------------------------------------- FRONT END --------------------------------------------------

1. Wat is het verschil tussen een gewone (function) component en een component die gewrapt is in React.memo()?Hoe heet deze laatste?Heb je deze laatste gebruikt? Waarom wel/niet?

* De memo functie wordt gebruikt om een component te creëren die enkel zal rerenderen als de props wijzigen.
* React.memo() vergelijkt de props van de componenten en voorkomt opnieuw renderen als de props niet zijn gewijzigd.
* Dit zorgt ervoor dat de component alleen opnieuw wordt gerenderd als de props veranderen, waardoor onnodige renders worden voorkomen en de prestaties worden verbeterd.
* Als je React.memo() niet gebruikt, zal een functionele component altijd opnieuw renderen wanneer deze wordt aangeroepen, ongeacht of de props of de staat zijn gewijzigd. Het gebruik van React.memo() helpt om onnodige renders te voorkomen door memoïsering toe te passen op basis van de props.
* VOORBEELD in FullGameslist.jsx op ln 16 gebruik ik useMemo voor het zoeken van een game, wanneer de searchvariabele verandert wordt de lijst van de games gefiltert en opnieuw gerendert

1. Leg,a.d.h.v. een voorbeeld uit jouw code, kort uit hoe routing werkt.

Pages.jsx: In dit bestand worden React-componenten gedefinieerd voor verschillende pagina's van de applicatie.

Main.jsx: In dit bestand wordt de routing geconfigureerd en de applicatie gestart. Op lijn 25 en 66 worden appart de adminroutes geconfigureerd.

Voorbeeld:

* Gebruiker wilt naar /lists navigeren
* De react router (ln 46 in main.jsx) controleert of de route gedefinieerd is, zo niet -> render het NotFound component(ln 103). Zo wel, dan wordt het bijbehorende component voor /lists gerenderd, in het geval van /lists is dit AllPlayLists(ln 87) hier wordt ook eerst gecontroleerd of de user is ingelogd via de PrivateRouter component(ln 83).

1. Wat is een context? Waarvoor kan dit gebruikt worden?Heb jij dit gebruikt? Zo ja, hoe werkt deze?Indien niet, hoe deel jij data tussen componenten, geef een voorbeeld.

* In React kan een context worden gebruikt om gegevens te delen tussen componenten zonder de tussenliggende componenten handmatig te doorlopen en de gegevens als props door te geven. Dit is vooral handig wanneer bepaalde gegevens op veel plaatsen in de componentenboom nodig zijn
* Voordbeeld: in auth.context.jsx is er een context aangemaakt voor deze gegevens: token, user, error, ready, loading, isAuthed, login, logout te delen tussen componenten. In NavBar.jsx wordt de useAuth functie gebruikt om de gegevens op te halen uit te context en te controleren of de gebruiker is ingelogd of niet

1. Wat zijn React Hooks? Waarom mogen deze niet gebruikt worden binnen iteraties of selecties? Toon a.d.h.v. een voorbeeld uit jouw code.

* React Hooks zijn functies die het mogelijk maken om state en andere React-functionaliteiten toe te voegen aan functionele componenten: useState, useEffect zijn voorbeelden.
* Gebruik hooks enkel op het top niveau. Gebruik hooks niet binnen een if, andere condities, loops of geneste functies.
* De reden hierachter is dat React de volgorde van Hooks bij elke render moet weten, zodat het de juiste associaties kan behouden tussen de hooks en de component-state. Als je Hooks binnen een lus of selectie zou gebruiken, zou de volgorde van de hooks kunnen variëren, wat leidt tot onverwacht gedrag en bugs.
  1. Reden: React valt terug op de volgorde waarin hooks worden aangeroepen om een waarde terug te geven. React houdt dit bij in een array. De volgorde moet dezelfde zijn bij elke render.
  2. Roep hooks enkel aan vanuit React functies. Dit wil zeggen: enkel vanuit function components of vanuit eigen geschreven hooks.

