Solution

1. Create a new class called UserActivityDto in the Profiles folder (already done this part).

```
using System;
using System.Text.Json.Serialization;

namespace Application.Profiles
{
    public class UserActivityDto
    {
        public Guid Id { get; set; }
        public string Title { get; set; }
        public string Category { get; set; }
        public DateTime Date { get; set; }

        [JsonIgnore]
        public string HostUsername { get; set; }
    }
}
```

2. Create a new class called ListActivities in the profile folder. This will be a handler and we want to return a list of activities based on a **predicate** and the **username** of the user whose profile we are looking at. Do not worry about paging this list to keep it simple. In this handler we want to return a list of **UserActivityDto** the user is attending and the predicate will either be:
   1. Activities in the **past**
   2. Activities the user is **hosting**
   3. The activities the user is going to in the future (default case)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using Application.Activities;
using Application.Core;
using AutoMapper;
using AutoMapper.QueryableExtensions;
using MediatR;
using Microsoft.EntityFrameworkCore;
using Persistence;

namespace Application.Profiles
{
    public class ListActivities
    {
        public class Query : IRequest<Result<List<UserActivityDto>>>
        {
            public string Username { get; set; }
            public string Predicate { get; set; }
        }

        public class Handler : IRequestHandler<Query,
Result<List<UserActivityDto>>>
        {
            private readonly DataContext _context;
            private readonly IMapper _mapper;
            public Handler(DataContext context, IMapper mapper)
            {
                _mapper = mapper;
                _context = context;
            }

            public async Task<Result<List<UserActivityDto>>> Handle(Query
request, CancellationToken cancellationToken)
            {
                var query = _context.ActivityAttendees
                    .Where(u => u.AppUser.UserName == request.Username)
                    .OrderBy(a => a.Activity.Date)
                    .ProjectTo<UserActivityDto>(_mapper.ConfigurationProvider)
                    .AsQueryable();
                query = request.Predicate switch
                {
                    "past" => query.Where(a => a.Date <= DateTime.Now),
                    "hosting" => query.Where(a => a.HostUsername ==
request.Username),
                    _ => query.Where(a => a.Date >= DateTime.Now)
                };

                var activities = await query.ToListAsync();

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

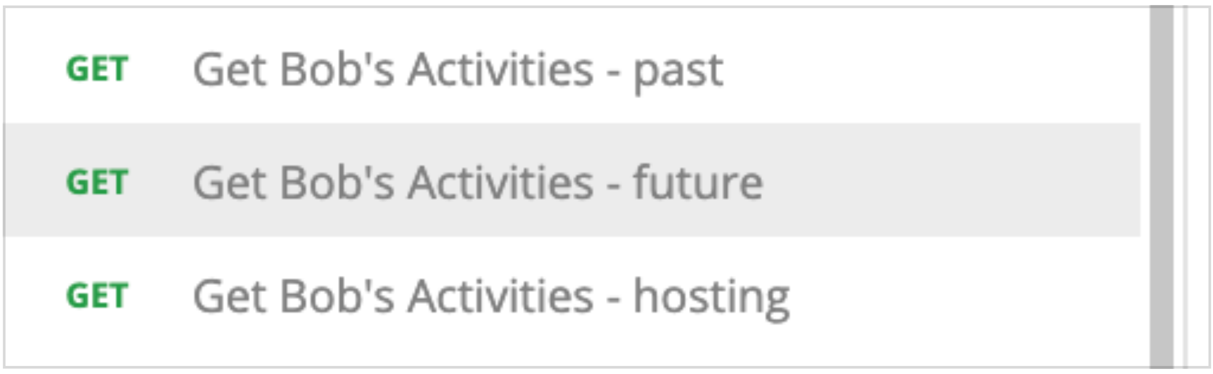
3. If you are using AutoMapper for your solution then you will need to map from an ActivityAttendee object to the UserActivityDto object. Note: We need to use the AutoMapper namespace here as we will get conflicts due to needing to add a using statement for our Profiles.

```

using System.Linq;
using Application.Activities;
using Application.Comments;
using Application.Profiles;
using AutoMapper;
using Domain;

namespace Application.Core
{
    public class MappingProfiles : AutoMapper.Profile
    {
        public MappingProfiles()
        {
            // other mappings omitted
            CreateMap<ActivityAttendee, UserActivityDto>()
                .ForMember(d => d.Id, o => o.MapFrom(s => s.Activity.Id))
                .ForMember(d => d.Date, o => o.MapFrom(s => s.Activity.Date))
                .ForMember(d => d.Title, o => o.MapFrom(s => s.Activity.Title))
                .ForMember(d => d.Category, o => o.MapFrom(s =>
s.Activity.Category))
                .ForMember(d => d.HostUsername, o => o.MapFrom(s =>
                    s.Activity.Attendees.FirstOrDefault(x =>
x.IsHost).AppUser.UserName));
        }
    }
}
```

