

Lab 2.2 – Modify Identity DB on ASP.NET Core

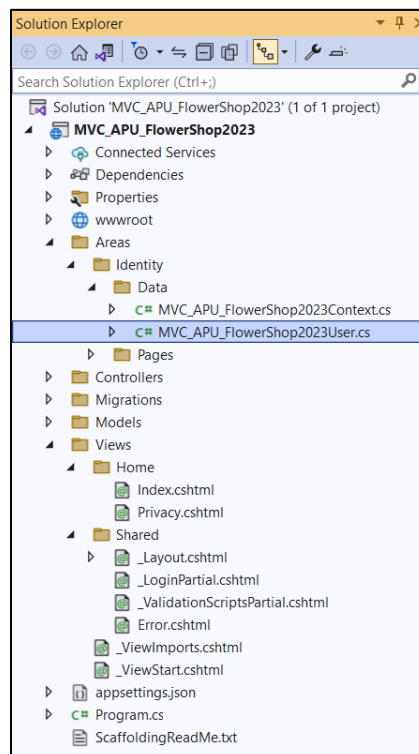
Estimated usage time: 1 Hour 30 Minutes

In this topic, you learn how to add custom user data to the Identity DB. You can also learn how to modify the user registration page and user profile page for your own use.

a. Add custom user data to the Identity DB

(Estimated Total Time Used: 30 minutes)

1. Continued from the **Lab 2.1** project.
2. Update the *IdentityUser* derived class with custom properties. If you named the project MVC_APU_FlowerShop2023, the file is named **Areas/Identity/Data/MVC_APU_FlowerShopUser2023.cs**.



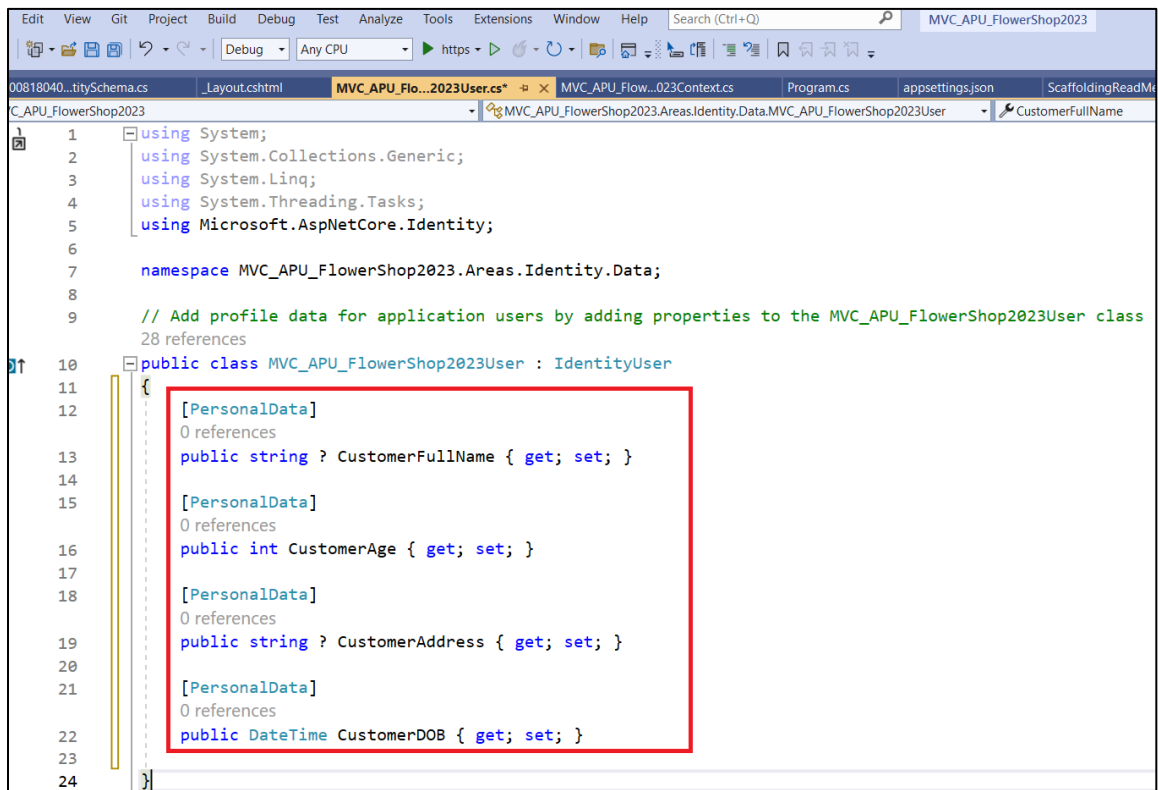
3. Update the file with the following code:

```
[PersonalData]
public string ? CustomerFullName { get; set; }

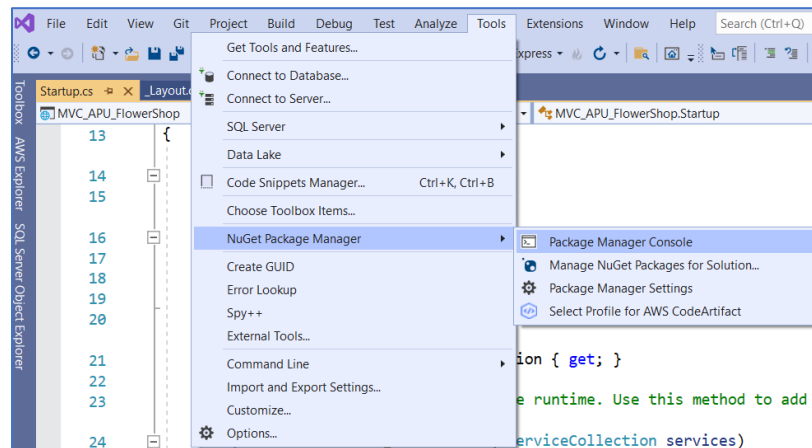
[PersonalData]
public int CustomerAge { get; set; }

[PersonalData]
public string ? CustomerAddress { get; set; }

[PersonalData]
public DateTime CustomerDOB { get; set; }
```