1. Leg de useState() hookuit a.d.h.v. een voorbeeld uit jouw code.

* De [useState](https://reactjs.org/docs/hooks-reference.html" \l "usestate" \t "_blank) hook wordt gebruikt om een component stateful te maken. Hij geeft een component met andere woorden state.
* Voorbeeld: gamedetails ln 22 tot 32 worde useStates gedeclareerd bv op lijn 23: game: een stukje state om informatie over het huidige spel bij te houden. Wanneer je naar de pagina GameDetails navigeert, worden de useState-variabelen ingevuld met de huidige waarden van het bijbehorende spel. Dit gebeurt via het gebruik van de useEffect-hook op lijn 34. De functie setGame (ln40) update hier de state met de bijhorende data van de huidige game.

1. Leg de useContext() hookuit a.d.h.v. een voorbeeld uit jouw code.

* In auth.context.jsx op lijn 18 Definieer een hook genaamd useAuth die de useContext-hook gebruikt om toegang te krijgen tot de waarden binnen de AuthContext. Wanneer je dan in een andere component useAuth gebruikt wordt telkens de context opgehaalt: user, isAuthed, login, logout, ….

1. Leg de useCallback() hookuit a.d.h.v. een voorbeeld uit jouw code.Deze hook krijgt altijd een array mee als parameter. Wat stelt deze array voor? Waarom is dit nodig?

In auth.context.jsx op ln 38

* useCallback wordt gebruikt om de login-functie te memoïseren.
* De functie doLogin is een callback-functie die afkomstig is van useSWRMutation. Aangezien het waarschijnlijk een callback-functie is die kan veranderen tussen renders, willen we voorkomen dat login opnieuw wordt gemaakt wanneer de component wordt hergerenderd.
* De afhankelijkheidsarray [doLogin] geeft aan dat als doLogin verandert, de gememoïseerde versie van login opnieuw wordt gemaakt. Als doLogin constant is tussen renders, zal de gememoïseerde versie van login dezelfde blijven, waardoor onnodige re-renders worden voorkomen.
* De login-functie wordt vaak doorgegeven aan geoptimaliseerde kindcomponenten als een prop. Door useCallback te gebruiken, kunnen we ervoor zorgen dat deze prop niet elke keer opnieuw wordt gemaakt, wat de prestaties kan verbeteren.
* Voorbeeld: in auth.context.jsx op lijn 38 wordt de functie login gedeckareerd waarmee een gebruiker kunnen inloggen, deze wordt gewrapped in een useCallback

1. Legde useMemo() hookuit a.d.h.v. een voorbeeld uit jouw code.Deze hook krijgt altijd een array mee als parameter. Wat stelt deze array voor? Waarom is dit nodig?

* VOORBEELD in fullgameslist.jsx op ln 16 wordt useMemo gebruikt om een gefilterde lijst van games te maken op basis van de zoektekst (search) en de lijst met alle games (games). Hier is de array [search, games] de afhankelijkheidsarray, als 1 van deze variabelen verandert, dan wordt de memo opnieuw uitgevoerd. Dit is nodig zodat de filtering enkel gebeurt wanneer het nodig is.

1. Legde useEffect() hookuit a.d.h.v. een voorbeeld uit jouw code.Deze hook krijgt altijd een array mee als parameter. Wat stelt deze array voor? Waarom is dit nodig?

* VOOREELD ln 34 in gamedetails.js de useEffect wordt gebruikt om de details van een game op te halen nadat de ID in de URL is gewijzigt. De array bevat dus de waarde , wanneer deze verandert wordt de useEffect uitgevoerd. Dus wanneer ik in de url van 5 naar 4 zou gaan, wordt de pagina geupdate naar game met id 4.

1. Wanneer zou je kiezen voor useState() en voor useContext()?Leg uit a.d.h.v. een voorbeeld uit jouw code.

* useContext voor het delen van een state over meerdere componenten
* useState als de state enkel binnen deze component nodig is
* VOORBEELD: in fullgameslist.js op ln 10 gebruik ik useState omdat de games enkel belangrijk zijn voor deze component, ik heb die nergens anders nodig, anders kon ik useContext gebruiken
* VOORBEELD in Auth.context.jsx op ln 18 heb ik useContext gebruikt omdat we de waarden zoals user,token,isAuthed,login,logout nodig heb in andere componenten

1. Wanneer zal een component opnieuw renderen? Kan een van deze gevallen overbodig zijn? Zo ja, welk geval en waarom?Leg uit a.d.h.v. een voorbeeld uit jouw code.

