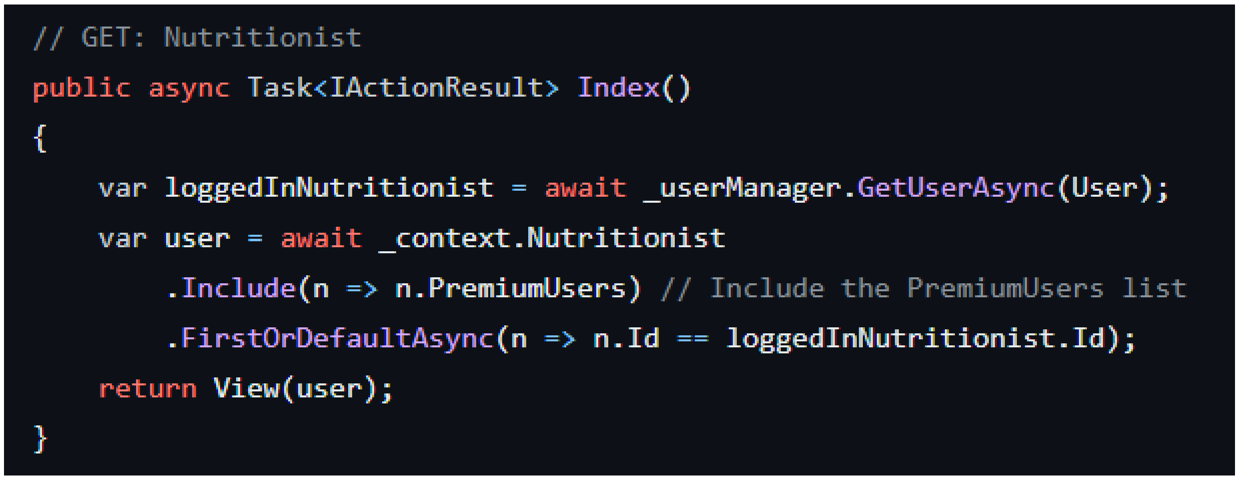
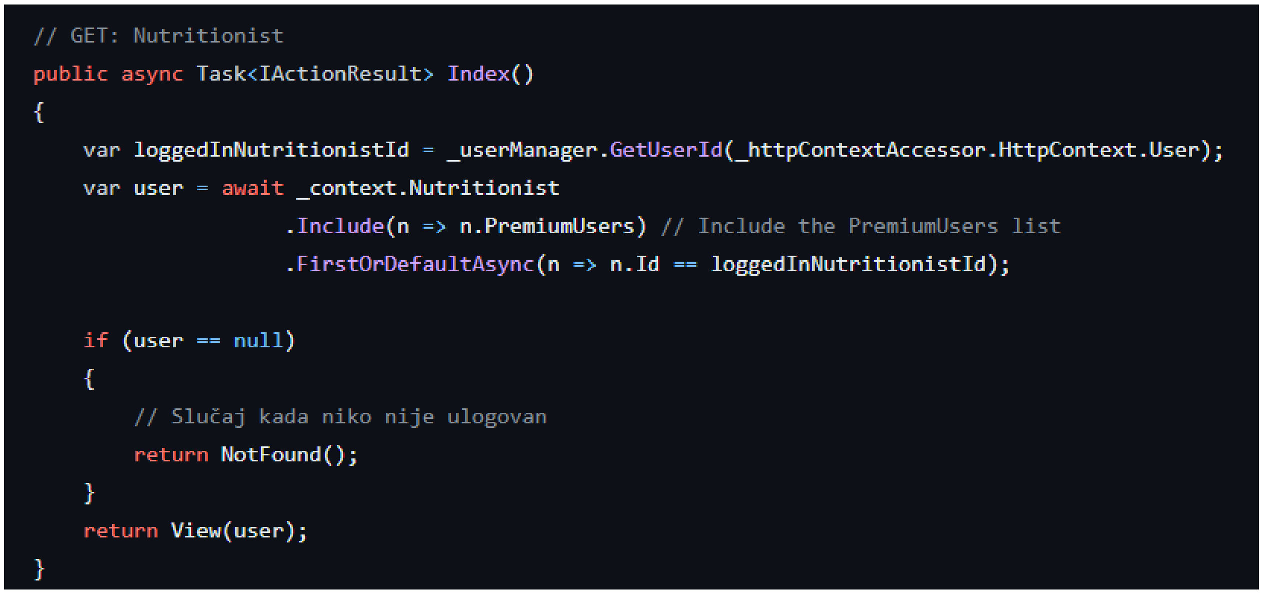
Dodan slučaj kada niko nije ulogovan za index – NutritionistController:

Prije:



```
// GET: Nutritionist
public async Task<IActionResult> Index()
{
    var loggedInNutritionist = await _userManager.GetUserAsync(User);
    var user = await _context.Nutritionist
        .Include(n => n.PremiumUsers) // Include the PremiumUsers list
        .FirstOrDefaultAsync(n => n.Id == loggedInNutritionist.Id);
    return View(user);
}
```

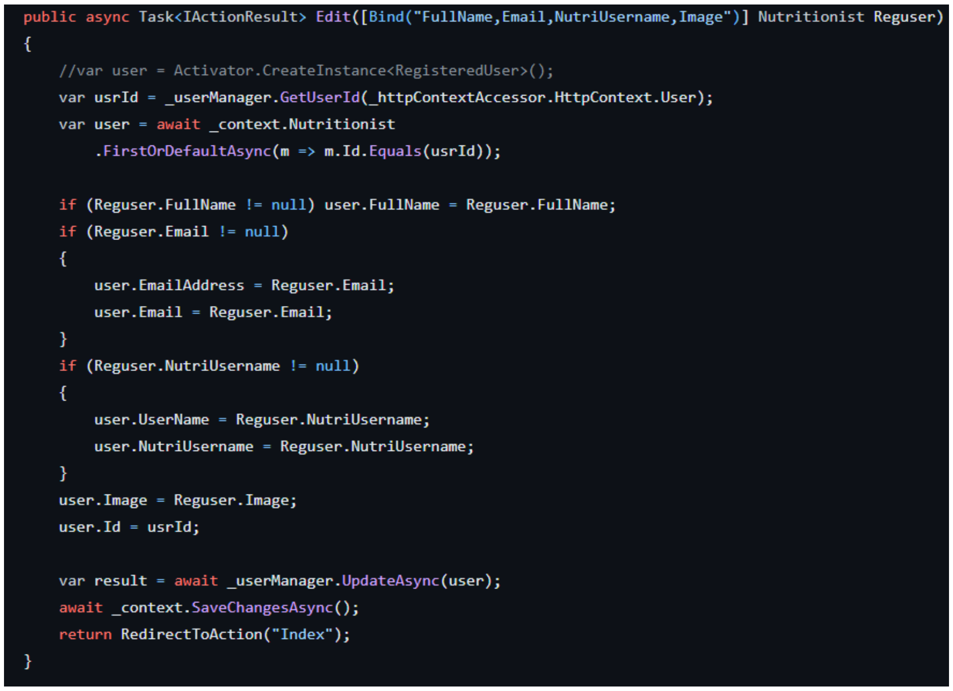
Poslije:



```
// GET: Nutritionist
public async Task<IActionResult> Index()
{
    var loggedInNutritionistId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
    var user = await _context.Nutritionist
                    .Include(n => n.PremiumUsers) // Include the PremiumUsers list
                    .FirstOrDefaultAsync(n => n.Id == loggedInNutritionistId);

    if (user == null)
    {
        // Slučaj kada niko nije ulogovan
        return NotFound();
    }
    return View(user);
}
```

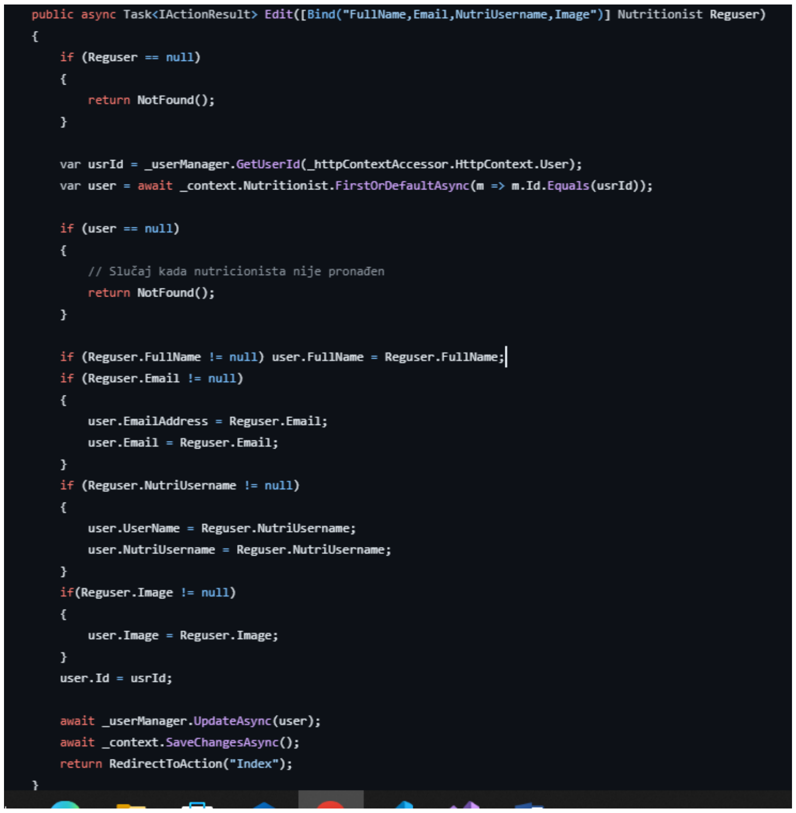
Provjera za null vrijednosti – Edit – NutritionistController:

Prije:



```
public async Task<IActionResult> Edit([Bind("FullName,Email,NutriUsername,Image")] Nutritionist Reguser)
{
    //var user = Activator.CreateInstance<RegisteredUser>();
    var usrId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
    var user = await _context.Nutritionist
        .FirstOrDefaultAsync(m => m.Id.Equals(usrId));

    if (Reguser.FullName != null) user.FullName = Reguser.FullName;
    if (Reguser.Email != null)
    {
        user.EmailAddress = Reguser.Email;
        user.Email = Reguser.Email;
    }
    if (Reguser.NutriUsername != null)
    {
        user.UserName = Reguser.NutriUsername;
        user.NutriUsername = Reguser.NutriUsername;
    }
    user.Image = Reguser.Image;
    user.Id = usrId;

    var result = await _userManager.UpdateAsync(user);
    await _context.SaveChangesAsync();
    return RedirectToAction("Index");
}
```

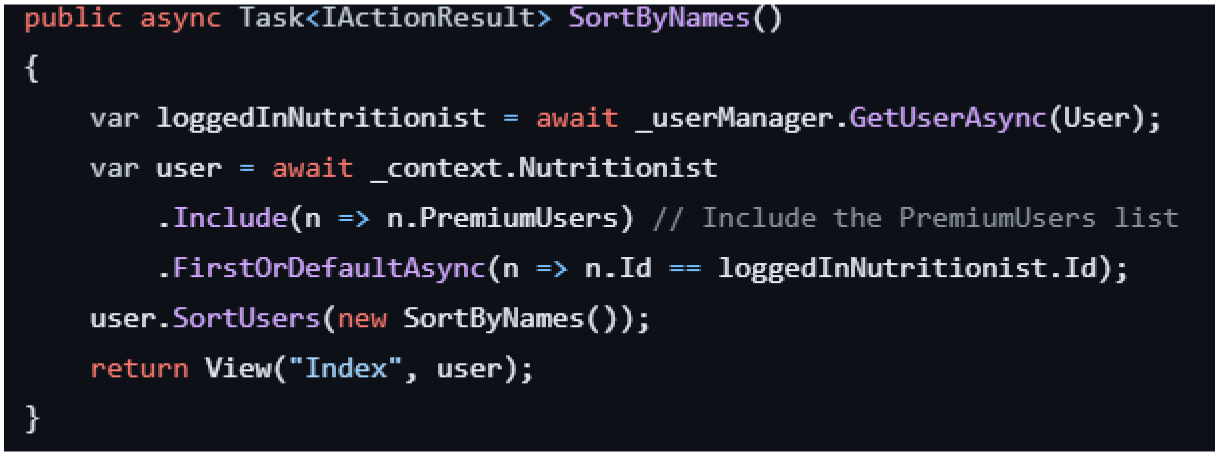
Poslije:



```
public async Task<IActionResult> Edit([Bind("FullName,Email,NutriUsername,Image")] Nutritionist Reguser)
{
    if (Reguser == null)
    {
        return NotFound();
    }

    var usrId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
    var user = await _context.Nutritionist.FirstOrDefaultAsync(m => m.Id.Equals(usrId));

    if (user == null)
    {
        // Slučaj kada nutricionista nije pronađen
        return NotFound();
    }

    if (Reguser.FullName != null) user.FullName = Reguser.FullName;
    if (Reguser.Email != null)
    {
        user.EmailAddress = Reguser.Email;
        user.Email = Reguser.Email;
    }
    if (Reguser.NutriUsername != null)
    {
        user.UserName = Reguser.NutriUsername;
        user.NutriUsername = Reguser.NutriUsername;
    }
    if(Reguser.Image != null)
    {
        user.Image = Reguser.Image;
    }
    user.Id = usrId;

    await _userManager.UpdateAsync(user);
    await _context.SaveChangesAsync();
    return RedirectToAction("Index");
}
```