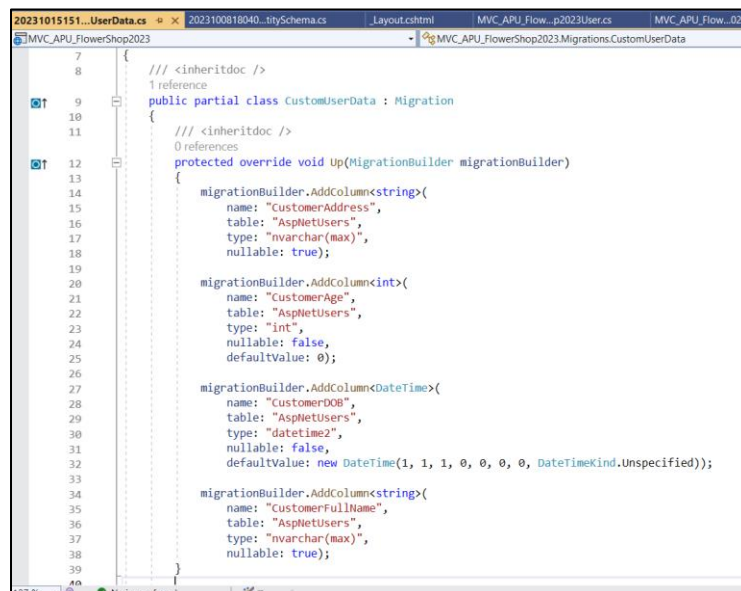
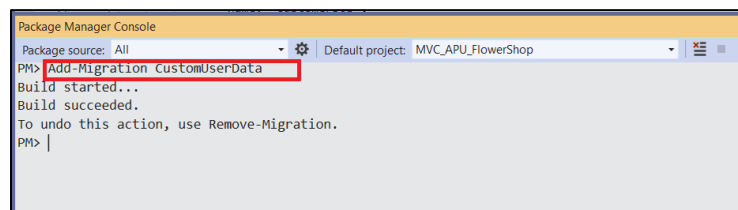


- To modify the current Identity database, you must use the Package Manager Console (PMC).
- To start the PMC, click on the **Tools > NuGet Package Manager > Package Manager Console**.



- In the Visual Studio **Package Manager Console**, type the below commands:

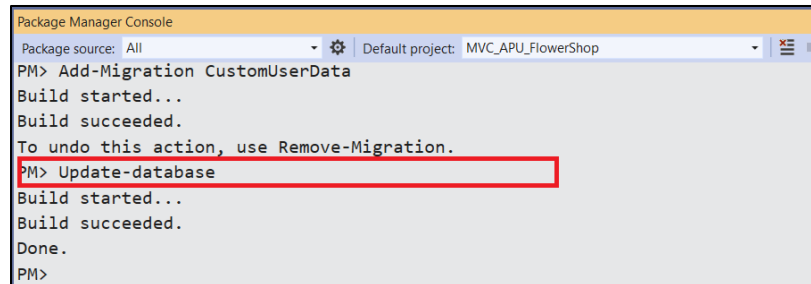
Add-Migration CustomUserData



A migration code will be generated after the add-migration command. This migration code will modify the existing structural schema of the tables to the latest schema.

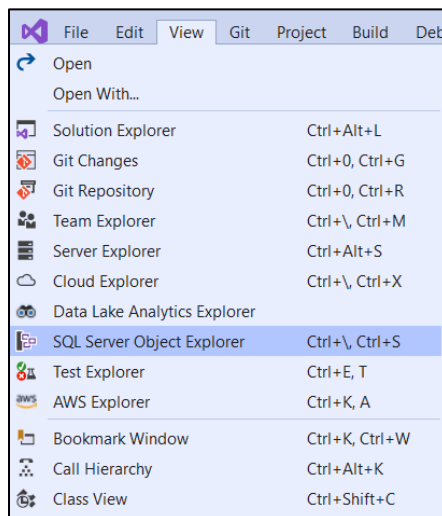
- After creating a migration file using the add-migration command, you must update the database. Execute the Update-Database command to create or modify a database schema.

Update-Database

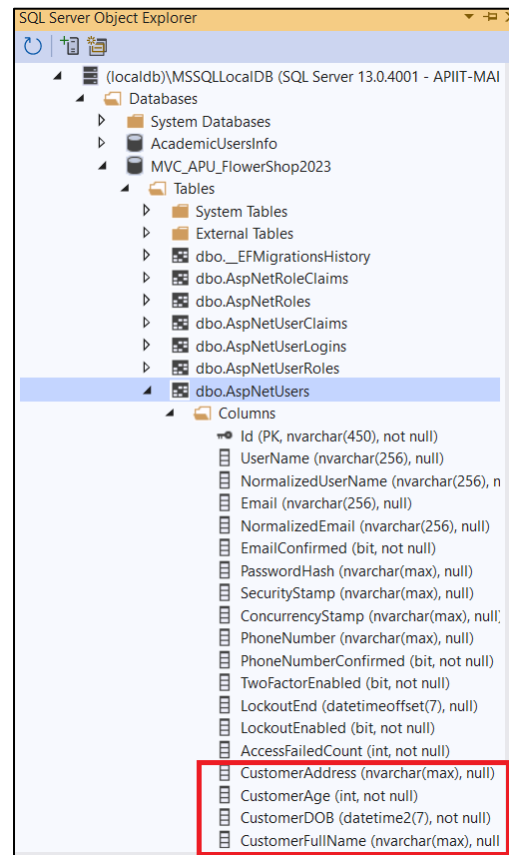


```
Package Manager Console
Package source: All Default project: MVC_APU_FlowerShop
PM> Add-Migration CustomUserData
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> Update-database
Build started...
Build succeeded.
Done.
PM>
```

- Now, in Visual Studio, open the **SQL Server Object Explorer (SSOX)**.