* Verandering van state
* Props veranderen
* Context veranderen
* Geforceerde herrenderen
* Met useEffect
* VOORBEELD: als een component een prop ontvangt dat geen enkele invloed heeft op het uiterlijk of gedrag, moet deze niet altijd herrenderen.
* Heb geen voorbeeld in code maar weet wel dat wanneer deze opnieuw renderen

1. Toon en bespreek 2 à 3 van de testen die je geschreven hebt.
2. Hoe zorg je ervoor dat niet-ingelogde gebruikers bepaalde delen van de applicatie niet zien?Leg uit a.d.h.v. een voorbeeld uit jouw code.

* In main.jsx kunnen we zien welke paginas een niet-ingelogde gebruiker niet kan bezoeken bv /profile, er wordt eerst gechecked via PrivateRoute component of de gebruiker is ingelogd of niet.

1. Leg uit hoe een entiteit uit jouw applicatie toegevoegd en verwijderd wordt. Wat gebeurt er allemaal?

* Bv het verwijderen van een game: in EditGames.jsx op ln 170 als de gebruiker op delete klikt, wordt de gameToDelete id op de correcte game gezet, en zet showDeleteConfirmation op true zodat er een confirmdisplay te zien komt (ln 179). Als je op confirm klikt in de popup, wwordt de confirmDelete functie (ln 182) uitgevoerd hierin wordt er op ln 93 een api call gedaan en de game wordt verwijderd. Als dit is gelukt, wordt de games state geupdate door de verwijderde game te verwijderen uit de array. De games worden opnieuw gerendert omdat de games state is verandert
* Bv bij het toevoegen van een review: in gameDetails.jsx op ln 189 wanneer je op add review klikt wordt de handleAddReviewClick functie uitgevoerd, de reviewModal wordt geopend door onOpen() uit te voeren deze zet de isOpen(ln232) state van useDiscolsure op true. Wanneer de gegevens zijn ingevoert en de gebruiker klikt op submit, wordt de handleReviewSubmit functie uitgevoerd als alles correct is (lengte,…) dan wordt de review aangemaakt met een api call createReview. De modal wordt terug leeggemaakt en geclosed. De functie addNewReview wordt uitgevoerd en doet api call om alle reviews opnieuw te verkijgen. De reviews state wordt geupdate via setReviews.

1. Wat is vite? Waarvoor wordt het gebruikt binnen een React applicatie?

* [**Vite**](https://vitejs.dev/) is een **buildtool** en **ontwikkelingsserver** die voornamelijk wordt gebruikt voor het bouwen van moderne webtoepassingen op een snelle en efficiente manier
* Bied een snelle ontwikkelingsserver aan waardoor developers wijzigingen in hun code direct kunnen zien zonder de applicatie opnieuw to runnen

1. Hoe werken FormProvider en useFormContext()van react-hook-form? Wat wordt achterliggend gebruikt?Leg uit a.d.h.v. een voorbeeld uit jouw code.

* FormProvider is een contextprovider die wordt gebruikt om het formuliercontext aan zijn kinderen door te geven. Het zorgt ervoor dat het formuliercontext beschikbaar is voor alle componenten binnen zijn bereik.
* In pages op ln 86 creëren we een formulier met login en gebruiken we FormProvider om het formuliercontext (methods in dit geval) door te geven aan alle kinderen binnen de <FormProvider>-tags.
* useFormContext() is een hook waarmee je toegang krijgt tot het formuliercontext binnen een component dat is ingesloten in een FormProvider.

1. Wat is jouw extra functionaliteit? Hoe werkt het?

* Ik heb chakra UI gebruikt, dit is een open source react-componentbibliotheek. Dit heeft mij geholpen omdat ik dan minder css moest schrijven, en zeker de modals en popups zijn super handig om te gebruiken

1. Waarom is het een slecht idee om objecten/functiesop volgende manier mee te geven aan props? Wat is een oplossing voor deze fouten?

