FINAL BUG FIXING

In this guide I'm going to show you how to fix some issues that we have in our Single Page Application. So the issues that we have here are:

- Having *Page undefined of undefined* on Pagination component
- Unable to delete the question on All Questions page
- Having Page not found error after successfully login or register

Alright, let's go ahead and open up our terminal then create a new branch:

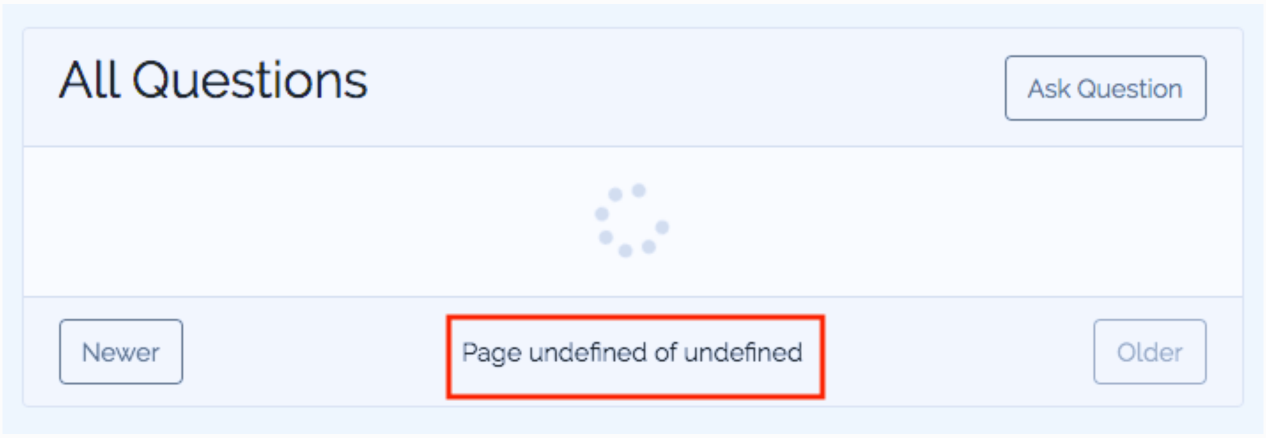
```
git checkout -b lesson-65
```

And don't forget to run Laravel mix:

```
npm run watch
```

FIXING ISSUE ON PAGINATION'S PAGE INFO

If you navigate to *All questions* page you'll see unwanted information on the pagination section during waiting for the response.



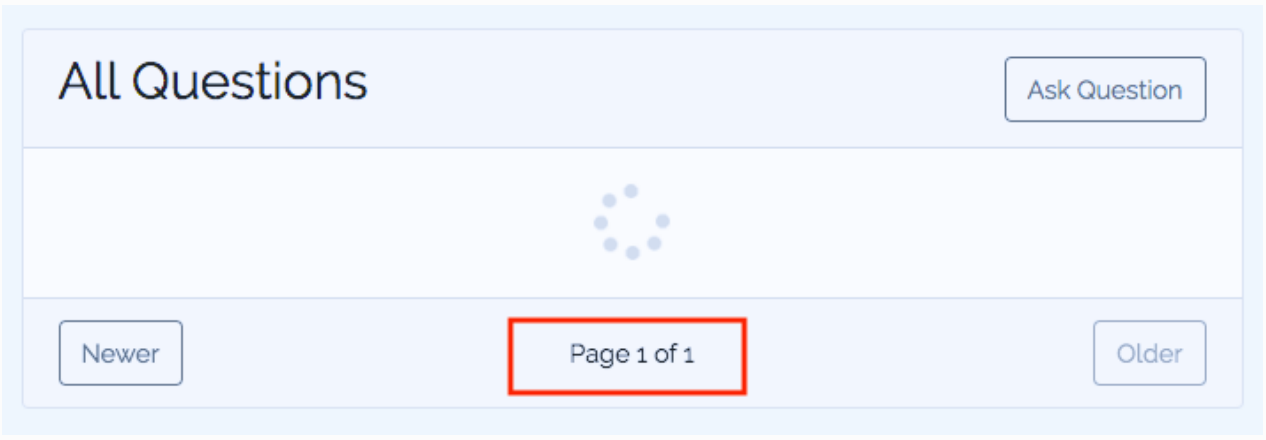
This could be not a big deal because it could happened in such a short amount of time. But in my opinion showing "Page 1 of 1" will be better.

