**EINDONTWERP**

|  |  |
| --- | --- |
| **Naam** | Xue Kun Zhou |
| **Studentnummer** | 20170173 |
| **Klas** | 6 |

**OPDRACHT**

Geef hieronder een korte beschrijving van je opdracht (een samenvatting van je productvisie) zodat je docent OPT3 ook begrijpt met welke opdracht jij bezig bent (neem dit onderdeel over uit je laatst ingeleverde opdracht en pas het aan aan de hand van je eigen bevindingen en aan de hand van feedback van medestudenten en je docent aan):

MyLibrary is een webapp dat bijhoudt welke films, series, boeken en games je hebt gezien, gelezen en gespeeld. De gebruiker heeft daarmee een digitale “collectie” van media die hij heeft geconsumeerd. Per item wordt er ook een score bijgehouden zodat wanneer de gebruiker op een later moment kan terugzien wat hij ervan vond. Door het bijhouden van deze items kan de gebruiker ook een overzicht krijgen van hoeveel tijd hij hierin besteed.

De app houdt niet bij welke films etc. er al bestaan en houdt dus geen gemiddelde score bij van alle gebruikers en geeft ook geen aanbevelingen van welke items de gebruiker misschien wel leuk vindt.

Vrienden kunnen wel de lijsten van elkaar zien.

**GITHUB**

Maak je GitHub-repo Private en geef je docent voor het werkcollege toegang tot je repo (vraag hem om zijn account als hij dat volgens jou nog niet aan jou gegeven heeft). Stuur niet steeds een nieuwe link/uitnodiging, maar bouw in feature-branches verder aan de applicatie die je eerder hebt opgeleverd:

https://github.com/XuekunZhou/OPT3\_MyLibrary (in branch main)

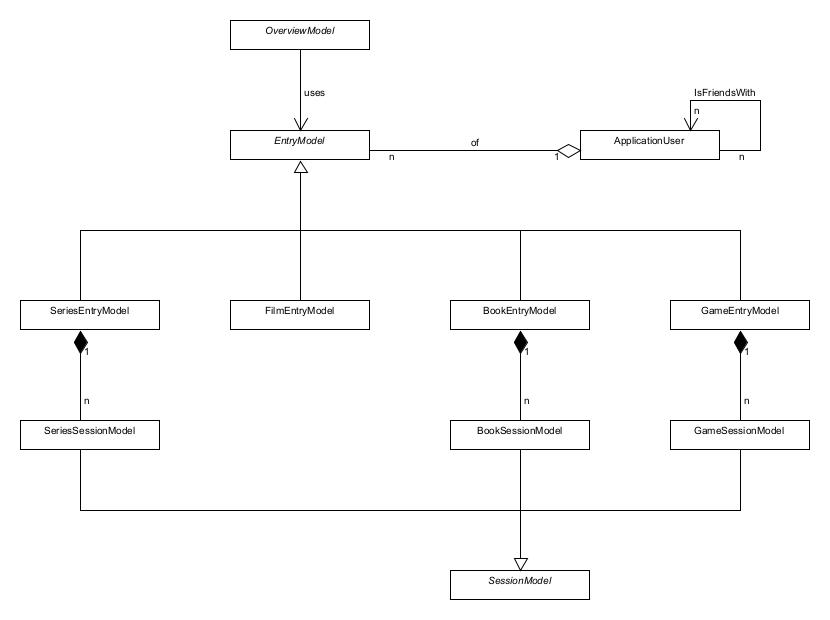
Bovenstaande repo is m.b.v. .gitignore ontdaan van IntelliJ-files en er is een Maven POM-file toegevoegd waarmee je applicatie in één keer gerund kan worden.

**DOMEINMODEL**

Bij de oplevering van het ontwerp en de realisatie hieronder ben ik uitgegaan van de volgende User Stories (gebruik de ‘+’ rechts van de tabel om een User Story toe te voegen):

|  |
| --- |
| Als Gebruiker wil ik film, series, etc. een score geven zodat ik kan bijhouden welke films, series, etc. ik gezien heb en wat ik er van vond. |

Kopieer hieronder (een kopie van) het (aangepaste) domeinmodel dat je als resultaat hebt opgeleverd bij de vorige opdracht (kopieer het model uit je vorige opdracht of kopieer het aangepaste model):

