Sprint4_SCRUM1_72_Documentation

Bryson Brandon

CSCI-4250-002

Mr. Kinser

12/5/23

1. **UserController.cs**

   Functionality: This added **[HttpPost]** action in the **UserController** updates the team name for the authenticated user. It retrieves the current user, checks for existence, updates the team name, and persists changes to the database using **userManager**. It returns an **Ok** result with a success message upon a successful update or a 500 Internal Server Error status with an error message if an exception occurs.

```
       [HttpPost]
      public async Task<IActionResult> UpdateTeamName(string teamName)
      {
      try
      {
      // Get the current user
      var currentUser = await _userManager.GetUserAsync(User);

     // Check if the user exists
       if (currentUser == null)
        {
            return NotFound();
        }

        // Update the TeamName property
        currentUser.TeamName = teamName;

        // Update the user in the database
        await _userManager.UpdateAsync(currentUser);

        // Optionally, you can return a success message or redirect to a different page
        return Ok(new { Message = "Team name updated successfully." });
    }
    catch (Exception ex)
    {
        // Handle exceptions if needed
        return StatusCode(StatusCodes.Status500InternalServerError, new { Message = "Error updating team name." });
    }
}
```

2. **ApplicationUser.cs**

- **Functionality:** This change adds a **TeamName** field to the **ApplicationUser** class, allowing the storage of team names for users.

```
public class ApplicationUser : IdentityUser
{
```

```
[Required]
[DisplayName("First Name")]
[StringLength(50)]
public string? FirstName { get; set; }
[DisplayName("Last Name")]
[StringLength(50)]
[Required]
public string? LastName { get; set; }
public AccessCode? AccessCode { get; set; }
[DisplayName("Registered Hunt")]
public Hunt? Hunt{ get; set; }
public ICollection<Location>? TasksCompleted { get; set; } = new List<Location>();
[NotMapped]
public ICollection<string> Roles { get; set; }
= new List<string>();
public Carriers Carrier { get; set; }
public string? TeamName {  get; set; }
}
```

3. **CreateOrJoinTeam.cshtml**

- **Functionality:** This file represents a page allowing users to create or join a team. The **submitTeam()** function sends a POST request to the **UpdateTeamName** action in the **UserController** to add a user to a team. The page also includes functionality to open a team-joining dialog and close it. The `createTeam()` function is a placeholder and needs to be implemented to call **generateCode** and assign the team name to the user.

```
@{
    ViewData["Title"] = "Create or Join a Team";
}

<h1>Create or Join a Team</h1>
<div class="btn-div">
    <button type="button" id="btn-Create" class="w-100 btn btn-lg btn-primary" onclick="createTeam()">Create Team</button>
</div>
<div class="btn-div">
    <button type="button" id="btn-Join" class="w-100 btn btn-lg btn-primary" onclick="openJoinTeamDialog()">Join Team</button>
</div>

    <!--Pop up window dialog  -->
<dialog id="joinTeamDialog">
    <label for="teamName">Enter Team Name:</label>
    <input type="text" id="teamName" name="teamName" />
    <br /><br />
    <button type="button" onclick="submitTeam()">Submit</button>
    <button type="button" onclick="closeJoinTeamDialog()">Cancel</button>
</dialog>

<script>
    function openJoinTeamDialog() {
        // Show the dialog
        document.getElementById('joinTeamDialog').showModal();
    }

    function closeJoinTeamDialog() {
        // Close the dialog
        document.getElementById('joinTeamDialog').close();
    }

    //
    // This method is designed to send a POST request to the usercontroller method "UpdateTeamName" to add a user to a team.
    //
```

```
    function submitTeam() {
    // Get the team name entered by the user
    var teamName = document.getElementById('teamName').value;
    if(teamName != null){
    // Close the dialog
    closeJoinTeamDialog();

    // Send an AJAX request to update the team name
    $.ajax({
        type: 'POST',
        url: '/User/UpdateTeamName',
        data: { teamName: teamName },
        success: function (response) {
            alert("Added to team: " + teamName);
            console.log(response);
            window.location.href = '/Hunt/TeamPage';
        },
        error: function (error) {
            alert("Invalid Team name! Please try again...")
            console.error(error);
        }
        }
    });
    //call generateCode and assign the team name to the user
    function createTeam(){

    }
}
</script>
```

4. **Migration Code (Add Column):**

    - **Functionality:** This code adds a new column named "TeamCode" to the "AspNetUsers" table in the database. The column is of type "nvarchar(max)" and allows null values. This is done using Entity Framework Core's **AddColumn** method in a migration, allowing the storage of team codes for users. This has currently not been migrated into the DB due to issues with the Entity Framework and the project build.

