

Api İle React Ödev Raporu

Tamamen servis yapısıyla yazdığım api ye react arayüzümü bağlayarak hem verileri api üzerinden veritabanından tamamen backendden aldım, hem de token üreterek authorize işlemlerini gerçekleştirdim. Json Web Token teknolojisini önce server tarafında nasıl hazırladığımı ve daha sonra client tarafından nasıl post isteğiyle istekte bulunduğumu bazı ekran görüntüleriyle göstereceğim

AuthService Classı:

```
using ProductManagementWebApi.Models.Interfaces;

namespace ProductManagementWebApi.Models.Services
{
    2 references
    public class AuthService : IAuthService
    {
        readonly ITokenService tokenService;
        DataContext _dataContext;
        0 references
        public AuthService(ITokenService _tokenService, DataContext dataContext)
        {
            this.tokenService = _tokenService;
            _dataContext = dataContext;
        }
        2 references
        public async Task<UserLoginResponse> LoginUserAsync(UserLoginRequest request)
        {
            UserLoginResponse response = new();

            if(String.IsNullOrEmpty(request.Username) || String.IsNullOrEmpty(request.Password))
            {
                throw new ArgumentNullException(nameof(request));
            }

            var user = _dataContext.User.FirstOrDefault(x=>x.Username == request.Username && x.Password ==request.Password);
            if (user!=null)
            {
                var generateTokenInformation = await tokenService.GenerateTokenAsync(new GenerateTokenRequest { Username = request.Username });
                response.AccessTokenExpireDate = generateTokenInformation.TokenExpireDate;
                response.AuthenticateResult = true;
                response.AuthToken = generateTokenInformation.Token;
            }

            return response;
        }
    }
}
```

Şimdi bu classı AuthControllerda nasıl kullandığımıza bakalım.

```

1  using Microsoft.AspNetCore.Http;
2  using Microsoft.AspNetCore.Mvc;
3  using ProductManagementWebApi.Models;
4  using ProductManagementWebApi.Models.Interfaces;
5
6  namespace ProductManagementWebApi.Controllers
7  {
8      [Route("api/[controller]")]
9      [ApiController]
10     public class AuthController : ControllerBase
11     {
12         readonly IAuthService authService;
13
14         public AuthController(IAuthService authService)
15         {
16             this.authService = authService;
17         }
18
19         [HttpPost("LoginUser")]
20         public async Task<ActionResult<UserLoginResponse>> LoginUserAsync([FromBody] UserLoginRequest request)
21         {
22             var result= await authService.LoginUserAsync(request);
23
24             return result;
25         }
26     }
27 }
28

```

Son olarak backend de token Service şu şekilde:

```
using Microsoft.IdentityModel.Tokens;
using ProductManagementWebApi.Models.Interfaces;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace ProductManagementWebApi.Models.Services
{
    2 references
    public class TokenService : ITokenService
    {
        0 references
        readonly IConfiguration configuration;

        0 references
        public TokenService(IConfiguration _configuration)
        {
            this.configuration = _configuration;
        }

        2 references
        public Task<GenerateTokenResponse> GenerateTokenAsync(GenerateTokenRequest request)
        {
            SymmetricSecurityKey symmetricSecurityKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(configuration["AppSettings:Secret"]));

            var dateTime = DateTime.Now;

            JwtSecurityToken jwt = new JwtSecurityToken(
                issuer: configuration["AppSettings:ValidIssuer"],
                audience: configuration["AppSettings:ValidAudience"],
                claims: new List<Claim>
                {
                    new Claim("userName", request.Username)
                },
                notBefore: dateTime,
                expires: dateTime.Add(TimeSpan.FromMinutes(60)),
                signingCredentials: new SigningCredentials(symmetricSecurityKey, SecurityAlgorithms.HmacSha256)
            );

            return Task.FromResult(new GenerateTokenResponse
            {
                Token = new JwtSecurityTokenHandler().WriteToken(jwt),
                TokenExpireDate = dateTime.Add(TimeSpan.FromMinutes(60)),
            });
        }
    }
}
```

Server tarafındaki temel kodlarımız bunlardı şimdi Clienttan nasıl istek attığımıza bakalım:

```
src > actions > JS authActions.js > ...
1  export const loginStart = () => ({
2    type: 'LOGIN_START'
3  });
4
5  export const loginSuccess = (user) => ({
6    type: 'LOGIN_SUCCESS',
7    payload: user
8  });
9
10 export const loginFailure = (error) => ({
11   type: 'LOGIN_FAILURE',
12   payload: error
13 });
14
15 export const loginUser = (credentials) => {
16   return (dispatch) => {
17     dispatch(loginStart());
18     fetch('https://localhost:7237/api/auth/loginuser', {
19       method: 'POST',
20       headers: {
21         'Content-Type': 'application/json'
22       },
23       body: JSON.stringify(credentials)
24     })
25       .then(res => res.json())
26       .then(data => {
27         // Handle your response here. Assuming data contains the user
28         dispatch(loginSuccess(data));
29       })
30       .catch(error => {
31         dispatch(loginFailure(error.message));
32       });
33   };
34 }
```