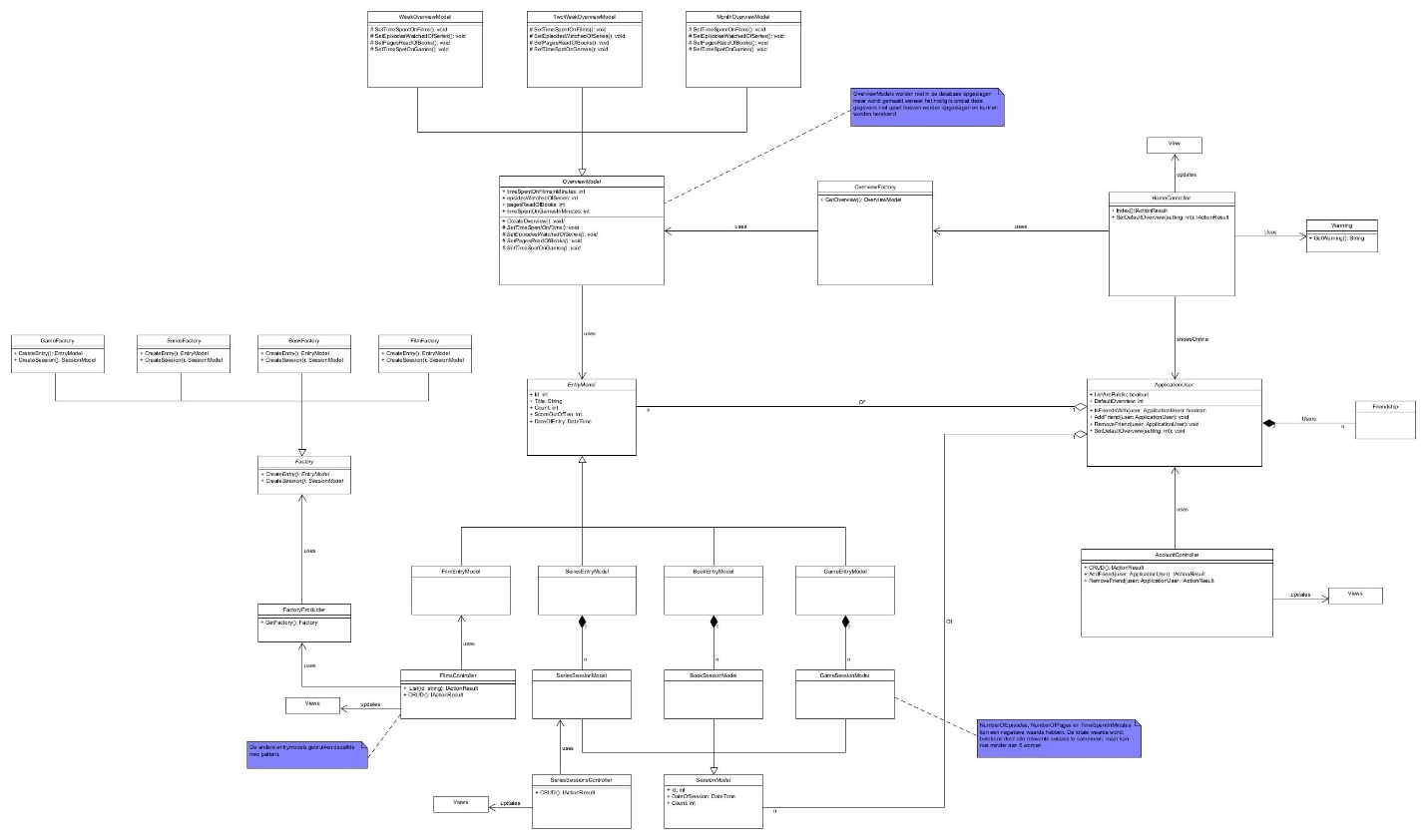


\* Controleer of het plaatje goed leesbaar is. Als je wilt dat je docent het domeinmodel met realisatie opnieuw beoordeelt, lever je het ingevulde template voor het domeinmodel in via de inleverbox voor de herkansing van het domeinmodel in BlackBoard.

**EINDONTWERP**

Vanuit dit domeinmodel heb je Patterns en SOLID-principes toegepast op je model en je hebt Classes toegevoegd die je nodig hebt om je applicatie te laten werken (zoals bijv. voor het tonen van een GUI of Command Line voor het tonen en afhandelen van een menu). Je hebt er kortom een technisch ontwerp van gemaakt waarin je ook de types van je properties, parameters en methodes hebt toegevoegd. Zie het volgende document voor voorbeelduitwerking en aanwijzingen:

‘[OPT3 16.1.3.2 - Eindontwerp – Toelichting.docx](https://blackboard.hhs.nl/bbcswebdav/courses/H-SE-S2-AOOP-2020/OPT3/OPT3%2014.1.3.2%20-%20Domeinontwerp%20en%20-realisatie%20-%20Toelichting.pdf)’



\* Controleer of het plaatje goed leesbaar is.

**PATTERNS**

Beschrijf hieronder hoe je het Template Method Pattern en een tweede Pattern hebt toegepast in bovenstaand eindontwerp:

|  |  |
| --- | --- |
| **Pattern** | **Beschrijving** |
| Template Method Pattern | Na(a)m(en) van template method (s): CreateOverview()  SetPeroidName() sets de variable van PeriodName, SetTimeSpentOnFilms() sets de variable van TimeSpentOnFilms, SetEpisodesWatchedOfSeries() sets de variable van EpisodesWatchedOfSeries, SetPagesReadOfBooks() sets de variable van PagesReadOfBooks, SetTimeSentOnGames() sets de variable van TimeSpentOnGames() |
| Observer Pattern | Observer pattern wordt gebruikt om veranderingen in het model bij te houden en om te weergeven in de views. |
| Abstract Factory Pattern | Een abstract factory voor het maken van entries en sessions factories zodat de de controllers niet afhankelijk zijn van de models. Ook is er een factory voor de OverviewModels |

**SOLID-PRINCIPES**

Beschrijf hieronder voor twee SOLID-principes hoe je die hebt toegepast in Bovenstaand eindontwerp:

|  |  |
| --- | --- |
| **SOLID-principe** | **Beschrijving (gebruik een spatie als je geen toelichting geeft)** |
| Single Responsibility | Scheiding van fucnties zoals data(model), weergave(views), bussiness rules(controller) door mvc architectuur. Het maken van objecten en het gebruik van de objecten is gescheiden. De factories zijn verantwoordelijk voor het maken van de objecten, er wordt dus geen ‘new’ geroepen in classes buiten de factories. |
| Open-Closed | EntryModel is abstract. Als er in de programma nieuwe type EntryModel word toegevoegd kan je de EntryModel extenden en hoeft hij niet worden bijgewerkt |
| Liskov | HomeController kan gebruik maken van alle subclasses van OverviewModel zonder raar gedrag tegen te komen |
| Interface Segragation |  |
| Dependency-Inversion | Hogere level classes zijn niet afhankelijk van lagere klasses, maar van hun abstracties. Zo is bijvoorbeeld de GamesController class niet afhankelijk van de concrete classes GameEntryModel en GameSessionModel maar de abstract classes EntryModel en SessionModel. Dit is wordt deels behaald doordat de class afhankelijk is van een nog hogere level class Factory. |

**REALISATIE**

Als laatste onderdeel voor deze opdracht kopieer je hieronder de tweede versie van de Javacode waarmee je het eindontwerp realiseert (hoewel je de code smells al wel uit deze code mag verwijderen wordt dat pas beoordeeld bij de eindoplevering):

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using MyLibrary.Models;

namespace MyLibrary.Controllers

{

    [Authorize]

    public class AccountController : Controller

    {

        private readonly ApplicationDbContext \_context;

        private readonly SignInManager<ApplicationUser> \_signInManager;

        private readonly UserManager<ApplicationUser> \_userManager;

        public AccountController(ApplicationDbContext context, SignInManager<ApplicationUser> signInManager, UserManager<ApplicationUser> userManager)

        {

            \_context = context;

            \_signInManager = signInManager;

            \_userManager = userManager;

        }

        public async Task<IActionResult> Index()

        {

            return View(await \_userManager.GetUserAsync(User));

        }

        [HttpGet]

        [AllowAnonymous]

        public IActionResult Register()

        {

            return View();

        }

        [HttpPost]

        [AllowAnonymous]

        public async Task<IActionResult> RegisterAsync(RegisterViewModel model)

        {

            if (ModelState.IsValid)

            {

                var user = new ApplicationUser(model.Email, model.Username);

                var result = await \_userManager.CreateAsync(user, model.Password);

                if (result.Succeeded)

                {

                    await \_signInManager.SignInAsync(user, true);

                    return RedirectToAction("Index", "Home");

                }

            }

            return View(model);

        }

        [HttpGet]

        [AllowAnonymous]

        public IActionResult Login()

        {

            return View();

        }

        [HttpPost]

        [AllowAnonymous]

        public async Task<IActionResult> LoginAsync(LoginViewModel model)

        {

            if (ModelState.IsValid)

            {

                var user = await \_userManager.FindByEmailAsync(model.Email);

                if (user != null)

                {

                    var result = await \_signInManager.PasswordSignInAsync(user, model.Password, model.RememberMe, false);

                    if (result.Succeeded)

                    {

                        return RedirectToAction("Index", "Home");

                    }

                }

            }

            ViewData["LoginError"] = "Incorrect email or password";

            return View(model);

        }

        [HttpGet]

        public async Task<IActionResult> LogoutAsync()

        {

            await \_signInManager.SignOutAsync();

            return RedirectToAction("Index", "Home");

