Solution

.Net code

1. Create a UserActivityDto in the Application/Profiles feature that contains the Id, Title, Category and Date properties.

```
namespace Application.Profiles.DTOs;

public class UserActivityDto
{
    public required string Id { get; set; }
    public required string Title { get; set; }
    public required string Category { get; set; }
    public DateTime Date { get; set; }
}
```

2. Create a GetUserActivities handler in the Profiles/Queries folder that returns a list of UserActivityDto. This list should be able to be filtered by past events that user has attended, future events that user is attending and events that user is hosting.

```
using System;
using Application.Core;
using Application.Profiles.DTOs;
using AutoMapper;
using AutoMapper.QueryableExtensions;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Persistence;

namespace Application.Profiles.Queries;

public class GetUserActivities
{
    public class Query : IRequest<Result<List<UserActivityDto>>>
    {
        public required string UserId { get; set; }
        public required string Filter { get; set; }
    }

    public class Handler(AppDbContext context, IMapper mapper)
        : IRequestHandler<Query, Result<List<UserActivityDto>>>
    {
        public async Task<Result<List<UserActivityDto>>> Handle(Query request,
            CancellationToken cancellationToken)
        {
            var query = context.ActivityAttendees
                .Where(u => u.User.Id == request.UserId)
                .OrderBy(a => a.Activity.Date)
                .Select(x => x.Activity)
                .AsQueryable();

            var today = DateTime.UtcNow;

            query = request.Filter switch
            {
                "past" => query.Where(a => a.Date <= today
```

```
                    && a.Attendees.Any(x => x.UserId == request.UserId)),
                "hosting" => query.Where(a => a.Attendees.Any(x => x.IsHost
                        && x.UserId == request.UserId)),
                _ => query.Where(a => a.Date >= today
                    && a.Attendees.Any(x => x.UserId == request.UserId))
            };

            var projectedActivities = query
                .ProjectTo<UserActivityDto>(mapper.ConfigurationProvider);

            var activities = await projectedActivities.ToListAsync(cancellationToken);

            return Result<List<UserActivityDto>>.Success(activities);
        }
    }
}
```

3. Create a MappingProfile to map from Activity to UserActivityDto

```
            CreateMap<Activity, UserActivityDto>();
```

4. Create an endpoint in the ProfilesController which has the following route and also accepts a filter parameter

```
using Application.Profiles;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace API.Controllers
{
    public class ProfilesController : BaseApiController
    {

       [HttpGet("{userId}/activities")]
        public async Task<IActionResult> GetUserActivities(string userId, string filter)
        {
            return HandleResult(await Mediator.Send(new GetUserActivities.Query
                { UserId = userId, Filter = filter }));
        }
    }
}
```

5. Restart the API server!

Client side code

1. Create a query in the useProfile hook to request the user activities. This query should only be enabled when we are on the events tab in the profile. Hint: useState() for the filter value that is initially set to null will help achieve this.

```
export const useProfile = (id?: string, predicate?: string) => {
    const [filter, setFilter] = useState<string | null>(null);
    const queryClient = useQueryClient();


    const {data: userActivities, isLoading: loadingUserActivities} = useQuery({
        queryKey: ['user-activities', filter],
        queryFn: async () => {
            const response = await agent.get<Activity[]>(`/profiles/${id}/activities`, {
                params: {
                    filter
```

```
                }
            });
            return response.data
        },
        enabled: !!id && !!filter
    });

    return {
                // other return elements omitted
        userActivities,
        loadingUserActivities,
        setFilter,
        filter
    }
}
```

2. Create a ProfileActivities.tsx component in the Profiles folder that uses tabs to allow the user to select the past, future or hosted activities that user is attending or hosting. Hint: Use a useEffect to set the initial filter value when the component mounts.

```
import { SyntheticEvent, useEffect, useState } from "react";
import { Box, Card, CardContent, CardMedia, Grid2, Tab, Tabs, Typography } from "@mui/mat
import { Link, useParams } from "react-router";
import { format } from "date-fns";
import { useProfile } from "../../lib/hooks/useProfile.ts";

export default function ProfileActivities() {
    const [activeTab, setActiveTab] = useState(0);
    const { id } = useParams();
    const { userActivities, setFilter, loadingUserActivities } = useProfile(id);

    useEffect(() => {
        setFilter('future')
    }, [setFilter])

    const tabs = [
        { menuItem: 'Future Events', key: 'future' },
        { menuItem: 'Past Events', key: 'past' },
        { menuItem: 'Hosting', key: 'hosting' }
    ];

    const handleTabChange = (_: SyntheticEvent, newValue: number) => {
        setActiveTab(newValue);
        setFilter(tabs[newValue].key);
    };

    return (
        <Box>
            <Grid2 container spacing={2}>
                <Grid2 size={12}>
                    <Tabs
                        value={activeTab}
                        onChange={handleTabChange}
                    >
                        {tabs.map((tab, index) => (
                            <Tab label={tab.menuItem} key={index} />
                        ))}
                    </Tabs>
                </Grid2>
```

```
            </Grid2>
            {(!userActivities || userActivities.length === 0)
                        && !loadingUserActivities ? (
                <Typography mt={2}>
                    No activities to show
                </Typography>
            ) : null}
            <Grid2
                container
                   spacing={2}
                   sx={{ marginTop: 2, height: 400, overflow: 'auto' }}
            >
                {userActivities && userActivities.map((activity: Activity) => (
                    <Grid2 size={2} key={activity.id}>
                        <Link to={`/activities/${activity.id}`}
                                    style={{ textDecoration: 'none' }}>
                            <Card elevation={4}>
                                <CardMedia
                                    component="img"
                                    height="100"
                                    image={
                                        `/images/categoryImages/${activity.category}.jpg
                                       }
                                    alt={activity.title}
                                    sx={{ objectFit: 'cover' }}
                                />
                                <CardContent>
                                    <Typography variant="h6" textAlign="center" mb={1}>
                                        {activity.title}
                                    </Typography>
                                    <Typography
                                        variant="body2"
                                        textAlign="center"
                                        display='flex'
                                        flexDirection='column'
                                    >
                                        <span>
                                            {format(activity.date, 'do LLL yyyy')}
                                           </span>
                                        <span>{format(activity.date, 'h:mm a')}</span>
                                    </Typography>
                                </CardContent>
                            </Card>
                        </Link>
                    </Grid2>
                ))}
            </Grid2>
        </Box>
    )
}
```

3. Update the Profile content to use the new ProfileActivities component.

```
  const tabContent = [
    {label: 'About', content: <ProfileAbout />},
    {label: 'Photos', content: <ProfilePhotos />},
    {label: 'Events', content: <ProfileActivities />},
    {label: 'Followers', content: <ProfileFollowings activeTab={value} />},
```

```
    {label: 'Following', content: <ProfileFollowings activeTab={value} />},
  ]
```

4. Test!

Challenge complete! Please commit your changes to GitHub. Suggested commit message “End of section 21”