To fix this issue you can open up the `Pagination` component. In the script section you can go to `computed` property and modify the `pageInfo` like so:

- Define a variable called `currentPage` and assign the `meta.current_page` to it. Add or operator and followed by `1`. This means when the `current_page` undefined the `currentPage` will be set to `1`.
- Define another variable called `lastPage` and do the similar thing as the first step.
- Replace the first number in the string (before 'of') with `currentPage` and replace the second number (after 'of') with `lastPage`.

```
pagesInfo () {
  let currentPage = this.meta.current_page || 1,
      lastPage = this.meta.last_page || 1;

  return `Page ${currentPage} of ${lastPage}`
},
```

Now if you visit the *All Questions* page again you'll see on the pagination section better information like this:



FIXING THE DELETE QUESTION ISSUE

If you now try to remove your own question in *All Questions* page you will see that the question is still in place. But if you reload the page it no longer in there.

After a couple minutes searching I finally found that the culprit of this problem is the `$root.loading`. It's hard to explain actually, but here I have two possible solutions to fix this issue.

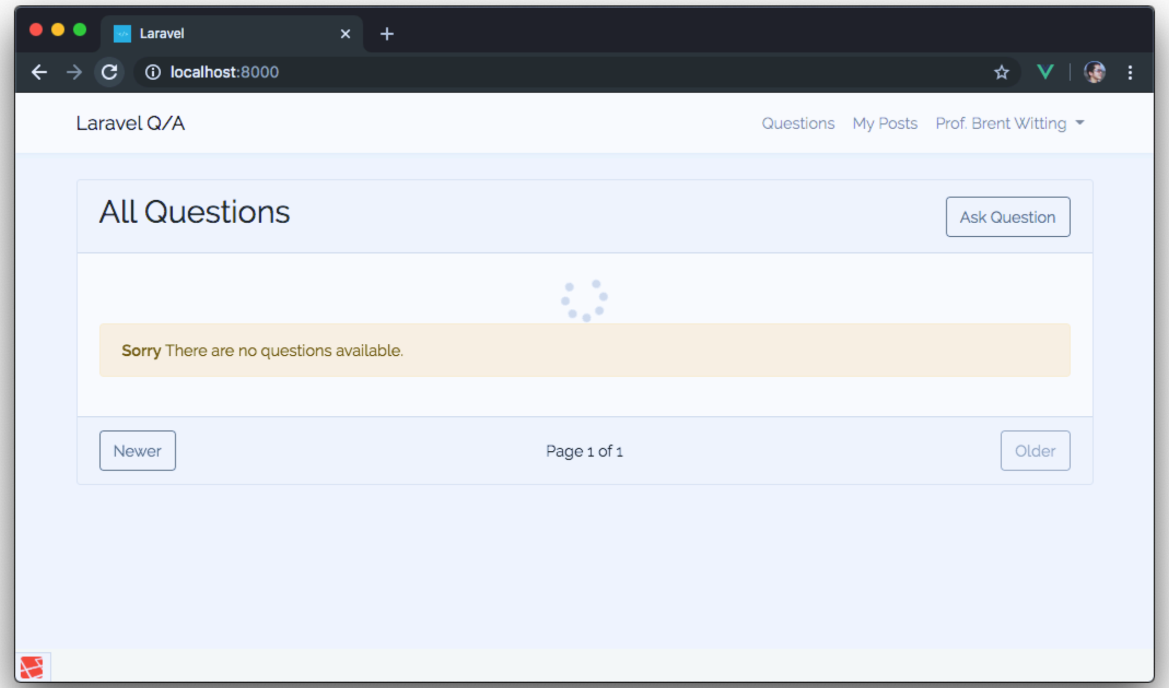
1st option: Put the spinner in different if directive

This is the very simple solution. All you have to do is to put the `spinner` and the `div` in different `if` directive. You can do that by changing the `v-else-if` in the `div` to `v-if` like this.

```
<div class="card-body">
  <spinner v-if="$root.loading"></spinner>
  <div v-if="questions.length">
    <question-excerpt @deleted="remove(index)" v-for="(question, index)
in questions" :question="question" :key="question.id"></question-
excerpt>
  </div>
  <div v-else class="alert alert-warning">
    <strong>Sorry</strong> There are no questions available.
  </div>
</div>
```

If you save that change, then try to remove your own question, you'll see the question will be removed from UI as soon as you confirm the deletion.