        }

        public async Task<IActionResult> SetPrivate()

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            loggedInUser.SetPrivacy(false);

            \_context.SaveChanges();

            return RedirectToAction("Index", "Account");

        }

        public async Task<IActionResult> SetPublic()

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            loggedInUser.SetPrivacy(true);

            \_context.SaveChanges();

            return RedirectToAction("Index", "Account");

        }

        public async Task<IActionResult> PeopleAsync()

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            return View(\_context.ApplicationUsers.Where(u => u.Id != loggedInUser.Id).ToList());

        }

        public async Task<IActionResult> FriendsAsync()

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            return View(\_context.ApplicationUsers.Where(u => u.Id != loggedInUser.Id).ToList());

        }

        public async Task<IActionResult> RemoveFriendAsync(string id)

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            var watchedUser = await \_userManager.FindByIdAsync(id);

            loggedInUser.RemoveFriend(watchedUser);

            \_context.SaveChanges();

            return RedirectToAction("Friends");

        }

        public async Task<IActionResult> AddFriendPAsync(string id)

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            var watchedUser = await \_userManager.FindByIdAsync(id);

            loggedInUser.AddFriend(watchedUser);

            \_context.SaveChanges();

            return RedirectToAction("People");

        }

        public async Task<IActionResult> RemoveFriendPAsync(string id)

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            var watchedUser = await \_userManager.FindByIdAsync(id);

            loggedInUser.RemoveFriend(watchedUser);

            \_context.SaveChanges();

            return RedirectToAction("People");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using MyLibrary.Models;

namespace MyLibrary.Controllers

{

    [Authorize]

    public class BooksController : Controller

    {

        private readonly ApplicationDbContext \_context;

        private readonly UserManager<ApplicationUser> \_userManager;

        public BooksController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)

        {

            \_context = context;

            \_userManager = userManager;

        }

        public async Task<IActionResult> ListAsync(string? id)

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            var watchedUser = await \_userManager.FindByIdAsync(id);

            if (id == null)

            {

                watchedUser = loggedInUser;

            }

            if (watchedUser == null)

            {

                return RedirectToAction("NotFound", "Error");

            }

            if (watchedUser.listsArePublic || loggedInUser.IsFriendsWith(watchedUser))

            {

                var books = \_context.BookEntries.Where(e => e.User == watchedUser).ToList();

                ViewData["Title"] = watchedUser.UserName + "'s books";

                return View(books);

            }

            return RedirectToAction("Private", "Error");

        }

        // GET: Book/Details/5

        public async Task<IActionResult> Details(int? id)

        {

            if (id == null || \_context.BookEntries == null)

            {

                return NotFound();

            }

            var bookEntryModel = await \_context.BookEntries

                .FirstOrDefaultAsync(m => m.Id == id);

            if (bookEntryModel == null)

            {

                return NotFound();

            }

            return View(bookEntryModel);

        }

        // GET: Book/Create

        public IActionResult Create()

        {

            return View();

        }

        // POST: Book/Create

        // To protect from overposting attacks, enable the specific properties you want to bind to.

        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> Create([Bind("Count,Id,Title,ScoreOutOfTen,DateOfEntry")] BookEntryModel bookEntryModel)

        {

            if (ModelState.IsValid)

            {

                bookEntryModel.User = await \_userManager.GetUserAsync(User);

                var session = FactoryProducer.GetFactory(EntryTypes.Book).GetSession();

                session.Count = bookEntryModel.Count;

                session.Entry = bookEntryModel;

                session.User = bookEntryModel.User;

                \_context.Add(bookEntryModel);

                \_context.Add(session);

                await \_context.SaveChangesAsync();

                return RedirectToAction("List");

            }

            return View(bookEntryModel);

        }

        // GET: Book/Edit/5

        public async Task<IActionResult> Edit(int? id)

        {

            if (id == null || \_context.BookEntries == null)

            {

                return NotFound();

            }

            var bookEntryModel = await \_context.BookEntries.FindAsync(id);

            if (bookEntryModel == null)

            {

                return NotFound();

            }

            return View(bookEntryModel);

        }

        // POST: Book/Edit/5

        // To protect from overposting attacks, enable the specific properties you want to bind to.

        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> Edit(int id, int oldTotal,[Bind("Count,Id,Title,ScoreOutOfTen,DateOfEntry")] BookEntryModel bookEntryModel)

        {

            if (id != bookEntryModel.Id)

            {

                return NotFound();

            }

            if (ModelState.IsValid)

            {

                bookEntryModel.User = await \_userManager.GetUserAsync(User);

                try

                {

                    \_context.Update(bookEntryModel);

                    await \_context.SaveChangesAsync();

                }

                catch (DbUpdateConcurrencyException)

                {

                    if (!BookEntryModelExists(bookEntryModel.Id))

                    {

                        return NotFound();

                    }

                    else

                    {

                        throw;

                    }

                }

                var dif = bookEntryModel.Count - oldTotal;

                if(dif != 0)

                {

                    var session = FactoryProducer.GetFactory(EntryTypes.Book).GetSession();

                        session.Count = dif;

                        session.Entry = bookEntryModel;

                        session.User = bookEntryModel.User;

                    \_context.Add(session);

                    \_context.SaveChanges();

                }

                return RedirectToAction("List");

            }

            return View(bookEntryModel);

        }

        // GET: Book/Delete/5

        public async Task<IActionResult> Delete(int? id)

        {

            if (id == null || \_context.BookEntries == null)

            {

                return NotFound();

            }

            var bookEntryModel = await \_context.BookEntries

                .FirstOrDefaultAsync(m => m.Id == id);

            if (bookEntryModel == null)

            {

                return NotFound();

            }

            return View(bookEntryModel);

        }

        // POST: Book/Delete/5

        [HttpPost, ActionName("Delete")]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> DeleteConfirmed(int id)

        {

            if (\_context.BookEntries == null)

            {

                return Problem("Entity set 'ApplicationDbContext.BookEntries'  is null.");

            }

            var bookEntryModel = await \_context.BookEntries.FindAsync(id);

            if (bookEntryModel != null)

            {

                \_context.BookEntries.Remove(bookEntryModel);

            }

            await \_context.SaveChangesAsync();

            return RedirectToAction("List");

        }

        private bool BookEntryModelExists(int id)

        {

          return (\_context.BookEntries?.Any(e => e.Id == id)).GetValueOrDefault();

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using MyLibrary.Models;

namespace MyLibrary.Controllers

{

    [Authorize]

    public class BookSessionsController : Controller

    {

        private readonly ApplicationDbContext \_context;

        private readonly UserManager<ApplicationUser> \_userManager;

        public BookSessionsController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)

        {

            \_context = context;

            \_userManager = userManager;

        }

        public async Task<IActionResult> AddPage(int id)

        {

            var book = \_context.BookEntries.Where(s => s.Id == id).FirstOrDefault();

            if (book != null)

