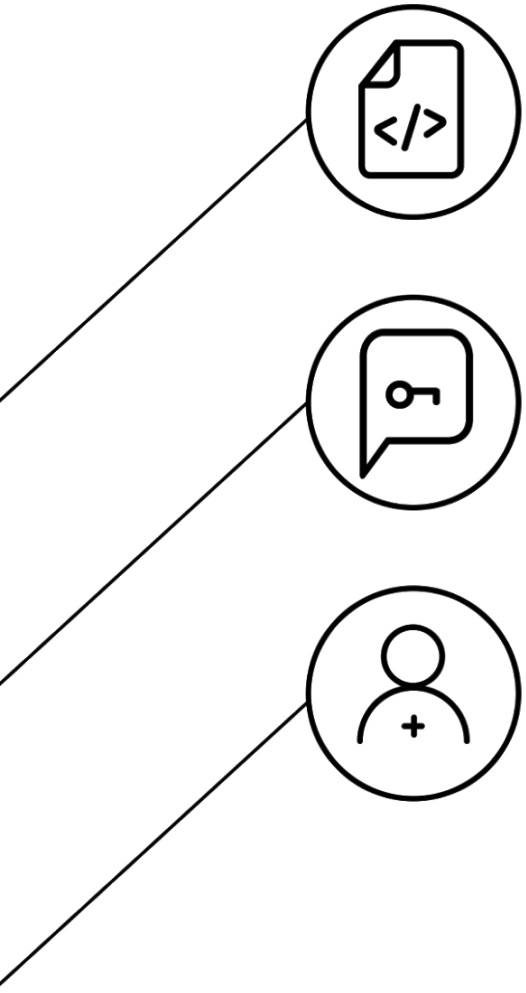

Eindopdracht

Hbo-bachelor Software Development

Leerlijn Frontend

Opdrachtbeschrijving en deelopdrachten

Randvoorwaarden, structuur en beoordeling

NOVI HOGESCHOOL

Inhoud

Integrale eindopdracht Frontend (30 EC)

De leerlijn Frontend bevat de cursussen HTML & CSS, JavaScript, React. Om deze leerlijn af te ronden dien je de volgende leeruitkomsten aan te tonen:

1. **HTML & CSS**
   De student bouwt statische webpagina's met HTML, voorziet deze van styling en layout door CSS en ontwerpt een prototype in Figma (of Adobe XD, deze software is echter niet langer gratis) waarmee hij een visueel design omzet naar een webpagina. (LU1)
2. **JavaScript**
   De student schrijft schone, gestructureerde JavaScript code waarmee hij webpagina's voorziet van interactie en haalt data op via een API. (LU2)
3. **React**
   De student bouwt een interactieve en modulaire webapplicatie in React en maakt hierbij gebruik van herbruikbare interface elementen, state management en de life cycles van React. (LU3)

Algemene opdrachtbeschrijving

Dagelijks gebruiken we online applicaties zoals Microsoft Teams, Google Drive en YouTube. Al deze applicaties zijn voorzien van zowel een frontend als een backend. Bij frontend development ligt de focus op wat gebruikers visueel te zien krijgen in hun browser. Je ontwikkelt het uiterlijk van een webpagina en bent daarmee verantwoordelijk voor het creëren van de gebruikerservaring van het product.

Opdracht: je gaat een applicatie bedenken en bouwen waarvan je alleen de frontend gaat programmeren. Je bedenkt welk probleem je wil oplossen met jouw product en aan welke eisen de applicatie moet voldoen. Hiervoor ga je eerst een plan schrijven en schermontwerpen maken. Daarna ga je de frontend code schrijven.

Door deze opdracht uit te voeren toon je aan een volwaardige webapplicatie te kunnen bouwen in de frontend. In het volgende hoofdstuk vind je meer informatie over de deelopdrachten.

Om de leerlijn Frontend met succes af te ronden is een voldoende nodig voor de integrale eindopdracht; met de deelopdrachten kunnen geen losse cursussen worden afgerond.

