CS6004NT
Application Development

WEEK - 08

Working with DB Transaction

```
await using var transaction = await _dbContext.Database.BeginTransactionAsync();

try
{
    // Perform database operations here...

    await _dbContext.SaveChangesAsync();
    await transaction.CommitAsync();
}
catch (Exception ex)
{
    await transaction.RollbackAsync();
    throw;
}
```

Working with Cookies

```
Response.Cookies.Append("mycookie", "myvalue", new CookieOptions
{
    HttpOnly = true, // In the browser, document.cookie (JS) won't be able to read the cookie
    SameSite = SameSiteMode.None, // Cookie is sent on cross-site requests (api.example.com, app.example.com)
    Secure = true, // Cookie is sent only over HTTPS, it is required when SameSite is None
    Path = "/my/path", // Cookie is sent only on specific path prefix
    Path = "/", // Cookie is sent on all paths
    Domain = "api.example.com", // Cookie is for specific domain
    Domain = ".example.com", // Cookie is for cross subdomains
});
// Reading a cookie
var myCookie = Request.Cookies["mycookie"];
// Removing a cookie from browser
if (Request.Cookies["mycookie"] != null)
{
    var myCookie = Request.Cookies["mycookie"];
    myCookie.Expires = DateTime.Now.AddDays(-1); // Set the Expiry date to past date
    Response.Cookies.Add(myCookie); // Update the Cookie in Browser
}
```

Profile Picture

1. **Blazor** Profile Page

```
<InputFile max-file-size="1536000" accept=".png,.jpg,.jpeg" OnChange="@OnInputFileChange" single />
```

```
private async Task OnInputFileChange(InputFileChangeEventArgs e){
    try{
        var file = e.File;
        var filename = file.Name;
        var fileContent = new StreamContent(file.OpenReadStream(maxAllowedSize: 1024 * 1000));
        fileContent.Headers.ContentDisposition = new("form-data") { Name = "file", FileName = filename };
        fileContent.Headers.ContentType = new MediaTypeHeaderValue(file.ContentType);

        using var content = new MultipartFormDataContent();
        content.Add(content: fileContent, name: "image");

        var response = await AssetService.UploadAssetAsync(content);
        ProfilePicturePreview = response?.Url ?? string.Empty;
        _userRequest.ProfilePicture = filename;
    }catch (Exception ex){
        ErrorMessage = ex.Message;
    }
}
```

Profile Picture

2. **Blazor** Service Method

```
public async Task<AssetUploadResponse?> UploadAssetAsync(MultipartFormDataContent content){
    var response = await _httpClient.AuthPostAsync("/api/assets", content);

    await CheckForErrorResponse(response);

    var result = await response.Content.ReadFromJsonAsync<AssetUploadResponse>();
    return result;
}
```

2. **API** Controller POST Action Method

```
[Authorize]
[HttpPost("/api/assets")]
public async Task<ActionResult<AssetUploadResponse>> UploadFile(IFormFile? file){
    if (file == null || file.Length == 0)
        return BadRequest("No file selected");

    var url = await _assetService.UploadAsync(file);

    return Ok(new AssetUploadResponse { Url = url });
}
```

Profile Picture

4. **API** Service Method

```
public async Task<string> UploadAsync(IFormFile file){
    var fileName = file.FileName;

    if (!IsAllowedFileType(Path.GetExtension(fileName)))
        throw new DomainException("Invalid file type");

    if (file.Length > 1 * 1024 * 1024) // 1MB
        throw new DomainException("File size exceeds the limit");

    return await _fileProvider.UploadFileAsync(file, fileName);
}
```

4. **API** Provider Method

```
public async Task<string> UploadFileAsync(IFormFile file, string fileName){
    var filePath = Path.Combine(UploadDirectory, fileName);
    await using var fileStream = new FileStream(filePath, FileMode.Create, FileAccess.Write);
    await file.CopyToAsync(fileStream);

    return $"https://localhost:5001/api/assets/{fileName}";
}
```

