Verantwoording

MoodFood React WebApp

Hayk Abrahamian

09-10-2024

Inleiding

In deze tweede deelopdracht, het verantwoordingsdocument, wordt er dieper ingegaan op de technische aspecten van de ontwikkeling van de MoodFood webapplicatie. De focus ligt hierbij op de programmeerkeuzes die ik bewust heb gemaakt tijdens het gebruik van **JavaScript** en **React**, en andere technieken die buiten het curriculum vallen. Dit document dient als een analyse van deze keuzes, met nadruk op waarom bepaalde technieken zijn toegepast en hoe ze bijdragen aan de functionaliteit en het succes van de applicatie.

Het is niet alleen belangrijk om te reflecteren op de keuzes die ik heb gemaakt, maar ook om te kijken naar mogelijke alternatieven en verbeteringen. Daarnaast zal ik reflecteren op mijn leerproces: wat ik heb geleerd van de technische uitdagingen en hoe ik mijn vaardigheden in de toekomst kan verbeteren. Dit document biedt dus zowel een technisch als persoonlijk overzicht van mijn ontwikkeling tijdens dit project.

Verantwoording Keuzen (Technische Programmeerkeuzes):

1. JavaScript

- **Waarom deze keuze?** Hoewel de keuze voor JavaScript in het curriculum was vastgelegd, wil ik reflecteren op hoe ik dit in mijn project heb toegepast. Aangezien mijn voorkennis met Python was, voelde ik aanvankelijk de overgang naar JavaScript als een uitdaging. JavaScript is een krachtige taal voor webontwikkeling vanwege de flexibiliteit, maar het was even wennen aan de andere syntax en asynchrone werking.

- **Reflectie:** De keuze om JavaScript intensief te gebruiken in de front-end van mijn applicatie heeft mijn begrip van moderne webontwikkeling sterk verbeterd. Het gebruik van functies zoals asynchrone verwerking (bijvoorbeeld met **async/await**) heeft me laten inzien hoe belangrijk het is om efficiënte en niet-blokkerende code te schrijven. Dit was vooral belangrijk toen ik API-verzoeken implementeerde om gegevens op te halen en te verwerken zonder dat de gebruikerservaring werd onderbroken.

2. React

- **Waarom deze keuze?** Hoewel React een verplichte keuze was, heb ik bewust gekozen om te werken met **functionele componenten** en **hooks** zoals

`useState` en `useEffect`. Dit omdat functionele componenten een duidelijkere en eenvoudigere manier van programmeren bieden dan de meer traditionele class-componenten.

- **Reflectie:** Functionele componenten in combinatie met hooks hebben mijn leerproces enorm verbeterd. Het stelde me in staat om state en lifecycle-methoden op een intuïtieve manier te beheren. In mijn project was dit cruciaal, vooral bij het dynamisch updaten van de UI op basis van gebruikersinput of API-gegevens. Bijvoorbeeld, met `useEffect` heb ik het herhaaldelijk ophalen van recepten op basis van gebruikerskeuzes kunnen beheren zonder dat de hele pagina opnieuw geladen hoefde te worden, wat zorgde voor een vlotte gebruikerservaring. Dit was een van de meest waardevolle lessen in het begrijpen van state management in React.

3. Axios voor API-verzoeken

- **Waarom deze keuze?** Ik heb ervoor gekozen om **Axios** te gebruiken voor het uitvoeren van API-verzoeken, ondanks dat er andere mogelijkheden waren zoals de native `fetch()` API. Axios bood echter een eenvoudigere en leesbaardere syntax, en de ingebouwde foutafhandelingsmogelijkheden maakten het gebruik ervan aantrekkelijker.

- **Reflectie:** In de MoodFood-app was het belangrijk om betrouwbare API-verzoeken te doen, vooral omdat ik data van externe bronnen ophaalde om recepten te genereren. Een voorbeeld is het gebruik van Axios om recepten op te halen op basis van zoekcriteria die door de gebruiker zijn ingevoerd. Dankzij Axios kon ik niet alleen de resultaten efficiënt verwerken, maar ook eenvoudig foutafhandeling toevoegen voor bijvoorbeeld netwerkproblemen. Deze foutafhandelingsfunctionaliteit verbeterde de robuustheid van mijn applicatie en gaf mij inzicht in hoe belangrijk fouttolerante systemen zijn in echte applicaties.