Op te leveren producten:

- Functioneel ontwerp met daarin o.a. de probleembeschrijving, functionele- en niet-functionele-eisen, use case tabellen, geschetste wireframes en schermontwerpen;
- De broncode van een React webapplicatie;
- Zelfgeschreven installatiehandleiding als README.md;
- Verantwoordingsdocument.

Deze producten worden ingeleverd als één ZIP-bestand.

Deelopdrachten

We gaan jouw vaardigheden als frontend ontwikkelaar toetsen door een applicatie te bouwen die gebruik maakt van externe data. Daarnaast zul je authenticatie implementeren door gebruik te maken van een NOVI-backend waarin je nieuwe gebruikers kunt toevoegen en bestaande gebruikers kunt valideren. Indien de functionaliteit van de NOVI-backend niet toereikend is voor jouw idee mag je in sommige gevallen zelf een backend maken. Hier zijn wel voorwaarden aan verbonden (zie 'Randvoorwaarden').

Voordat je begint met programmeren, maak je een gestructureerd plan van aanpak en tijdens het ontwikkelen documenteer je jouw keuzes. Al deze stappen zijn opgedeeld in deelopdrachten.

Voordat je kunt starten met de eerste deelopdracht lever je eerst jouw casus in ter goedkeuring bij de docent. Je mag ervoor kiezen om de wensen van de voorbeeldcasus te vertalen naar een idee voor jouw webapplicatie, of om een eigen casus te bedenken. In dat geval ga je eerst opzoek naar een passende API en bedenk je hoe jij deze data kunt gebruiken binnen jouw casus.

Je beschrijft in maximaal 250 woorden:

- Welk probleem je wil oplossen met deze applicatie;
- Welke API je hiervoor gaat gebruiken (inclusief link naar de documentatie);
- Wat de 5 belangrijkste functionaliteiten van jouw applicatie zullen zijn;
- Of je de NOVI backend zult gebruiken voor registratie en inlog of, indien je een andere backend wil gebruiken, welke dit zal zijn.

Na goedkeuring van de docent over jouw probleemstelling en de oplossing die jij daarvoor in gedachte hebt, kun je aan de slag met de eerste deelopdracht.

Voorbeeldcasus

Het onderstaande voorbeeld beschrijft het probleem wat de fictieve student zou willen oplossen en een globale beschrijving van hoe hij dat zou willen doen.

Weet jij ook nooit wat je vanavond moet koken? Lijkt het alsof jouw creativiteit en besluitvaardigheid rond etenstijd ook ineens verdwenen is? Persoonlijk ervaar ik dit ongeveer iedere dag. Daarom ga ik een applicatie programmeren waar ik een aantal grappige vragen kan beantwoorden over o.a. mijn stemming, eetgezelschap en motivatie-om-te-koken-niveau, waarna deze applicatie een aantal passende recepten voorstelt om uit te kiezen. Wanneer iemand een rotdag heeft gehad, zal de applicatie bijvoorbeeld meer comfort-food recepten voorstellen. Wanneer iemand geen zin heeft om te koken, worden er snelle en makkelijke recepten getoond. Op de dagen dat de gebruiker zich wel besluitvaardig voelt, is er ook een mogelijkheid om door alle beschikbare recepten te browsen en op zoek te gaan naar specifieke recepten met behulp van een zoekfunctie. Ten slotte heeft de gebruiker ook de mogelijkheid om recepten te bekijken op basis van de ingrediënten die hij of zij nog in de koelkast heeft liggen. Om dit te doen maak gebruik ik de *Edamam API* (https://www.edamam.com/) om zo recepten op te vragen en te verwerken in mijn applicatie.

Wil je liever een eigen casus bedenken? Kijk eens op https://github.com/hogeschoolnovi/frontend-api-list om je te oriënteren op mogelijke API's die je kunt gebruiken voor een eigen casus. Deze lijst is door NOVI samengesteld en gecontroleerd.