            {

                var session = FactoryProducer.GetFactory(EntryTypes.Book).GetSession();

                session.Count = 1;

                session.Entry = book;

                session.User = await \_userManager.GetUserAsync(User);

                \_context.Add(session);

                book.Count ++;;

                \_context.SaveChanges();

            }

            return RedirectToAction("List", "Books");

        }

        public async Task<IActionResult> RemovePage(int id)

        {

            var book = \_context.BookEntries.Where(s => s.Id == id).FirstOrDefault();

            if (book != null)

            {

                if (book.Count < 1)

                {

                    book.Count = 0;;

                    \_context.SaveChanges();

                    return RedirectToAction("List", "Books");

                }

                var session = FactoryProducer.GetFactory(EntryTypes.Book).GetSession();

                session.Count = -1;

                session.Entry = book;

                session.User = await \_userManager.GetUserAsync(User);

                \_context.Add(session);

                book.Count--;

                \_context.SaveChanges();

            }

            return RedirectToAction("List", "Books");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using MyLibrary.Models;

namespace MyLibrary.Controllers

{

    [Authorize]

    public class FilmsController : Controller

    {

        private readonly ApplicationDbContext \_context;

        private readonly UserManager<ApplicationUser> \_userManager;

        public FilmsController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)

        {

            \_context = context;

            \_userManager = userManager;

        }

        [AllowAnonymous]

        public async Task<IActionResult> ListAsync(string? id)

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            if (loggedInUser == null)

            {

                return RedirectToAction("Login", "Account");

            }

            var watchedUser = await \_userManager.FindByIdAsync(id);

            if (id == null)

            {

                watchedUser = loggedInUser;

            }

            if (watchedUser == null)

            {

                return RedirectToAction("NotFound", "Error");

            }

            if (watchedUser.listsArePublic || loggedInUser.IsFriendsWith(watchedUser))

                {

                    var films = \_context.FilmEntries.Where(e => e.User == watchedUser).ToList();

                    ViewData["Title"] = watchedUser.UserName + "'s films";

                    return View("List", films);

                }

            return RedirectToAction("Private", "Error");

        }

        // GET: Film/Details/5

        public async Task<IActionResult> Details(int? id)

        {

            if (id == null || \_context.FilmEntries == null)

            {

                return NotFound();

            }

            var filmEntryModel = await \_context.FilmEntries

                .FirstOrDefaultAsync(m => m.Id == id);

            if (filmEntryModel == null)

            {

                return NotFound();

            }

            return View(filmEntryModel);

        }

        // GET: Film/Create

        public IActionResult Create()

        {

            return View();

        }

        // POST: Film/Create

        // To protect from overposting attacks, enable the specific properties you want to bind to.

        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> Create([Bind("Count,Id,Title,ScoreOutOfTen,DateOfEntry")] FilmEntryModel filmEntryModel)

        {

            if (ModelState.IsValid)

            {

                filmEntryModel.User = await \_userManager.GetUserAsync(User);

                \_context.Add(filmEntryModel);

                await \_context.SaveChangesAsync();

                return RedirectToAction("List", new {id = \_userManager.GetUserId(User)});

            }

            return View(filmEntryModel);

        }

        // GET: Film/Edit/5

        public async Task<IActionResult> Edit(int? id)

        {

            if (id == null || \_context.FilmEntries == null)

            {

                return NotFound();

            }

            var filmEntryModel = await \_context.FilmEntries.FindAsync(id);

            if (filmEntryModel == null)

            {

                return NotFound();

            }

            return View(filmEntryModel);

        }

        // POST: Film/Edit/5

        // To protect from overposting attacks, enable the specific properties you want to bind to.

        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> Edit(int id, [Bind("Count,Id,Title,ScoreOutOfTen,DateOfEntry")] FilmEntryModel filmEntryModel)

        {

            if (id != filmEntryModel.Id)

            {

                return NotFound();

            }

            if (ModelState.IsValid)

            {

                try

                {

                    \_context.Update(filmEntryModel);

                    await \_context.SaveChangesAsync();

                }

                catch (DbUpdateConcurrencyException)

                {

                    if (!FilmEntryModelExists(filmEntryModel.Id))

                    {

                        return NotFound();

                    }

                    else

                    {

                        throw;

                    }

                }

                return RedirectToAction("List");

            }

            return View(filmEntryModel);

        }

        // GET: Film/Delete/5

        public async Task<IActionResult> Delete(int? id)

        {

            if (id == null || \_context.FilmEntries == null)

            {

                return NotFound();

            }

            var filmEntryModel = await \_context.FilmEntries

                .FirstOrDefaultAsync(m => m.Id == id);

            if (filmEntryModel == null)

            {

                return NotFound();

            }

            return View(filmEntryModel);

        }

        // POST: Film/Delete/5

        [HttpPost, ActionName("Delete")]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> DeleteConfirmed(int id)

        {

            if (\_context.FilmEntries == null)

            {

                return Problem("Entity set 'ApplicationDbContext.FilmEntries'  is null.");

            }

            var filmEntryModel = await \_context.FilmEntries.FindAsync(id);

            if (filmEntryModel != null)

            {

                \_context.FilmEntries.Remove(filmEntryModel);

            }

            await \_context.SaveChangesAsync();

            return RedirectToAction("List");

        }

        private bool FilmEntryModelExists(int id)

        {

          return (\_context.FilmEntries?.Any(e => e.Id == id)).GetValueOrDefault();

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using MyLibrary.Models;

namespace MyLibrary.Controllers

{

    [Authorize]

    public class GamesController : Controller

    {

        private readonly ApplicationDbContext \_context;

        private readonly UserManager<ApplicationUser> \_userManager;

        public GamesController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)

        {

            \_context = context;

            \_userManager = userManager;

        }

        // GET: Games

        public async Task<IActionResult> ListAsync(string? id)

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            var watchedUser = await \_userManager.FindByIdAsync(id);

            if (id == null)

            {

                watchedUser = loggedInUser;

            }

            if (watchedUser == null)

            {

                return RedirectToAction("NotFound", "Error");

            }

            if (watchedUser.listsArePublic || loggedInUser.IsFriendsWith(watchedUser))

            {

                var games = \_context.GameEntries.Where(e => e.User == watchedUser).ToList();

                ViewData["Title"] = watchedUser.UserName + "'s games";

                return View(games);

            }

            return RedirectToAction("Private", "Error");

        }

        // GET: Games/Details/5

        public async Task<IActionResult> Details(int? id)

        {

            if (id == null || \_context.GameEntries == null)

            {

                return NotFound();

            }

            var gameEntryModel = await \_context.GameEntries

                .FirstOrDefaultAsync(m => m.Id == id);

            if (gameEntryModel == null)

            {

                return NotFound();

            }

            return View(gameEntryModel);

        }

        // GET: Games/Create

        public IActionResult Create()

        {

            return View();

        }

        // POST: Games/Create

        // To protect from overposting attacks, enable the specific properties you want to bind to.