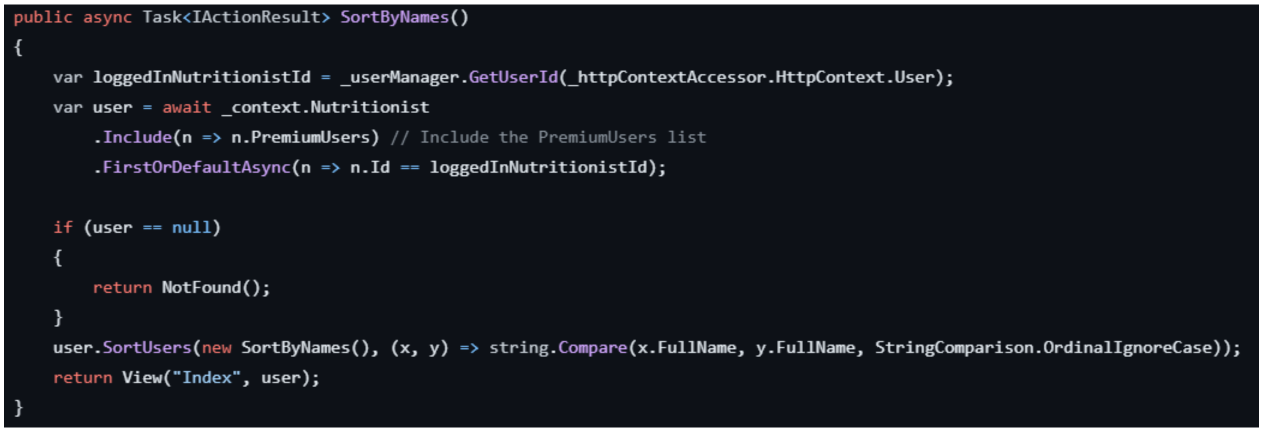
Ispravke u SortByNames – NutritionistController:

Prije:



```
public async Task<IActionResult> SortByNames()
{
    var loggedInNutritionist = await _userManager.GetUserAsync(User);
    var user = await _context.Nutritionist
        .Include(n => n.PremiumUsers) // Include the PremiumUsers list
        .FirstOrDefaultAsync(n => n.Id == loggedInNutritionist.Id);
    user.SortUsers(new SortByNames());
    return View("Index", user);
}
```

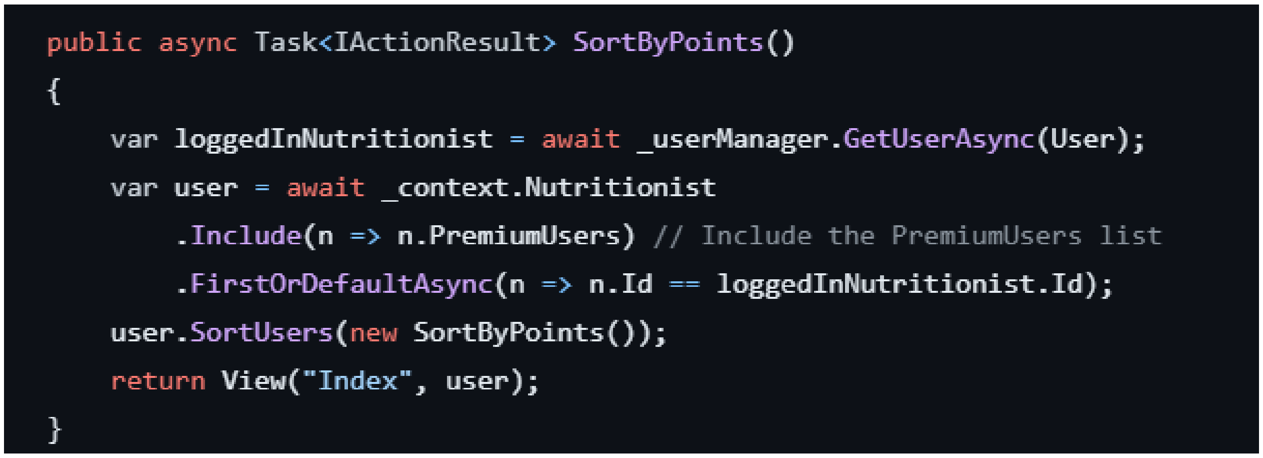
Poslije:



```
public async Task<IActionResult> SortByNames()
{
    var loggedInNutritionistId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
    var user = await _context.Nutritionist
        .Include(n => n.PremiumUsers) // Include the PremiumUsers list
        .FirstOrDefaultAsync(n => n.Id == loggedInNutritionistId);

    if (user == null)
    {
        return NotFound();
    }
    user.SortUsers(new SortByNames(), (x, y) => string.Compare(x.FullName, y.FullName, StringComparison.OrdinalIgnoreCase));
    return View("Index", user);
}
```

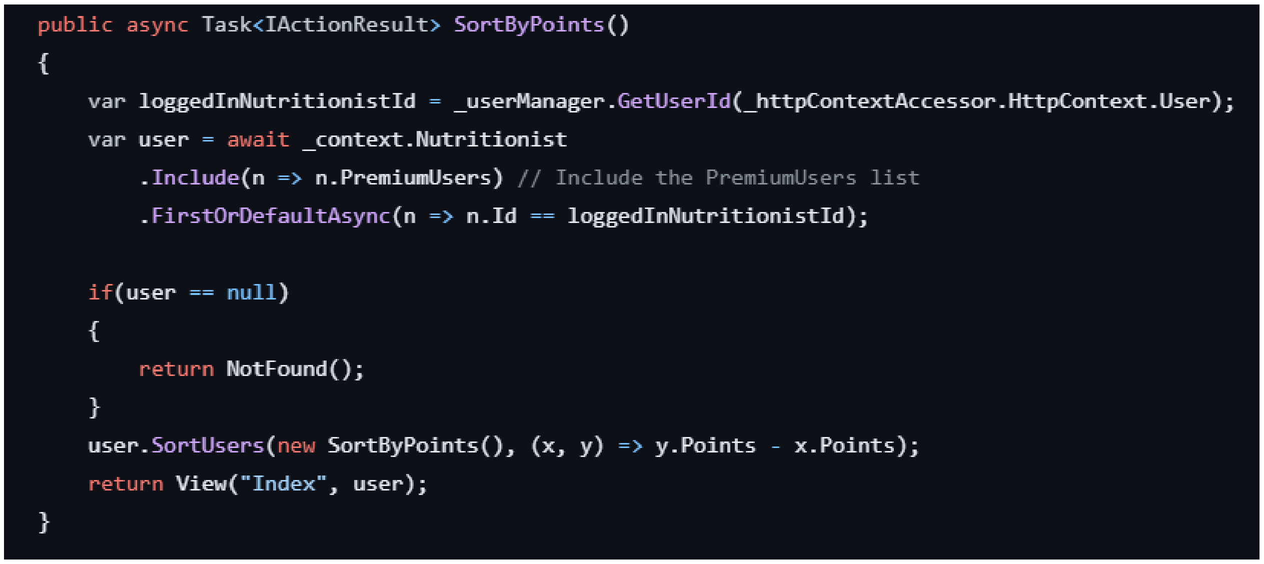
Ispravke u SortByPoints – NutritionistController:

Prije:



```
public async Task<IActionResult> SortByPoints()
{
    var loggedInNutritionist = await _userManager.GetUserAsync(User);
    var user = await _context.Nutritionist
        .Include(n => n.PremiumUsers) // Include the PremiumUsers list
        .FirstOrDefaultAsync(n => n.Id == loggedInNutritionist.Id);
    user.SortUsers(new SortByPoints());
    return View("Index", user);
}
```

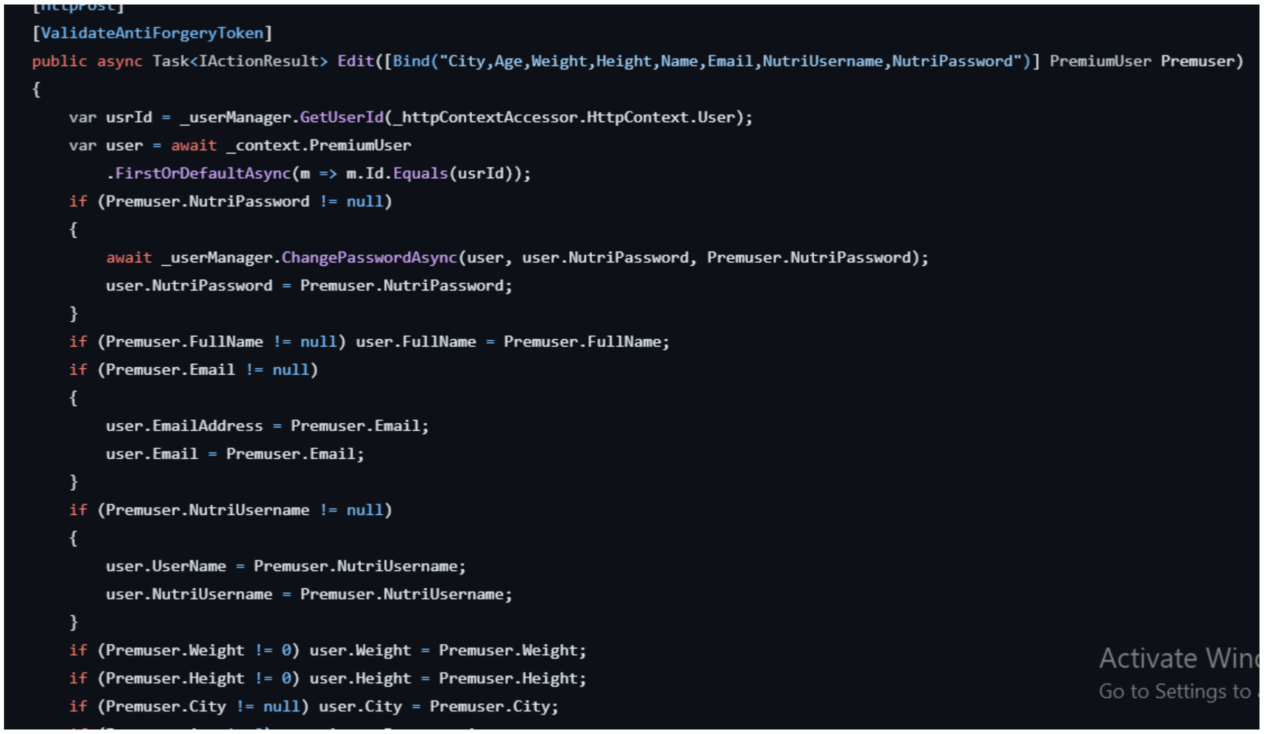
Poslije:



```
public async Task<IActionResult> SortByPoints()
{
    var loggedInNutritionistId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
    var user = await _context.Nutritionist
        .Include(n => n.PremiumUsers) // Include the PremiumUsers list
        .FirstOrDefaultAsync(n => n.Id == loggedInNutritionistId);

    if(user == null)
    {
        return NotFound();
    }
    user.SortUsers(new SortByPoints(), (x, y) => y.Points - x.Points);
    return View("Index", user);
}
```