Deelopdracht 1. Functioneel ontwerp

Wanneer je een webapplicatie gaat bouwen kun je niet zomaar beginnen met programmeren. Het is belangrijk eerst te inventariseren hoe het product moet werken (door use case tabellen op te stellen) en te bepalen welke functionele en niet-functionele eisen hieraan verbonden zijn. Vervolgens kun je visuele inspiratie op gaan doen voor het uiterlijk van het eindproduct en ruwe schetsen maken de indeling van de belangrijkste pagina's (*wireframes*). Ten slotte vertaal je deze naar uitgewerkte digitale schermontwerpen.

Je gaat een functioneel ontwerp maken dat voldoet aan de volgende eisen:

- Bevat een titelblad, inleiding en inhoudsopgave. In het documenten zitten geen verwijzingen naar afbeeldingen en informatie buiten het document zelf.
- Beschrijft het probleem en de manier waarop deze applicatie dat probleem oplost.
- Beschrijft wat de applicatie moet kunnen middels een sommering van tenminste 50 functionele- en niet-functionele eisen. Dit hoeft niet evenredig verdeeld te zijn.
- Bevat een verzameling van minimaal 3 verschillende inspiratiebronnen (screenshots of foto's van andere applicaties) die gebruikt zijn ter inspiratie voor de visuele stijl van de applicatie.
- Beschrijf de belangrijkste gebruikershandelingen in minimaal vier use case tabellen: één voor authenticatie (registreren *of* inloggen) en drie voor zelfbedachte functionaliteit.
- Bevat de gescande (of gefotografeerde) wireframes, geschetst op papier, van tenminste vijf pagina's. Deze dienen goed leesbaar te zijn, inclusief begeleidende tekst waarin duidelijk wordt over welke pagina het gaat.
- Bevat screenshots van ten minste 5 schermontwerpen met titels en beschrijvingen. Deze bevatten dusdanig veel uitgewerkte details zodat je tijdens het programmeren geen ontwerpbeslissingen meer hoeft te maken. Aanvullend lever je ook de link naar het originele Figma/Adobe XD project aan.

Op te leveren:

- Het functioneel ontwerp van de webapplicatie in PDF (.pdf).
- Link naar het (openbare) Figma project óf bijgevoegde Adobe XD project in .xd formaat.

Deelopdracht 2. Verantwoordingsdocument

Zodra je begint met programmeren, ga je ook beginnen aan het verantwoordingsdocument. Hierin leg je vast welke technische ontwerpbeslissingen je maakt en *waarom* je deze keuzes gemaakt hebt. Dit werkt het beste als je dit al tijdens het ontwikkelen van jouw product begint te maken. Waarom heb je ervoor gekozen om CSS Grid te gebruiken voor je pagina layout? Waarom heb je deze specifieke npm package gebruikt en niet een andere? Welke doorontwikkelingen zijn er mogelijk of misschien zelfs wenselijk, en waarom heb je deze zelf niet door kunnen voeren? Heb je bijvoorbeeld iets achterwege gelaten in verband met een tekort aan tijd? Leg dan uit wat je liever had willen doen als je meer tijd had gehad. Reflecteer hierbij ook op je eigen leerproces: wat ging goed en wat kan de volgende keer beter? Let op: technieken die als randvoorwaarden gesteld zijn in de eindopdracht tellen niet mee.

Op te leveren:

- Verantwoordingsdocument in PDF (.pdf) met daarin:
    - Minimaal 10 beargumenteerde technische keuzes.
    - Een beschrijving van limitaties van de applicatie en beargumentatie van mogelijke doorontwikkelingen.