Devamında burda tanımladığımız loginUser fonksiyonun loginde kullanımı :

```
import React, { useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { loginUser } from '../actions/authActions';

const Login = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const dispatch = useDispatch();
  const { isFetching, error } = useSelector(state => state.auth);

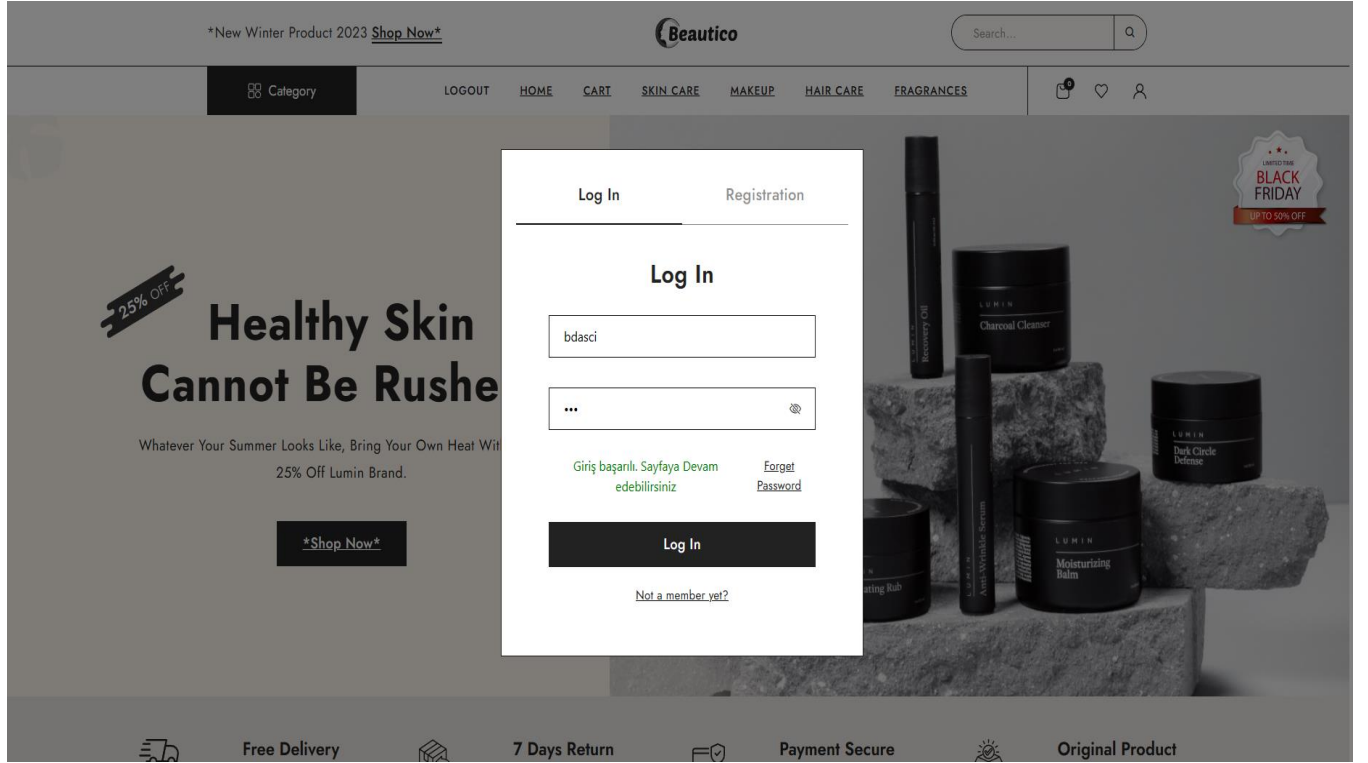
  const handleSubmit = (e) => {
    e.preventDefault();
    dispatch(loginUser({ username, password }));
  };

  return (
    <div>
      <form onSubmit={handleSubmit}>
        <input
          type="text"
          placeholder="Username"
          value={username}
          onChange={(e) => setUsername(e.target.value)}
        />
        <input
          type="password"
          placeholder="Password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />
        <button type="submit" disabled={isFetching}>Login</button>
        {error && <p>{error}</p>}
      </form>
    </div>
  );
};
```

Ve kod tarafında son olarak tokeni local storage(session veya cookie de olabilirdi.) a atıp Bearer ile yaptığımız kontrol bölümü:

```
const submitCart = async () => {
  if (localStorage.getItem("token") != null) {
    const response = await axios.post('https://localhost:7237/api/cart/fillcart', {
      username: localStorage.getItem('username'),
      totalprice: total
    },
    {
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${localStorage.getItem("token")}`
      }
    })
  }
  else {
    alert("Lütfen Giriş Yapınız.")
  }
  dispatch(clearCart())
}
```

Sağ üstte , kalp simgesinin yanında bulunan login simgesine basıp giriş yaparız. Çıkış için iste navbardaki logout linki kullanılabilir.