Profile Picture

6. **API** Controller GET Action Method

```
[HttpGet("/api/assets/{fileName}")]
public IActionResult GetAsset(string fileName){
    var (file, contentType) = _assetService.GetFileAsStream(fileName);
    return File(file, contentType);
}
```

6. **API** Provider Method

```
public (FileStream file, string contentType) GetFileAsStream(string fileName){
    var filePath = Path.Combine(UploadDirectory, fileName);
    if (!File.Exists(filePath))
        throw new DomainException("File not found", 404);

    var file = File.OpenRead(filePath);
    var contentTypeProvider = new FileExtensionContentTypeProvider();
    var contentType = contentTypeProvider.TryGetContentType(filePath, out var type) ? type :
"application/octet-stream";
    return (file, contentType);
}
```

Forgot Password

1. Prerequisites
    - A. Log-in into Gmail with your account
    - B. Navigate to https://security.google.com/settings/security/apppasswords
    - C. In select **Mail** and **Other (custom name)**, give it a name and press **Generate**
    - D. It will give your password
    - E. Update appsetting.json in API project

```
...// other settings
"App": {
    "ApiBaseUrl": "https://localhost:5001/api",
    "WebAppBaseUrl": "https://localhost:3001"
  },
  "GmailCredentials": {
    "UserName": "himalay.sunuwar@islingtoncollege.edu.np",
    "Password": "passwordFromPreviousSteps"
}
```

Forgot Password

2. **API** Email Provider

```
...// using statements

namespace BookReview.Api.Infrastructure.Email;
public class GmailEmailProvider : IGmailEmailProvider {
    private readonly string _from;
    private readonly SmtpClient _client;

    public GmailEmailProvider(IConfiguration configuration) {
        var userName = configuration.GetSection("GmailCredentials:UserName").Value!;
        var password = configuration.GetSection("GmailCredentials:Password").Value!;

        _from = userName;
        _client = new SmtpClient("smtp.gmail.com", 587) {
            Credentials = new NetworkCredential(userName, password),
            UseDefaultCredentials = false,
            EnableSsl = true
        };
    }
```

Forgot Password

```
    public async Task SendEmailAsync(EmailMessage message) {
        var mailMessage = new MailMessage(_from, message.To, message.Subject, message.Body);
        foreach (var attachment in message.AttachmentPaths.Select(a => new Attachment(a))) {
            mailMessage.Attachments.Add(attachment);
        }
        await _client.SendMailAsync(mailMessage);
    }
}
```

3. **API** Email Service Method

```
public async Task SendForgotPasswordEmailAsync(string name, string toEmail, string passwordResetToken) {
        var passwordRestUrl = $"{_webAppBaseUrl}/reset-password?token={passwordResetToken}";
        var message = new EmailMessage {
        Subject = "Password Reset Request",
        To = toEmail,
        Body = @$"Dear {name},
To reset your password, please click on the following link:
{passwordRestUrl}"
        };

        await _emailProvider.SendEmailAsync(message);
}
```

Forgot Password

4. **API** Auth Service Methods

```
public async Task ForgotPassword(string email) {
        var user = await _userManager.FindByEmailAsync(email);
        if (user != null) {
        var passwordResetToken = await _userManager.GeneratePasswordResetTokenAsync(user);
        var token = ToUrlSafeBase64(passwordResetToken);
        await _emailService.SendForgotPasswordEmailAsync(user.Name, email, token);
        }
}

public async Task ResetPassword(string email, string token, string password) {
        var user = await _userManager.FindByEmailAsync(email);
        if (user != null) {
        var passwordResetToken = FromUrlSafeBase64(token);
        var result = await _userManager.ResetPasswordAsync(user, passwordResetToken, password);

        ValidateIdentityResult(result);
        }
}
```