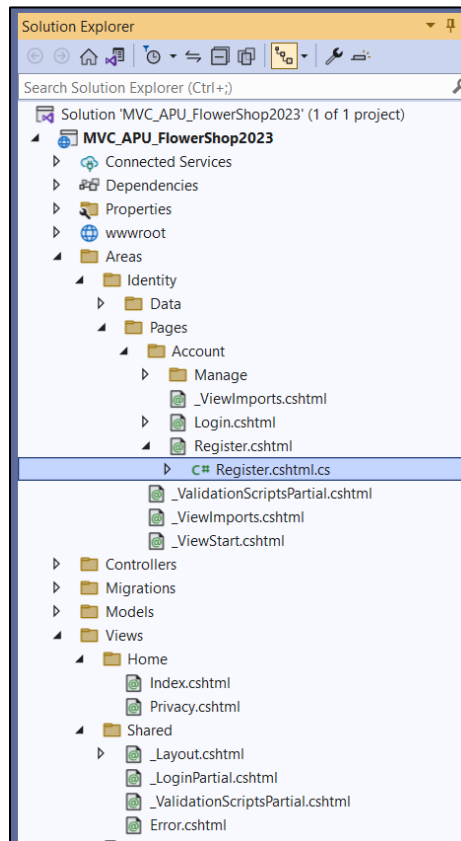
9. You will see your database “MVC_APU_FlowerShop2023” located in the SQL Server and all the default tables are under the database. Now, open the **dbo.AspNetUsers**, you will see the new columns appear in the table.



b. Update the Account/Register.cshtml page.

(Estimated Total Time Used: 30 minutes)

1. Update the *InputModel* in **Areas/Identity/Pages/Account/Register.cshtml.cs** with the following highlighted code:



```
[Required(ErrorMessage = "You must enter the name first before
submitting your form!")]
[StringLength(256, ErrorMessage = "You must enter the value between 6
- 256 chars", MinimumLength = 6)]
[Display(Name = "Customer Full Name")] //label
public string customerfullname { get; set; }

[Required]
[Display(Name = "Customer DOB")]
[DataType(DataType.Date)]
public DateTime DoB { get; set; }
```

Register.cshtml.cs* 202310151513...mUserData.cs 2023100818040...itySchema.cs _Layout.cshtml MVC_APU_Flow...p2023User... MVC_APU_FlowerShop2023 MVC_APU_FlowerShop2023.Areas.Identity.Pages.Account.Register.cshtml DoB

```
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111
```

```
/// <summary>  
/// This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used  
/// directly from your code. This API may change or be removed in future releases.  
/// </summary>  
1 reference  
public class InputModel  
{  
    /// <summary> This API supports the ASP.NET Core Identity default UI infrastru...  
    [Required]  
    [EmailAddress]  
    [Display(Name = "Email")]  
    7 references  
    public string Email { get; set; }  
  
    /// <summary> This API supports the ASP.NET Core Identity default UI infrastru...  
    [Required]  
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 6)]  
    [DataType(DataType.Password)]  
    [Display(Name = "Password")]  
    4 references  
    public string Password { get; set; }  
  
    /// <summary> This API supports the ASP.NET Core Identity default UI infrastru...  
    [DataType(DataType.Password)]  
    [Display(Name = "Confirm password")]  
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]  
    3 references  
    public string ConfirmPassword { get; set; }  
  
    [Required(ErrorMessage = "You must enter the name first before submitting your form!")]  
    [StringLength(256, ErrorMessage = "You must enter the value between 6 - 256 chars", MinimumLength = 6)]  
    [Display(Name = "Customer Full Name")] //label  
    0 references  
    public string customerfullname { get; set; }  
  
    [Required]  
    [Display(Name = "Customer DOB")]  
    [DataType(DataType.Date)]  
    0 references  
    public DateTime DoB { get; set; }  
}
```

