Solution

.NET Project

1. Create an **EditProfile** handler to update the profile. We only want to validate against the **DisplayName** here as we do not ask the user for a **Bio** when they register so we want to allow this to be optional.

```
// Application/Profiles/Commands/EditProfile.cs

using System;
using Application.Core;
using Application.Interfaces;
using MediatR;
using Persistence;

namespace Application.Profiles.Commands;

public class EditProfile
{
    public class Command : IRequest<Result<Unit>>
    {
        public string DisplayName { get; set; } = string.Empty;
        public string Bio { get; set; } = string.Empty;
    }

    public class Handler(AppDbContext context, IUserAccessor userAccessor)
        : IRequestHandler<Command, Result<Unit>>
    {
        public async Task<Result<Unit>> Handle(Command request,
            CancellationToken cancellationToken)
        {
            var user = await userAccessor.GetUserAsync();

            user.DisplayName = request.DisplayName;
            user.Bio = request.Bio;

            var result = await context.SaveChangesAsync(cancellationToken) > 0;

            return result
                ? Result<Unit>.Success(Unit.Value)
                : Result<Unit>.Failure("Failed to update profile", 400);
        }
    }
}
```

```
// Application/Profiles/Validators/EditProfileValidator.cs

using System;
using Application.Profiles.Commands;
using FluentValidation;

namespace Application.Profiles.Validators;

public class EditProfileValidator : AbstractValidator<EditProfile.Command>
{
    public EditProfileValidator()
    {
```

```
        RuleFor(x => x.DisplayName).NotEmpty();
    }
}
```

2. Update the **ProfilesController** and add an endpoint for editing the profile.

```
public class ProfilesController : BaseApiController
{
    [HttpPut]
    public async Task<ActionResult> UpdateProfile(EditProfile.Command command)
    {
        return HandleResult(await Mediator.Send(command));
    }
```

3. Test the endpoint in Postman using the 3 requests in the Module 18 folder.
    a. If we try and update the profile again without changing it we will get a 400 bad request as there were no changes to save to the Database.
    b. Is there a way we can accept the same data coming in and tell EF the entity has been modified even if it has not?

If you did want the entity to appear modified even if it is not then you can use the following code in the handler to set a flag on the entity to say it has been modified. This will result in the SaveChanges method returning > 0.

```
public async Task<Result<Unit>> Handle(Command request,
    CancellationToken cancellationToken)
{
    var user = await userAccessor.GetUserAsync();

    user.DisplayName = request.DisplayName;
    user.Bio = request.Bio;

    context.Entry(user).State = EntityState.Modified;

    var result = await context.SaveChangesAsync(cancellationToken) > 0;

    return result
        ? Result<Unit>.Success(Unit.Value)
        : Result<Unit>.Failure("Failed to update profile", 400);
}
```

Client Project

1. Create a Zod schema called editProfileSchema.ts so we can validate the 2 fields the user can edit. DisplayName should be required and Bio is optional.

```
// lib/schemas/editProfileSchema.ts
import {z} from "zod";
import {requiredString} from "../util/util.ts";

export const editProfileSchema = z.object({
    displayName: requiredString('Display Name'),
    bio: z.string().optional()
});
```

```
export type EditProfileSchema = z.infer<typeof editProfileSchema>;
```

2. Add a new mutation function in the useProfile hook. Ensure that the DisplayName property is updated in both the [’profile’] and [’user’] queries. Choose whichever approach you prefer as invalidating, setting on success or optimistic updating are all viable options

```
    // useProfile.ts

    const updateProfile = useMutation({
       mutationFn: async (profile: EditProfileSchema) => {
           await agent.put(`/profiles`, profile);
       },
       onSuccess: (_, profile) => {
           queryClient.setQueryData(['profile', id], (data: Profile) => {
               if (!data) return data;
               return {
                   ...data,
                   displayName: profile.displayName,
                   bio: profile.bio
               }
           });
           queryClient.setQueryData(['user'], (userData: User) => {
               if (!userData) return userData;
               return {
                   ...userData,
                   displayName: profile.displayName
               }
           });
       }
    })

    return {
      // other returns omitted
        updateProfile
    }
```

3. Create a **ProfileEditForm** component in the Profiles feature folder and ensure this can be opened and cancelled in it’s parent component (**ProfileAbout**) as well as being closed after successful submission of the form**.** We need a normal text input and a multi-line text input for the 2 fields.

```
import { useForm } from "react-hook-form";
import { editProfileSchema, EditProfileSchema } from "../../lib/schemas/editProfileSchema
import { zodResolver } from "@hookform/resolvers/zod";
import { useProfile } from "../../lib/hooks/useProfile.ts";
import { useEffect } from "react";
import { Box, Button } from "@mui/material";
import TextInput from "../../app/shared/components/TextInput.tsx";
import { useParams } from "react-router";

type Props = {
    setEditMode: (editMode: boolean) => void;
}

export default function ProfileEdit({ setEditMode }: Props) {
    const { id } = useParams();
    const { updateProfile, profile } = useProfile(id);
    const { control, handleSubmit, reset, formState: { isDirty, isValid } }
```

```
        = useForm<EditProfileSchema>({
            resolver: zodResolver(editProfileSchema),
            mode: 'onTouched'
    });

    const onSubmit = (data: EditProfileSchema) => {
        updateProfile.mutate(data, {
            onSuccess: () => setEditMode(false)
        });
    }

    useEffect(() => {
        reset({
            displayName: profile?.displayName,
            bio: profile?.bio || ''
        });
    }, [profile, reset]);

    return (
        <Box component='form'
            onSubmit={handleSubmit(onSubmit)}
            display='flex'
            flexDirection='column'
            alignContent='center'
            gap={3}
            mt={3}
        >
            <TextInput label='Display Name' name='displayName' control={control} />
            <TextInput
                label='Add your bio'
                name='bio'
                control={control}
                multiline
                rows={4}
            />
            <Button
                type='submit'
                variant='contained'
                disabled={!isValid || !isDirty || updateProfile.isPending}
            >
                Update profile
            </Button>
        </Box>
    );
}
```

```
// ProfileAbout.tsx

import { useParams } from "react-router"
import { useProfile } from "../../lib/hooks/useProfile";
import { Box, Button, Divider, Typography } from "@mui/material";
import { useState } from "react";
import ProfileEdit from "./ProfileEditForm";

export default function ProfileAbout() {
    const { id } = useParams();
    const { profile, isCurrentUser } = useProfile(id);
    const [editMode, setEditMode] = useState(false);
```

```
    return (
        <Box>
            <Box display='flex' justifyContent='space-between'>
                <Typography variant="h5">About {profile?.displayName}</Typography>
                {isCurrentUser &&
                <Button onClick={() => setEditMode(!editMode)}>
                    Edit profile
                </Button>}
            </Box>
            <Divider sx={{my: 2}} />
            <Box sx={{overflow: 'auto', maxHeight: 350}}>
                <Typography variant="body1" sx={{whiteSpace: 'pre-wrap'}}>
                    {profile?.bio || 'No description added yet'}
                </Typography>
                {editMode && (
                    <ProfileEdit setEditMode={setEditMode} />
                )}
            </Box>
        </Box>
    )
}
```

Feature complete!

You can now commit your changes. Suggested commit message: “End of section 18”.