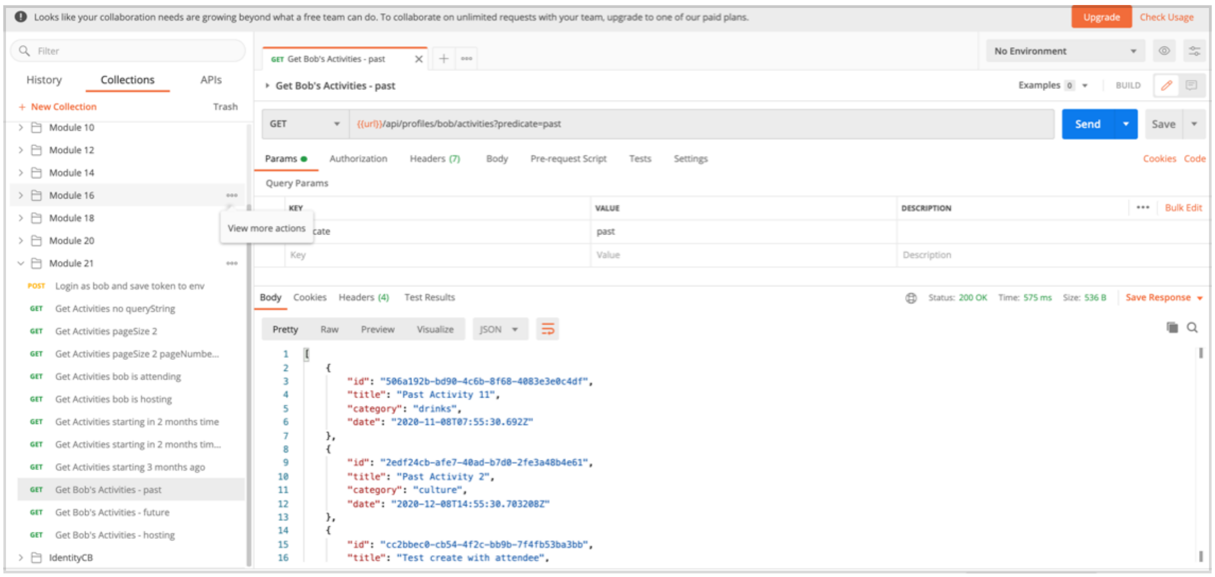
4. Add an endpoint in the Profiles Controller so the client can send a get request to "/api/profiles/{username}/activities?predicate='thePredicate'".

```
        [HttpGet("{username}/activities")]
        public async Task<IActionResult> GetUserActivities(string username,
string predicate)
        {
            return HandleResult(await Mediator.Send(new ListActivities.Query
                {Username = username, Predicate = predicate}));
        }
```

4. Test the results using the 3 following pre-written requests in Postman.


GET Get Bob's Activities - past
GET Get Bob's Activities - future
GET Get Bob's Activities - hosting

Should see the following and the tests should pass in Postman:


Looks like your collaboration needs are growing beyond what a free team can do. To collaborate on unlimited requests with your team, upgrade to one of our paid plans.

