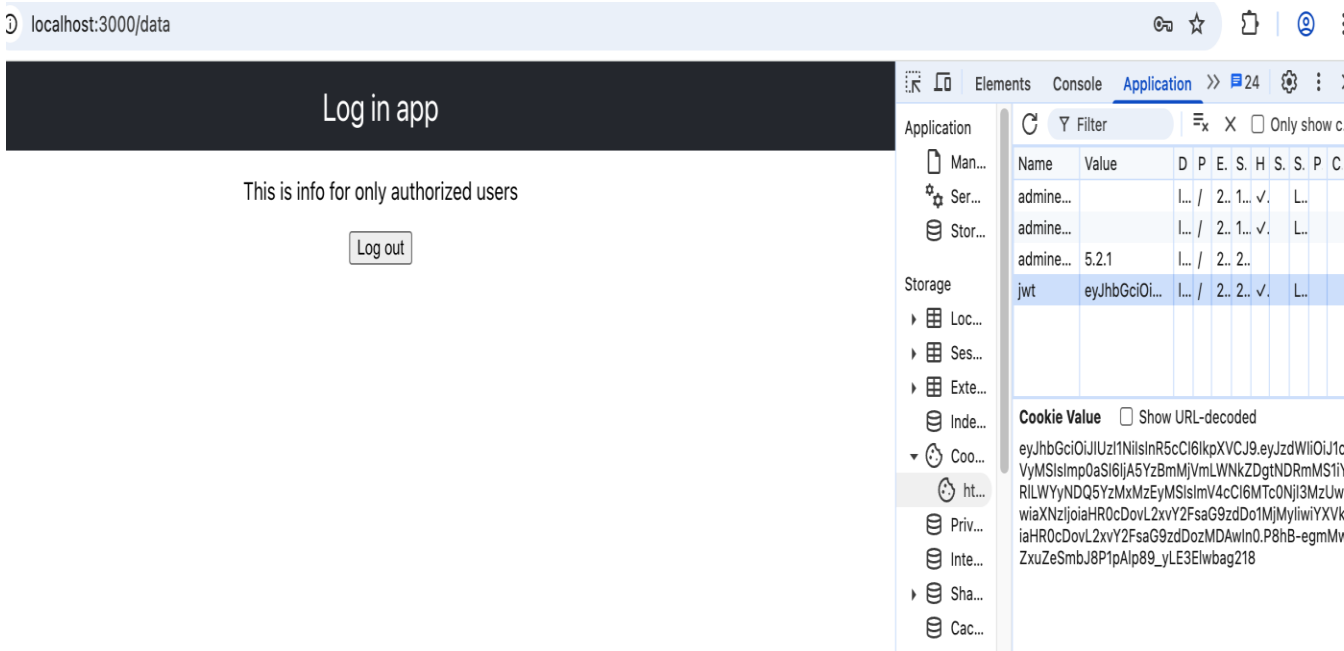
Application for demonstrating a simple login using password hashing and jwt-tokens

Aim of the program

- The aim of the program is to demonstrate a simple login functionality.
- The login asks for username and password and if they are correct, the backend provides a jwt-token for the frontend to access authorized content.