        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> Create([Bind("Count,Id,Title,ScoreOutOfTen,DateOfEntry")] GameEntryModel gameEntryModel)

        {

            if (ModelState.IsValid)

            {

                gameEntryModel.User = await \_userManager.GetUserAsync(User);

                var session = FactoryProducer.GetFactory(EntryTypes.Game).GetSession();

                session.Count = gameEntryModel.Count;

                session.Entry = gameEntryModel;

                session.User = gameEntryModel.User;

                \_context.Add(gameEntryModel);

                \_context.Add(session);

                await \_context.SaveChangesAsync();

                return RedirectToAction("List");

            }

            return View(gameEntryModel);

        }

        // GET: Games/Edit/5

        public async Task<IActionResult> Edit(int? id)

        {

            if (id == null || \_context.GameEntries == null)

            {

                return NotFound();

            }

            var gameEntryModel = await \_context.GameEntries.FindAsync(id);

            if (gameEntryModel == null)

            {

                return NotFound();

            }

            return View(gameEntryModel);

        }

        // POST: Games/Edit/5

        // To protect from overposting attacks, enable the specific properties you want to bind to.

        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> Edit(int id, int oldTotal, [Bind("Count,Id,Title,ScoreOutOfTen,DateOfEntry")] GameEntryModel gameEntryModel)

        {

            if (id != gameEntryModel.Id)

            {

                return NotFound();

            }

            if (ModelState.IsValid)

            {

                gameEntryModel.User = await \_userManager.GetUserAsync(User);

                try

                {

                    \_context.Update(gameEntryModel);

                    await \_context.SaveChangesAsync();

                }

                catch (DbUpdateConcurrencyException)

                {

                    if (!GameEntryModelExists(gameEntryModel.Id))

                    {

                        return NotFound();

                    }

                    else

                    {

                        throw;

                    }

                }

                var dif = gameEntryModel.Count - oldTotal;

                if(dif != 0)

                {

                    var session = FactoryProducer.GetFactory(EntryTypes.Game).GetSession();

                    session.Count = dif;

                    session.Entry = gameEntryModel;

                    session.User = gameEntryModel.User;

                    \_context.Add(session);

                    \_context.SaveChanges();

                }

                return RedirectToAction("Index");

            }

            return View(gameEntryModel);

        }

        // GET: Games/Delete/5

        public async Task<IActionResult> Delete(int? id)

        {

            if (id == null || \_context.GameEntries == null)

            {

                return NotFound();

            }

            var gameEntryModel = await \_context.GameEntries

                .FirstOrDefaultAsync(m => m.Id == id);

            if (gameEntryModel == null)

            {

                return NotFound();

            }

            return View(gameEntryModel);

        }

        // POST: Games/Delete/5

        [HttpPost, ActionName("Delete")]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> DeleteConfirmed(int id)

        {

            if (\_context.GameEntries == null)

            {

                return Problem("Entity set 'ApplicationDbContext.GameEntries'  is null.");

            }

            var gameEntryModel = await \_context.GameEntries.FindAsync(id);

            if (gameEntryModel != null)

            {

                \_context.GameEntries.Remove(gameEntryModel);

            }

            await \_context.SaveChangesAsync();

            return RedirectToAction("Index");

        }

        private bool GameEntryModelExists(int id)

        {

          return (\_context.GameEntries?.Any(e => e.Id == id)).GetValueOrDefault();

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using MyLibrary.Models;

namespace MyLibrary.Controllers

{

    [Authorize]

    public class GameSessionsController : Controller

    {

        private readonly ApplicationDbContext \_context;

        private readonly UserManager<ApplicationUser> \_userManager;

        public GameSessionsController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)

        {

            \_context = context;

            \_userManager = userManager;

        }

        public async Task<IActionResult> AddSession(int id)

        {

            var game = \_context.GameEntries.Where(s => s.Id == id).FirstOrDefault();

            if (game != null)

            {

                var session = FactoryProducer.GetFactory(EntryTypes.Game).GetSession();

                session.Count = 30;

                session.Entry = game;

                session.User = await \_userManager.GetUserAsync(User);

                \_context.Add(session);

                game.Count += 30;;

                \_context.SaveChanges();

            }

            return RedirectToAction("List", "Games");

        }

        public async Task<IActionResult> RemoveSession(int id)

        {

            var game = \_context.GameEntries.Where(s => s.Id == id).FirstOrDefault();

            int time = 30;

            if (game != null)

            {

                if (game.Count <= 0)

                {

                    game.Count = 0;

                    \_context.SaveChanges();

                    return RedirectToAction("List", "Games");

                }

                if (game.Count < 30)

                {

                    time = game.Count;

                }

                var session = FactoryProducer.GetFactory(EntryTypes.Game).GetSession();

                session.Count = -time;

                session.Entry = game;

                session.User = await \_userManager.GetUserAsync(User);

                \_context.Add(session);

                game.Count -= time;;

                \_context.SaveChanges();

            }

            return RedirectToAction("List", "Games");

        }

    }

}

using System.Diagnostics;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using MyLibrary.Models;

namespace MyLibrary.Controllers

{

    [Authorize]

    public class HomeController : Controller

    {

        private readonly ApplicationDbContext \_context;

        private readonly UserManager<ApplicationUser> \_userManager;

        public HomeController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)

        {

            \_context = context;

            \_userManager = userManager;

        }

        [AllowAnonymous]

        public async Task<IActionResult> IndexAsync()

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            OverviewModel? model = null;

            ViewData["Warning"] = "";

            if (loggedInUser != null)

            {

                switch(loggedInUser.defaultOverview)

                {

                    case 1: model = OverviewFactory.GetTwoWeekOverview(\_context, loggedInUser); break;

                    case 2: model = OverviewFactory.GetMonthOverview(\_context, loggedInUser); break;

                    default: model = OverviewFactory.GetWeekOverview(\_context, loggedInUser); break;

                }

                ViewData["Warning"] = Warning.GetWarning(OverviewFactory.GetWeekOverview(\_context, loggedInUser));

                return View(model);

            }

            return View();

        }

        public async Task<IActionResult> SetOverviewAsync(int id)

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            loggedInUser.SetDefaultOverview(id);

            \_context.SaveChanges();

            return RedirectToAction("Index");

        }

        [AllowAnonymous]

        public IActionResult Privacy()

        {

            return View();

        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

        public IActionResult Error()

        {

            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using MyLibrary.Models;

namespace MyLibrary.Controllers

{

    [Authorize]

    public class SeriesController : Controller

    {

        private readonly ApplicationDbContext \_context;

        private readonly UserManager<ApplicationUser> \_userManager;

        public SeriesController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)

        {

            \_context = context;

            \_userManager = userManager;

        }

        // GET: Series

        public async Task<IActionResult> ListAsync(string? id)

        {

            var loggedInUser = await \_userManager.GetUserAsync(User);

            var watchedUser = await \_userManager.FindByIdAsync(id);

            if (id == null)

                {

                    watchedUser = loggedInUser;

                }

            if (watchedUser == null)

            {

                return RedirectToAction("NotFound", "Error");

            }

            if (watchedUser.listsArePublic || loggedInUser.IsFriendsWith(watchedUser))

                {

                    var series = \_context.SeriesEntries.Where(e => e.User == watchedUser).ToList();

                    ViewData["Title"] = watchedUser.UserName + "'s series";

                    return View(series);

                }

            return RedirectToAction("Private", "Error");

        }

        // GET: Series/Details/5

        public async Task<IActionResult> Details(int? id)

        {

            if (id == null || \_context.SeriesEntries == null)

            {

                return NotFound();

            }

            var seriesEntryModel = await \_context.SeriesEntries

                .FirstOrDefaultAsync(m => m.Id == id);

            if (seriesEntryModel == null)

            {

                return NotFound();

            }

            return View(seriesEntryModel);

        }

        // GET: Series/Create

        public IActionResult Create()

        {

            return View();

        }

        // POST: Series/Create

        // To protect from overposting attacks, enable the specific properties you want to bind to.

