Vedlegg 1

Hjelp med GROQ-spørringer fra chatGPT

Metoder som er basert på denne chatten:

```
fetchGenresForUsers(user1, user2)
fetchWishlistForUsers(user1, user2)
fetchFavoritesForUsers(user1, user2)
fetchFavoritesUser1AndWishlistUser2(user1, user2)
fetchWishlistUser1AndFavoritesUser2(user1, user2)
```

Bruker input til ChatGPT markert i RØD tekst
ChatGPT respons er markert i SVART tekst

CHATGPT:

i need you to help me with a groq query in sanity. I will provide you with my schemas.

ChatGPT

Sure, I can help with that! Please go ahead and share your Sanity schemas or describe what you need the Groq query to accomplish.

Dont generate any answer yet. here are my schemas: export const genre = { title: "Sjanger", name: "genre", type: "document", fields: [{ title: "Sjanger", name: "genre", type: "string" }, { title: "Sjangerlink", name: "genreurl", type: "slug", options: { source: "genre", slugify: input => input.toLowerCase().replace(/\s+/g, '-') } }] } export const users = { title: "Brukere", name: "users", type: "document", fields: [{ title: "Navn", name: "name", type: "string" }, { title: "Id", name: "id", type: "number" }, { title: "Favoritt Sjangere", name: "genrelist", type: "array", of: [{ type: "string", name: "genre" }] }, { title: "Ønskeliste Filmer", name: "wishlist", type: "array", of: [{ type: "string", name: "movie" }] },] }

ChatGPT

Thanks for providing your schemas. Now, please tell me what you want the GROQ query to achieve using these schemas. For example, are you looking to fetch all users with their favorite genres, or something specific like filtering by a particular genre? Let me know your exact requirements!

I am trying to enter to usernames as parameters to the query, where the paramateres are two different usernames. I want the groq query to compare the two genrelists of the users and return only the genres that are in both users lists.

ChatGPT

To create a GROQ query that compares the genre lists of two users based on usernames and returns only the genres common to both, you will need to use a few advanced GROQ features like joins and filters. Here’s how you can construct such a query:

groq

Kopier kode

```
*[_type == "users" && (name == $username1 || name == $username2)]{ "commonGenres": genrelist[]->genre[title in *[_type == "users" && name == $username1].genrelist[]->genre[title] && title in *[_type == "users" && name == $username2].genrelist[]->genre[title]] }
```

This query works by:

1. **Selecting Users:** It selects users where the **name** matches either of the two usernames (**$username1** or **$username2**).
2. **Fetching Common Genres:** It then checks the **genrelist** of these users and includes only those genres that are present in both users' **genrelist**.

Explanation:

- **`*[_type == "users" && (name == $username1 || name == $username2)]`** retrieves documents of type **users** where the user's name matches either of the two specified usernames.
- **`genrelist[]->genre[title in ...]`** dereferences the genres in each user's genre list and filters them by checking if the genre titles appear in the genre lists of both users. The inner queries (**`*[_type == "users" && name == $username1/2].genrelist[]->genre[title]`**) fetch the titles of genres favored by each user, allowing for a comparison.

Parameters:

- You would replace **$username1** and **$username2** with the actual usernames you want to compare when running this query.

This query assumes that `genrelist[]->genre[title]

this didnt work. Is there any other way accomplish this?

ChatGPT

Let's simplify the approach. Since the **genrelist** fields in the **users** schema store arrays of strings directly (not references), the querying method will be a bit different. You would want to compare these strings directly rather than attempting to dereference them.

Given that you're trying to find genres that are common to the **genrelist** of two specific users, you can use the following GROQ query to achieve this:

groq

Kopier kode

```
{ "user1": *[_type == "users" && name == $username1][0].genrelist, "user2": *[_type == "users" && name == $username2][0].genrelist, "commonGenres": intersection(*[_type == "users" && name == $username1][0].genrelist, *[_type == "users" && name == $username2][0].genrelist) }
```

How this works:

1. **Fetch each user's genre list:** The query retrieves the **genrelist** of each of the two users separately (**user1** and **user2**).
2. **Calculate intersection:** It uses the **intersection()** function to find common elements between the **genrelist** of the two users. This function computes the set intersection of the arrays provided.

Considerations:

- Ensure that **$username1** and **$username2** are replaced with the actual usernames when using this query in your application or testing environment.
- Make sure your Sanity Studio or the version of the GROQ server you're using supports the **intersection()** function, as some implementations may not have this function by default.

If this approach still doesn't work, we may need to consider whether there's a version mismatch or specific configuration in your Sanity setup that affects the availability of certain functions like **intersection()**.

is says intersection is not defined as a function

ChatGPT

I apologize for the confusion; it seems that I provided a function (**intersection()**) that doesn't exist natively in GROQ. We'll need to approach this with basic GROQ functionalities.

Since GROQ doesn't directly support array intersection as a built-in operation, we can accomplish this task by manually comparing the arrays using a query. Here's a more appropriate way to formulate your query to find common genres between two users:

groq

Kopier kode

```
{ "user1Genres": *[_type == "users" && name == $username1][0].genrelist, "user2Genres": *[_type == "users" && name == $username2][0].genrelist, "commonGenres": *[_type == "users" && name in [$username1, $username2]] { genrelist }[0].genrelist[(@ in *[_type == "users" && name == $username1][0].genrelist) && (@ in *[_type == "users" && name == $username2][0].genrelist)] }
```

Breakdown of the query:

- **Retrieve Genre Lists**: The query initially fetches the **genrelist** for the two users specified.
- **Manual Array Comparison**: For the **commonGenres**, it iterates over the genre list of the first user (**$username1**) and checks each genre to see if it also appears in the genre list of the second user (**$username2**). It does this using the GROQ conditional and the **in** keyword, which checks for presence in an array.

This query is a bit more complex and assumes that each user's **genrelist** isn't too large, as performance might degrade with very large lists. Additionally, make sure to replace **$username1** and **$username2** with actual username parameters when you execute the query.

If you encounter further issues or if this approach doesn't suit your needs, you might need to handle the comparison outside of GROQ, such as in your application logic after fetching the genre lists separately.