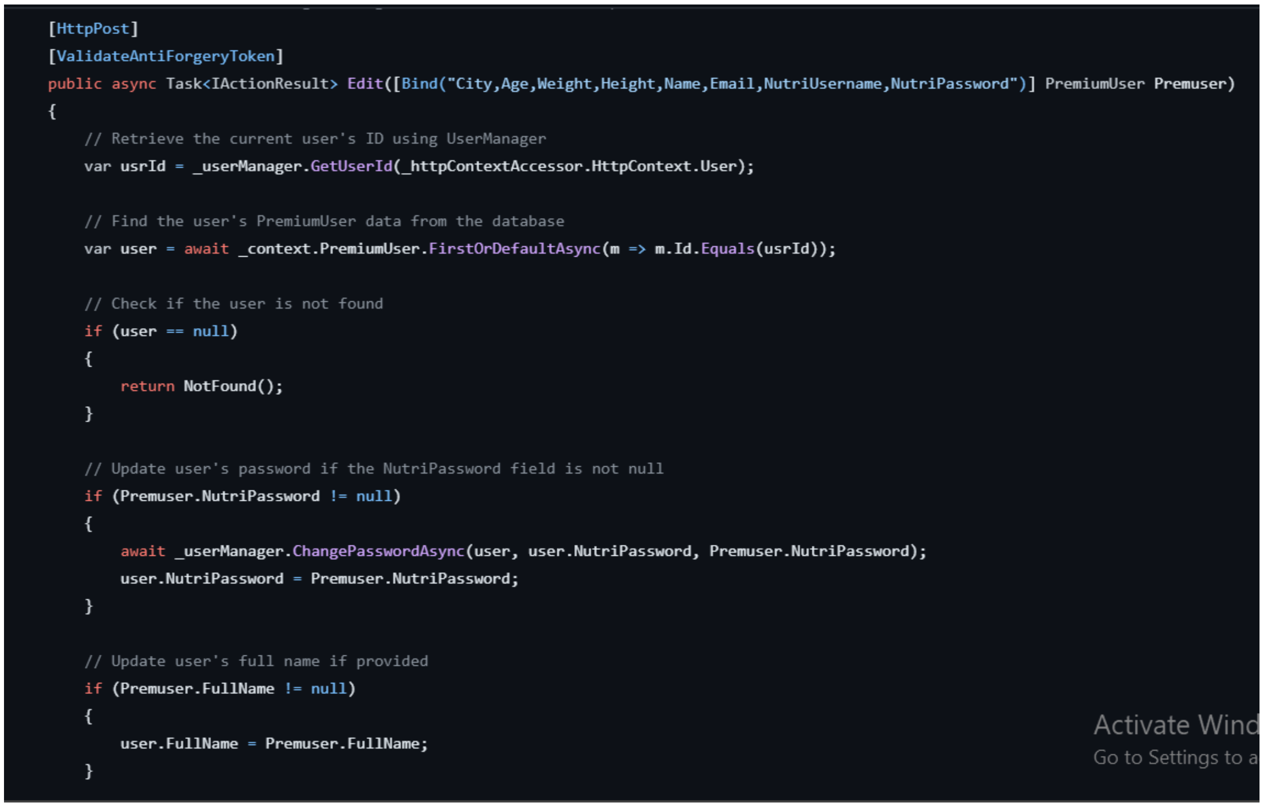
Nedostatak promjene validnosti u metodama – PremiumUsersController:

Prije:



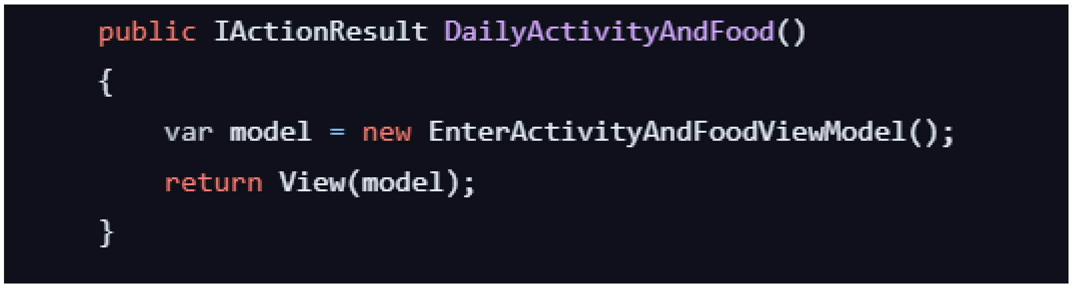
```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit([Bind("City,Age,Weight,Height,Name,Email,NutriUsername,NutriPassword")] PremiumUser Premuser)
{
    var usrId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
    var user = await _context.PremiumUser
        .FirstOrDefaultAsync(m => m.Id.Equals(usrId));
    if (Premuser.NutriPassword != null)
    {
        await _userManager.ChangePasswordAsync(user, user.NutriPassword, Premuser.NutriPassword);
        user.NutriPassword = Premuser.NutriPassword;
    }
    if (Premuser.FullName != null) user.FullName = Premuser.FullName;
    if (Premuser.Email != null)
    {
        user.EmailAddress = Premuser.Email;
        user.Email = Premuser.Email;
    }
    if (Premuser.NutriUsername != null)
    {
        user.UserName = Premuser.NutriUsername;
        user.NutriUsername = Premuser.NutriUsername;
    }
    if (Premuser.Weight != 0) user.Weight = Premuser.Weight;
    if (Premuser.Height != 0) user.Height = Premuser.Height;
    if (Premuser.City != null) user.City = Premuser.City;
```

Poslije:



```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit([Bind("City,Age,Weight,Height,Name,Email,NutriUsername,NutriPassword")] PremiumUser Premuser)
{
    // Retrieve the current user's ID using UserManager
    var usrId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);

    // Find the user's PremiumUser data from the database
    var user = await _context.PremiumUser.FirstOrDefaultAsync(m => m.Id.Equals(usrId));

    // Check if the user is not found
    if (user == null)
    {
        return NotFound();
    }

    // Update user's password if the NutriPassword field is not null
    if (Premuser.NutriPassword != null)
    {
        await _userManager.ChangePasswordAsync(user, user.NutriPassword, Premuser.NutriPassword);
        user.NutriPassword = Premuser.NutriPassword;
    }

    // Update user's full name if provided
    if (Premuser.FullName != null)
    {
        user.FullName = Premuser.FullName;
    }
```

Nedostižan kod – PremiumUsersController:

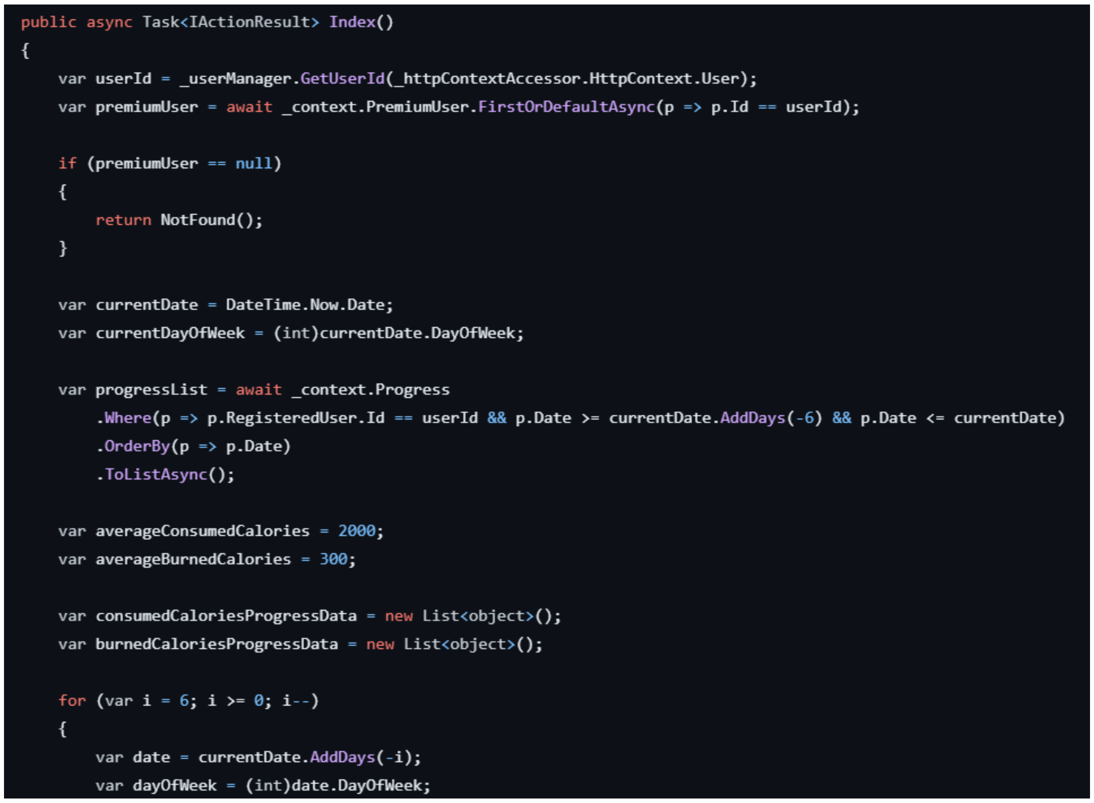


```
public IActionResult DailyActivityAndFood()
{
    var model = new EnterActivityAndFoodViewModel();
    return View(model);
}
```

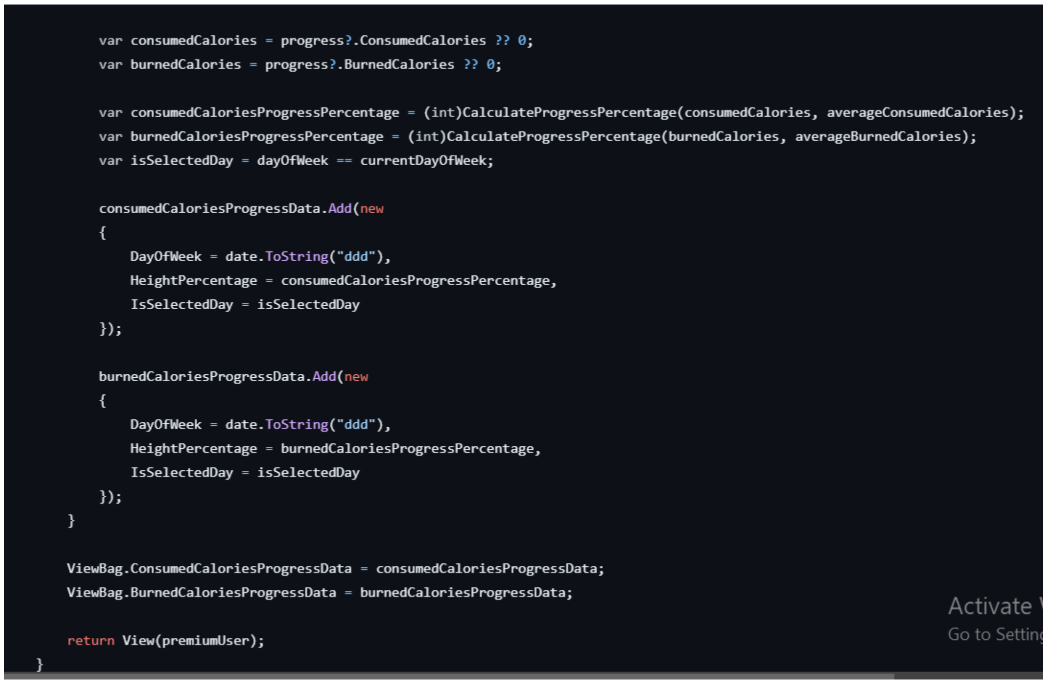
Obrisano

Refaktoring index I details metoda – PremiumUsersController

Prije:



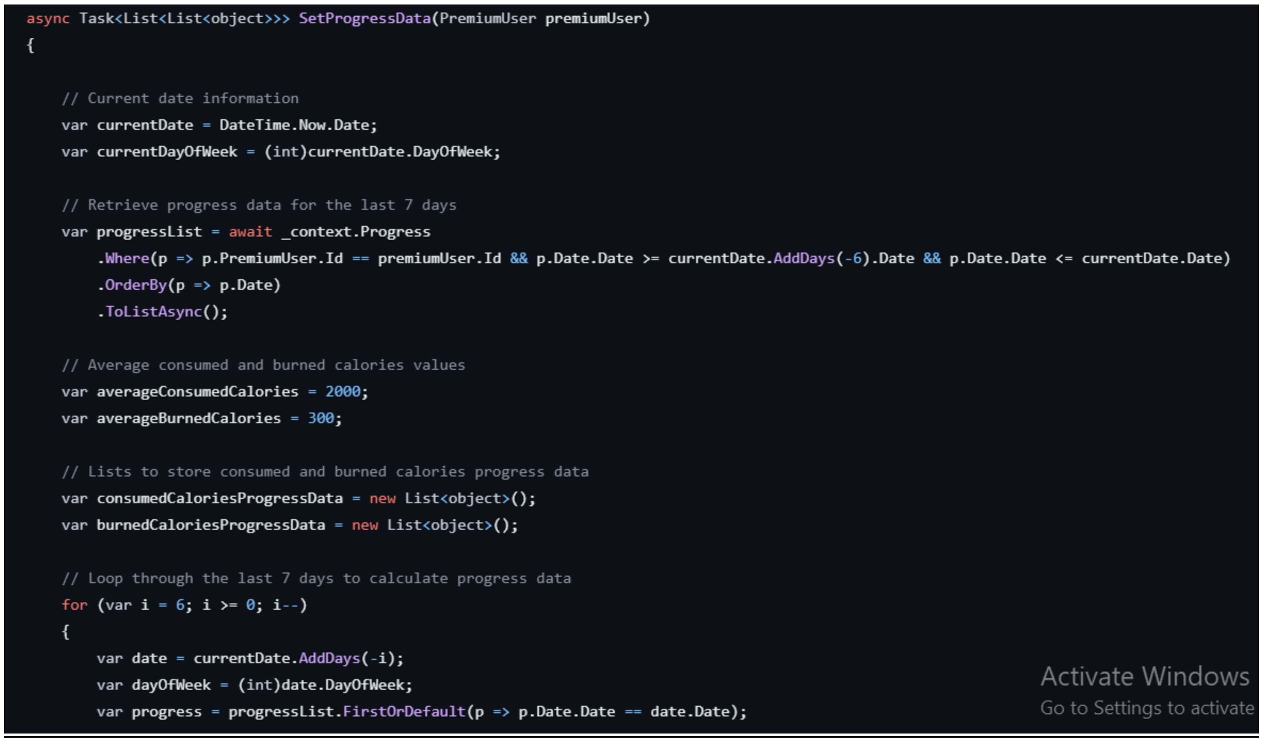
```
public async Task<IActionResult> Index()
{
    var userId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
    var premiumUser = await _context.PremiumUser.FirstOrDefaultAsync(p => p.Id == userId);

    if (premiumUser == null)
    {
        return NotFound();
    }

    var currentDate = DateTime.Now.Date;
    var currentDayOfWeek = (int)currentDate.DayOfWeek;

    var progressList = await _context.Progress
        .Where(p => p.RegisteredUser.Id == userId && p.Date >= currentDate.AddDays(-6) && p.Date <= currentDate)
        .OrderBy(p => p.Date)
        .ToListAsync();

    var averageConsumedCalories = 2000;
    var averageBurnedCalories = 300;

    var consumedCaloriesProgressData = new List<object>();
    var burnedCaloriesProgressData = new List<object>();

    for (var i = 6; i >= 0; i--)
    {
        var date = currentDate.AddDays(-i);
        var dayOfWeek = (int)date.DayOfWeek;
```



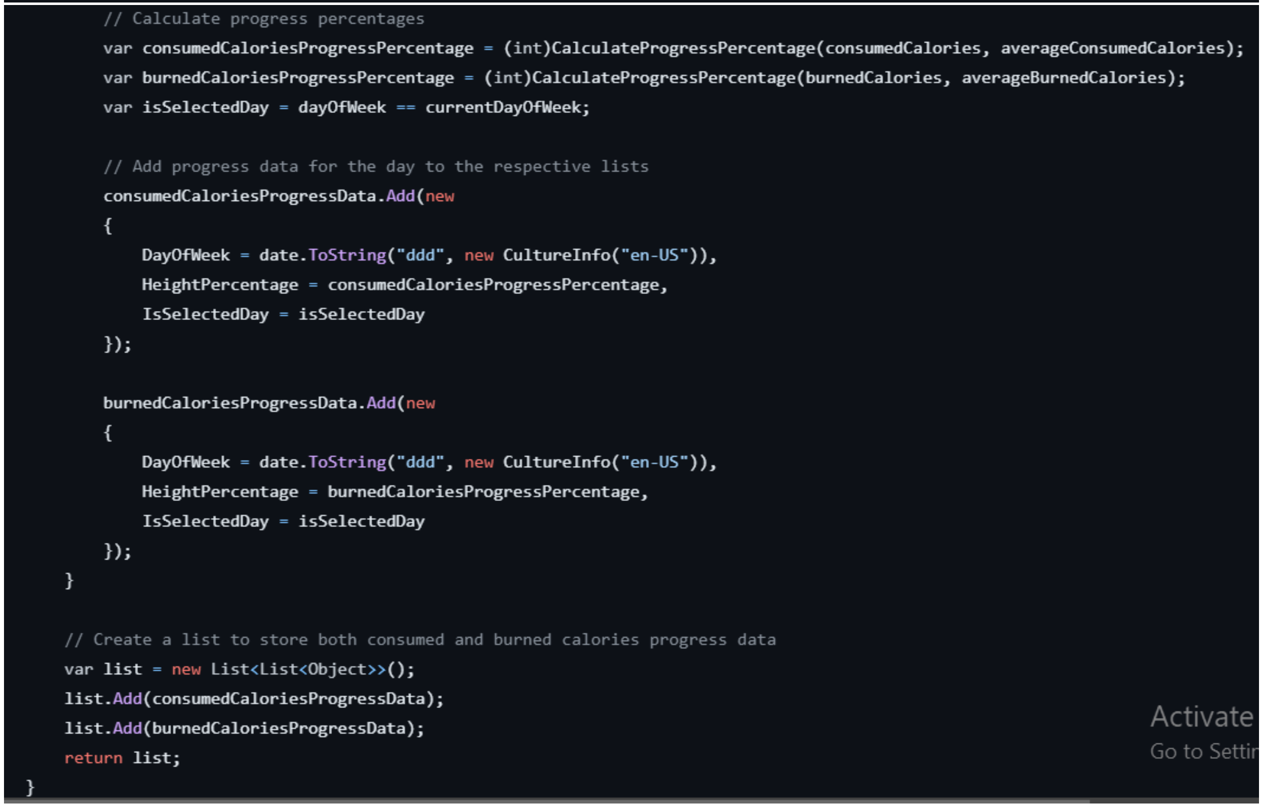
```
        var consumedCalories = progress?.ConsumedCalories ?? 0;
        var burnedCalories = progress?.BurnedCalories ?? 0;

        var consumedCaloriesProgressPercentage = (int)CalculateProgressPercentage(consumedCalories, averageConsumedCalories);
        var burnedCaloriesProgressPercentage = (int)CalculateProgressPercentage(burnedCalories, averageBurnedCalories);
        var isSelectedDay = dayOfWeek == currentDayOfWeek;

        consumedCaloriesProgressData.Add(new
        {
            DayOfWeek = date.ToString("ddd"),
            HeightPercentage = consumedCaloriesProgressPercentage,
            IsSelectedDay = isSelectedDay
        });

        burnedCaloriesProgressData.Add(new
        {
            DayOfWeek = date.ToString("ddd"),
            HeightPercentage = burnedCaloriesProgressPercentage,
            IsSelectedDay = isSelectedDay
        });
    }

    ViewBag.ConsumedCaloriesProgressData = consumedCaloriesProgressData;
    ViewBag.BurnedCaloriesProgressData = burnedCaloriesProgressData;

    return View(premiumUser);
}
```

-isto I u details.

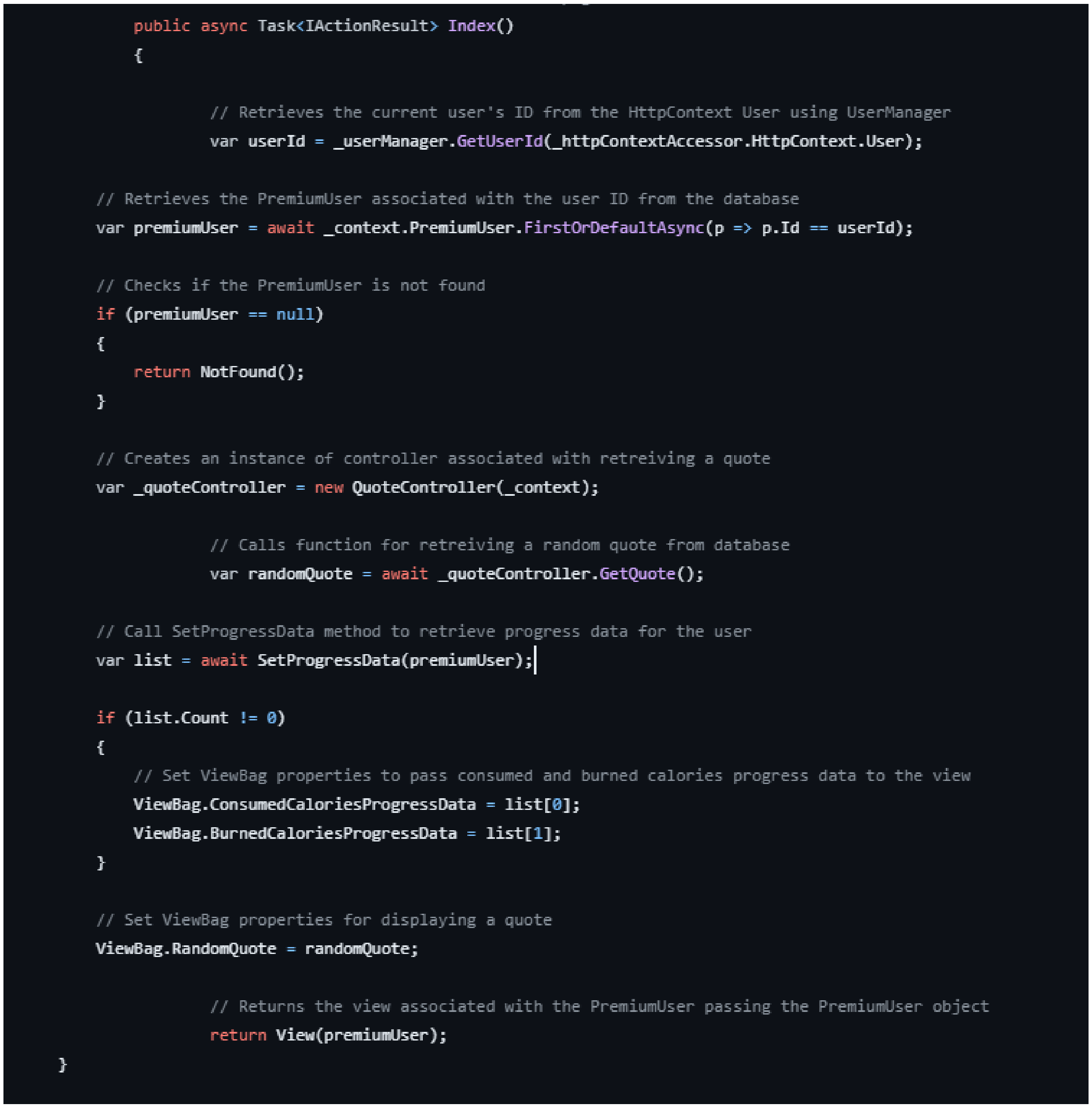
Poslije:



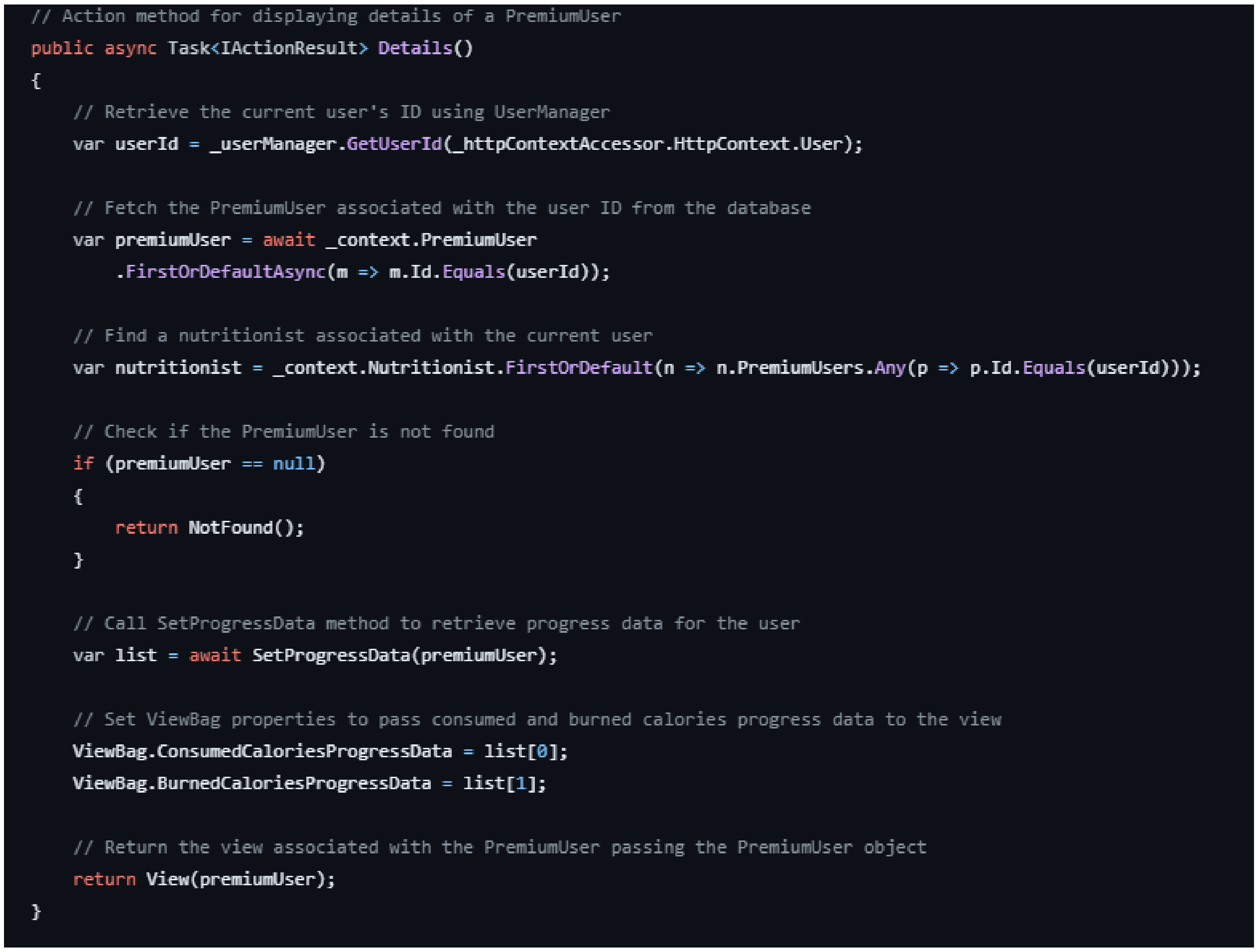
```
async Task<List<List<object>>> SetProgressData(PremiumUser premiumUser)
{

    // Current date information
    var currentDate = DateTime.Now.Date;
    var currentDayOfWeek = (int)currentDate.DayOfWeek;

    // Retrieve progress data for the last 7 days
    var progressList = await _context.Progress
        .Where(p => p.PremiumUser.Id == premiumUser.Id && p.Date.Date >= currentDate.AddDays(-6).Date && p.Date.Date <= currentDate.Date)
        .OrderBy(p => p.Date)
        .ToListAsync();

    // Average consumed and burned calories values
    var averageConsumedCalories = 2000;
    var averageBurnedCalories = 300;

    // Lists to store consumed and burned calories progress data
    var consumedCaloriesProgressData = new List<object>();
    var burnedCaloriesProgressData = new List<object>();

    // Loop through the last 7 days to calculate progress data
    for (var i = 6; i >= 0; i--)
    {
        var date = currentDate.AddDays(-i);
        var dayOfWeek = (int)date.DayOfWeek;
        var progress = progressList.FirstOrDefault(p => p.Date.Date == date.Date);
```



```
        // Calculate progress percentages
        var consumedCaloriesProgressPercentage = (int)CalculateProgressPercentage(consumedCalories, averageConsumedCalories);
        var burnedCaloriesProgressPercentage = (int)CalculateProgressPercentage(burnedCalories, averageBurnedCalories);
        var isSelectedDay = dayOfWeek == currentDayOfWeek;

        // Add progress data for the day to the respective lists
        consumedCaloriesProgressData.Add(new
        {
            DayOfWeek = date.ToString("ddd", new CultureInfo("en-US")),
            HeightPercentage = consumedCaloriesProgressPercentage,
            IsSelectedDay = isSelectedDay
        });

        burnedCaloriesProgressData.Add(new
        {
            DayOfWeek = date.ToString("ddd", new CultureInfo("en-US")),
            HeightPercentage = burnedCaloriesProgressPercentage,
            IsSelectedDay = isSelectedDay
        });
    }

    // Create a list to store both consumed and burned calories progress data
    var list = new List<List<Object>>();
    list.Add(consumedCaloriesProgressData);
    list.Add(burnedCaloriesProgressData);
    return list;
}
```



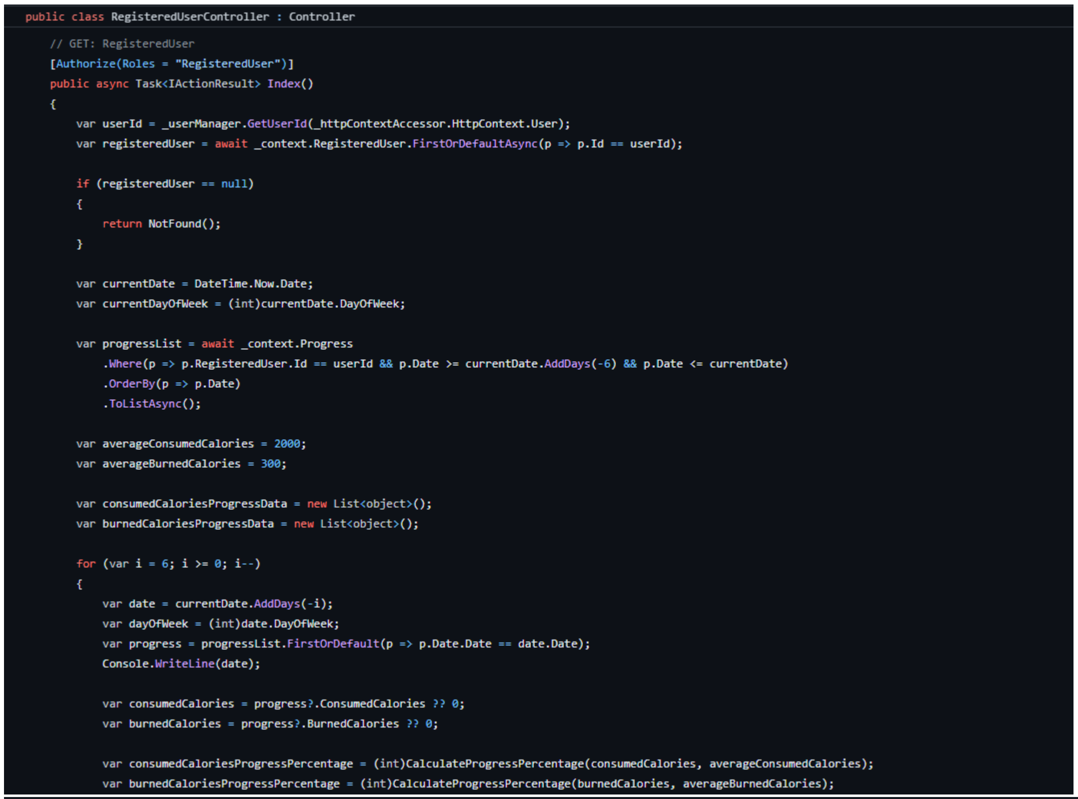
```
        public async Task<IActionResult> Index()
        {

                // Retrieves the current user's ID from the HttpContext User using UserManager
                var userId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);

    // Retrieves the PremiumUser associated with the user ID from the database
    var premiumUser = await _context.PremiumUser.FirstOrDefaultAsync(p => p.Id == userId);

    // Checks if the PremiumUser is not found
    if (premiumUser == null)
    {
        return NotFound();
    }

    // Creates an instance of controller associated with retreiving a quote
    var _quoteController = new QuoteController(_context);

                // Calls function for retreiving a random quote from database
                var randomQuote = await _quoteController.GetQuote();

    // Call SetProgressData method to retrieve progress data for the user
    var list = await SetProgressData(premiumUser);

    if (list.Count != 0)
    {
        // Set ViewBag properties to pass consumed and burned calories progress data to the view
        ViewBag.ConsumedCaloriesProgressData = list[0];
        ViewBag.BurnedCaloriesProgressData = list[1];
    }

    // Set ViewBag properties for displaying a quote
    ViewBag.RandomQuote = randomQuote;

                // Returns the view associated with the PremiumUser passing the PremiumUser object
                return View(premiumUser);
}
```



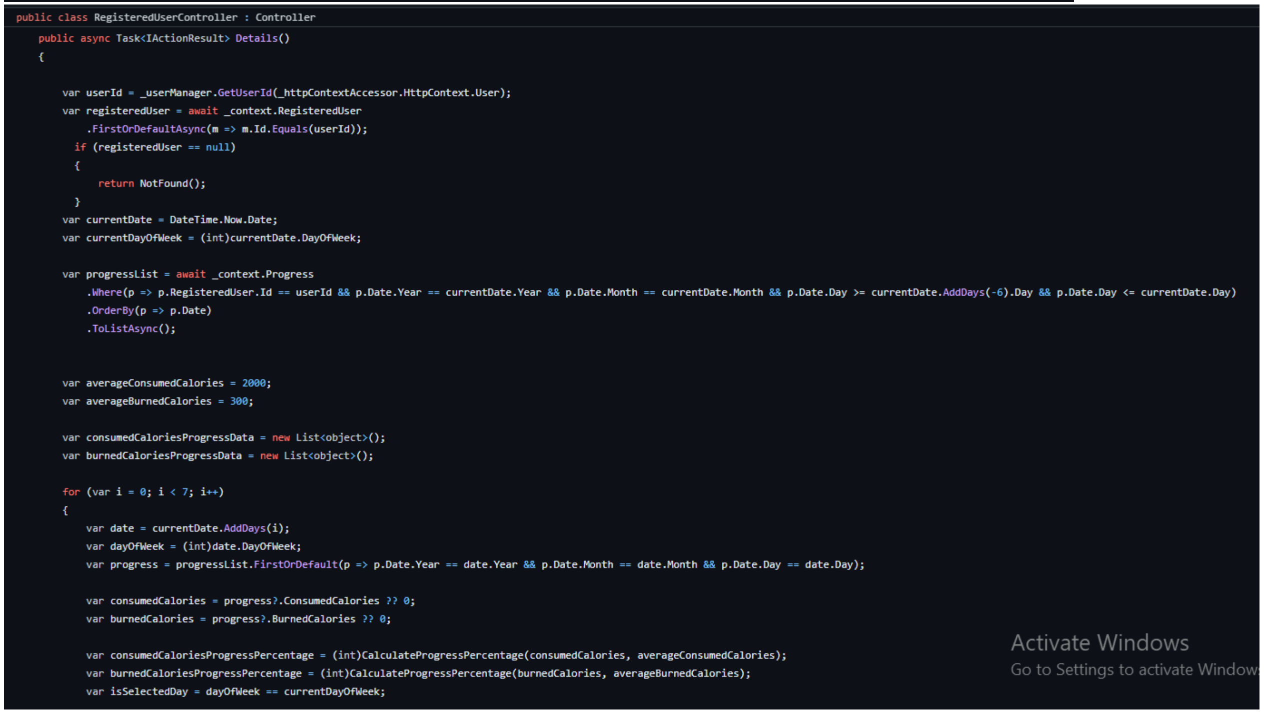
```csharp
// Action method for displaying details of a PremiumUser
public async Task<IActionResult> Details()
{
    // Retrieve the current user's ID using UserManager
    var userId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);

    // Fetch the PremiumUser associated with the user ID from the database
    var premiumUser = await _context.PremiumUser
        .FirstOrDefaultAsync(m => m.Id.Equals(userId));

    // Find a nutritionist associated with the current user
    var nutritionist = _context.Nutritionist.FirstOrDefault(n => n.PremiumUsers.Any(p => p.Id.Equals(userId)));

    // Check if the PremiumUser is not found
    if (premiumUser == null)
    {
        return NotFound();
    }

    // Call SetProgressData method to retrieve progress data for the user
    var list = await SetProgressData(premiumUser);

    // Set ViewBag properties to pass consumed and burned calories progress data to the view
    ViewBag.ConsumedCaloriesProgressData = list[0];
    ViewBag.BurnedCaloriesProgressData = list[1];

    // Return the view associated with the PremiumUser passing the PremiumUser object
    return View(premiumUser);
}
```

Ponovljen kod – RegisteredUserController – index I details:

Prije:

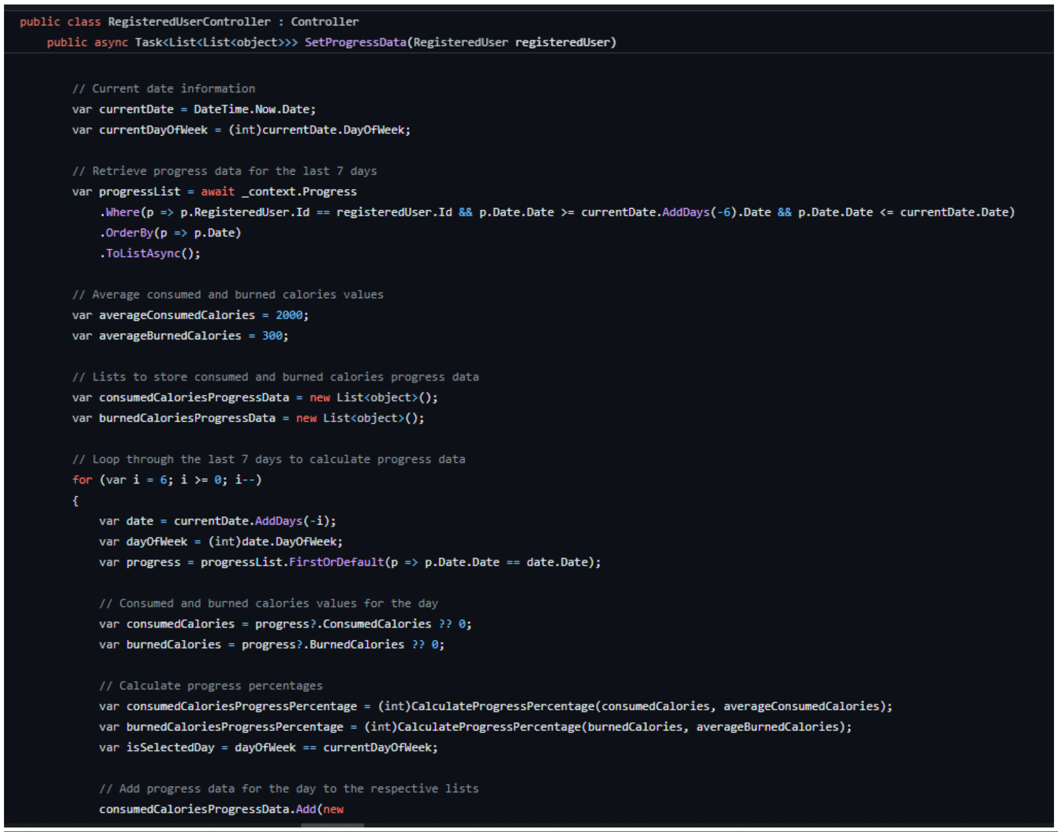


```csharp
public class RegisteredUserController : Controller

    // GET: RegisteredUser
    [Authorize(Roles = "RegisteredUser")]
    public async Task<IActionResult> Index()
    {
        var userId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
        var registeredUser = await _context.RegisteredUser.FirstOrDefaultAsync(p => p.Id == userId);

        if (registeredUser == null)
        {
            return NotFound();
        }

        var currentDate = DateTime.Now.Date;
        var currentDayOfWeek = (int)currentDate.DayOfWeek;

        var progressList = await _context.Progress
            .Where(p => p.RegisteredUser.Id == userId && p.Date >= currentDate.AddDays(-6) && p.Date <= currentDate)
            .OrderBy(p => p.Date)
            .ToListAsync();

        var averageConsumedCalories = 2000;
        var averageBurnedCalories = 300;

        var consumedCaloriesProgressData = new List<object>();
        var burnedCaloriesProgressData = new List<object>();

        for (var i = 6; i >= 0; i--)
        {
            var date = currentDate.AddDays(-i);
            var dayOfWeek = (int)date.DayOfWeek;
            var progress = progressList.FirstOrDefault(p => p.Date.Date == date.Date);
            Console.WriteLine(date);

            var consumedCalories = progress?.ConsumedCalories ?? 0;
            var burnedCalories = progress?.BurnedCalories ?? 0;

            var consumedCaloriesProgressPercentage = (int)CalculateProgressPercentage(consumedCalories, averageConsumedCalories);
            var burnedCaloriesProgressPercentage = (int)CalculateProgressPercentage(burnedCalories, averageBurnedCalories);
```

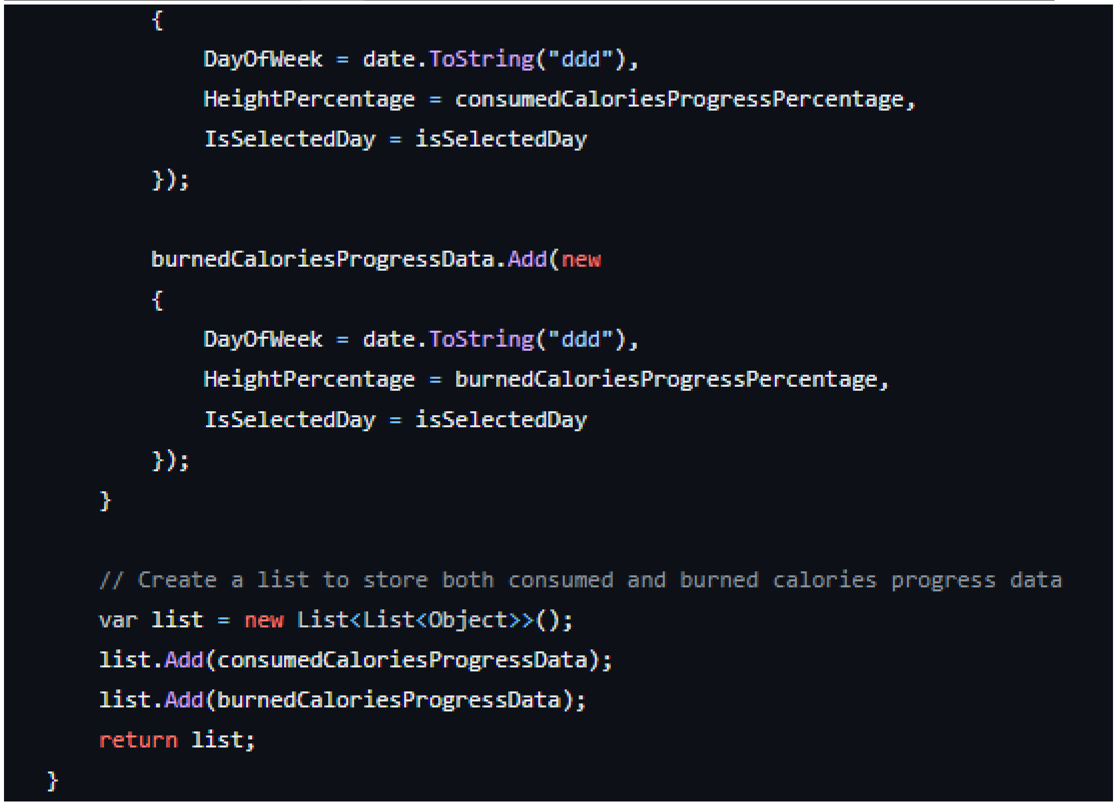


```csharp
public class RegisteredUserController : Controller
    public async Task<IActionResult> Details()
    {

        var userId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
        var registeredUser = await _context.RegisteredUser
            .FirstOrDefaultAsync(m => m.Id.Equals(userId));
          if (registeredUser == null)
          {
              return NotFound();
          }
        var currentDate = DateTime.Now.Date;
        var currentDayOfWeek = (int)currentDate.DayOfWeek;

        var progressList = await _context.Progress
            .Where(p => p.RegisteredUser.Id == userId && p.Date.Year == currentDate.Year && p.Date.Month == currentDate.Month && p.Date.Day >= currentDate.AddDays(-6).Day && p.Date.Day <= currentDate.Day)
            .OrderBy(p => p.Date)
            .ToListAsync();


        var averageConsumedCalories = 2000;
        var averageBurnedCalories = 300;

        var consumedCaloriesProgressData = new List<object>();
        var burnedCaloriesProgressData = new List<object>();

        for (var i = 0; i < 7; i++)
        {
            var date = currentDate.AddDays(i);
            var dayOfWeek = (int)date.DayOfWeek;
            var progress = progressList.FirstOrDefault(p => p.Date.Year == date.Year && p.Date.Month == date.Month && p.Date.Day == date.Day);

            var consumedCalories = progress?.ConsumedCalories ?? 0;
            var burnedCalories = progress?.BurnedCalories ?? 0;

            var consumedCaloriesProgressPercentage = (int)CalculateProgressPercentage(consumedCalories, averageConsumedCalories);
            var burnedCaloriesProgressPercentage = (int)CalculateProgressPercentage(burnedCalories, averageBurnedCalories);
            var isSelectedDay = dayOfWeek == currentDayOfWeek;
```

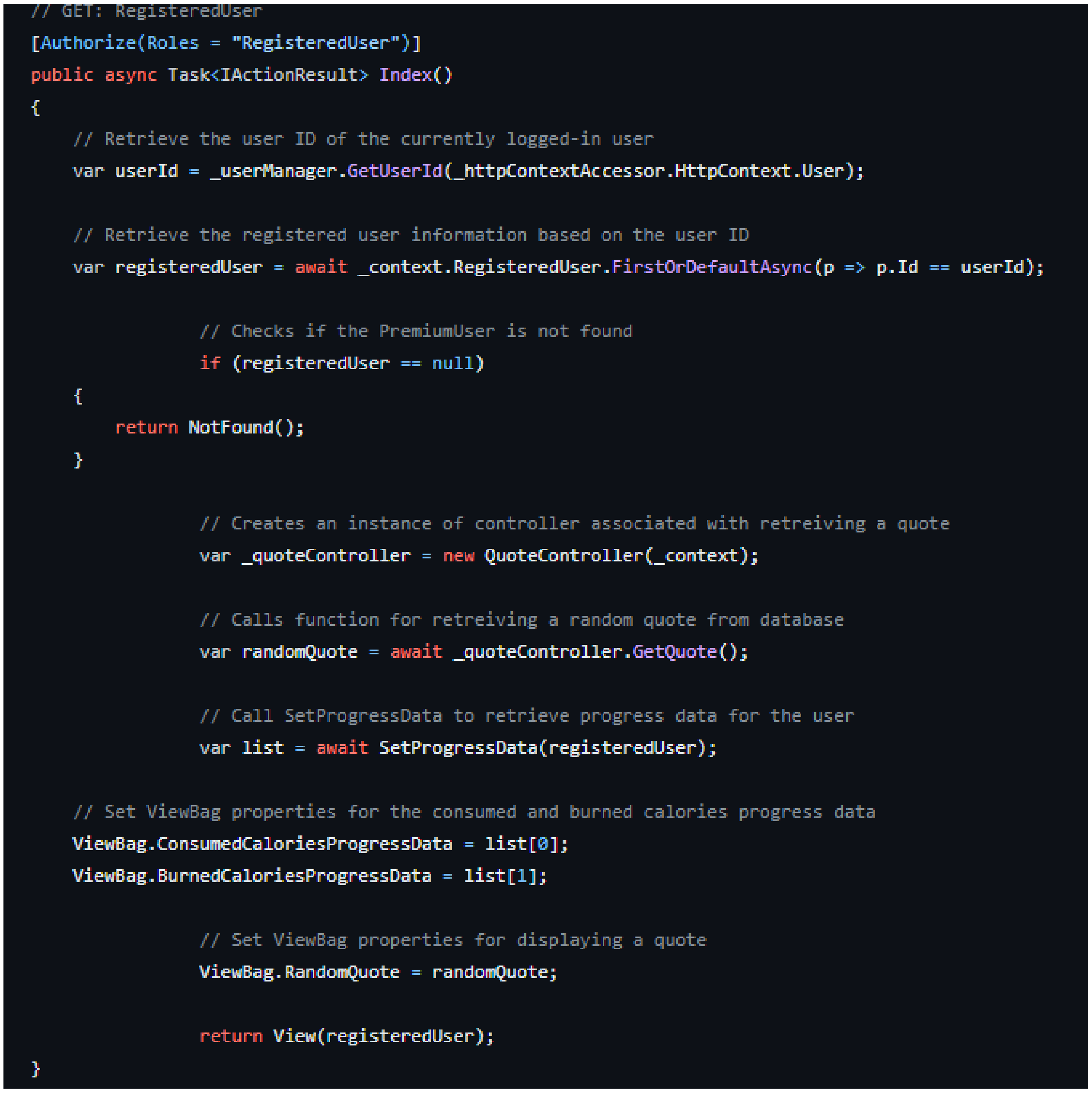
Poslije:



```
public class RegisteredUserController : Controller
    public async Task<List<List<object>>> SetProgressData(RegisteredUser registeredUser)

        // Current date information
        var currentDate = DateTime.Now.Date;
        var currentDayOfWeek = (int)currentDate.DayOfWeek;

        // Retrieve progress data for the last 7 days
        var progressList = await _context.Progress
            .Where(p => p.RegisteredUser.Id == registeredUser.Id && p.Date.Date >= currentDate.AddDays(-6).Date && p.Date.Date <= currentDate.Date)
            .OrderBy(p => p.Date)
            .ToListAsync();

        // Average consumed and burned calories values
        var averageConsumedCalories = 2000;
        var averageBurnedCalories = 300;

        // Lists to store consumed and burned calories progress data
        var consumedCaloriesProgressData = new List<object>();
        var burnedCaloriesProgressData = new List<object>();

        // Loop through the last 7 days to calculate progress data
        for (var i = 6; i >= 0; i--)
        {
            var date = currentDate.AddDays(-i);
            var dayOfWeek = (int)date.DayOfWeek;
            var progress = progressList.FirstOrDefault(p => p.Date.Date == date.Date);

            // Consumed and burned calories values for the day
            var consumedCalories = progress?.ConsumedCalories ?? 0;
            var burnedCalories = progress?.BurnedCalories ?? 0;

            // Calculate progress percentages
            var consumedCaloriesProgressPercentage = (int)CalculateProgressPercentage(consumedCalories, averageConsumedCalories);
            var burnedCaloriesProgressPercentage = (int)CalculateProgressPercentage(burnedCalories, averageBurnedCalories);
            var isSelectedDay = dayOfWeek == currentDayOfWeek;

            // Add progress data for the day to the respective lists
            consumedCaloriesProgressData.Add(new
```



```
            {
                DayOfWeek = date.ToString("ddd"),
                HeightPercentage = consumedCaloriesProgressPercentage,
                IsSelectedDay = isSelectedDay
            });

            burnedCaloriesProgressData.Add(new
            {
                DayOfWeek = date.ToString("ddd"),
                HeightPercentage = burnedCaloriesProgressPercentage,
                IsSelectedDay = isSelectedDay
            });
        }

        // Create a list to store both consumed and burned calories progress data
        var list = new List<List<Object>>();
        list.Add(consumedCaloriesProgressData);
        list.Add(burnedCaloriesProgressData);
        return list;
    }
```



```
// GET: RegisteredUser
[Authorize(Roles = "RegisteredUser")]
public async Task<IActionResult> Index()
{
    // Retrieve the user ID of the currently logged-in user
    var userId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);

    // Retrieve the registered user information based on the user ID
    var registeredUser = await _context.RegisteredUser.FirstOrDefaultAsync(p => p.Id == userId);

                // Checks if the PremiumUser is not found
                if (registeredUser == null)
    {
        return NotFound();
    }

                // Creates an instance of controller associated with retreiving a quote
                var _quoteController = new QuoteController(_context);

                // Calls function for retreiving a random quote from database
                var randomQuote = await _quoteController.GetQuote();

                // Call SetProgressData to retrieve progress data for the user
                var list = await SetProgressData(registeredUser);

    // Set ViewBag properties for the consumed and burned calories progress data
    ViewBag.ConsumedCaloriesProgressData = list[0];
    ViewBag.BurnedCaloriesProgressData = list[1];

                // Set ViewBag properties for displaying a quote
                ViewBag.RandomQuote = randomQuote;

                return View(registeredUser);
}
```

```
[Authorize(Roles = "RegisteredUser")]
public async Task<IActionResult> Details()
{

    var userId = _userManager.GetUserId(_httpContextAccessor.HttpContext.User);
    var registeredUser = await _context.RegisteredUser
        .FirstOrDefaultAsync(m => m.Id.Equals(userId));
    if (registeredUser == null)
    {
        return NotFound();
    }

    // Call SetProgressData to retrieve progress data for the user
    var list = await SetProgressData(registeredUser);

    // Set ViewBag properties for the consumed and burned calories progress data
    ViewBag.ConsumedCaloriesProgressData = list[0];
    ViewBag.BurnedCaloriesProgressData = list[1];

    return View(registeredUser);
}
```

