```
 1  using DatabaseConnect;
 2  using Microsoft.AspNetCore.Authorization;
 3  using Microsoft.AspNetCore.Mvc;
 4  using Microsoft.EntityFrameworkCore;
 5  using Microsoft.Extensions.Configuration;
 6  using Microsoft.Extensions.Logging;
 7  using Microsoft.IdentityModel.Tokens;
 8  using System;
 9  using System.Collections.Generic;
10  using System.Linq;
11  using System.Security.Claims;
12  using System.Threading.Tasks;
13  using static LibraryAppMVC.Models.Models;
14  using DatabaseConnect.Entities;
15  using System.IdentityModel.Tokens.Jwt;
16  using System.Text;
17  using Microsoft.AspNetCore.Cryptography.KeyDerivation;
18
19  namespace LibraryAppMVC.Controllers
20  {
21      [Route("/user/")] // All endpoints checked 2/25/18, logout not working but
          not important (token dumped client side at logout)
22      public class UserController : Controller
23      {
24          private IConfiguration _config;
25          private Context _ctx;
26
27          public UserController(IConfiguration config, Context context)
28          {
29              _config = config;
30              _ctx = context;
31          }
32
33
34          [Route("login")]
35          [AllowAnonymous]
36          [HttpPost]
37          public IActionResult CreateToken([FromBody]LoginModel login) // Checked
              2/24/18 working
38          {
39              IActionResult response = Unauthorized();
40              var user = Authenticate(login);
41
42              if (user != null)
43              {
44                  response = BuildToken(user);
45              }
46              return response;
47          }
48
49          [Route("logout")]
50          [Authorize]
```

```
51          [HttpPost]
52          public IActionResult Logout() // Checked 2/24/18 NOT working TODO, maybe
              not important because client dumps token on logout
53          {
54              string schoolID = User.Claims.FirstOrDefault(c => c.Type ==
                  ClaimTypes.NameIdentifier).Value;
55              int userID = _ctx.Users
56                  .Single(u => u.SchoolID == schoolID)
57                  .UserID;
58              _ctx.Users
59                  .Single(u => u.UserID == userID);
60              _ctx.SaveChanges();
61              return Ok();
62          }
63
64          [Route("info")]
65          [Authorize]
66          [HttpGet]
67          public IActionResult UserInfo()
68          {
69              string schoolID = User.Claims.FirstOrDefault(c => c.Type ==
                  ClaimTypes.NameIdentifier).Value;
70              var user = _ctx.Users
71                  .Single(u => u.SchoolID == schoolID);
72              int userID = user.UserID;
73
74              var checkouts = _ctx.Checkouts
75                  .Where(c => c.Active)
76                  .Where(c => c.UserID == userID)
77                  .Include(c => c.Book)
78                  .ToList();
79
80              var reservations = _ctx.Reservations
81                  .Where(r => r.Active)
82                  .Where(r => r.UserID == userID)
83                  .Include(r => r.Book)
84                  .ToList();
85
86              foreach (Checkout c in checkouts)
87              {
88                  c.User = null;
89              }
90              foreach (Reservation r in reservations)
91              {
92                  r.User = null;
93              }
94              user.PasswordHash = null;
95              user.Salt = null;
96              var resp = new { checkouts, reservations, user };
97              return Json(resp);
98          }
99
```

```
100         private IActionResult BuildToken(UserModel user)
101         {
102             var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config
                  ["Jwt:Key"]));
103             var creds = new SigningCredentials(key,
                  SecurityAlgorithms.HmacSha256);
104             var claims = new[]
105             {
106                 new Claim(JwtRegisteredClaimNames.Sub, user.StudentID),
107                 new Claim(JwtRegisteredClaimNames.Jti, user.TokenVersion.ToString
                      ())
108             };
109
110             var token = new JwtSecurityToken(
111                 _config["Jwt:Issuer"],
112                 _config["Jwt:Issuer"],
113                 expires: DateTime.Now.AddMinutes(Convert.ToDouble(_config
                      ["LoginDurationMinutes"])),
114                 signingCredentials: creds,
115                 claims: claims
116                 );
117             return Ok(
118                 new
119                 {
120                     token = new JwtSecurityTokenHandler().WriteToken(token),
121                     expiration = token.ValidTo
122                 });
123         }
124
125         private UserModel Authenticate(LoginModel login)
126         {
127             User User;
128             UserModel usermodel = null;
129             try
130             {
131                 User = _ctx.Users
132                     .Single(u => u.SchoolID.Equals(login.Username));
133             }
134             catch
135             {
136                 return null;  // No user found with specified school ID
137             }
138             if (VerifyPass(login.Password, User.Salt, User.PasswordHash))
139             {
140                 usermodel = new UserModel { Name = User.FullName, StudentID =
                      User.SchoolID, TokenVersion = User.TokenVersion };
141             }
142             return usermodel;
143         }
144
145         private Boolean VerifyPass(String RawPass, String Salt, String
              PasswordHash)
```

```
146         {
147             byte[] salt_array = Convert.FromBase64String(Salt);
148             String hashed = Convert.ToBase64String(KeyDerivation.Pbkdf2(
149                 password: RawPass,
150                 salt: salt_array,
151                 prf: KeyDerivationPrf.HMACSHA1,
152                 iterationCount: 10000,
153                 numBytesRequested: 256 / 8));
154             return hashed.Equals(PasswordHash);
155         }
156     }
157 }
158
```