Deelopdracht 3. Broncode React

In de eerste opdracht heb je uitgewerkt wat de applicatie moet kunnen en hoe deze eruit moet zien. In deze opdracht ga je jouw schermontwerpen als basis gebruiken om de webapplicatie te implementeren in React. Jouw webapplicatie heeft minimaal de volgende eigenschappen:

- Een gedeelte van de content is alleen beschikbaar voor ingelogde gebruikers (zoals bijvoorbeeld een profiel-pagina). Gebruikers kunnen zich zowel registreren als inloggen. Hiervoor wordt gebruik gemaakt van React Context en de NOVI-backend*. De documentatie voor het gebruik van deze backend vind je hier.
- Er wordt gebruik gemaakt van externe data om zo, naast eigen content, ook andere informatie te kunnen gebruiken. Deze data moet worden opgehaald door middel van **netwerk requests naar een API**. Voorbeelden hiervan zijn weersvoorspelling data, de IMDB-database, een receptendatabase, de data van Sky Scanner, etc.
- Er wordt gebruik gemaakt van routing waardoor gebruikers naar verschillende webpagina's in de applicatie kunnen navigeren.
- Je schrijft jouw styling zelf en maakt géén gebruik van out-of-the-box styling systemen zoals Bootstrap, Material-UI of Tailwind. Het gebruik van iconpacks zoals Fontawesome is wel toegestaan.

* Tenzij je een andere backend gebruikt. Zie Randvoorwaarden.

Op te leveren:

- Projectmap met daarin de broncode, inclusief de link naar de Github repository.

Deelopdracht 4. Installatiehandleiding

In de voorgaande opdrachten heb je jouw ontwikkelwerk afgerond. Om ervoor te zorgen dat ook andere ontwikkelaars jouw project kunnen gebruiken, is het belangrijk een installatiehandleiding te schrijven waarin beschreven wordt wat zij hiervoor nodig hebben. Je schrijft jouw installatiehandleiding voor een mede-developer, maar zorgt ervoor dat dit ook te volgen is wanneer deze persoon geen enkele ervaring heeft binnen het frontend-landschap.

Het bevat:

- Een inleiding met korte beschrijving van de functionaliteit van de applicatie en screenshot van de belangrijkste pagina van de applicatie.
- Lijst van benodigdheden om de applicatie te kunnen runnen (zoals runtime environments, een API key of gegevens van een externe backend). Let op: je vraagt de nakijkende docent nooit zelf een API key aan te maken. Jij levert zelf jouw API key aan.
- Een stappenplan met daarin installatie instructies.
- Met welke gegevens er ingelogd kan worden indien er al accounts beschikbaar zijn.
- Welke andere npm commando's er nog beschikbaar zijn in deze applicatie en waar deze voor dienen.

Op te leveren:

- Zelfgeschreven README.md in de root van de React projectmap

Randvoorwaarden

Hieronder vind je een aantal randvoorwaarden waaraan je eindopdracht moet voldoen naast de gestelde eisen aan de applicatie (zie 'Deelopdracht 3'). Deze randvoorwaarden hebben een verplicht karakter.

- Alleen React met JavaScript wordt geaccepteerd als programmeertaal (geen TypeScript).
- Er mag geen Redux gebruikt worden.
- Het project is geüpload naar een GitHub repository: deze repository staat op *public*.
- Het project wordt ingeleverd *zonder* node_modules-map/.idea-map.
- Het project bevat de JavaScript linter ESLint (Bij gebruik van create-react-app is dit standaard meegeleverd).
- De wireframes zijn getekend op papier.
- Het prototype is ontworpen met Figma. Indien je liever Adobe XD wil gebruiken, mag dit ook. Deze software is helaas niet langer gratis.
- De applicatie start op zonder crashes.
- Er is géén gebruik gemaakt van out-of-the-box styling systemen zoals Bootstrap, Material-UI of Tailwind.
- Er wordt gebruik gemaakt van de NOVI-backend. Wanneer er gebruik gemaakt wordt van een eigen backend mag dit alléén:
    - een Firebase backend zijn en dienen de inloggegevens ook te worden verstrekt aan de docent
    - óf de backend zijn die eerder is gemaakt in de Backend-leerlijn. Deze moet dan opnieuw worden ingeleverd samen met deze opdracht.
- Het wordt aangeleverd d.m.v. een ZIP-bestand van maximaal 50 MB. (Met de extensie .zip. Projecten die als .rar worden aangeleverd, worden niet geaccepteerd.) Tip: blijkt jouw bestand veel groter dan 50 MB, dan ben je waarschijnlijk vergeten jouw node_modules-map te verwijderen.