- Update the *OnPostAsync()* function in **Areas/Identity/Pages/Account/Register.cshtml.cs** with the following highlighted code:

```

user.CustomerFullName = Input.customerfullname;

user.CustomerDOB = Input.DoB;

user.EmailConfirmed = true;

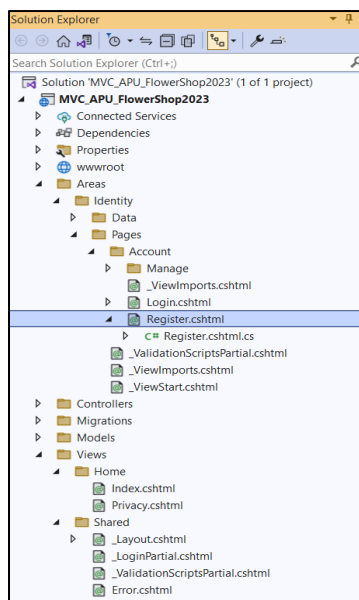
```

```

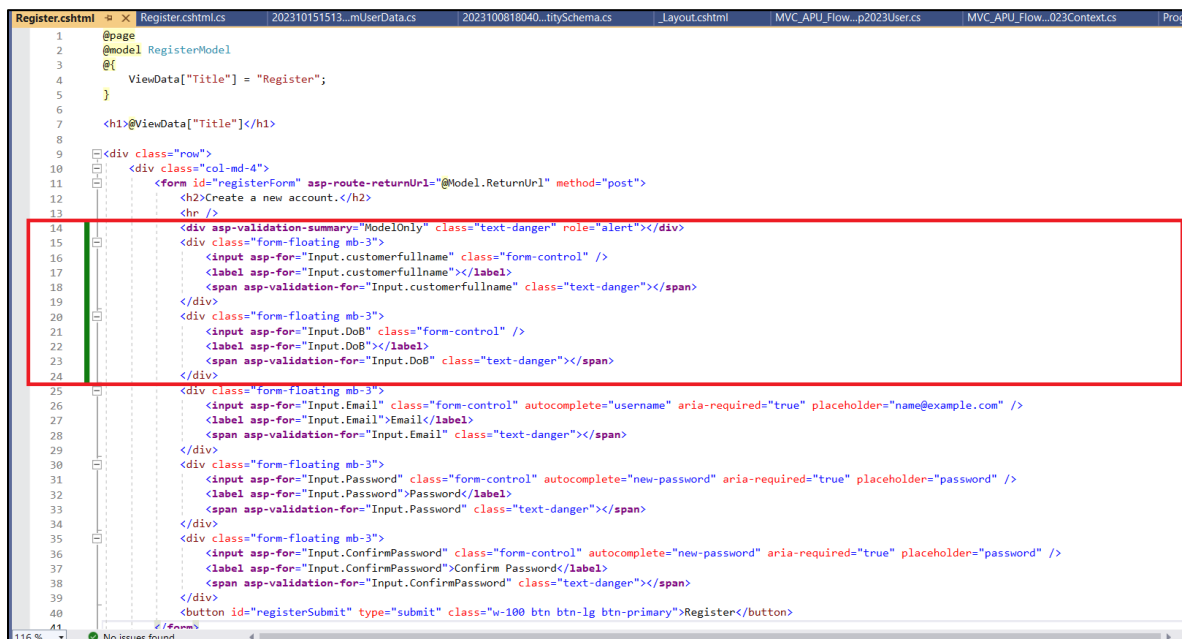
120 public async Task<IActionResult> OnPostAsync(string returnUrl = null)
121 {
122     returnUrl ??= Url.Content("~/");
123     ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
124     if (ModelState.IsValid)
125     {
126         var user = CreateUser();
127         user.CustomerFullName = Input.customerfullname;
128         user.CustomerDOB = Input.DoB;
129         user.EmailConfirmed = true;
130
131         await _userManager.SetUserNameAsync(user, Input.Email, CancellationToken.None);
132         await _emailSender.SetEmailAsync(user, Input.Email, CancellationToken.None);
133         var result = await _userManager.CreateAsync(user, Input.Password);
134
135         if (result.Succeeded)
136         {
137             //_logger.LogInformation("User created a new account with password.");
138
139             //var userId = await _userManager.GetUserIdAsync(user);
140             //var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
141             //code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));
142             //var callbackUrl = Url.Page(
143             //    "/Account/ConfirmEmail",
144             //    pageHandler: null,
145             //    values: new { area = "Identity", userId = userId, code = code, returnUrl = returnUrl },
146             //    protocol: Request.Scheme);
147
148             //await _emailSender.SendEmailAsync(Input.Email, "Confirm your email",
149             //    $"Please confirm your account by clicking here<a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");
150
151             if (_userManager.Options.SignIn.RequireConfirmedAccount)
152             {
153                 // return RedirectToPage("RegisterConfirmation", new { email = Input.Email, returnUrl = returnUrl });
154                 return RedirectToPage("Login");
155             }
156             else
157             {

```


3. Update the `Areas/Identity/Pages/Account/Register.cshtml` with the following highlighted markup:



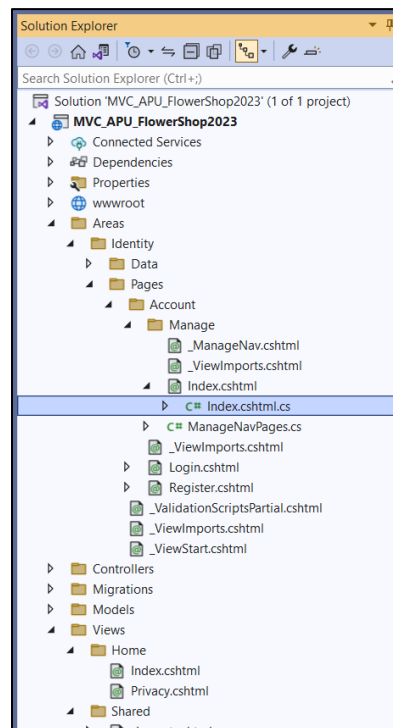
```
<div class="form-floating mb-3">
  <input asp-for="Input.customerfullname" class="form-control" />
  <label asp-for="Input.customerfullname"></label>
  <span asp-validation-for="Input.customerfullname" class="text-
danger"></span>
</div>
<div class="form-floating mb-3">
  <input asp-for="Input.DoB" class="form-control" />
  <label asp-for="Input.DoB"></label>
  <span asp-validation-for="Input.DoB" class="text-danger"></span>
</div>
```



c. Update the Account/Manage/Index.cshtml page.