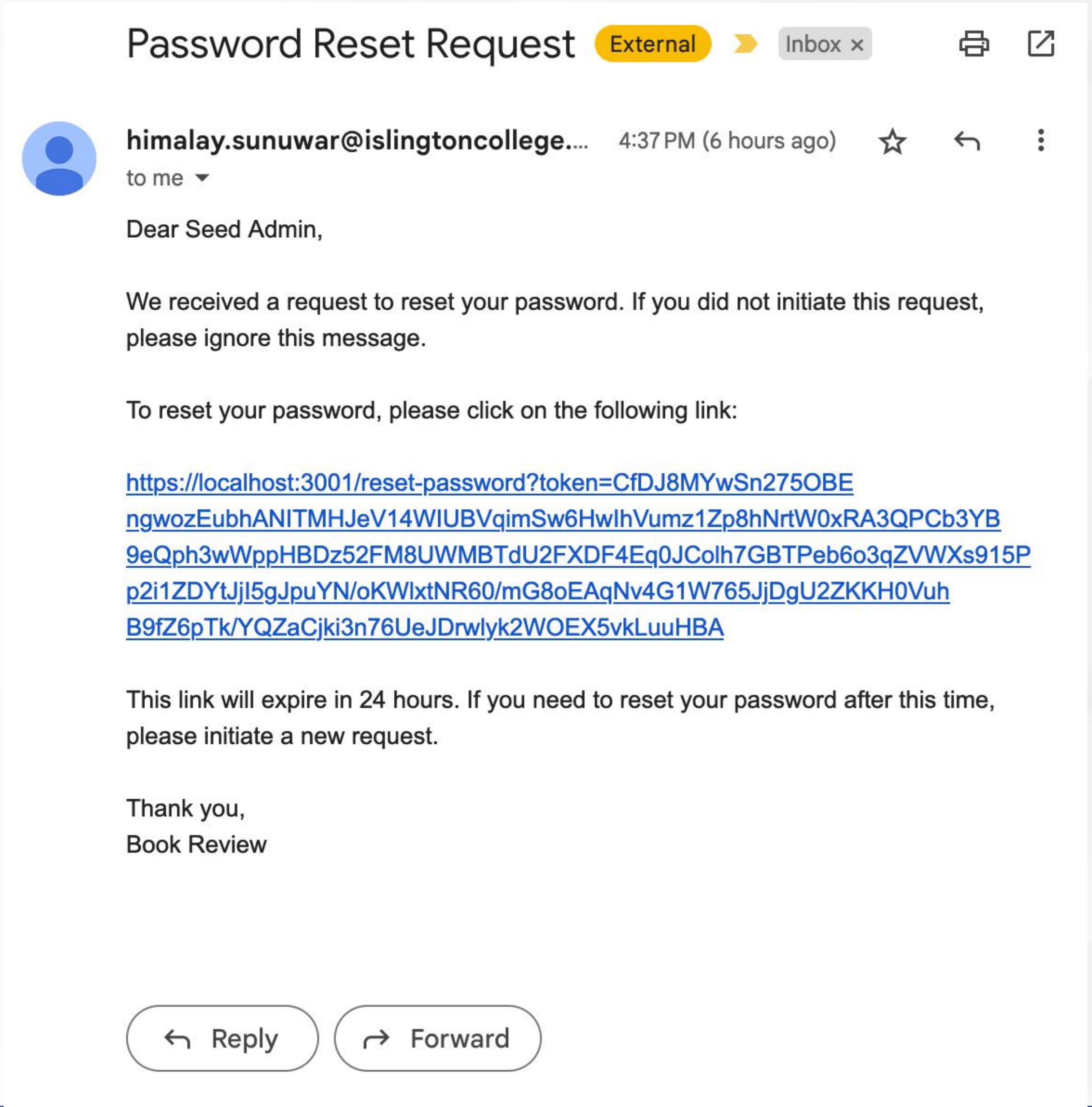
Forgot Password

```
private void ValidateIdentityResult(IdentityResult result) {
      if (result.Succeeded) return;
      var errors = result.Errors.Select(x => x.Description);
      throw new DomainException(string.Join('\n', errors));
}

private static string ToUrlSafeBase64(string base64String) {
      return base64String.Replace('+', '-').Replace('/', '~').Replace('=', '_');
}

private static string FromUrlSafeBase64(string urlSafeBase64String) {
      return urlSafeBase64String.Replace('-', '+').Replace('~', '/').Replace('_', '=');
}
```

Forgot Password



Password Reset Request External Inbox ×

himalay.sunuwar@islingtoncollege.... 4:37 PM (6 hours ago)

to me

Dear Seed Admin,

We received a request to reset your password. If you did not initiate this request, please ignore this message.