4. CSS Modules voor Stijlbeheer

- **Waarom deze keuze?** Voor het stijlbeheer van de applicatie koos ik voor **CSS Modules**. Dit was een bewuste keuze omdat het lokale scoping biedt, waardoor stijlconflicten tussen componenten worden vermeden. Het gaf mij ook de mogelijkheid om herbruikbare stijlen te creëren en te gebruiken zonder dat ze de globale namespace vervuilden.

- **Reflectie:** Het gebruik van CSS Modules verbeterde niet alleen de structuur van mijn code, maar ook de onderhoudbaarheid van de applicatie. In het MoodFood-

project moest ik verschillende componenten zoals knoppen, invoervelden en navigatie-elementen consistent stylen. Dankzij CSS Modules kon ik de stijlen specifiek houden voor elk component, zonder dat ze elkaar onbedoeld beïnvloedden. Dit was cruciaal bij het opschalen van de applicatie, vooral toen ik steeds meer UI-componenten toevoegde. Hierdoor heb ik geleerd hoe belangrijk het is om stijlen modulair en schaalbaar te houden.

5. React Router voor Routing

- **Waarom deze keuze?** Ik heb gekozen voor **React Router** om de navigatie in de applicatie te beheren. Dit omdat React Router een naadloze integratie biedt met React en krachtige mogelijkheden heeft zoals dynamische routes en dieplinks, waardoor gebruikers makkelijk tussen verschillende pagina's kunnen navigeren zonder dat de pagina volledig opnieuw hoeft te laden.

- **Reflectie:** Deze keuze heeft goed gewerkt in de MoodFood-app. Met React Router kon ik eenvoudig pagina’s zoals de **Home**, **Zoek**, en **Fridge** pagina's beheren. Een specifiek voorbeeld is hoe ik dynamische routes heb gebruikt om individuele receptpagina’s te genereren op basis van de recepten-ID, wat de flexibiliteit van de applicatie aanzienlijk heeft verhoogd. Dit gaf mij een dieper inzicht in het opzetten van een single-page applicatie (SPA) en het belang van routing voor een soepele gebruikerservaring.

Limitaties

Limitatie 1: Geen donkere modus

- **Argumentatie voor doorontwikkeling:** Het toevoegen van een donkere modus zou de gebruikerservaring aanzienlijk kunnen verbeteren, vooral in situaties waarbij de gebruiker de applicatie in een omgeving met weinig licht gebruikt, bijvoorbeeld 's nachts. Veel moderne applicaties ondersteunen een donkere modus omdat het minder vermoeiend is voor de ogen bij langdurig gebruik. Door CSS-variabelen te implementeren, kan er eenvoudig tussen een licht en donker thema gewisseld worden. Daarnaast verhoogt een donkere modus de toegankelijkheid van de applicatie voor gebruikers met visuele beperkingen zoals lichtgevoeligheid. Deze verbetering zou de applicatie gebruiksvriendelijker maken en aansluiten bij de huidige trends in UI/UX-ontwerp.

Limitatie 2: Geen ondersteuning voor meerdere talen

- **Argumentatie voor doorontwikkeling:** In een steeds meer geglobaliseerde wereld is het aanbieden van een meertalige ondersteuning een logische stap om de toegankelijkheid van de applicatie te vergroten. Door de applicatie te vertalen naar meerdere talen, kunnen gebruikers met verschillende taalachtergronden de applicatie gebruiken, wat het bereik van de applicatie aanzienlijk kan vergroten. Het gebruik van internationale bibliotheken zoals `react-intl` of `i18next` zou het mogelijk maken om eenvoudig meerdere talen te ondersteunen, wat vooral van belang is voor gebruikers in verschillende landen of met verschillende taalvoorkeuren. Dit draagt niet alleen bij aan een betere gebruikerservaring, maar ook aan de inclusiviteit van de applicatie.

Limitatie 3: Beperkte browsercompatibiliteit

- **Argumentatie voor doorontwikkeling:** Een brede browsercompatibiliteit is essentieel om ervoor te zorgen dat alle gebruikers, ongeacht de browser die ze gebruiken, een consistente ervaring hebben. Hoewel moderne webbrowsers zoals Chrome en Firefox veel worden gebruikt, zijn er ook nog steeds gebruikers van browsers zoals Safari, Edge of zelfs Internet Explorer. Het optimaliseren van de applicatie voor deze browsers zou ervoor zorgen dat gebruikers niet tegen technische problemen aanlopen. Dit kan gerealiseerd worden door grondige cross-browser tests en het gebruik van polyfills om ontbrekende functionaliteiten te ondersteunen in oudere browsers. Door deze doorontwikkeling kunnen we garanderen dat de applicatie bruikbaar blijft voor een breed scala aan gebruikers, ongeacht hun voorkeur voor browser.