(Estimated Total Time Used: 30 minutes)

1. Update the *InputModel* in **Areas/Identity/Pages/Account/Manage/Index.cshtml.cs** with the following highlighted code:

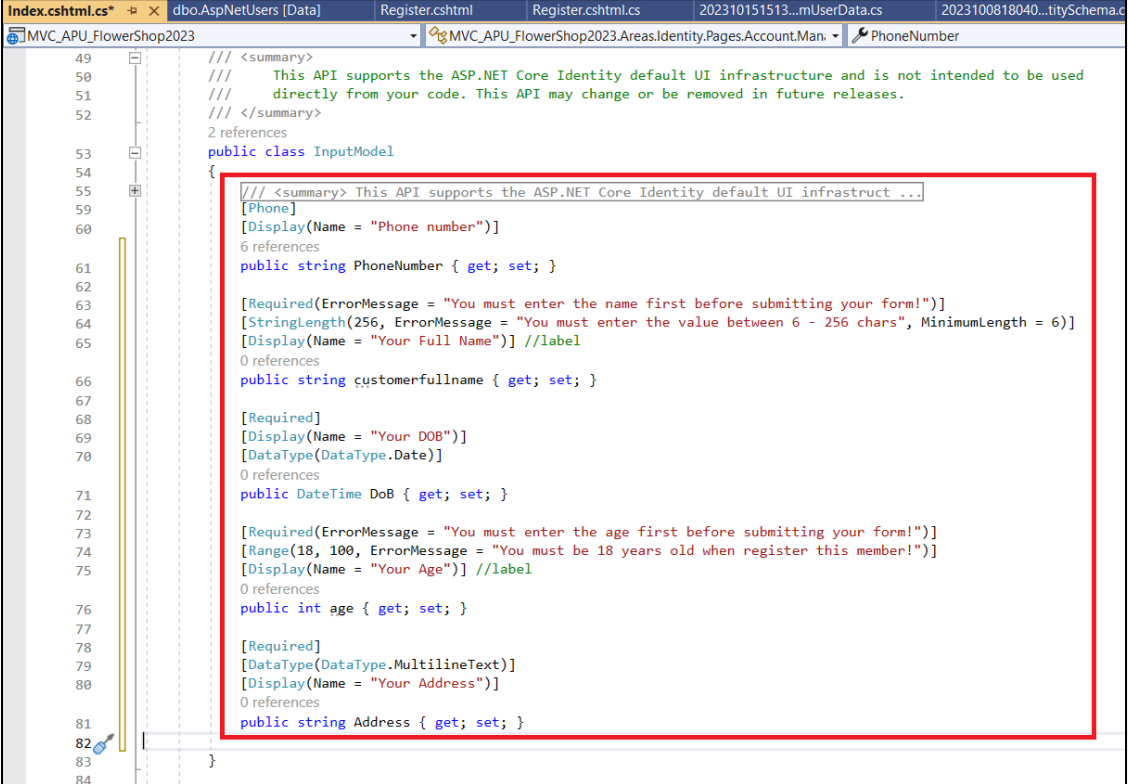


```
[Required(ErrorMessage = "You must enter the name first before submitting your
form!")]
[StringLength(256, ErrorMessage = "You must enter the value between 6 - 256
chars", MinimumLength = 6)]
[Display(Name = "Your Full Name")] //label
public string customerfullname { get; set; }

[Required]
[Display(Name = "Your DOB")]
[DataType(DataType.Date)]
public DateTime DoB { get; set; }

[Required(ErrorMessage = "You must enter the age first before submitting your
form!")]
[Range(18, 100, ErrorMessage = "You must be 18 years old when register this
member!")]
[Display(Name = "Your Age")] //label
public int age { get; set; }

[Required]
[DataType(DataType.MultilineText)]
[Display(Name = "Your Address")]
public string Address { get; set; }
```