Structuur

Functioneel ontwerp
Het functioneel ontwerp is een verzorgd document met een titelblad, inhoudsopgave en inleiding. Het verantwoordingsdocument mag toegevoegd worden als laatste hoofdstuk in het functioneel ontwerp. Het bevat geen verwijzingen naar afbeeldingen of documenten buiten het document zelf. Naast de bijgevoegde losse schermontwerpen introduceer je deze schermontwerpen ook al te bij het functioneel ontwerp. Ook wireframes behoren voorzien te worden van paginatitels en beschrijving zodat het duidelijk is over welke pagina het gaat. De functionele- en niet-functionele eisen zijn genummerd en gecategoriseerd. Er is geen harde richtlijn met betrekking tot het aantal woorden, maar zorg ervoor dat je beknopt en bondig schrijft.

Installatiehandleiding
De installatiehandleiding is een verzorgd document met een inhoudsopgave en inleiding.

Beoordelingscriteria

De eindopdracht wordt beoordeeld op basis van de volgende beoordelingscriteria. Per criterium kent de beoordelaar een aantal punten toe. Hierbij wordt zowel gekeken naar de feitelijke realisatie van de leeruitkomst als naar het door de student getoonde inzicht in de bijbehorende theorie.

# Deelopdracht	**Leeruitkomsten en beoordelingscriteria**	**Weging**	**Score in cijfers van 1 t/m 10**	**Weging maal score**
***1.* Functioneel ontwerp**	***Met deze opdracht worden aspecten van de leeruitkomst van de cursus HTML & CSS (LU1) getoetst.***	**25%**		
Criterium 1.1	De student schrijft een compleet functioneel ontwerp, inclusief: heldere omschrijving probleem én oplossing, minimaal 50 passende functionele en niet-functionele eisen, minimaal 4 kwalitatieve use-case tabellen, verzameling van minimaal 3 verschillende inspiratiebronnen en ten minste 5 duidelijk vormgegeven wireframes. (LU1)	20%		
Criterium 1.2	De student ontwerpt schermontwerpen met Figma/Adobe XD door middel van minimaal 5 schermontwerpen. (LU1)	5%		
2. Verantwoordingsdocument	***Met deze opdracht worden aspecten van de leeruitkomsten van de cursussen JavaScript (LU2) en React (LU3) getoetst.***	**10%**		
Criterium 2.1	De student reflecteert op zijn (schone en gestructureerde) JavaScript/React code en beargumenteert minimaal 10 gemaakte technische keuzes. (LU2, LU3)	5%		
Criterium 2.2	De student beschrijft realistische limitaties van zijn applicatie en beargumenteerd welke doorontwikkelingen mogelijk en/of wenselijk zijn. (LU2 en LU3)	5%		
3. Broncode React	***Met deze opdracht worden aspecten van de leeruitkomsten van de cursussen HTML & CSS (LU1), JavaScript (LU2) en React (LU3) getoetst.***	**60%**		
Criterium 3.1	De student bouwt zijn applicatie, op basis van het functioneel ontwerp en schermontwerpen, met semantische HTML-elementen en maakt gebruik van zelfgeschreven, schone, gestructureerde CSS	10%		