<div style={{ color: ‘black’ }}onClick={() => console.log(‘div clicked’)}/>

* Het gebruik van arrow functies in de onClick-prop kan interfereren met memoization-technieken, zoals React.memo of useMemo. Deze technieken zijn bedoeld om te voorkomen dat een component opnieuw wordt weergegeven als de props niet zijn gewijzigd

----------------------------------------- BACKEND -------------------------------------------------------------------------

1. Toon en bespreek 2 a 3 van de testen die je geschreven hebt
2. Wat zijn de conventies om een API RESTful te noemen? Welke heb je toegepast in jouw projecten waarom?Waarom heb je de andere niet toegepast?

- Uniforme interfance aan de hand van: Gebruik van de http methoden voor CRUD operaties: GET, POST, PUT, DELETE.

-Client-server: Dit scheidt de presentatielaag van de data-laag, wat betekent dat de server alleen zorgt voor het verwerken van verzoeken en het terugsturen van responses, terwijl de client zich richt op het weergeven van deze data aan de gebruiker.

-Nesten van resources bv in rest->playlist.js->ln160 worden er 2 id’s genest in de url

-Gebruik van JSON bodys als communicatie

1. Wat zijn NodeJS en Koa? Leg kort de werking van middlewares uita.d.h.v. een voorbeeld uit jouw code. Wat is er speciaal aan middlewares in Koa?Wat is bv. het verschil tussen await next() en return next() in een middleware?

-NodeJS is een omgeving om JavaScript te gebruiken om server-side applications te maken.

-Koa is een framework voor NodeJS om het maken van webapps en APIs gemakkelijker maken.

-Middlewares zijn functies die worden uitgevoerd tussen het ontvangen van een verzoek en het terugsturen van een response

-Voorbeeld: in games.js op ln 142 ontvangen we een get / request, wanneer we op ln 24 de functie uitvoeren wordt dit gezien als het starten van de middleware, dab gebruiken we de servicelaag om de data op te halen, die dan in de ctx.body op ln 34 wordt gestoken. Dit is het einde van de middleware en zo gaanwe door naar de volgende.

-Als je await next() gebruikt, zal de middleware wachten totdat de volgende middleware in de stapel is uitgevoerd voordat de uitvoering wordt voortgezet.

- Als je return next() gebruikt, zal de middleware direct doorgaan naar de volgende middleware zonder te wachten totdat deze is uitgevoerd.

1. Leg, a.d.h.v. een voorbeeld uit jouw code,kort uit hoe routing werkt.

-Routing is het proces waarbij wordt bepaald welk pad inkomend netwerkverkeer moet volgen om zijn bestemming te bereiken.

-We maken gebruik van Koa framework om routes te bepalen. VOORBEELD: ln 135 in rest-> games.js maken we een router aan met prefix ‘/games’, de routes voor de verschillende requests (get, post, …) worden hier ook gedefinieerd. Wanneer we bv een request ontvangen met adres GET /games/5, wordt de functie op ln 143 uitgevoerd met id 5.

1. Welke mogelijkheden zijn er op het gebied van datalaag in NodeJS? Wanneer kies je voor elke mogelijkheid?Welke heb jij gekozenen waarom?Hoe heb je dit dan geïmplementeerd?

-       Zelf queries schrijven :SQL in stringvorm opgebouwd; resultaat uit databank moet ook terug verwerkt worden

-       Query builder (bijvoorbeeld knex) :Gebruik van knex als framework om met de database te praten

-       Object Relational Mapper (ORM) Bv prisma als communicatie met DB

- ik heb voor ORM gekozen omdat dit een leuke extra mogelijkheid leek voor mijn project en na een beetje research te doen leek dit handiger dan knex aangezien ik geen repo laag meer nodig heb. Ik heb gebruik gemaakt van prisma en heb de prisma quickstart guide gebruikt om deze te implementeren.

1. Hoe zorg je ervoor dat enkel ingelogde gebruikers/gebruikers met een bepaalde rol een resource kunnen raadplegen?Leg uit a.d.h.v. een voorbeeld uit jouw code.

-VOORBEELD op ln 142 in rest->games.js wordt er gebruik gemaakt van requireAuthentication,dit controleert of de gebruiker is ingelogd of niet. Zoniet, dan kan dit request niet uitgevoerd worden.