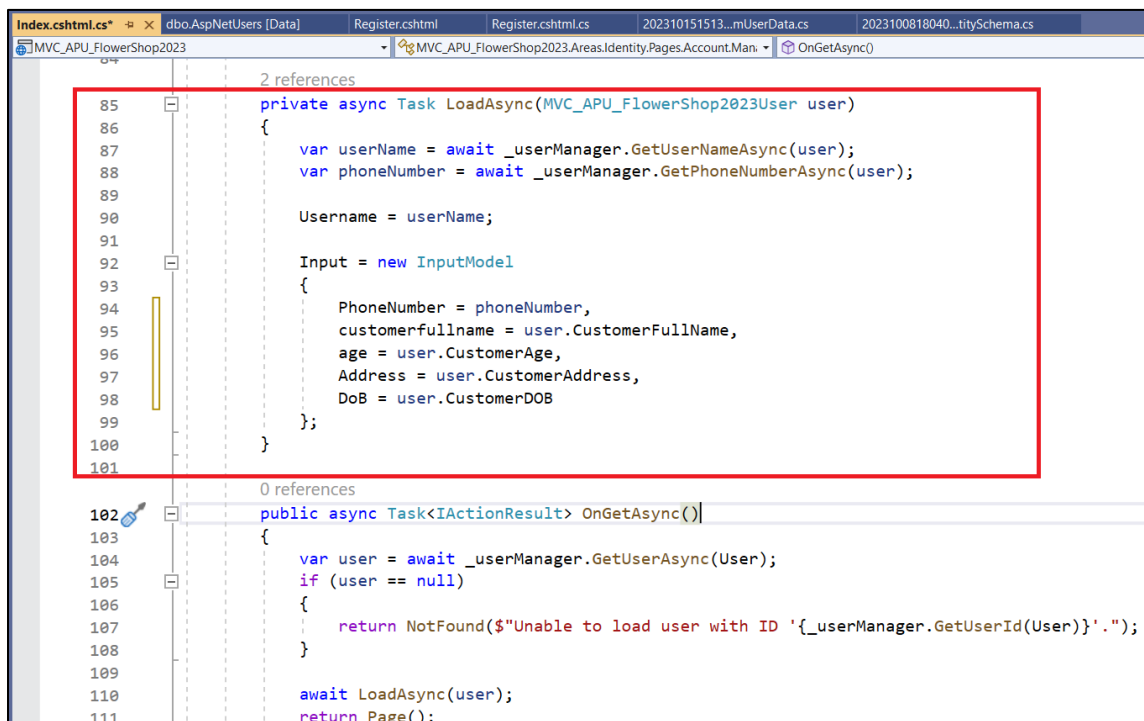
```

49  /// <summary>
50  ///     This API supports the ASP.NET Core Identity default UI infrastructure and is not intended to be used
51  ///     directly from your code. This API may change or be removed in future releases.
52  /// </summary>
53  2 references
54  public class InputModel
55  {
56      /// <summary> This API supports the ASP.NET Core Identity default UI infrastru...
57      [Phone]
58      [Display(Name = "Phone number")]
59      6 references
60      public string PhoneNumber { get; set; }
61
62      [Required(ErrorMessage = "You must enter the name first before submitting your form!")]
63      [StringLength(256, ErrorMessage = "You must enter the value between 6 - 256 chars", MinimumLength = 6)]
64      [Display(Name = "Your Full Name")] //label
65      0 references
66      public string customerfullname { get; set; }
67
68      [Required]
69      [Display(Name = "Your DOB")]
70      [DataType(DataType.Date)]
71      0 references
72      public DateTime Dob { get; set; }
73
74      [Required(ErrorMessage = "You must enter the age first before submitting your form!")]
75      [Range(18, 100, ErrorMessage = "You must be 18 years old when register this member!")]
76      [Display(Name = "Your Age")] //label
77      0 references
78      public int age { get; set; }
79
80      [Required]
81      [DataType(DataType.MultilineText)]
82      [Display(Name = "Your Address")]
83      0 references
84      public string Address { get; set; }
85  }

```

2. Update the *LoadAsync()* in **Areas/Identity/Pages/Account/Manage/Index.cshtml.cs** with the following highlighted code:

```
Input = new InputModel
{
    PhoneNumber = phoneNumber,
    customerfullname = user.CustomerFullName,
    age = user.CustomerAge,
    Address = user.CustomerAddress,
    DoB = user.CustomerDOB
};
```



```
Index.cshtml.cs* x  dbo.AspNetUsers [Data]  Register.cshtml  Register.cshtml.cs  202310151513...mUserData.cs  2023100818040...titySchema.cs
MVC_APU_FlowerShop2023  MVC_APU_FlowerShop2023.Areas.Identity.Pages.Account.Manage.Index.cshtml  OnGetAsync()

2 references
85  private async Task LoadAsync(MVC_APU_FlowerShop2023User user)
86  {
87      var userName = await _userManager.GetUserNameAsync(user);
88      var phoneNumber = await _userManager.GetPhoneNumberAsync(user);
89
90      Username = userName;
91
92      Input = new InputModel
93      {
94          PhoneNumber = phoneNumber,
95          customerfullname = user.CustomerFullName,
96          age = user.CustomerAge,
97          Address = user.CustomerAddress,
98          DoB = user.CustomerDOB
99      };
100 }
101

0 references
102 public async Task<IActionResult> OnGetAsync()
103 {
104     var user = await _userManager.GetUserAsync(User);
105     if (user == null)
106     {
107         return NotFound($"Unable to load user with ID '{_userManager.GetUserId(User)}'.");
108     }
109
110     await LoadAsync(user);
111     return Page();
```

- Update the *OnPostAsync()* in **Areas/Identity/Pages/Account/Manage/Index.cshtml.cs** with the following highlighted code:

```

        if (Input.customerfullname != user.CustomerFullName)
        {
            user.CustomerFullName = Input.customerfullname;
        }

        if (Input.DoB != user.CustomerDOB)
        {
            user.CustomerDOB = Input.DoB;
        }

        if (Input.Address != user.CustomerAddress)
        {
            user.CustomerAddress = Input.Address;
        }

        if (Input.age != user.CustomerAge)
        {
            user.CustomerAge = Input.age;
        }
        await _userManager.UpdateAsync(user);