Sepeti doldurmak için add to Cart butonları kullanılır.

Best Selling Product

[*View All Product](#) →

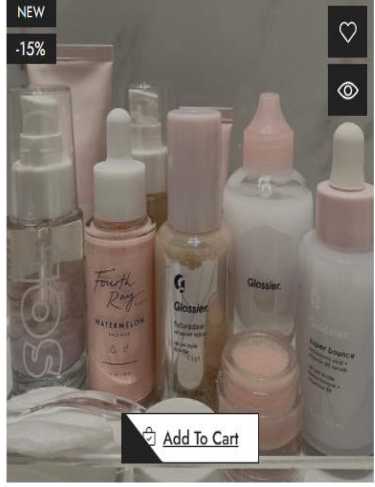
Foundation
REVLON
\$150.00 ~~\$200.00~~
★★★★★ (50)

NEW
-15%



Lipstick
REVLON
\$150.00 ~~\$200.00~~
★★★★★ (50)

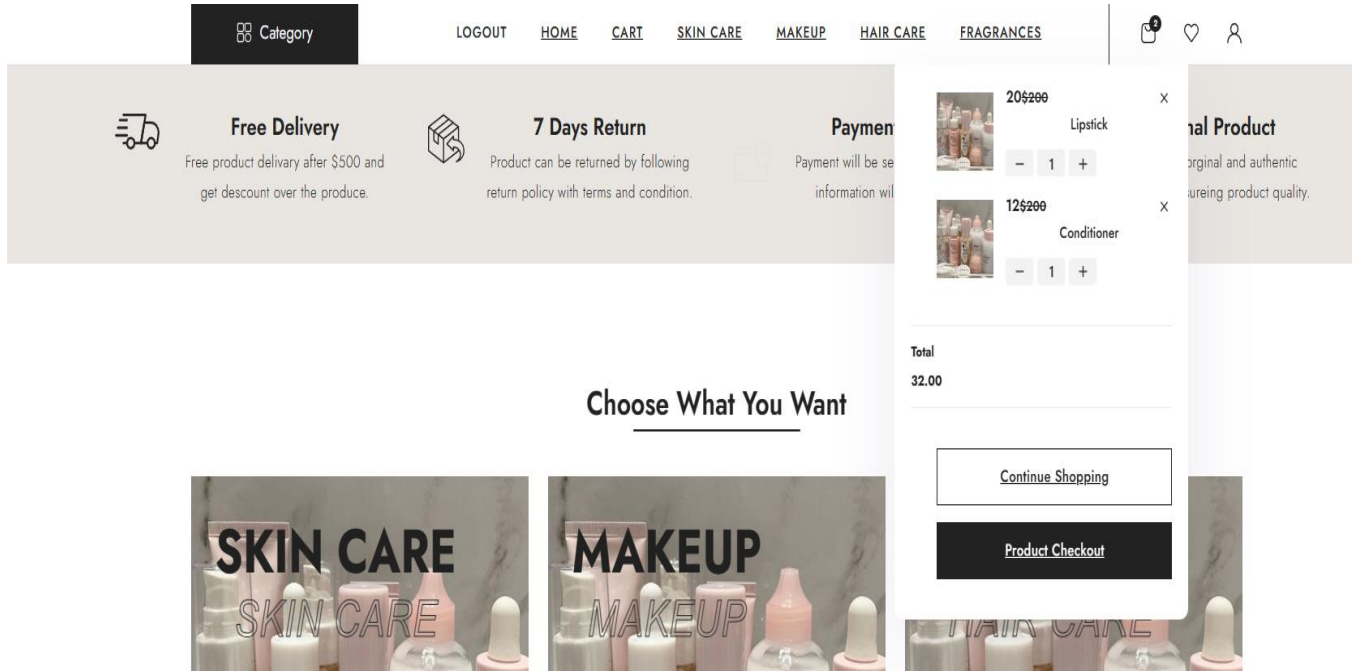
NEW
-15%



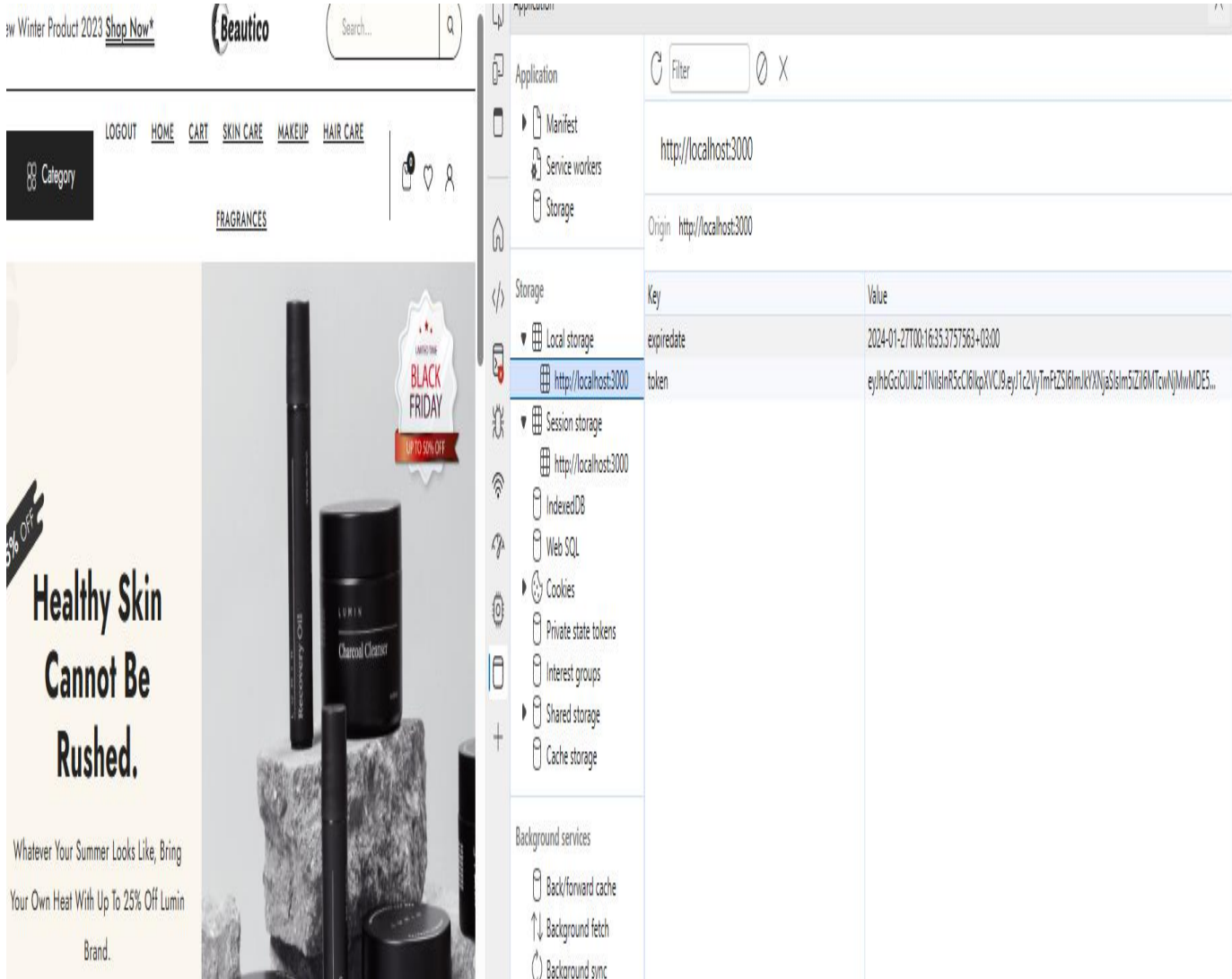
Moisturizing Cream
REVLON
\$150.00 ~~\$200.00~~
★★★★★ (50)

Add To Cart

Sepet simgesine tıkladıktan sonra açılan sepette Product Checkouta tıklayıp sipariş tamamlama ekranına geçilir.




Tokeni alıp alamadığımızı local storage üzerinden kontrol edelim:



Tokeni aldığımız için yukarıdaki submitCart fonksiyonunun koşulu sağlandı ve artık detayları db ye kaydebilir.

Order Summary




Lipstick

- 1 +

× 20

×



Conditioner

- 1 +

× 12





×

Subtotal	\$32.00
Tax	\$1.60
Total (tax excl.)	\$32.00
Total (tax incl.)	\$33.60

Total	\$33.60
--------------	----------------

☐**Check payments**
Please send a check to Store Name, Store Street, Store State / Country,
Store Postcode.

☐**Cash on delivery**
Pay with cash upon delivery.

☐ **Paypal**     [What is PayPal?](#)

☐**Card**

☐ I have read and agree to the website [Terms and conditions](#)

Place Order

Yukarıdaki görselde "Place Order" butonuna tıkladığında aktif kullanıcı ve totalPrice gibi değerlerle beraber veri tabanında Cart tablosuna order kaydı yapılır. Token yoksa veya expire olduysa hata alerti vererek işlemi iptal eder.