        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> Create([Bind("Id,Title,ScoreOutOfTen,Count,DateOfEntry")] SeriesEntryModel seriesEntryModel)

        {

            if (ModelState.IsValid)

            {

                seriesEntryModel.User = await \_userManager.GetUserAsync(User);

                var session = FactoryProducer.GetFactory(EntryTypes.Series).GetSession();

                session.Count = seriesEntryModel.Count;

                session.Entry = seriesEntryModel;

                session.User = seriesEntryModel.User;

                \_context.Add(seriesEntryModel);

                \_context.Add(session);

                await \_context.SaveChangesAsync();

                return RedirectToAction("List");

            }

            return View(seriesEntryModel);

        }

        // GET: Series/Edit/5

        public async Task<IActionResult> Edit(int? id)

        {

            if (id == null || \_context.SeriesEntries == null)

            {

                return NotFound();

            }

            var seriesEntryModel = await \_context.SeriesEntries.FindAsync(id);

            if (seriesEntryModel == null)

            {

                return NotFound();

            }

            return View(seriesEntryModel);

        }

        // POST: Series/Edit/5

        // To protect from overposting attacks, enable the specific properties you want to bind to.

        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> Edit(int id, int oldTotal, [Bind("Id,Title,ScoreOutOfTen,Count,DateOfEntry")] SeriesEntryModel seriesEntryModel)

        {

            if (id != seriesEntryModel.Id)

            {

                return NotFound();

            }

            if (ModelState.IsValid)

            {

                seriesEntryModel.User = await \_userManager.GetUserAsync(User);

                try

                {

                    \_context.Update(seriesEntryModel);

                    await \_context.SaveChangesAsync();

                }

                catch (DbUpdateConcurrencyException)

                {

                    if (!SeriesEntryModelExists(seriesEntryModel.Id))

                    {

                        return NotFound();

                    }

                    else

                    {

                        throw;

                    }

                }

                var dif = seriesEntryModel.Count - oldTotal;

                if(dif != 0)

                {

                    var session = FactoryProducer.GetFactory(EntryTypes.Series).GetSession();

                    session.Count = dif;

                    session.Entry = seriesEntryModel;

                    session.User = seriesEntryModel.User;

                    \_context.Add(session);

                    \_context.SaveChanges();

                }

                return RedirectToAction("List");

            }

            return View(seriesEntryModel);

        }

        // GET: Series/Delete/5

        public async Task<IActionResult> Delete(int? id)

        {

            if (id == null || \_context.SeriesEntries == null)

            {

                return NotFound();

            }

            var seriesEntryModel = await \_context.SeriesEntries

                .FirstOrDefaultAsync(m => m.Id == id);

            if (seriesEntryModel == null)

            {

                return NotFound();

            }

            return View(seriesEntryModel);

        }

        // POST: Series/Delete/5

        [HttpPost, ActionName("Delete")]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> DeleteConfirmed(int id)

        {

            if (\_context.SeriesEntries == null)

            {

                return Problem("Entity set 'ApplicationDbContext.SeriesEntries'  is null.");

            }

            var seriesEntryModel = await \_context.SeriesEntries.FindAsync(id);

            if (seriesEntryModel != null)

            {

                \_context.SeriesEntries.Remove(seriesEntryModel);

            }

            await \_context.SaveChangesAsync();

            return RedirectToAction("List");

        }

        private bool SeriesEntryModelExists(int id)

        {

          return (\_context.SeriesEntries?.Any(e => e.Id == id)).GetValueOrDefault();

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using MyLibrary.Models;

namespace MyLibrary.Controllers

{

    [Authorize]

    public class SeriesSessionsController : Controller

    {

        private readonly ApplicationDbContext \_context;

        private readonly UserManager<ApplicationUser> \_userManager;

        public SeriesSessionsController(ApplicationDbContext context, UserManager<ApplicationUser> userManager)

        {

            \_context = context;

            \_userManager = userManager;

        }

        public async Task<IActionResult> AddEpisode(int id)

        {

            var series = \_context.SeriesEntries.Where(s => s.Id == id).FirstOrDefault();

            if (series != null)

            {

                var session = FactoryProducer.GetFactory(EntryTypes.Series).GetSession();

                session.Count = 1;

                session.Entry = series;

                session.User = await \_userManager.GetUserAsync(User);

                \_context.Add(session);

                series.Count ++;;

                \_context.SaveChanges();

            }

            return RedirectToAction("List", "Series");

        }

        public async Task<IActionResult> RemoveEpisode(int id)

        {

            var series = \_context.SeriesEntries.Where(s => s.Id == id).FirstOrDefault();

            if (series != null)

            {

                if (series.Count < 1)

                {

                    series.Count = 0;;

                    \_context.SaveChanges();

                    return RedirectToAction("List", "Series");

                }

                var session = FactoryProducer.GetFactory(EntryTypes.Series).GetSession();

                session.Count = -1;

                session.Entry = series;

                session.User = await \_userManager.GetUserAsync(User);

                \_context.Add(session);

                series.Count--;

                \_context.SaveChanges();

            }

            return RedirectToAction("List", "Series");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

using MyLibrary.Models;

    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>

    {

        public ApplicationDbContext (DbContextOptions<ApplicationDbContext> options)

            : base(options)

        {

        }

        public DbSet<ApplicationUser> ApplicationUsers { get; set; }

        public DbSet<Friendship> Friendships { get; set; }

        public DbSet<FilmEntryModel> FilmEntries { get; set; }

        public DbSet<BookEntryModel> BookEntries { get; set; }

        public DbSet<SeriesEntryModel> SeriesEntries { get; set; }

        public DbSet<GameEntryModel> GameEntries { get; set; }

        public DbSet<SeriesSessionModel> SeriesSessions { get; set; }

        public DbSet<BookSessionModel> BookSessions { get; set; }

        public DbSet<GameSessionModel> GameSessions { get; set; }

    }

using System.ComponentModel.DataAnnotations;

namespace MyLibrary.Models

{

    public class BookEntryModel: EntryModel

    {

        public ICollection<SessionModel>? Sessions { get; set; }

    }

}

#nullable disable

using System.ComponentModel.DataAnnotations;

namespace MyLibrary.Models

{

    public abstract class EntryModel

    {

        [Key]

        public int Id { get; set; }

        public String Title { get; set; }

        [Range(0, 10)]

