Summary

Information

Class:	FuelQuoteApp_p1.Controllers.AccountController
Assembly:	FuelQuoteApp_p1
File(s):	C:\Users\Dell\OneDrive\Desktop\FuelQuoteApp_p1\FuelQuoteApp_p1\Controllers\AccountController.cs

Line coverage

100%

Covered lines:	22
Uncovered lines:	0
Coverable lines:	22
Total lines:	150
Line coverage:	100%

Branch coverage

100%

Covered branches:	10
Total branches:	10
Branch coverage:	100%

Method coverage

Method coverage is only available for sponsors.

Upgrade to PRO version

Methods/Properties

- .ctor(Microsoft.AspNetCore.Identity.User…
- RegisterDataValidation(FuelQuoteApp_p…

Metrics

Method	Branch coverage	Cyclomatic complexity	Line coverage
.ctor(...)	100%	1	100%
RegisterDataValidation(...)	100%	10	100%

File(s)

C:\Users\Dell\OneDrive\Desktop\FuelQuoteApp_p1\FuelQuoteApp_p1\Controllers\AccountController.cs

#	Line	Line coverage
	1	`using FuelQuoteApp_p1.EntModels.Models;`
	2	`using FuelQuoteApp_p1.Models.Account;`
	3	`using Microsoft.AspNetCore.Authorization;`
	4	`using Microsoft.AspNetCore.Http;`
	5	`using Microsoft.AspNetCore.Identity;`
	6	`using Microsoft.AspNetCore.Mvc;`
	7	`using Newtonsoft.Json;`
	8	`using System;`
	9	`using System.Diagnostics.CodeAnalysis;`
	10	`using System.Text.RegularExpressions;`
	11	`using System.Threading.Tasks;`
	12	
	13	`namespace FuelQuoteApp_p1.Controllers`
	14	`{`
	15	`    public class AccountController : Controller`
	16	`    {`
	17	`        private readonly UserManager<IdentityUser> userManager;`
	18	`        private readonly SignInManager<IdentityUser> signInManager;`
	19	
1	20	`        public AccountController(UserManager<IdentityUser> userManager, SignInManager<IdentityUser> signInManager)`
1	21	`        {`
1	22	`            this.userManager = userManager;`
1	23	`            this.signInManager = signInManager;`
1	24	`        }`
	25	
	26	`        [HttpGet]`
	27	`        [ExcludeFromCodeCoverage]`
	28	`        public IActionResult Display()`
	29	`        {`
	30	`            return View();`
	31	`        }`
	32	
	33	`        [HttpGet]`
	34	`        [AllowAnonymous]`
	35	`        [ExcludeFromCodeCoverage]`
	36	`        public IActionResult Login()`
	37	`        {`
	38	`            return View();`
	39	`        }`
	40	
	41	`        [HttpPost]`
	42	`        [AllowAnonymous]`
	43	`        [ExcludeFromCodeCoverage]`
	44	`        public async Task<IActionResult> Login(Login model)`
	45	`        {`
	46	`            if (ModelState.IsValid)`
	47	`            {`
	48	
	49	`                var result = await signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, fals`
	50	
	51	`                if (result.Succeeded)`
	52	`                {`
	53	`                    int userID = 1;   //Hard Coding`
	54	`                    User userinfo = new User`
	55	`                    {`
	56	`                        Id = userID,`
	57	`                        Email = model.Email`

```
                    };

                    HttpContext.Session.SetString("SessionUser", JsonConvert.SerializeObject(userinfo));

                    bool clientinfo = true; //Hard Coding
                    if (clientinfo)
                    {
                        return RedirectToAction("ClientDashBoard", "Client");
                    }
                    else
                    {
                        TempData["ClientProfileInfo"] = "Fill the profile details before requesting for a quote";
                        return RedirectToAction("ClientProfile", "Client");
                    }

                }

                ModelState.AddModelError(string.Empty, "Invalid Credentials!");
            }
            return View(model);
        }

        [HttpGet]
        [AllowAnonymous]
        [ExcludeFromCodeCoverage]
        public IActionResult Register()
        {
            return View();
        }

        [HttpPost]
        [AllowAnonymous]
        [ExcludeFromCodeCoverage]
        public async Task<IActionResult> Register(Register registerInfo)
        {
            if (ModelState.IsValid)
            {
                var user = new IdentityUser { UserName = registerInfo.Email, Email = registerInfo.Email };
                var result = await userManager.CreateAsync(user, registerInfo.Password);

                if (result.Succeeded)
                {
                    await signInManager.SignInAsync(user, isPersistent: false);
                    User userinfo = new User
                    {
                        UserName = registerInfo.UserName,
                        Email = registerInfo.Email
                    };


                    TempData["RegistrationSuccessful"] = "You're registered succesfully!";
                    return RedirectToAction("Login", "Account");
                }

                foreach (var error in result.Errors)
                {
                    ModelState.AddModelError("", error.Description);
                }
            }
            return View();
        }

        [HttpPost]
        [ExcludeFromCodeCoverage]
        public IActionResult Logout()
        {
            signInManager.SignOutAsync();
            return RedirectToAction("Index", "Home");

        }

        public bool RegisterDataValidation(Register registerinfo)
        {
            bool flag = false;
            if ((registerinfo.UserName.Length <= 50) && (registerinfo.UserName != String.Empty))
            {
                if ((Regex.IsMatch(registerinfo.Email, @"^[\w!#$%&'*+\-/=?\^_`{|}~]+(\.[\w!#$%&'*+\-/=?\^_`{|}~]+)*")) &
                {
                    if (registerinfo.Password == registerinfo.ConfirmPassword)
                    {
                        flag = true;
                    }
                }
            }
            else
            {
                flag = false;
            }

            return flag;
        }
    }
}
```