Uvijek uneseno 0 za cosumedCalories, Save – RegisteredUserController

Prije:

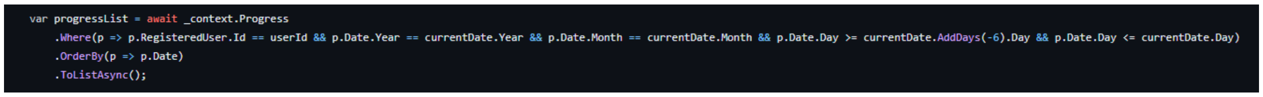
```
    var progress = new Progress
    {
        Date = DateTime.Now,
        BurnedCalories = 0,
        ConsumedCalories = 0,
        RegisteredUser = currentUser,
        PremiumUser = null
    };
```

Poslije:

```
var progress = new Progress {
    Date = DateTime.Now,
    BurnedCalories = 0,
    ConsumedCalories = consumedCalories,
    RegisteredUser = registeredUser,
    PremiumUser = premiumUser
};
```

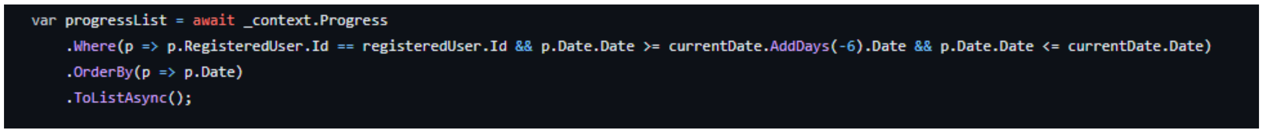
Neispravno prikazan progress za trenutni datum – RegisteredUserControler, index I details:

Prije:



```
var progressList = await _context.Progress
    .Where(p => p.RegisteredUser.Id == userId && p.Date.Year == currentDate.Year && p.Date.Month == currentDate.Month && p.Date.Day >= currentDate.AddDays(-6).Day && p.Date.Day <= currentDate.Day)
    .OrderBy(p => p.Date)
    .ToListAsync();
```

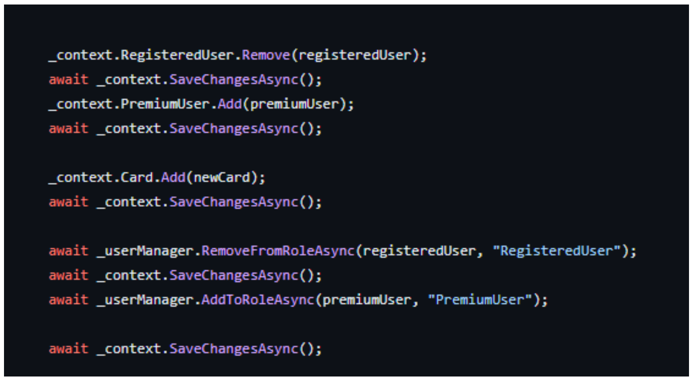
Poslije:



```
var progressList = await _context.Progress
    .Where(p => p.RegisteredUser.Id == registeredUser.Id && p.Date.Date >= currentDate.AddDays(-6).Date && p.Date.Date <= currentDate.Date)
    .OrderBy(p => p.Date)
    .ToListAsync();
```

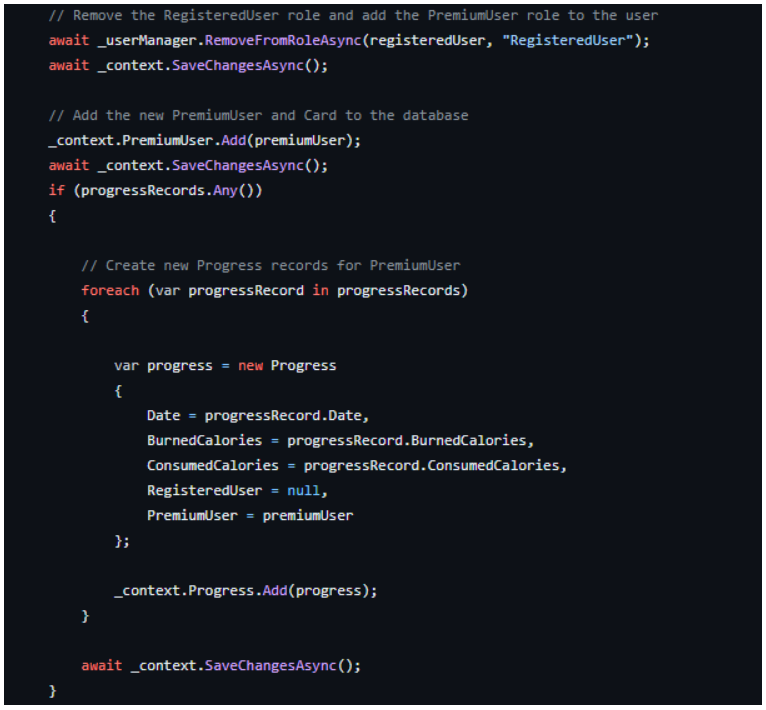
Exception kod upgrade u premium – RegisteredUserController, editCard:

Prije:



```
_context.RegisteredUser.Remove(registeredUser);
await _context.SaveChangesAsync();
_context.PremiumUser.Add(premiumUser);
await _context.SaveChangesAsync();

_context.Card.Add(newCard);
await _context.SaveChangesAsync();

await _userManager.RemoveFromRoleAsync(registeredUser, "RegisteredUser");
await _context.SaveChangesAsync();
await _userManager.AddToRoleAsync(premiumUser, "PremiumUser");

await _context.SaveChangesAsync();
```

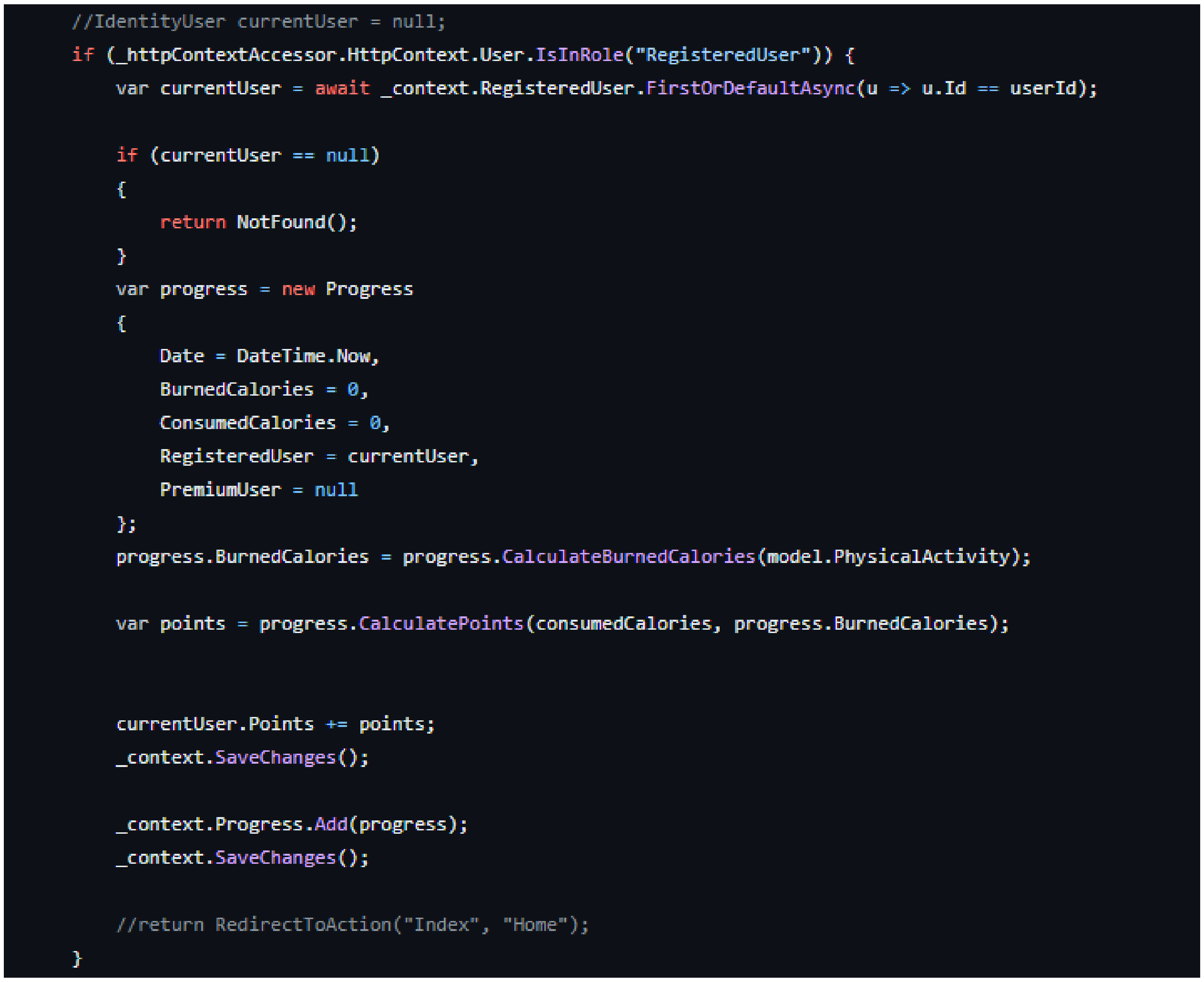
Poslije:



```
// Remove the RegisteredUser role and add the PremiumUser role to the user
await _userManager.RemoveFromRoleAsync(registeredUser, "RegisteredUser");
await _context.SaveChangesAsync();

// Add the new PremiumUser and Card to the database
_context.PremiumUser.Add(premiumUser);
await _context.SaveChangesAsync();
if (progressRecords.Any())
{

    // Create new Progress records for PremiumUser
    foreach (var progressRecord in progressRecords)
    {

        var progress = new Progress
        {
            Date = progressRecord.Date,
            BurnedCalories = progressRecord.BurnedCalories,
            ConsumedCalories = progressRecord.ConsumedCalories,
            RegisteredUser = null,
            PremiumUser = premiumUser
        };

        _context.Progress.Add(progress);
    }

    await _context.SaveChangesAsync();
}
```