Migration Code (Drop Column):

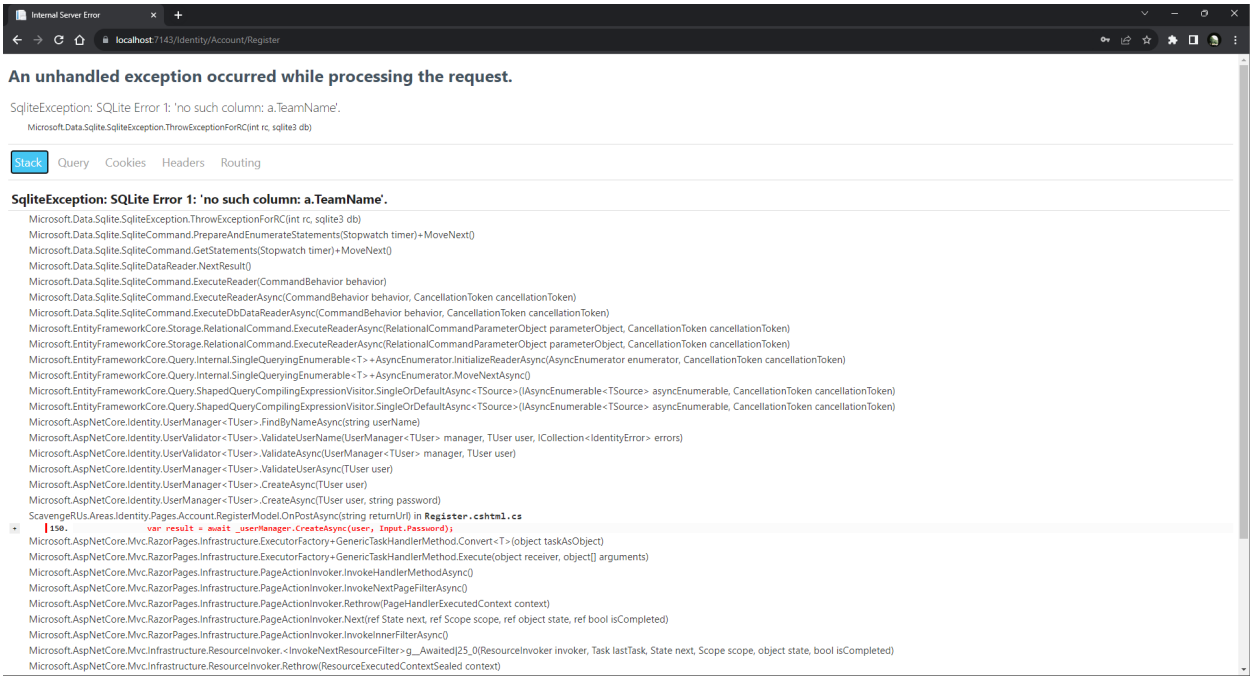
- **Functionality:** This code removes the "TeamCode" column from the "AspNetUsers" table in the database. This is done using Entity Framework Core's **DropColumn** method in a migration, indicating the removal of the specified column. This has currently not been migrated into the DB due to issues with the Entity Framework and the project build.

```
migrationBuilder.AddColumn<string>(
                name: "TeamCode",
                table: "AspNetUsers",
                type: "nvarchar(max)",
                nullable: true);

migrationBuilder.DropColumn(
                name: "TeamCode",
                table: "AspNetUsers");
```

5. **Testing:**

   For the testing of this task I attempted to register as a new user, which resulted in an unhandled exception due to the DB migration not being migrated.



I also attempted to create a new user on the admin side which resulted in the same error as soon as I logged in as an admin.