Limitatie 4: Gebrek aan geavanceerde zoekmogelijkheden

- **Argumentatie voor doorontwikkeling:** Het toevoegen van geavanceerde zoekfuncties zou de gebruiksvriendelijkheid van de applicatie aanzienlijk verbeteren. Gebruikers zouden in staat zijn om recepten te filteren op basis van specifieke criteria zoals ingrediënten, dieetvoorkeuren, kooktijd, of allergieën. Deze functionaliteit maakt de applicatie flexibeler en beter afgestemd op de individuele behoeften van de gebruiker. Daarnaast kan het gebruik van externe API's voor het verrijken van zoekresultaten, zoals het integreren van een voedingsinformatie-API, gebruikers helpen om beter geïnformeerde keuzes te maken. Geavanceerde zoekalgoritmen zorgen ervoor dat de zoekresultaten relevanter worden, wat uiteindelijk leidt tot een hogere tevredenheid bij de gebruiker.

Limitatie 5: Niet geoptimaliseerd voor mobiele apparaten

- **Argumentatie voor doorontwikkeling:** Het optimaliseren van de applicatie voor mobiele apparaten is tegenwoordig een noodzaak gezien het toenemende gebruik van smartphones en tablets voor online activiteiten. Door gebruik te maken van responsive designprincipes kan de interface zich aanpassen aan verschillende schermformaten, wat de gebruikservaring op mobiele apparaten aanzienlijk verbetert. Dit omvat ook het optimaliseren van de laadsnelheid en het verbeteren van de touch-interactie, zodat de applicatie intuïtief en snel blijft op mobiele platforms. Het implementeren van deze aanpassingen zou ervoor zorgen dat de applicatie niet alleen aantrekkelijk is voor desktopgebruikers, maar ook voor de grote groep mobiele gebruikers.

Conclusie

De doorontwikkelingen die voortkomen uit de bovengenoemde beperkingen zijn niet alleen haalbaar, maar dragen ook bij aan een betere gebruikerservaring en bredere toegankelijkheid van de applicatie. Door verbeteringen zoals een donkere modus, meertalige ondersteuning en mobiele optimalisatie door te voeren, kan de applicatie niet alleen functioneler, maar ook inclusiever en moderner worden. Daarnaast zorgen geavanceerde zoekopties en browsercompatibiliteit ervoor dat de applicatie meer gebruikers kan bereiken en hen een gepersonaliseerde ervaring biedt. Deze ontwikkelingen zijn essentieel voor de lange termijn groei en het succes van de applicatie.

Persoonlijke ervaringen/evaluatie

Het ontwerpen van wireframes en het opstellen van de Figma-modellen heeft mij enorm geholpen bij het visualiseren van zowel het ontwerp als de functionaliteit van de applicatie. Ondanks de uitdaging van het opstellen van de functionele en niet-functionele vereisten, was dit een waardevolle oefening die me dwong om gestructureerd na te denken over het project.

Een groot leermoment was het integreren van een API in mijn applicatie. De uitdagingen rondom het ophalen en verwerken van externe data hebben mij niet alleen technische vaardigheden opgeleverd, maar me ook geleerd hoe belangrijk het is om een gedegen analyse vooraf te doen. Online bronnen zoals tutorials en documentatie waren hierbij cruciaal, en ik heb geleerd om niet alleen direct in de code te duiken, maar eerst goed na te denken over het ontwerp en de architectuur van de oplossing.

Door deze opdracht heb ik mijn technische vaardigheden aanzienlijk kunnen verbeteren, vooral met betrekking tot React en de keuze van externe tools zoals Axios en CSS Modules. Deze bewuste keuzes hebben bijgedragen aan een beter schaalbare, onderhoudbare en robuuste applicatie. Reflectie op deze keuzes en de geleerde lessen zal mij in toekomstige projecten helpen om nog efficiënter te werken en meer diepgang te creëren in mijn ontwikkelprocessen.