Technologies

Backend

Dotnet 8.0 + ASP.NET Core - library

Frontend

React

CI/CD pipeline

GitHub Actions

SonarQube

OWASP dependency check

OWASP Zap scan

CycloneDX

Trivy

Docker

Secure programming solutions

1. Implementing a CORS-policy

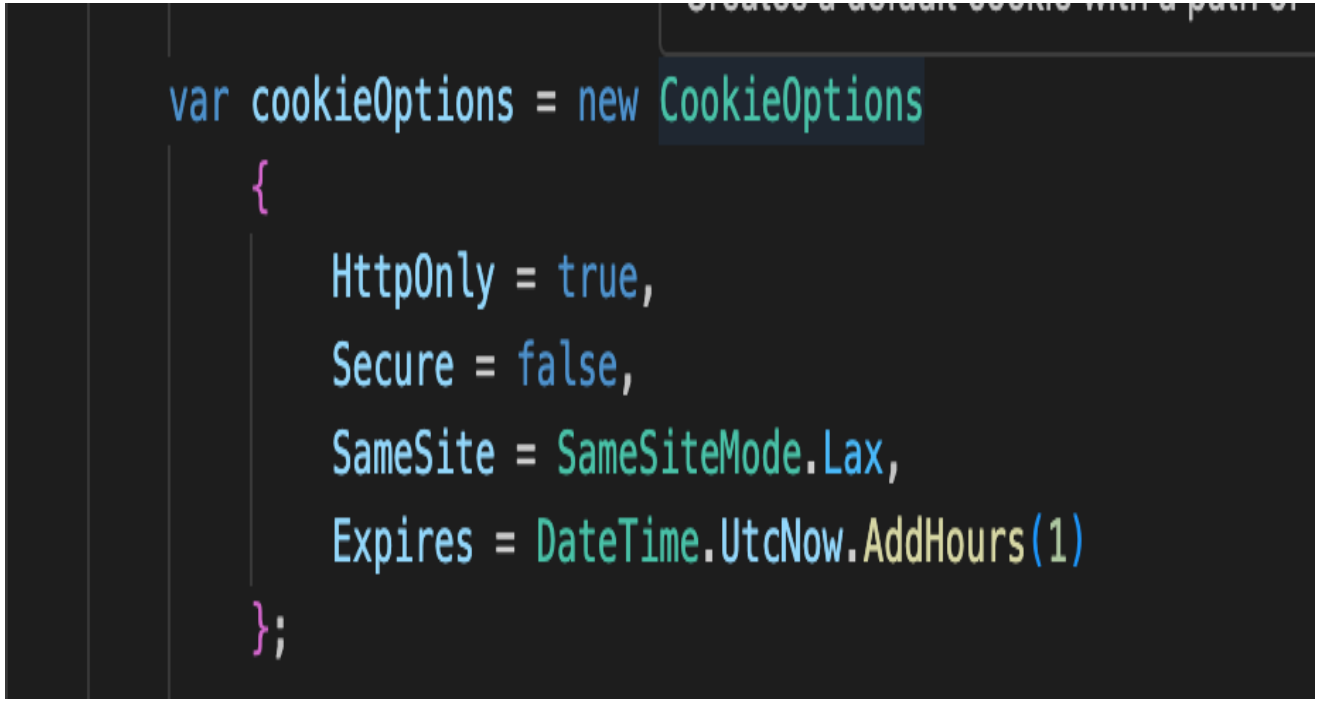
Implementing a CORS-policy prevents requests from any other endpoint than the specifies URL for the frontend

```
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowSpecificOrigin",
        policy =>
        {
            policy.WithOrigins("http://localhost:3000")
                  .AllowAnyHeader()
                  .AllowAnyMethod()
                  .AllowCredentials();
        });
});
```

Secure programming solutions

2. HTTP-only JWT-tokens

The JWT-token is HTTP only which means that it can be set and modified only by the backend and the user cannot tamper the token



Secure programming solutions

3. Authorizing endpoints

The data controller has a authorized annotation which means that the controller is only accessible by authorized users.

```
[HttpGet("data")]
[Authorize]
0 references
public IActionResult GetData (){
    var responseData = new { message = "
    return Ok(responseData);
}
```

Secure programming solutions

4. Data encryption

Any vulnerable data is encrypted before storing, which in this case means encrypting the password.