        [Display(Name ="Score")]

        public int ScoreOutOfTen { get; set; }

        [Range(0, int.MaxValue)]

        public int Count { get; set; }

        [DataType(DataType.Date)]

        [Display(Name ="Date of entry")]

        public DateTime DateOfEntry { get; set; }

        public ApplicationUser User { get; set; }

    }

}

using System.ComponentModel.DataAnnotations;

namespace MyLibrary.Models

{

    public class FilmEntryModel : EntryModel

    {

    }

}

using System.ComponentModel.DataAnnotations;

namespace MyLibrary.Models

{

    public class GameEntryModel: EntryModel

    {

        public ICollection<SessionModel>? Sessions { get; set; }

    }

}

using System.ComponentModel.DataAnnotations;

namespace MyLibrary.Models

{

    public class SeriesEntryModel: EntryModel

    {

        public ICollection<SessionModel>? Episodes { get; set; }

    }

}

namespace MyLibrary.Models

{

    public class BookFactory : Factory

    {

        public override EntryModel GetEntry()

        {

            return new BookEntryModel{DateOfEntry = DateTime.UtcNow};

        }

        public override SessionModel GetSession()

        {

            return new BookSessionModel{DateOfSession = DateTime.UtcNow};

        }

    }

}

namespace MyLibrary.Models

{

    public abstract class Factory

    {

        public abstract EntryModel GetEntry();

        public abstract SessionModel GetSession();

    }

    public enum EntryTypes

    {

        Film,

        Series,

        Book,

        Game

    }

}

namespace MyLibrary.Models

{

    public static class FactoryProducer

    {

        public static Factory GetFactory(EntryTypes type)

        {

            if (type == EntryTypes.Film)

                return new FilmFactory();

            if (type == EntryTypes.Book)

                return new BookFactory();

            if (type == EntryTypes.Game)

                return new GameFactory();

            if (type == EntryTypes.Series)

                return new SeriesFactory();

            throw new ArgumentException("Incorrect factory type specified");

        }

    }

}

namespace MyLibrary.Models

{

    public class FilmFactory : Factory

    {

        public override EntryModel GetEntry()

        {

            return new FilmEntryModel{DateOfEntry = DateTime.UtcNow};

        }

        public override SessionModel GetSession()

        {

            throw new NotImplementedException();

        }

    }

}

namespace MyLibrary.Models

{

    public class GameFactory : Factory

    {

        public override EntryModel GetEntry()

        {

            return new GameEntryModel{DateOfEntry = DateTime.UtcNow};

        }

        public override SessionModel GetSession()

        {

            return new GameSessionModel{DateOfSession = DateTime.UtcNow};

        }

    }

}

using Microsoft.AspNetCore.Identity;

namespace MyLibrary.Models

{

    public static class OverviewFactory

    {

        public static OverviewModel GetWeekOverview( ApplicationDbContext \_context,  ApplicationUser \_user)

        {

            return new WeekOverviewModel(\_context, \_user);

        }

        public static OverviewModel GetTwoWeekOverview( ApplicationDbContext \_context,  ApplicationUser \_user)

        {

            return new TwoWeekOverviewModel(\_context, \_user);

        }

        public static OverviewModel GetMonthOverview( ApplicationDbContext \_context,  ApplicationUser \_user)

        {

            return new MonthOverviewModel(\_context, \_user);

        }

    }

}

namespace MyLibrary.Models

{

    public class SeriesFactory : Factory

    {

        public override EntryModel GetEntry()

        {

            return new SeriesEntryModel{DateOfEntry = DateTime.UtcNow};

        }

        public override SessionModel GetSession()

        {

            return new SeriesSessionModel{DateOfSession = DateTime.UtcNow};

        }

    }

}

using System.ComponentModel.DataAnnotations;

using Microsoft.AspNetCore.Identity;

namespace MyLibrary.Models

{

    public class ApplicationUser: IdentityUser

    {

        [Range(0,2)]

        public int defaultOverview { get; set; }

        public Boolean listsArePublic { get; set; }

        public ICollection<ApplicationUser>? Friends { get; set; }

        public ApplicationUser()

        {

        }

        public ApplicationUser(string email, string username)

        {

            this.Email = email;

            this.UserName = username;

        }

        public bool IsFriendsWith(ApplicationUser user)

        {

            if (this != user && Friends != null)

            {

                return Friends.Contains(user);

            }

             return true;

        }

        public void SetPrivacy(bool set)

        {

            listsArePublic = set;

        }

        public void SetDefaultOverview(int period)

        {

            defaultOverview = period;

        }

        public void AddFriend(ApplicationUser user)

        {

            if(Friends != null)

            {

                Friends.Add(user);

            }

        }

        public void RemoveFriend(ApplicationUser user)

        {

            if (Friends != null)

            {

                Friends.Remove(user);

            }

        }

    }

}

#nullable disable

using Microsoft.EntityFrameworkCore;

namespace MyLibrary.Models

{

    public class Friendship

    {

        public int Id { get; set; }

        public ApplicationUser UserOne { get; set; }

        public ApplicationUser UserTwo { get; set; }

    }

}

#nullable disable

using System.ComponentModel.DataAnnotations;

namespace MyLibrary.Models

{

    public class LoginViewModel

    {

        [Required]

        [DataType(DataType.EmailAddress)]

        public string Email { get; set; }

        [Required]

        [DataType(DataType.Password)]

        public string Password { get; set; }

        [Display(Name = "Remember me")]

        public bool RememberMe { get; set; }

    }

}

#nullable disable

using System.ComponentModel.DataAnnotations;

namespace MyLibrary.Models

{

    public class RegisterViewModel

    {

        [Required]

        [DataType(DataType.EmailAddress)]

        public string Email { get; set; }

        [Required]

        [MaxLength(16)]

        public string Username { get; set; }

        [Required]

        [DataType(DataType.Password)]

        public string Password { get; set; }

        [DataType(DataType.Password)]

        [Compare("Password", ErrorMessage = "Passwords do not match")]

        [Display(Name = "Confirm password")]

        public string ConfirmPassword { get; set; }

    }

}

namespace MyLibrary.Models

{

    public class MonthOverviewModel : OverviewModel

    {

        public MonthOverviewModel(ApplicationDbContext context, ApplicationUser user)

        {

            \_context = context;

            \_user = user;

            CreateOverview();

        }

        protected override void SetTimeSpentOnFilms()

        {

            var date = DateTime.UtcNow.AddDays(-28);

            var totalTime = \_context.FilmEntries.Where(u => u.User == \_user).Where(f => f.DateOfEntry >= date).Sum(x => x.Count);

            base.TimeSpentOnFilmsInMinutes = totalTime;