	door modulaire styling aan specifieke componenten te koppelen. (LU1, LU2 en LU3)			
Criterium 3.2	De student schrijft schone, gestructureerde JavaScript en React code waarin hij op correcte wijze functies, variabelen, arrays en objecten gebruikt en clean code principes toepast. (LU2 en LU3)	10%		
Criterium 3.3	De student werkt op een juiste manier met asynchrone functies om externe data op te halen via een API en handelt errors af. (LU2)	5%		
Criterium 3.4	De student implementeert vier belangrijke kern-functionaliteiten op correcte wijze, waaronder het authenticatie-proces van registratie/inloggen en drie zelfbedachte, complete use cases.	10%		
Criterium 3.5	De student deelt zijn applicatie op logische wijze op in herbruikbare componenten waarbij data op juiste wijze wordt doorgegeven via properties en callback properties. (LU3)	10%		
Criterium 3.6	De student voorziet componenten van interactie en externe data door correct gebruik te maken van state management en alle React Life Cycle hooks. (LU3)	10%		
Criterium 3.7	De student beheert diens code met gebruik van Git om met versiebeheer de voortgang van het project vast te leggen. De student zorgt voor kleine commits, maakt pull requests en mergt regelmatig.	5%		
4. Installatiehandleiding	***Met deze opdracht worden aspecten van de leeruitkomst van de cursus React (LU3) getoetst.***	**5%**		
Criterium 4.1	De student schrijft met gebruik van de clean code principes een duidelijke installatiehandleiding waarmee iemand zonder kennis van het project de gebouwde applicatie zelfstandig kan draaien. (LU3)	5%		
	Totaal	**100%**		

Bijlage A: Template herkansingsfeedback

Verantwoordingsdocument

Heb je jouw eindopdracht ingeleverd en heb je een onvoldoende gehaald? Dan verwerk je een extra hoofdstuk in jouw verantwoordingsdocument waarin je aantoont wat je precies hebt verbeterd naar aanleiding van de feedback van de docent.
Dit doe je per beoordelingscriterium, waarin je ook de feedback en originele cijfer benoemt. Gebruik hiervoor onderstaand template. In het geval dat je feedback *niet* verbeterd hebt, geef je dit ook aan.

Criterium 1.1 – De student schrijft een compleet functioneel ontwerp, inclusief: heldere omschrijving probleem én oplossing, minimaal 50 passende functionele en niet-functionele eisen, minimaal 4 kwalitatieve use-case tabellen, verzameling van minimaal 3 verschillende inspiratiebronnen en ten minste 5 duidelijk vormgegeven wireframes.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 1.2 – De student ontwerpt schermontwerpen met Figma/Adobe XD door middel van minimaal 5 schermontwerpen.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 2.1 – De student reflecteert op diens (schone en gestructureerde) JavaScript/React code en beargumenteert minimaal 10 gemaakte technische keuzes.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 2.2 – De student beschrijft realistische limitaties van zijn applicatie en beargumenteerd welke doorontwikkelingen mogelijk en/of wenselijk zijn.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 3.1 – De student bouwt diens applicatie, op basis van het functioneel ontwerp en schermontwerpen, met semantische HTML-elementen en maakt gebruik van zelfgeschreven, schone, gestructureerde CSS door modulaire styling aan specifieke componenten te koppelen.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 3.2 – De student schrijft schone, gestructureerde JavaScript en React code waarin die op correcte wijze functies, variabelen, arrays en objecten gebruikt en clean code principes toepast.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 3.3 – De student werkt op een juiste manier met asynchrone functies om externe data op te halen via een API en handelt errors af.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 3.4 – De student implementeert vier belangrijke kern-functionaliteiten op correcte wijze, waaronder het authenticatie-proces van registratie/inloggen en drie zelfbedachte, complete use cases.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 3.5 – De student deelt zijn applicatie op logische wijze op in herbruikbare componenten waarbij data op juiste wijze wordt doorgegeven via properties en callback properties.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 3.6 – De student voorziet componenten van interactie en externe data door correct gebruik te maken van state management en alle React Life Cycle hooks.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 3.7 – De student beheert diens code met gebruik van Git om met versiebeheer de voortgang van het project vast te leggen. De student zorgt voor kleine commits, maakt pull requests en merget regelmatig.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	

Criterium 4.1 – De student schrijft met gebruik van de clean code principes een duidelijke installatiehandleiding waarmee iemand zonder kennis van het project de gebouwde applicatie zelfstandig kan gebruiken.	
Feedback docent:	*[cijfer]*
Toelichting verbeteringen student	