LAB 3
Account Management

❖ CONTENT

- Display user list
- Reset user password
- Block/Unblock user

❖ INSTRUCTION

A. Display user list

1. Create Admin Controller to handle user management actions

```
[Authorize(Roles = "Admin")]
public class AdminController : Controller
{
    private readonly UserManager<IdentityUser> _userManager;

    public AdminController(UserManager<IdentityUser> userManager)
    {
        _userManager = userManager;
    }

    public async Task<IActionResult> UserList()
    {
        var users = await _userManager.Users.ToListAsync();
        var userList = new List<UserViewModel>();

        foreach (var user in users)
        {
            var roles = await _userManager.GetRolesAsync(user);
            userList.Add(new UserViewModel
            {
                Id = user.Id,
                Email = user.Email,
                Roles = roles.ToList()
            });
        }
        return View(userList);
    }
}
```

*Figure 1 – **Controllers/AdminController.cs***

2. Create UserViewModel to store user details

```
public class UserViewModel
{
    public string Id { get; set; }
    public string Email { get; set; }
    public List<string> Roles { get; set; }
    public bool IsBlocked { get; set; }
}
```

*Figure 2 - **ViewModels/UserViewModel.cs***

3. Create UserList view to display account list with roles and menu to reset pass & block/unblock user for *non-admin* accounts

```
@model List<web.ViewModels.UserViewModel>

<h2>User Management</h2>
<table class="table table-bordered">
    <thead>
        <tr>
            <th>ID</th>
            <th>Email</th>
            <th>Role</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var user in Model)
        {
            <tr>
                <td>@user.Id</td>
                <td>@user.Email</td>
                <td>@string.Join(", ", user.Roles)</td>
                <td>
                    @{
                        if (!user.Roles.Contains("Admin"))
                        {
                            <form asp-action="ResetPass" asp-route-id="@user.Id" method="post" style="display:inline;">
                                <button type="submit" class="btn btn-warning">Reset Password</button>
                            </form>

                            @if (!user.IsBlocked)
                            {
                                <form asp-action="BlockUser" asp-route-id="@user.Id" method="post" style="display:inline;">
                                    <button type="submit" class="btn btn-danger">Block User</button>
                                </form>
                            }
                            else
                            {
                                <form asp-action="UnblockUser" asp-route-id="@user.Id" method="post" style="display:inline;">
                                    <button type="submit" class="btn btn-primary">Unblock User</button>
                                </form>
                            }
                        }
                    }
                </td>
            </tr>
        }
    </tbody>
</table>
```

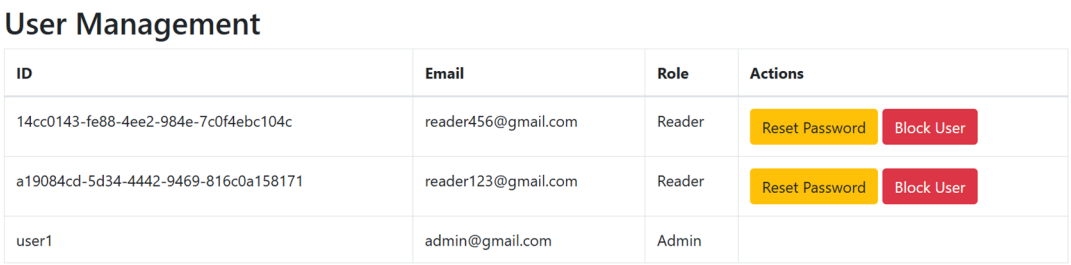
*Figure 3 - **Views/Admin/UserList.cshtml***

4. Add new menu to navigation bar for only admin role

```
@if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))
{
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Admin" asp-action="UserList">User List</a>
    </li>
}
```

*Figure 4 - **Views/Shared/_Layout.cshtml***

❖ Result



User Management

ID	Email	Role	Actions
14cc0143-fe88-4ee2-984e-7c0f4ebc104c	reader456@gmail.com	Reader	Reset Password Block User
a19084cd-5d34-4442-9469-816c0a158171	reader123@gmail.com	Reader	Reset Password Block User
user1	admin@gmail.com	Admin	