To reset your password, please click on the following link:

https://localhost:3001/reset-password?token=CfDJ8MYwSn275OBE
ngwozEubhANITMHJeV14WIUBVqimSw6HwIhVumz1Zp8hNrtW0xRA3QPCb3YB
9eQph3wWppHBDz52FM8UWMBTdU2FXDF4Eq0JColh7GBTPeb6o3qZVWXs915P
p2i1ZDYtJjI5gJpuYN/oKWlxtNR60/mG8oEAqNv4G1W765JjDgU2ZKKH0Vuh
B9fZ6pTk/YQZaCjki3n76UeJDrwlyk2WOEX5vkLuuHBA

This link will expire in 24 hours. If you need to reset your password after this time, please initiate a new request.

Thank you,
Book Review

User Registration

1. **API** Auth Service Methods

```
public async Task Register(string name, string email, string password) {
        var newUser = new AppUser { Name = name, UserName = email, Email = email };
        var result = await _userManager.CreateAsync(newUser, password);
        ValidateIdentityResult(result);

        await _userManager.AddToRoleAsync(newUser, "User");
        var emailConfirmationToken = await _userManager.GenerateEmailConfirmationTokenAsync(newUser);
        var token = ToUrlSafeBase64(emailConfirmationToken);
        await _emailService.SendEmailConfirmationEmailAsync(name, newUser.Id, email, token);
}

public async Task ConfirmEmail(string token, string userId) {
        var user = await _userManager.FindByIdAsync(userId);
        var emailConfirmationToken = FromUrlSafeBase64(token);
        var result = await _userManager.ConfirmEmailAsync(user, emailConfirmationToken);
        ValidateIdentityResult(result);
}
```

JWT Token Authentication

1. **API** Prerequisites

   A. Install the necessary NuGet packages:

```
dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer --version 6.0.15
dotnet add package Microsoft.IdentityModel.Tokens --version 6.27.0
```

   A. Add JWT settings to appsettings.json

```
"Jwt": {
    "Key": "tZXiWWJeqXSwezvUFTDSwMkB$4xyPRpk$zeP^ytBU%FqUi&hVG@nDzMExTDDik%c",
    "Issuer": "book-review",
    "Audience": "book-review-app"
  }
```

   A. Configure JWT authentication in the Program.cs

```
var key = Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]);
builder.Services.AddAuthentication(auth => {
    auth.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    auth.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
```

JWT Token Authentication

```
.AddJwtBearer(options => {
    options.RequireHttpsMetadata = false;
    options.SaveToken = true;
    options.TokenValidationParameters = new TokenValidationParameters {
        ValidIssuer = builder.Configuration["Jwt:Issuer"],
        ValidAudience = builder.Configuration["Jwt:Audience"],
        IssuerSigningKey = new SymmetricSecurityKey(key),
        ValidateIssuerSigningKey = true,
        ValidateLifetime = false
    };
});
builder.Services.AddAuthorization();
```

JWT Token Authentication

2. **API** Token Service

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

namespace BookReview.Api.Infrastructure.Identity;
public class TokenService : ITokenService {
    private readonly string _key;
    private readonly string _issuer;
    private readonly string _audience;

    public TokenService(IConfiguration configuration) {
        _key = configuration.GetSection("JWT:Key").Value!;
        _issuer = configuration.GetSection("JWT:Issuer").Value!;
        _audience = configuration.GetSection("JWT:Audience").Value!;
    }
```

JWT Token Authentication