History | Collections | APIs

Module 10, Module 12, Module 14, Module 16, Module 18, Module 20, Module 21

- POST Login as bob and save token to env
- GET Get Activities no queryString
- GET Get Activities pageSize 2
- GET Get Activities pageSize 2 pageNumbe...
- GET Get Activities bob is attending
- GET Get Activities bob is hosting
- GET Get Activities starting in 2 months time
- GET Get Activities starting in 2 months tim...
- GET Get Activities starting 3 months ago
- GET Get Bob's Activities - past
- GET Get Bob's Activities - future
- GET Get Bob's Activities - hosting

IdentityCB

Get Bob's Activities - past

GET {{url}}/api/profiles/bob/activities?predicate=past

KEY	VALUE	DESCRIPTION
cate	past	

Status: 200 OK Time: 575 ms Size: 536 B

```
[
    {
        "id": "506a192b-bd90-4c6b-8f68-4083e3e0c4df",
        "title": "Past Activity 11",
        "category": "drinks",
        "date": "2020-11-08T07:55:30.692Z"
    },
    {
        "id": "2edf24cb-afe7-40ad-b7d0-2fe3a48b4e61",
        "title": "Past Activity 2",
        "category": "culture",
        "date": "2020-12-08T14:55:30.703208Z"
    },
    {
        "id": "cc2bbec0-cb54-4f2c-bb9b-7f4fb53ba3bb",
        "title": "Test create with attendee",
```

5. Add a method in the agent.ts to get the activities for a user based on the predicate

```
const Profiles = {
    get: (username: string) => requests.get<Profile>(`/profiles/${username}`),
    uploadPhoto: (file: Blob) => {
        let formData = new FormData();
        formData.append('File', file);
        return axios.post<Photo>('photos', formData, {
            headers: { 'Content-type': 'multipart/form-data' }
        })
    },
    setMainPhoto: (id: string) => requests.post(`/photos/${id}/setMain`, {}),
    deletePhoto: (id: string) => requests.del(`/photos/${id}`),
    updateProfile: (profile: Partial<Profile>) => requests.put(`/profiles`, 
profile),
    updateFollowing: (username: string) => requests.post(`/follow/${username}`, 
{}),
    listFollowings: (username: string, predicate: string) =>
        requests.get<Profile[]>(`/follow/${username}?predicate=${predicate}`),
    listActivities: (username: string, predicate: string) =>
        requests.get<UserActivity[]>(`/profiles/${username}/activities?
predicate=${predicate}`)
}
```

6. Add an interface called UserActivity in the profile.ts class that matches the properties we return in this object from the API

```
export interface UserActivity {
    id: string;
    title: string;
    category: string;
    date: Date;
}
```

7. Add a property in the profile store for the UserActivities as well as a loading flag called 'loadingActivities'

```
// imports omitted

export default class ProfileStore {
    currentUserProfile: Profile | null = null;
    profile: Profile | null = null;
    loadingProfile = false;
    uploading = false;
    loading = false;
    followings: Profile[] = [];
    loadingFollowings = false;
    activeTab: number = 0;
    userActivities: UserActivity[] = [];
    loadingActivities = false;

// omitted
```

8. Add a method in the profiles store to load activities for a user that takes a username and the predicate as a parameter.

```
  // profileStore.ts
```

```
    loadUserActivities = async (username: string, predicate?: string) => {
        this.loadingActivities = true;
        try {
            const activities = await agent.Profiles.listActivities(username,
predicate!);
            runInAction(() => {
                this.userActivities = activities;
                this.loadingActivities = false;
            })
        } catch (error) {
            console.log(error);
            runInAction(() => {
                this.loadingActivities = false;
            })
        }
    }
```

