FIXING THE LOAD MORE ANSWERS BUTTON

To fix duplicate answers on the answers list when the *Load more answers* button being hit, we need to record the new answers in the component then exclude them when we hit our API endpoint. In the server we can take them and exclude them in the query.

THE FRONTEND

1. Store the new answers

Let's reopen the `Answers.vue` then go to script section. In `data` method let's add another property to hold the new answers. Let's call it `excludeAnswers` and set it with empty array.

```
data () {
    return {
        // ...
        excludeAnswers: []
    }
},
```

Now we can go to `add` method. And in this method we can also store new answer created to the `excludeAnswers`.

```
add (answer) {
    this.excludeAnswers.push(answer);
    // ...
},
```

2. Adding the excluded answers in query string

Now let's jump to `computed` property and define a new method. Let's call it `theNextUrl`. Basically what this method does is to return the next URL (that is stored in `nextUrl` varriable) along with the list of answer id being excluded in query string like so:

```
http://localhost:8000/questions/1/answers&page=2&excludes[]=1&excludes[
]=2&...
```

So to do that we need to make sure that the `nextUrl` is not empty and we have excluded answers. If both conditions were met return the `nextUrl` along with the query string that we get from `excludeAnswers`.

Otherwise if one of the conditions is not met we simply return the `nextUrl`.

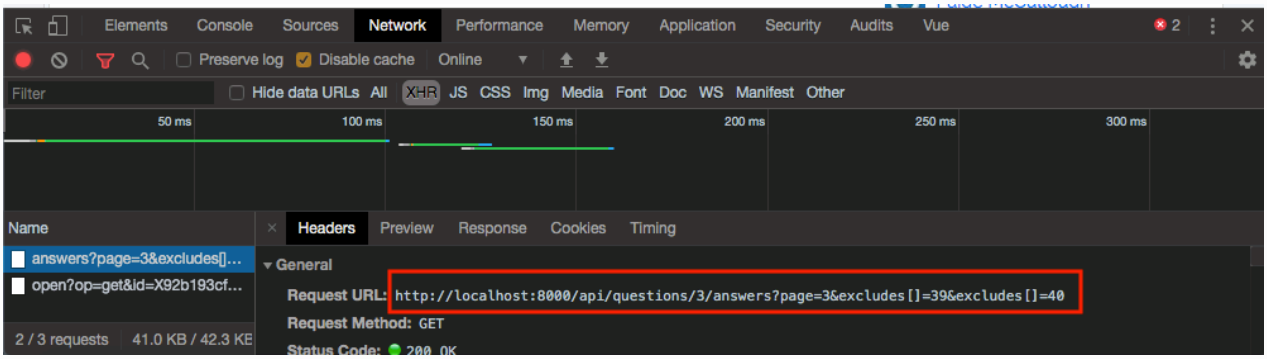
```
computed: {
    // ...
    theNextUrl () {
        if (this.nextUrl && this.excludeAnswers.length) {
            return this.nextUrl +
                   this.excludeAnswers.map(a => '&excludes[]=' +
a.id).join('');
        }
        return this.nextUrl;
    }
},
```

Last, let's go to template and make a bit change on the *Load more answers* button. Here we need to change the value on `v-if` directive as well as `fetch` method call from `nextUrl` to `theNexUrl`.

```
<div class="text-center mt-3" v-if="theNextUrl">
    <button @click.prevent="fetch(theNextUrl)" ...>Load more
answers</button>
</div>
```

3. Testing

Now if you go to your browser, add some answers and then hit the *Load more answers* button. You still see duplicate answers in the list. But if you open your browser console you'll see the's a query string being added in the Request URL.



Now let's work on the backend and exclude the excluded answers in the query.

THE BACKEND

Let's open `Api/AnswersController.php` file. In the `index` method we can add `where` method before `simplePaginate` method. In that method let's pass in an empty closure.

```
$answers = $question->answers()
                    ->with('user')
                    ->where(function ($q) {})
                    ->simplePaginate(3);
```

Now in the closure body we can check the `excludes` existence from the request. If the condition was met we can exclude the answers that we get in the query by making use the `whereNotIn` method.

```
function ($q) {
  if (request()->has('excludes')) {
    $q->whereNotIn('id', request()->query('excludes'));
  }
}
```

So here the `index` method will look like.

```
public function index(Question $question)
{
  $answers = $question->answers()->with('user')->where(function ($q) {
    if (request()->has('excludes')) {
      $q->whereNotIn('id', request()->query('excludes'));
    }
  })->simplePaginate(3);

  return AnswerResource::collection($answers);
}
```

Now if you go to your browser and doing some test like before you won't see any duplicate answers on the list.