```csharp
    public string GenerateToken(AppUser user, string role) {
        var tokenHandler = new JwtSecurityTokenHandler();
        var key = Encoding.ASCII.GetBytes(_key);
        var tokenDescriptor = new SecurityTokenDescriptor {
            Subject = new ClaimsIdentity(new[] {
                new Claim(ClaimTypes.NameIdentifier, user.Id),
                new Claim(ClaimTypes.Email, user.Email),
                new Claim(ClaimTypes.Role, role)
            }),
            Expires = DateTime.UtcNow.AddHours(12),
            Issuer = _issuer,
            Audience = _audience,
            SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
        };
        var securityToken = tokenHandler.CreateToken(tokenDescriptor);
        return tokenHandler.WriteToken(securityToken);
    }
}
```

JWT Token Authentication

3. **API** Auth Service Method

```
public async Task<string> TokenLoginAsync(string email, string password) {
    var user = await _userManager.FindByEmailAsync(email);
    if (user == null)
        throw new DomainException("Invalid email or password", 401);

    var result = await _signInManager.CheckPasswordSignInAsync(user, password, false);
    if (!result.Succeeded)
        throw new DomainException("Invalid email or password", 401);

    var roles = await _userManager.GetRolesAsync(user);
    var role = roles.FirstOrDefault();
    return _tokenService.GenerateToken(user, role!);
}
```

JWT Token Authentication

4. **API** Auth Controller Action Method

```
[HttpPost("/api/auth/login")]
public async Task<IActionResult> LoginAsync([FromBody] LoginRequest login) {
    var token = await _authService.TokenLoginAsync(login.Username, login.Password);

    return Ok(new TokenLoginResponse {
            Token = token
        });
}
```

5. **Blazor** Auth Service Methods

```
public async Task LoginAsync(LoginRequest request) {
    var response = await _httpClient.AuthPostAsJsonAsync("/api/auth/login", request);

    await CheckForErrorResponse(response);