9. Add a new component called 'ProfileActivities' where each profile activity is contained in its own card.   This component should have 3 tabs to allow the user to select from:
   1. Future activities
   2. Past activities
   3. Activities the user is hosting

```
import React, { SyntheticEvent, useEffect } from 'react';
import { observer } from 'mobx-react-lite';
import { Tab, Grid, Header, Card, Image, TabProps } from 'semantic-ui-react';
import { Link } from 'react-router-dom';
import { UserActivity } from '../../app/models/profile';
import { format } from 'date-fns';
import { useStore } from "../../app/stores/store";

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

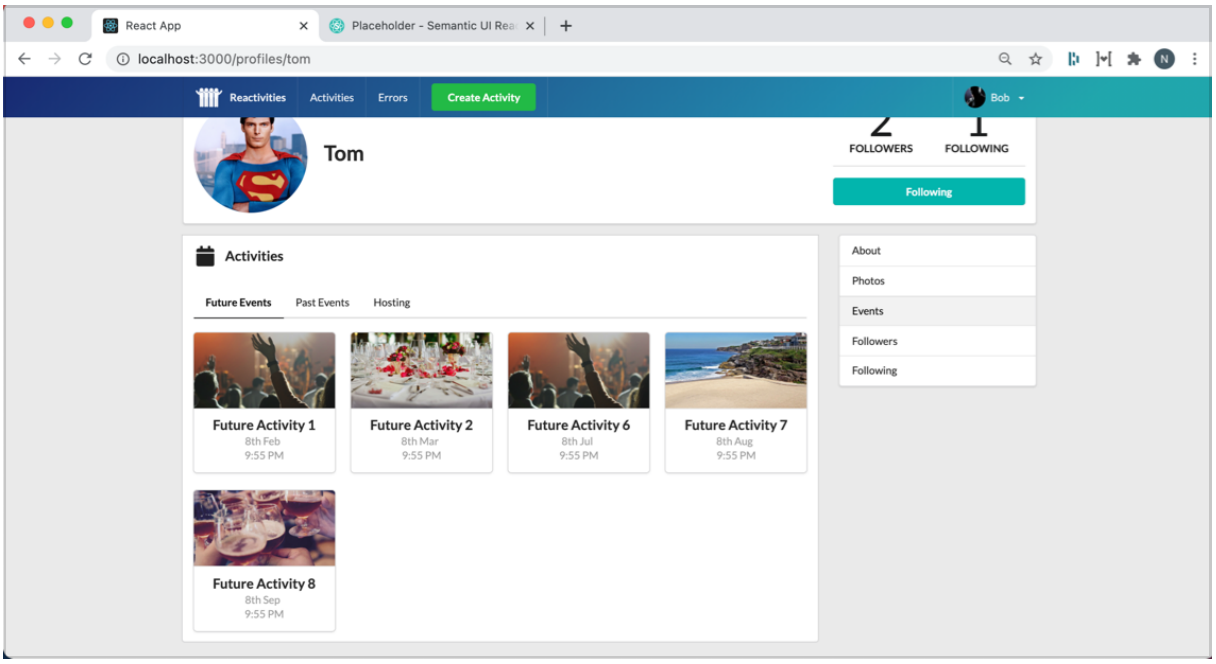
```
export default observer(function ProfileActivities() {
    const { profileStore } = useStore();
    const {
        loadUserActivities,
        profile,
        loadingActivities,
        userActivities
    } = profileStore;

    useEffect(() => {
        loadUserActivities(profile!.username);
    }, [loadUserActivities, profile]);

    const handleTabChange = (e: SyntheticEvent, data: TabProps) => {
        loadUserActivities(profile!.username, panes[data.activeIndex as
number].pane.key);
    };