-VOORBEELD op ln 144 in rest)> games.js wordt er gebruik gemaakt van requireAdmin om een game aan te maken. Enkel admin kan dit doen. We checken of de gebruiker in de array Roles, de rol ADMIN bevat.

1. Wat is async/await? Wat is de relatie met Promises?Leg uit a.d.h.v. jouw code.

-VOORBEELD: rest-> games.js ln 24 wordt aync gebruikt, dit is een asynchrone functie die een promise returned. De await op ln 25 wordt gebruikt om te wachten op voltooiing van de promise. Het pauzeert de werking van de asyncrhrone functie tot de promise is opgelost of afgewezen. Dus async/await is een makkelijkere manier om asyncrhone code te schrijven, terwijl je eigenlijk nog gebruik maakt van promises. Het maakt de code leesbaarder

1. Toon en bespreekjouw package.json. Wat staat hier allemaal in? Wat is het verschil tussen dependencies en devDependencies?

-Depencencies: De modules die hier worden vermeld, zijn de externe bibliotheken en frameworks die nodig zijn voor de werking van de applicatie in een productieomgeving.

-devDependencies: De modules hier zijn ook externe bibliotheken en tools, maar ze zijn alleen nodig tijdens de ontwikkeling van de applicatie. VOORBEELD: prisma is een devdependencie want prisma wordt gebruikt tijdens de onwikkeling van het project. Tijdens productie hebben we enkel het prisma schema nodig schema.prisma

1. Wat is een linter? Wat is het nut ervan?

- Een linter is een tool die wordt gebruikt om de broncode van een programma te analyseren op mogelijke programmeerfouten, stijlproblemen en andere problemen volgens vooraf gedefinieerde regels of conventies.

- ik kan bv instellen dat ik enkel ‘ haakjes heb in mijn project en nooit “ met een linter.

1. Hoe werkt jouw datalaag?Hoe krijg je data in de databank in development mode? Hoe wordt het databankschema up to date gehouden?Leg uit a.d.h.v. voorbeelden uit jouw code.

-Ik maak gebruik van een ORM met prisma. Het bestand migrations.sql legt de structuur vast van de database en maakt deze aan. Dan wordt het prisma schema gegenereerd op basis van de database. Dit bestand wordt tijdens productie gebruikt door de Prisma Client om aanpassingen te doen bv data ophalen, verwijderen, toevoegen. Het seed bestand zorgt het vullen van de DB met data.

- Als er wijzigingen zijn gebeurt in de database kunnen we het prisma schema update met prisma db pull

1. Wat is jouw extra functionaliteit? Hoe werkt het?

Zie vraag 10

1. Welke lagen heb je typisch in een gelaagde applicatie? Welke lagen heb jij geimplementeerd? Wat is het nut/doel van elke laag?Hoe projecteer je de lagen presentatie/domein/persistentie hierop?

-rest laag = Is het aanspreekpunt van de API kan dus gezien worden als de presentatie laag

-servicelaag = Wordt alle logica in uitgevoerd, kan worden beschouwd als de domein laag

-repository laag = de tussenpersoon tussen domein en databank dus een abstractie van de datalaag, handig wanneer data moet worden omgevormd

-datalaag = handelt communicatie af met de databank en kan dus worden gezien als presistentie laag

-Ik heb alle lagen geimplementeerd, maar de repository laag is bijmij dus enkel prisma, aangezien prisma de tussenpersoon is tussen domein en databank

1. Hoe pak je logging aan in een applicatie?Waarom zijn console.log, console.error, enz. niet voldoende?Leg uit a.d.h.v. een voorbeelduit jouw code.

-In createserver.js initializeren we de logger met bestand logging.js we maken gebruik van winston. Met deze logger kunnen we bv elke request zien, elk antwoord en statuscode, of we in development of production zitten, dit is veel beter dan telkens in de code waar je iets wilt loggen gebruik te maken van console.log

-Een logger laat toe om eenvoudig meer of minder te loggen naargelang we in productie of development draaien.

-Het probleem met console.log en console.error is dat deze altijd zullen loggen. Bij het gebruik van bijvoorbeeld winston kan het loglevel worden meegegeven en zal er dus enkel maar gelogd worden naar de hoeveelheid informatie die in die environment gepast is.