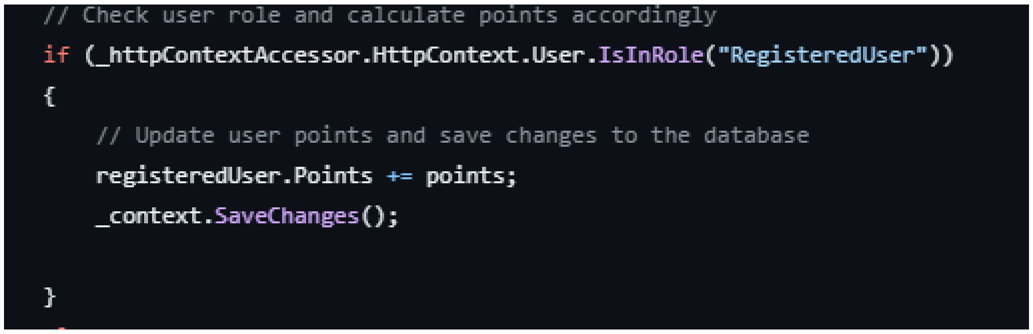
Bespotrebno korištenje httpContextAccessor, Save – RegisteredUserController:

Prije:



```
    //IdentityUser currentUser = null;
    if (_httpContextAccessor.HttpContext.User.IsInRole("RegisteredUser")) {
        var currentUser = await _context.RegisteredUser.FirstOrDefaultAsync(u => u.Id == userId);

        if (currentUser == null)
        {
            return NotFound();
        }
        var progress = new Progress
        {
            Date = DateTime.Now,
            BurnedCalories = 0,
            ConsumedCalories = 0,
            RegisteredUser = currentUser,
            PremiumUser = null
        };
        progress.BurnedCalories = progress.CalculateBurnedCalories(model.PhysicalActivity);

        var points = progress.CalculatePoints(consumedCalories, progress.BurnedCalories);


        currentUser.Points += points;
        _context.SaveChanges();

        _context.Progress.Add(progress);
        _context.SaveChanges();

        //return RedirectToAction("Index", "Home");
    }
```

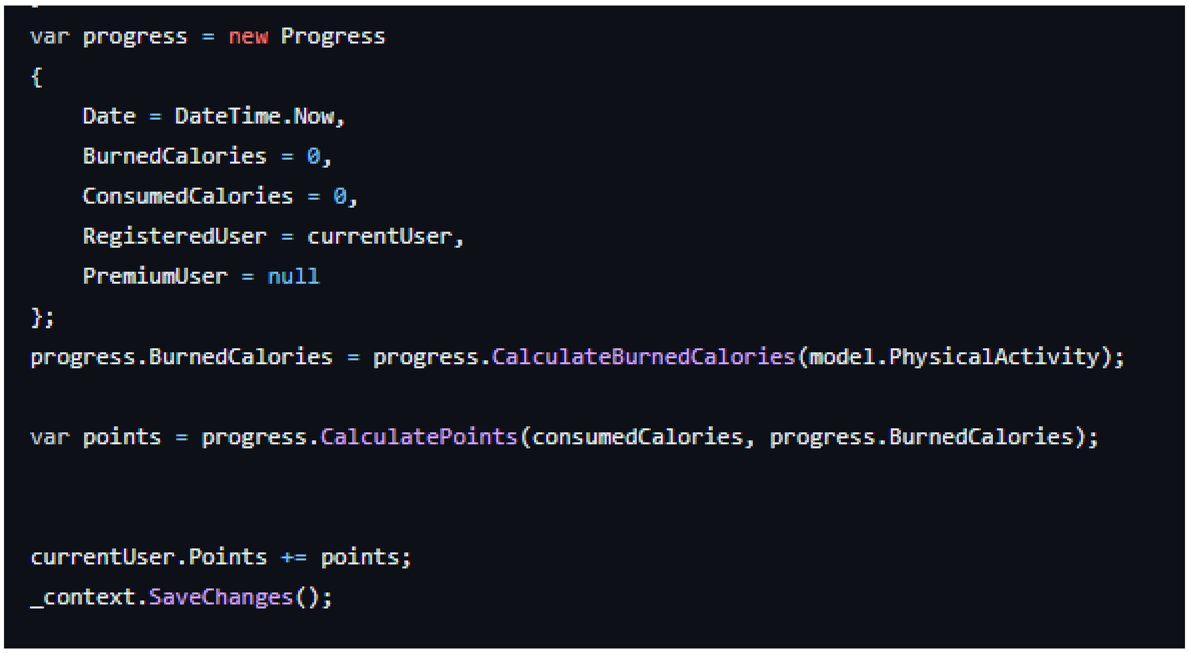
Poslije:



```
 // Check user role and calculate points accordingly
 if (_httpContextAccessor.HttpContext.User.IsInRole("RegisteredUser"))
 {
     // Update user points and save changes to the database
     registeredUser.Points += points;
     _context.SaveChanges();


 }
```

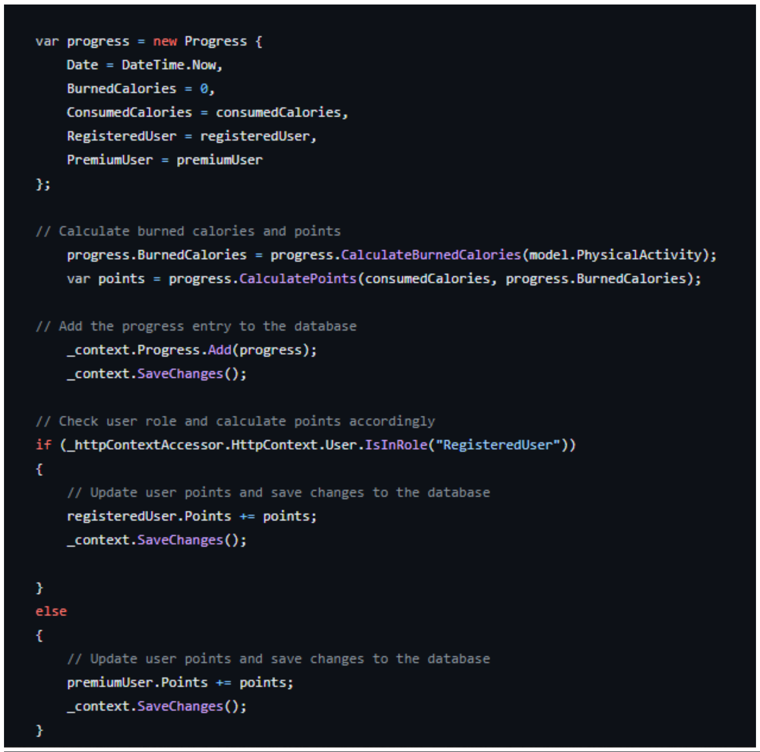
Prenos bodova – Save – RegisteredUserController:

Prije:

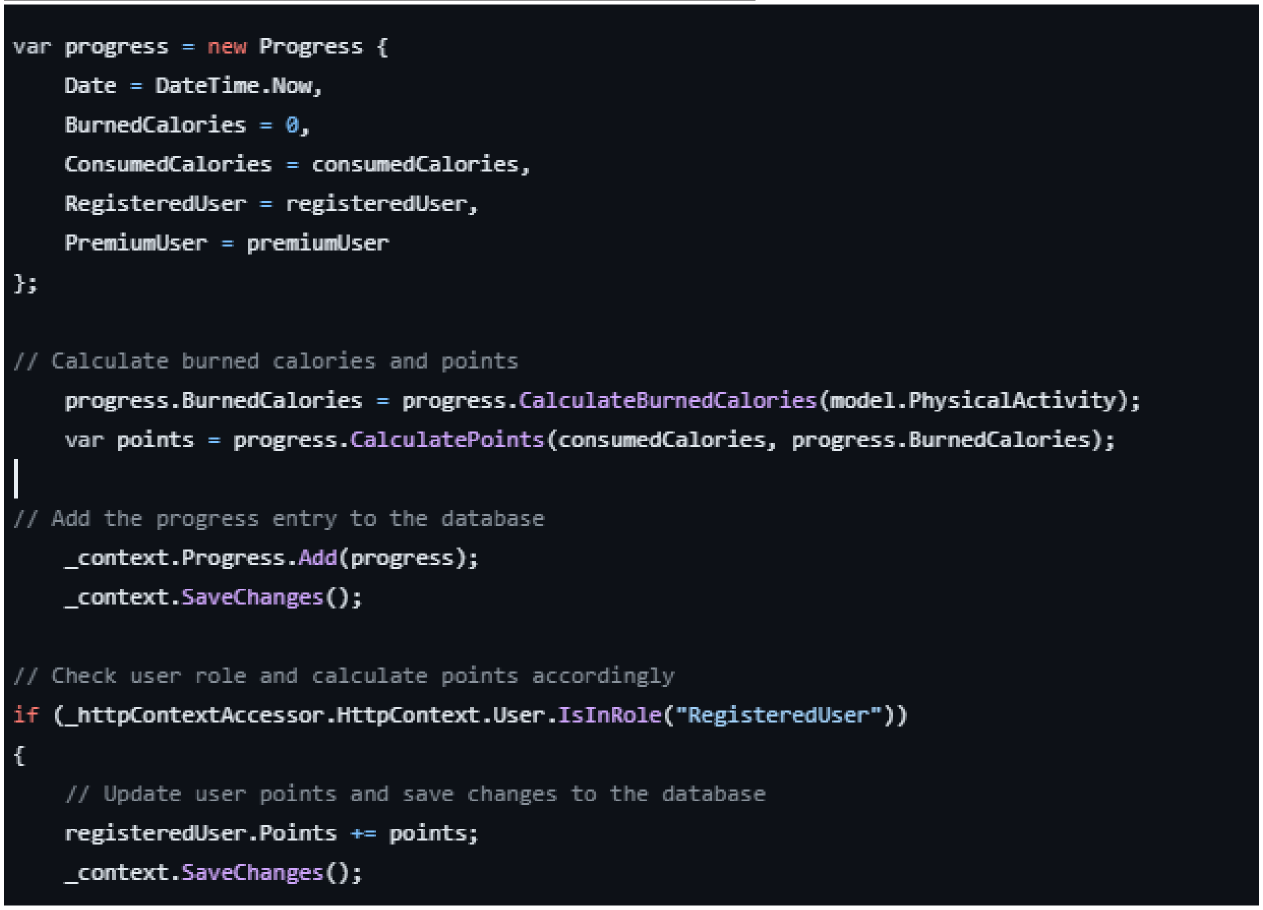


```
var progress = new Progress
{
    Date = DateTime.Now,
    BurnedCalories = 0,
    ConsumedCalories = 0,
    RegisteredUser = currentUser,
    PremiumUser = null
};
progress.BurnedCalories = progress.CalculateBurnedCalories(model.PhysicalActivity);

var points = progress.CalculatePoints(consumedCalories, progress.BurnedCalories);



currentUser.Points += points;
_context.SaveChanges();
```

Poslije:



```
var progress = new Progress {
    Date = DateTime.Now,
    BurnedCalories = 0,
    ConsumedCalories = consumedCalories,
    RegisteredUser = registeredUser,
    PremiumUser = premiumUser
};

// Calculate burned calories and points
    progress.BurnedCalories = progress.CalculateBurnedCalories(model.PhysicalActivity);
    var points = progress.CalculatePoints(consumedCalories, progress.BurnedCalories);

// Add the progress entry to the database
    _context.Progress.Add(progress);
    _context.SaveChanges();

// Check user role and calculate points accordingly
if (_httpContextAccessor.HttpContext.User.IsInRole("RegisteredUser"))
{
    // Update user points and save changes to the database
    registeredUser.Points += points;
    _context.SaveChanges();

}
else
{
    // Update user points and save changes to the database
    premiumUser.Points += points;
    _context.SaveChanges();
}
```



```
var progress = new Progress {
    Date = DateTime.Now,
    BurnedCalories = 0,
    ConsumedCalories = consumedCalories,
    RegisteredUser = registeredUser,
    PremiumUser = premiumUser
};

// Calculate burned calories and points
    progress.BurnedCalories = progress.CalculateBurnedCalories(model.PhysicalActivity);
    var points = progress.CalculatePoints(consumedCalories, progress.BurnedCalories);

// Add the progress entry to the database
    _context.Progress.Add(progress);
    _context.SaveChanges();

// Check user role and calculate points accordingly
if (_httpContextAccessor.HttpContext.User.IsInRole("RegisteredUser"))
{
    // Update user points and save changes to the database
    registeredUser.Points += points;
    _context.SaveChanges();
```