    return (
        <Tab.Pane loading={loadingActivities}>
            <Grid>
                <Grid.Column width={16}>
                    <Header floated='left' icon='calendar'
content={'Activities'} />
                </Grid.Column>
                <Grid.Column width={16}>
                    <Tab
                        panes={panes}
                        menu={{ secondary: true, pointing: true }}
                        onTabChange={(e, data) => handleTabChange(e, data)}
                    />
                    <br />
                    <Card.Group itemsPerRow={4}>
                        {userActivities.map((activity: UserActivity) => (
                            <Card
                                as={Link}
                                to={`/activities/${activity.id}`}
                                key={activity.id}
```

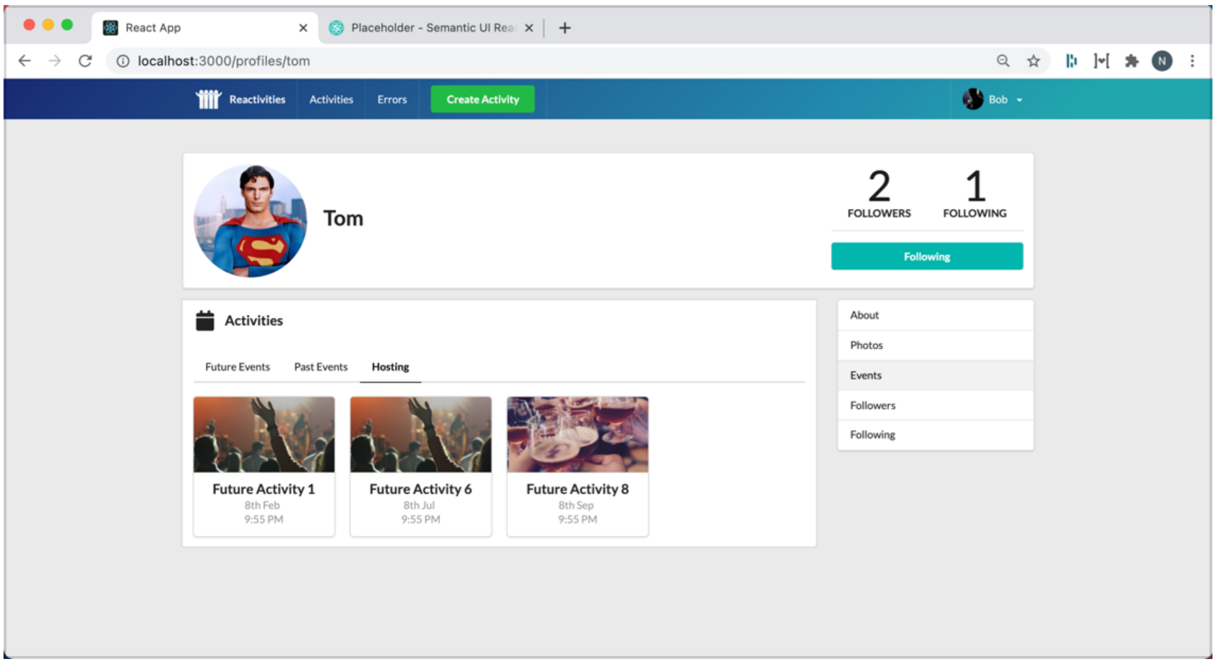
```
                              >
                                  <Image
                                      src={`/assets/categoryImages/$
{activity.category}.jpg`}
                                      style={{ minHeight: 100, objectFit:
'cover' }}
                                  />
                                  <Card.Content>
                                      <Card.Header
textAlign='center'>{activity.title}</Card.Header>
                                      <Card.Meta textAlign='center'>
                                          <div>{format(new Date(activity.date),
'do LLL')}</div>
                                          <div>{format(new Date(activity.date),
'h:mm a')}</div>
                                      </Card.Meta>
                                  </Card.Content>
                              </Card>
                          ))}
                      </Card.Group>
                  </Grid.Column>
              </Grid>
          </Tab.Pane>
    );
});
```

10. Add the ProfileActivities to the ProfileContent component.

```
export default observer(function ProfileContent({profile}: Props) {
    const {profileStore} = useStore();

    const panes = [
        {menuItem: 'About', render: () => <ProfileAbout />},
        {menuItem: 'Photos', render: () => <ProfilePhotos profile={profile} /
>},
        {menuItem: 'Events', render: () => <ProfileActivities />},
        {menuItem: 'Followers', render: () => <ProfileFollowings />},
        {menuItem: 'Following', render: () => <ProfileFollowings />},
    ];
```

11. Test to make sure this works in the client.





12. Finished! Commit changes to source control.

#reactivities/section 21 - Paging#