Figure 5 - User Management page

B. Reset User Password

1. Add new method to reset password in AdminController

```
[HttpPost]
public async Task<IActionResult> ResetPass(string id)
{
    var user = await _userManager.FindByIdAsync(id);
    if (user == null)
    {
        TempData["Error"] = "User not found.";
        return RedirectToAction("UserList");
    }

    // Generate a password reset token
    var token = await _userManager.GeneratePasswordResetTokenAsync(user);
    var newPassword = "123@Abc";

    // Reset the password using the token
    var result = await _userManager.ResetPasswordAsync(user, token, newPassword);

    if (result.Succeeded)
    {
        TempData["Message"] = $"Password for {user.Email} has been reset to default (123@Abc)";
    }
    else
    {
        TempData["Error"] = $"Failed to reset password for {user.Email}";
    }

    return RedirectToAction("UserList");
}
```

*Figure 6 - **Controllers/AdminController.cs***

2. Update UserList view to display message after resetting password (and also for blocking/unblocking user – we do later)

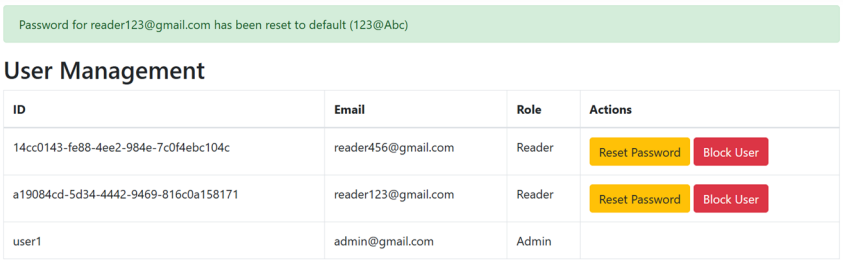
```
@model List<web.ViewModels.UserViewModel>

@if (TempData["Message"] != null)
{
    <div class="alert alert-success">
        @TempData["Message"]
    </div>
}

@if (TempData["Error"] != null)
{
    <div class="alert alert-danger">
        @TempData["Error"]
    </div>
}
```

*Figure 7 - **Views/Admin/UserList.cshtml***

❖ Result



C. Block/Unblock User

1. Add new method to block user in AdminController

```
[HttpPost]
public async Task<IActionResult> BlockUser(string id)
{
    var user = await _userManager.FindByIdAsync(id);
    if (user == null)
    {
        TempData["Error"] = "User not found.";
        return RedirectToAction("UserList");
    }

    // Set a very far lockout end date to block the user
    var lockoutEndDate = DateTimeOffset.UtcNow.AddYears(100);
    var result = await _userManager.SetLockoutEndDateAsync(user, lockoutEndDate);

    if (result.Succeeded)
    {
        TempData["Message"] = $"User {user.Email} has been blocked.";
    }
    else
    {
        TempData["Error"] = $"Failed to block user {user.Email}.";
    }

    return RedirectToAction("UserList");
}
```

*Figure 8 - **Controllers/AdminController.cs***

2. Add another method to unblock user

```
[HttpPost]
public async Task<IActionResult> UnblockUser(string id)
{
    var user = await _userManager.FindByIdAsync(id);
    if (user == null)
    {
        TempData["Error"] = "User not found.";
        return RedirectToAction("UserList");
    }

    // Set lockout end date to null to unblock the user
    var result = await _userManager.SetLockoutEndDateAsync(user, null);

    if (result.Succeeded)
    {
        TempData["Message"] = $"User {user.Email} has been unblocked.";
    }
    else
    {
        TempData["Error"] = $"Failed to unblock user {user.Email}.";
    }

    return RedirectToAction("UserList");
}
```

*Figure 9 - **Controllers/AdminController.cs***

3. Prevent user login to system after blocking

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));

    services.AddDefaultIdentity<IdentityUser>(options =>
    {
        options.SignIn.RequireConfirmedAccount = true;

        // Configure lockout settings for blocking users
        options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromDays(365);
        options.Lockout.MaxFailedAccessAttempts = 3;
        options.Lockout.AllowedForNewUsers = true;
    })
    .AddRoles<IdentityRole>()
```

*Figure 10 - **Startup.cs***

❖ Result

User reader456@gmail.com has been blocked.

User Management

ID	Email	Role	Actions
14cc0143-fe88-4ee2-984e-7c0f4ebc104c	reader456@gmail.com	Reader	Reset Password Unblock User
a19084cd-5d34-4442-9469-816c0a158171	reader123@gmail.com	Reader	Reset Password Block User
user1	admin@gmail.com	Admin	