```

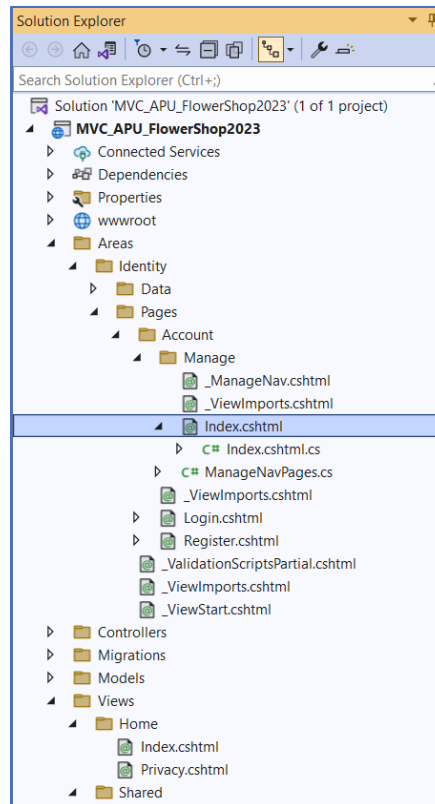
The screenshot shows the Visual Studio IDE with the file `Index.cshtml.cs` open. The code is for the `OnPostAsync()` method in the `Account.Manage.Index` page. A red rectangular box highlights the code that updates the user's profile information, which matches the code provided in the previous block. The code includes checks for full name, date of birth, address, and age, and then calls `_userManager.UpdateAsync(user)` to save the changes. The status message is set to "Your profile has been updated" and the user is redirected to the page.

```

114 public async Task<IActionResult> OnPostAsync()
115 {
116     var user = await _userManager.GetUserAsync(User);
117     if (user == null) ...
121
122     if (!ModelState.IsValid) ...
127
128     var phoneNumber = await _userManager.GetPhoneNumberAsync(user);
129     if (Input.PhoneNumber != phoneNumber)
130     {
131         var setPhoneResult = await _userManager.SetPhoneNumberAsync(user, Input.PhoneNumber);
132         if (!setPhoneResult.Succeeded) ...
137
138     if (Input.customerfullname != user.CustomerFullName)
139     {
140         user.CustomerFullName = Input.customerfullname;
141     }
142
143     if (Input.DoB != user.CustomerDOB)
144     {
145         user.CustomerDOB = Input.DoB;
146     }
147
148     if (Input.Address != user.CustomerAddress)
149     {
150         user.CustomerAddress = Input.Address;
151     }
152
153     if (Input.age != user.CustomerAge)
154     {
155         user.CustomerAge = Input.age;
156     }
157
158     await _userManager.UpdateAsync(user);
159     await _signInManager.RefreshSignInAsync(user);
160     StatusMessage = "Your profile has been updated";
161     return RedirectToPage();
162 }
163

```

4. Update the **Areas/Identity/Pages/Account/Manage/Index.cshtml** with the following highlighted markup:



```
<div class="form-floating mb-3">
  <input asp-for="Input.customerfullname" class="form-control" />
  <label asp-for="Input.customerfullname"></label>
  <span asp-validation-for="Input.customerfullname" class="text-danger"></span>
</div>
<div class="form-floating mb-3">
  <input asp-for="Input.age" class="form-control" />
  <label asp-for="Input.age"></label>
  <span asp-validation-for="Input.age" class="text-danger"></span>
</div>
<div class="form-floating mb-3">
  <input asp-for="Input.DoB" class="form-control" />
  <label asp-for="Input.DoB"></label>
  <span asp-validation-for="Input.DoB" class="text-danger"></span>
</div>
<div class="form-floating mb-3">
  <input asp-for="Input.Address" class="form-control" />
  <label asp-for="Input.Address"></label>
  <span asp-validation-for="Input.Address" class="text-danger"></span>
</div>
```