        }

        protected override void SetEpisodesWatchedOfSeries()

        {

            var date = DateTime.UtcNow.AddDays(-28);

            var totalEpisodes = \_context.SeriesSessions.Where(u => u.User == \_user).Where(f => f.DateOfSession >= date).Sum(x => x.Count);

            EpisodesWatchedOfSeries = totalEpisodes;

        }

        protected override void SetTimeSpentOnGames()

        {

            var date = DateTime.UtcNow.AddDays(-28);

            var totalTime = \_context.GameSessions.Where(u => u.User == \_user).Where(f => f.DateOfSession >= date).Sum(x => x.Count);

            TimeSpentOnGamesInMinutes = totalTime;

        }

        protected override void SetPagesReadOfBooks()

        {

            var date = DateTime.UtcNow.AddDays(-28);

            var totalTime = \_context.BookSessions.Where(u => u.User == \_user).Where(f => f.DateOfSession >= date).Sum(x => x.Count);

            PagesReadOfBooks = totalTime;

        }

        protected override void SetPeriod()

        {

            Period = "month";

        }

    }

}

namespace MyLibrary.Models

{

    public abstract class OverviewModel

    {

        protected ApplicationDbContext \_context;

        protected ApplicationUser \_user;

        public string? Period { get; protected set; }

        public int TimeSpentOnFilmsInMinutes { get; protected set; }

        public int EpisodesWatchedOfSeries { get; protected set; }

        public int PagesReadOfBooks { get; protected set; }

        public int TimeSpentOnGamesInMinutes { get; protected set; }

        public void CreateOverview()

        {

            SetPeriod();

            SetTimeSpentOnFilms();

            SetEpisodesWatchedOfSeries();

            SetPagesReadOfBooks();

            SetTimeSpentOnGames();

            CheckNegatives();

        }

        protected abstract void SetPeriod();

        protected abstract void SetTimeSpentOnFilms();

        protected abstract void SetEpisodesWatchedOfSeries();

        protected abstract void SetTimeSpentOnGames();

        protected abstract void SetPagesReadOfBooks();

        protected void CheckNegatives()

        {

            if (TimeSpentOnFilmsInMinutes < 0)

            {

                TimeSpentOnFilmsInMinutes = 0;

            }

            if (EpisodesWatchedOfSeries < 0)

            {

                EpisodesWatchedOfSeries = 0;

            }

            if (PagesReadOfBooks < 0)

            {

                PagesReadOfBooks = 0;

            }

            if (TimeSpentOnGamesInMinutes < 0)

            {

                TimeSpentOnGamesInMinutes = 0;

            }

        }

    }

}

namespace MyLibrary.Models

{

    public class TwoWeekOverviewModel : OverviewModel

    {

        public TwoWeekOverviewModel(ApplicationDbContext context, ApplicationUser user)

        {

            \_context = context;

            \_user = user;

            CreateOverview();

        }

        protected override void SetTimeSpentOnFilms()

        {

            var date = DateTime.UtcNow.AddDays(-14);

            var totalTime = \_context.FilmEntries.Where(u => u.User == \_user).Where(f => f.DateOfEntry >= date).Sum(x => x.Count);

            base.TimeSpentOnFilmsInMinutes = totalTime;

        }

        protected override void SetEpisodesWatchedOfSeries()

        {

            var date = DateTime.UtcNow.AddDays(-14);

            var totalEpisodes = \_context.SeriesSessions.Where(u => u.User == \_user).Where(f => f.DateOfSession >= date).Sum(x => x.Count);

            EpisodesWatchedOfSeries = totalEpisodes;

        }

        protected override void SetTimeSpentOnGames()

        {

            var date = DateTime.UtcNow.AddDays(-14);

            var totalTime = \_context.GameSessions.Where(u => u.User == \_user).Where(f => f.DateOfSession >= date).Sum(x => x.Count);

            TimeSpentOnGamesInMinutes = totalTime;

        }

        protected override void SetPagesReadOfBooks()

        {

            var date = DateTime.UtcNow.AddDays(-14);

            var totalTime = \_context.BookSessions.Where(u => u.User == \_user).Where(f => f.DateOfSession >= date).Sum(x => x.Count);

            PagesReadOfBooks = totalTime;

        }

        protected override void SetPeriod()

        {

            Period = "two weeks";

        }

    }

}

namespace MyLibrary.Models

{

    public class WeekOverviewModel : OverviewModel

    {

        public WeekOverviewModel(ApplicationDbContext context, ApplicationUser user)

        {

            \_context = context;

            \_user = user;

            CreateOverview();

        }

        protected override void SetTimeSpentOnFilms()

        {

            var date = DateTime.UtcNow.AddDays(-7);

            var totalTime = \_context.FilmEntries.Where(u => u.User == \_user).Where(f => f.DateOfEntry >= date).Sum(x => x.Count);

            base.TimeSpentOnFilmsInMinutes = totalTime;

        }

        protected override void SetEpisodesWatchedOfSeries()

        {

            var date = DateTime.UtcNow.AddDays(-7);

            var totalEpisodes = \_context.SeriesSessions.Where(u => u.User == \_user).Where(f => f.DateOfSession >= date).Sum(x => x.Count);

            EpisodesWatchedOfSeries = totalEpisodes;

        }

        protected override void SetTimeSpentOnGames()

        {

            var date = DateTime.UtcNow.AddDays(-7);

            var totalTime = \_context.GameSessions.Where(u => u.User == \_user).Where(f => f.DateOfSession >= date).Sum(x => x.Count);

            TimeSpentOnGamesInMinutes = totalTime;

        }

        protected override void SetPagesReadOfBooks()

        {

            var date = DateTime.UtcNow.AddDays(-7);

            var totalTime = \_context.BookSessions.Where(u => u.User == \_user).Where(f => f.DateOfSession >= date).Sum(x => x.Count);

            PagesReadOfBooks = totalTime;

        }

        protected override void SetPeriod()

        {

            Period = "week";

        }

    }

}

namespace MyLibrary.Models

{

    public class BookSessionModel: SessionModel

    {

    }

}

namespace MyLibrary.Models

{

    public class GameSessionModel: SessionModel

    {

    }

}

namespace MyLibrary.Models

{

    public class SeriesSessionModel: SessionModel

    {

    }

}

#nullable disable

namespace MyLibrary.Models

{

    public abstract class SessionModel

    {

        public int Id { get; set; }

        public int Count { get; set; }

        public DateTime DateOfSession { get; set; }

        public EntryModel Entry { get; set; }

        public ApplicationUser User { get; set; }

    }

}

namespace MyLibrary.Models

{

    public static class Warning

    {

        public static string GetWarning(OverviewModel model)

        {

            var totalTime = model.TimeSpentOnFilmsInMinutes + model.TimeSpentOnGamesInMinutes + model.EpisodesWatchedOfSeries \* 45;

            if (totalTime > 1320)

                return "You REALLY should spent less time on this";

            if (totalTime > 850)

                return "You should spent less time on this";

            return "";

        }

    }

}