    var result = await response.Content.ReadFromJsonAsync<TokenLoginResponse>();
    await _jsRuntime.InvokeAsync<string>("localStorage.setItem", "token", result?.Token);
}
```

JWT Token Authentication

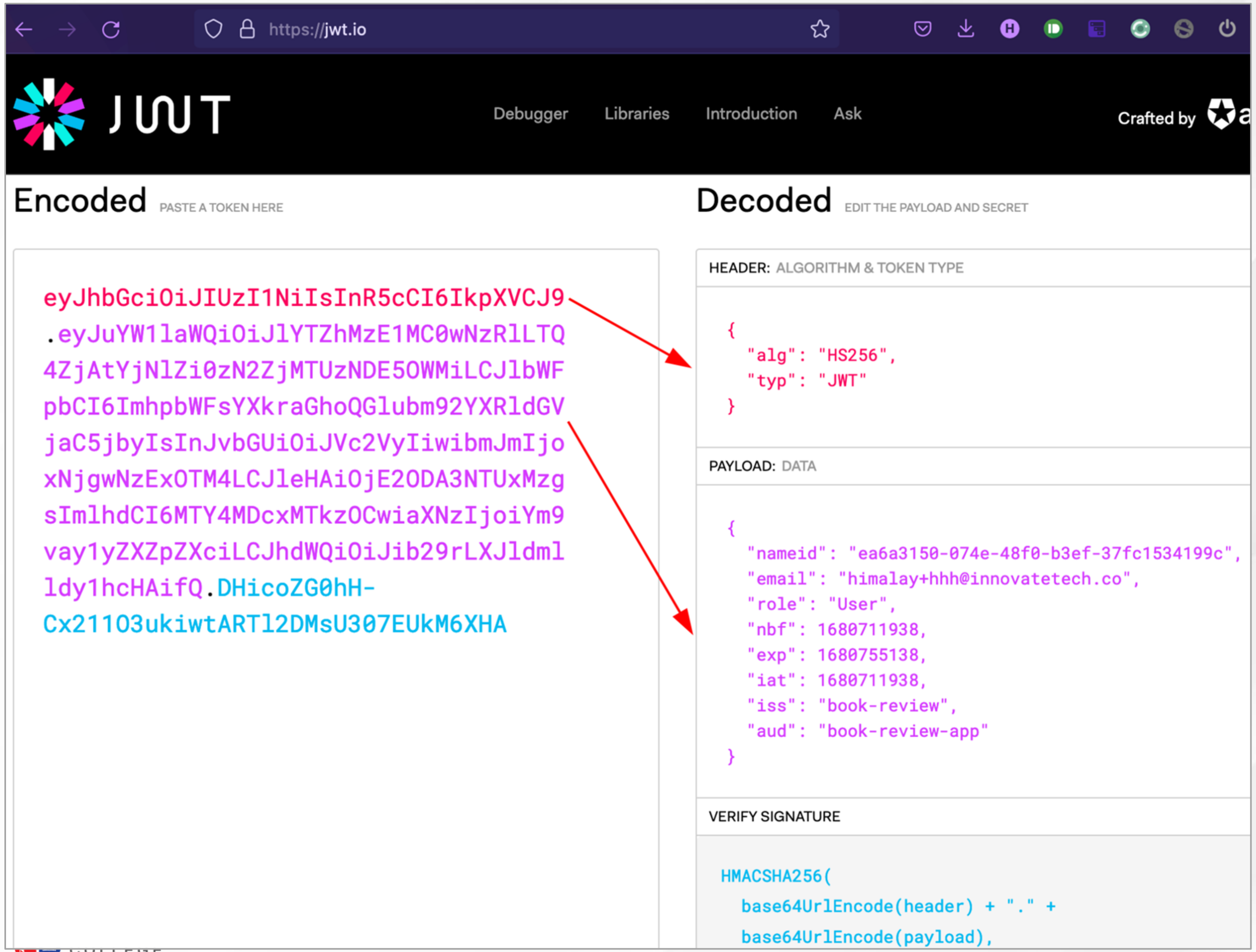
```
public async Task<UserResponse?> GetProfileAsync() {
    var token = await _jsRuntime.InvokeAsync<string>("localStorage.getItem", "token");
    _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);
    var response = await _httpClient.AuthGetAsync("/api/auth/profile");

    await CheckForErrorResponse(response);

    var result = await response.Content.ReadFromJsonAsync<UserResponse>();
    return result;
}

public async Task LogoutAsync() {
    var token = await _jsRuntime.InvokeAsync<string>("localStorage.getItem", TokenKey);
    _httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);
    await _jsRuntime.InvokeVoidAsync("localStorage.removeItem", "token");
    var response = await _httpClient.AuthPostAsync("/api/auth/logout", null);
    await CheckForErrorResponse(response);
}
```

JWT Token



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
.eyJuYW1laWQiOiJlYTZhMzE1MC0wNzRlLTQ
4ZjAtYjNlZi0zN2ZjMTUzNDE5OWMiLCJlbWF
pbCI6ImhpbWFsYXkraGhoQGlubm92YXRldGV
jaC5jbyIsInJvbGUiOiJVc2VyIiwibmJmIjo
xNjgwNzExOTM4LCJleHAiOjE2ODA3NTUxMzg
sImlhdCI6MTY4MDcxMTkzOCwiaXNzIjoiYm9
vay1yZXZpZXciLCJhdWQiOiJib29rLXJldml
ldy1hcHAifQ.DHicoZG0hH-
Cx21l03ukiwtARTl2DMsU307EUkM6XHA
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "nameid": "ea6a3150-074e-48f0-b3ef-37fc1534199c",
  "email": "himalay+hhh@innovatetech.co",
  "role": "User",
  "nbf": 1680711938,
  "exp": 1680755138,
  "iat": 1680711938,
  "iss": "book-review",
  "aud": "book-review-app"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
```

Thank You