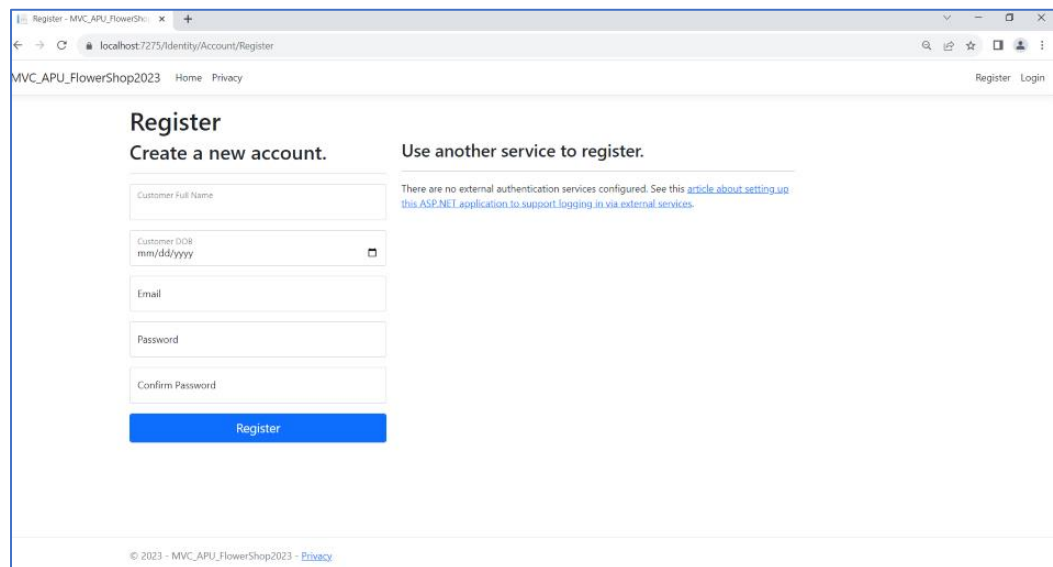
```

Index.cshhtml*  Index.cshhtml.cs  dbo.AspNetUsers [Data]  Register.cshhtml  Register.cshhtml.cs  202310151513...mUserData.cs
1  @page
2  @model IndexModel
3  @{
4      ViewData["Title"] = "Profile";
5      ViewData["ActivePage"] = ManageNavPages.Index;
6  }
7
8  <h3>@ViewData["Title"]</h3>
9  <partial name="_StatusMessage" for="StatusMessage" />
10 <div class="row">
11     <div class="col-md-6">
12         <form id="profile-form" method="post">
13             <div asp-validation-summary="ModelOnly" class="text-danger" role="alert"></div>
14             <div class="form-floating mb-3">
15                 <input asp-for="Username" class="form-control" placeholder="Please choose your username." disabled />
16                 <label asp-for="Username" class="form-label"></label>
17             </div>
18             <div class="form-floating mb-3">
19                 <input asp-for="Input.PhoneNumber" class="form-control" placeholder="Please enter your phone number."/>
20                 <label asp-for="Input.PhoneNumber" class="form-label"></label>
21                 <span asp-validation-for="Input.PhoneNumber" class="text-danger"></span>
22             </div>
23             <div class="form-floating mb-3">
24                 <input asp-for="Input.customerfullname" class="form-control" />
25                 <label asp-for="Input.customerfullname"></label>
26                 <span asp-validation-for="Input.customerfullname" class="text-danger"></span>
27             </div>
28             <div class="form-floating mb-3">
29                 <input asp-for="Input.age" class="form-control" />
30                 <label asp-for="Input.age"></label>
31                 <span asp-validation-for="Input.age" class="text-danger"></span>
32             </div>
33             <div class="form-floating mb-3">
34                 <input asp-for="Input.DoB" class="form-control" />
35                 <label asp-for="Input.DoB"></label>
36                 <span asp-validation-for="Input.DoB" class="text-danger"></span>
37             </div>
38             <div class="form-floating mb-3">
39                 <input asp-for="Input.Address" class="form-control" />
40                 <label asp-for="Input.Address"></label>
41                 <span asp-validation-for="Input.Address" class="text-danger"></span>
42             </div>
43             <button id="update-profile-button" type="submit" class="w-100 btn btn-lg btn-primary">Save</button>
44         </form>
45     </div>

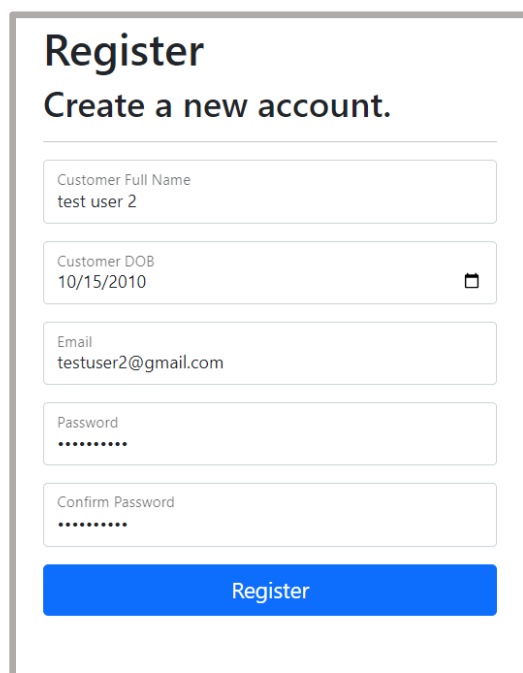
```

d. Test, create, update, delete the custom data in the table.

(Estimated Total Time Used: 20 minutes)

1. Once every above step is done, let's test the application now.**a. Register a new user**

The screenshot shows a web browser window with the URL `localhost:7275/Identity/Account/Register`. The page title is "Register - MVC_APU_FlowerShop2023". The main heading is "Register" with the subheading "Create a new account." Below this are five input fields: "Customer Full Name", "Customer DOB" (with a calendar icon), "Email", "Password", and "Confirm Password". A blue "Register" button is at the bottom. To the right, under "Use another service to register:", there is a message: "There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services](#)." The footer shows "© 2023 - MVC_APU_FlowerShop2023 - [Privacy](#)".



The mockup shows the "Register" form with the following data entered:

- Customer Full Name: test user 2
- Customer DOB: 10/15/2010
- Email: testuser2@gmail.com
- Password:
- Confirm Password:

The "Register" button is highlighted in blue.

- b. View the custom user data on the /Identity/Account/Manage page.

MVC_APU_FlowerShop2023 Home Privacy

Manage your account

Change your account settings

Profile

Email

Password

Two-factor authentication

Personal data

Profile

Username
testuser2@gmail.com

Phone number

Your Full Name
test user 2

Your Age
0

Your DOB
10/15/2010

Your Address

Save

- c. Update data in User table.

Manage your account

Change your account settings

Profile

Email

Password

Two-factor authentication

Personal data

Profile

Username
testuser2@gmail.com

Phone number
0123456789

Your Full Name
test user 2

Your Age
22

Your DOB
10/15/2001

Your Address
Selangor

Save

Profile - MVC_APU_FlowerShop23 x +

localhost:7275/Identity/Account/Manage

MVC_APU_FlowerShop2023 Home Privacy Hello testuser2@gmail.com! Logout

Manage your account

Change your account settings

Profile

Profile

Your profile has been updated

Username
testuser2@gmail.com

Phone number
0123456789

Your Full Name
test user 2

Your Age
22

Your DOB
10/15/2001

Your Address
Selangor

Save

- d. Select the Personal Data tab.
- e. Select the Download button and examined the PersonalData.json file.

Personal Data - MVC_APU_Flowe x +

localhost:7275/Identity/Account/Manage/PersonalData

MVC_APU_FlowerShop2023 Home Privacy Hello testuser2@gmail.com! Logout

Manage your account

Change your account settings

Personal Data

Your account contains personal data that you have given us. This page allows you to download or delete that data.

Deleting this data will permanently remove your account, and this cannot be recovered.

Download

Delete

- f. Test the Delete button, which deletes the logged-on user.

Summary:

In this tutorial, we learned how to build add, download, and delete custom user data to Identity DB in an ASP.NET Core project. In the next tutorial, we will learn how to create a simple CRUD operation in an ASP.NET Core project.
