|  |
| --- |
| Peerfeedback formulier voor programmeeropdracht |
| Naam student die beoordeeld wordt: Ensar Isiktas |
| Naam beoordelaar: Eleonora Avaleva |
| Datum: 22-03-2025 |
| Onderwerp: Review code backend API-architectuur |

|  |  |
| --- | --- |
| Criteria | Opmerkingen |
| Code Efficiëntie | Helaas heb ik hier onvoldoende kennis om te bepalen of de code efficienter kan. Wel loop je risico dat een onverwachtte type coersion kunt krijgen door het gebruikt te maken van alleen 2 gelijktekens (==). Gebruik in plaats daarvan 3 gelijktekens (===).  De volgende functie (line 485, script)verwacht 1 parameter maar je geeft het 2 mee.  Weet je zeker of het dan nog steeds werkt?  *document*.querySelectorAll('.sort-btn')?.forEach(btn => {  btn.addEventListener('click', () => sortCollection(btn.dataset.field, *currentSort*.ascending)); }); |
| Leesbaarheid en Stijl | Het zal leesbaarder zijn als je de frontend en de backend code in twee verschillende directories zet.  De functies zijn kort en duidelijk, wel kun je ze verdelen over aparte files (met oog op toekomstige uitbreiding van de applicatie) |
| Functionaliteit | De applicatie lijkt te werken met de server APIs. |
| Modulariteit en Herbruikbaarheid | Je zal kunnen overwegen om je frontend code op te splitsen naar, functies die data ophalen, filters en handlers. |
| Documentatie en Commentaren | Er zijn weinig tot geen comments in de code toegevoegd. Wel is de naamgeving van de functies goed, dit maakt dat ik de code makkelijk leesbaar vind. |
| Foutafhandeling en Robuustheid | Je maakt gebruik van statuscodes 400, 404 en 201 aan de server kant. Dat vind ik netjes. Wel kun je nog status 500 (server issue), 200 (request succes) er aan toe voegen. |

|  |
| --- |
| Algemene Opmerkingen: |
| Ik vind het een leuke oplossing! Het is interessant om te zien dat je met zo weinig code toch een mooie en werkende applicatie hebt gemaakt. |

**Tips voor specifieke feedback**

**Code Efficiëntie**

* Hoe snel of langzaam de applicatie draait.
* Code die onnodig hetzelfde werk meerdere keren uitvoert.
* Het gebruik van passende datatypes en datastructuren voor de bewerkingen die worden uitgevoerd.

**Leesbaarheid en Stijl**

* De code is begrijpelijk en goed leesbaar.
* Variabelen en functies hebben betekenisvolle namen.
* Functies zijn kort, duidelijk en begrijpelijk.
* Constanten worden bovenaan het script gedefinieerd.
* Hardcoded waarden worden vermeden omdat ze moeilijk te onderhouden zijn.

**Functionaliteit**

* Werkt de functie of applicatie zoals bedoeld, ook in randgevallen?
* Hoe gaat de code om met ongeldige, onverwachte of onvoorspelbare invoer?

**Modulariteit en Herbruikbaarheid**

* Elke module, klasse of functie dient een duidelijk doel.
* De code is opgesplitst in modules, klassen en functies, en kan in verschillende contexten worden hergebruikt.

**Documentatie en Commentaar**

* De code is voorzien van voldoende commentaar en bruikbaar.
* De documentatie komt overeen met wat de code daadwerkelijk doet.
* Er zijn duidelijke instructies voor het gebruik van externe bibliotheken.

**Foutafhandeling en Robuustheid**

* Invoer wordt gevalideerd.
* Het juiste gebruik van try-except blokken waar nodig om fouten af te handelen.