Initial State

Start with a new Asp.Net Core Web Application using the MVC template with Individual User Accounts.

Add a new model class for custom users.

1. Add a class in the model folder
   ```
   public class CustomIdentityUser
   ```
2. Add a using directive
   ```
   using Microsoft.AspNetCore.Identity;
   ```
3. Inherit from IdentityUser
   ```
   public class CustomIdentityUser : IdentityUser
   ```
4. Add additional properties like “Notes”.
   ```
   public string Notes { get; set; }
   ```
5. Check for errors by building.

Update the database context class to use the new model.

1. Open Data\ApplicationDbContext
2. Add using directive
   ```
   using WebApplication.Models;
   ```
3. Modify the base type to use a generic.
   ```
   public class ApplicationDbContext : IdentityDbContext<CustomIdentityUser>
   ```
4. Check for errors by building.

Update web application configuration

1. open Startup.cs
2. Update ConfigureServices
   ```
   services.AddDefaultIdentity<CustomIdentityUser>(options =>
   options.SignIn.RequireConfirmedAccount = true)
   ```

Update partial view

1. open views\shared_LoginPartial.cshtml
2. update DI directives.
   ```
   @inject SignInManager<CustomIdentityUser> SignInManager
   @inject UserManager<CustomIdentityUser> UserManager
   ```

Migrate changes to the database

1. Tools – NuGet Package Manager – Package Manager Console
2. add-migration initialization
3. update-database

Add customizable register page.

1. Right click project – add – New Scaffolded Item…
2. In the “Add New Scaffolded Item” dialog, select identity. Click “Add”.
3. Check “Account\Register”. Click “Add”

Modify register page to include notes

1. Open Areas\Identity\Pages\Account\Register.cshtml
2. Add notes after confirm password. After line 29.

```
<div class="form-group">
    <label asp-for="Input.Notes"></label>
    <input asp-for="Input.Notes" class="form-control" />
    <span asp-validation-for="Input.Notes" class="text-danger"></span>
```

```
</div>
```

Modify register page code behind to include notes

1. Open Areas\Identity\Pages\Account\Register.cshtml.cs
1. Add notes after confirm password. After line 63.

```
[DataType(DataType.MultilineText)]
[Display(Name = "User Notes")]
public string Notes { get; set; }
```

2. Modify the post behavior to bind to note. Update around line 82.

```
var user = new CustomIdentityUser { UserName = Input.Email, Email =
Input.Email, Notes = Input.Notes };
```

Update test controller to use notes

1. Open Controllers\HomeController.cs
2. add a using directive.

```
using Microsoft.AspNetCore.Identity;
```

3. add a field to reference a user manager.

```
private readonly UserManager<CustomIdentityUser> _userManager;
```

4. Modify the constructor to support dependency injection.

```
public HomeController(ILogger<HomeController> logger,
UserManager<CustomIdentityUser> userManager)
{
        _logger = logger;
        _userManager = userManager;
}
```

5. Modify the index action to retrieve user.

```
public async Task<IActionResult> Index()
{
        var user = await _userManager.GetUserAsync(User);
        ViewBag.User = user?.UserName;
        ViewBag.Notes = user?.Notes;
        return View();
}
```

6. Modify the index view to show user.

```
@{
    ViewData["Title"] = "Home Page";
}
<div class="text-center">
    <h1 class="display-4">Welcome @ViewBag.User</h1>
    <p>@ViewBag.Notes</p>
</div>
```

Test your work

1. Run the application
2. Register a new user with notes
3. Verify the user's email and notes appear on the note page