The password are stored with strong and adaptive hashing functions (PBKDF2)

```
1 reference
public async Task<byte[]> EncryptPassword (string password, byte[] salt){
    using Aes aes = Aes.Create();

    aes.Key = CreateKey(password, salt);
    aes.IV = CreateSalt(16);

    using MemoryStream output = new();
    using (CryptoStream cryptoStream = new(output, aes.CreateEncryptor(), CryptoStreamMode.Wr

    await cryptoStream.WriteAsync(salt);
    await cryptoStream.WriteAsync(aes.IV);

    byte[] passwordBytes = Encoding.Unicode.GetBytes(password);
    await cryptoStream.WriteAsync(passwordBytes, 0, passwordBytes.Length);
    await cryptoStream.FlushFinalBlockAsync();}

    return output.ToArray();
}
```

```
return Rfc2898DeriveBytes.Pbkdf2(Encoding.Unicode.GetBytes(password),
                                 salt,
                                 iterations,
                                 hashMethod,
                                 desiredKeyLength);
}
```

Secure programming solutions

5. Enforcing strong passwords

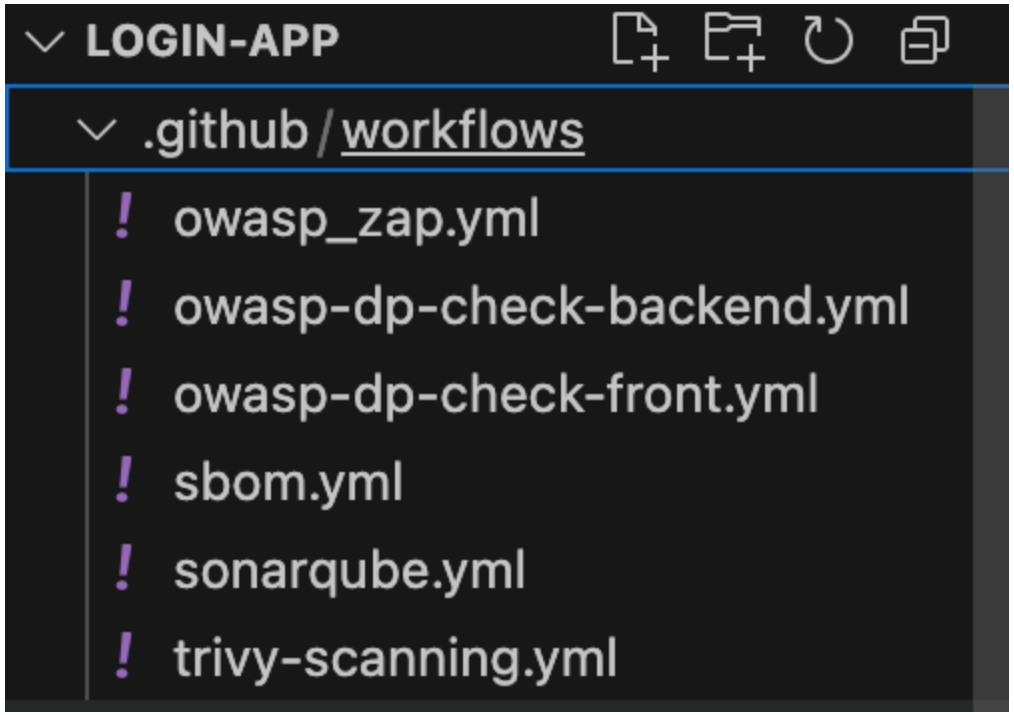
The user is enforced to create a strong password before storing the user credentials. This prevents from idetification failures.

```
Regex regex = new Regex(@"^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[
Match match = regex.Match(password);



if(!match.Success){
    return BadRequest(new {message = "Invalid password format. Passwor
        + " and one of special characters (#?!@$%^&*-)"});
}
```

Secure programming solutions

6. Implementing a DevSecOps-pipeline

The project has a DevSecOps-pipeline that reports any vulnerable packages, dependencies or other vulnerabilities in the application.



Security Testing

- Security testing was implemented by a DevSecOps-pipeline with the following tests and results:

SonarQube – Static application security testing	OWASP Dependency check	OWAS Zap – Dynamic application security testing	Trivy
• Storing the JWT secret keys in the config-file • Passing a timeout to the regex-check • Recursive copying in the docker file • Using the root as a default user in docker • Cookies with secure attribute as false	• After cleaning and updating dependencies two libraries were remained with known vulnerabilities	• Missing Content-Security-Policy header • Cross-Domain Configuration (implemented in backend) • Missing Anti-Clickjacking header	• Same two vulnerabilities as in the dependency check