But the disadvantage of this method is that you could see on the first load the spinner and alert will appear at the same time.



All Questions

Sorry There are no questions available.

Page 1 of 1

If you don't like that then you can see the second option.

2nd option: Utilizing global event on delete question

The second option that you can choose to fix issue on delete question is to use the global event or event Bus in this case to emit and listen the `deleted` event.

You can open `QuestionExcerpt` component, then go to `delete` method in the script section. Replace the `this.$emit` with `eventBus.$emit` and pass in the `question.id` as a payload. Also don't forget to import the `eventBus` file.

```
import eventBus from '../event-bus'

export default {
    // ...

    methods: {
        // ...

        delete () {
            axios.delete(`/questions/${this.question.id}`)
                .then(res => {
                    this.$toast.success(res.data.message, "Success",
                                        { timeout: 2000 });
                    eventBus.$emit('deleted', this.question.id)
                });
        }
    },
}
```

Now in `Questions` component you don't need to touch the `spinner` and the `div`. All you have to do is to remove the `@deleted` event hanling in the `question-except` component calling.

```
<question-excerpt
    v-for="question in questions"
    :question="question"
    :key="question.id">
</question-excerpt>
```

You can now listen to the `deleted` event in the `mounted` hook in the script section. Because you send `question.id` as a payload, you need to find the index of the deleted question by making use the javascript `findIndex` method. Once the index found you can remove it using `remove` method.

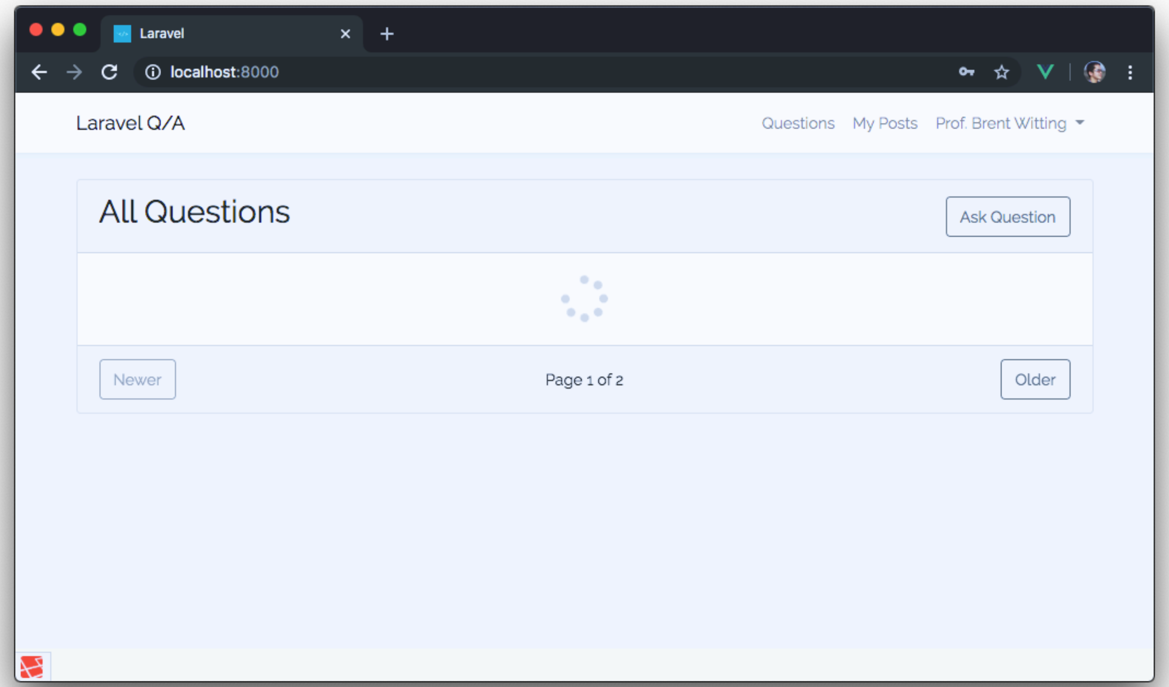
```
// ...
import eventBus from '../event-bus'

export default {
    // ...
    mounted () {
        this.fetchQuestions();

        eventBus.$on('deleted', (id) => {
            let index = this.questions.findIndex(question => id ===
question.id)
            this.remove(index)
        })
    },
    // ...
}
```

Now if you try to delete your own question again, you'll see the question will get removed from the UI as soon as you hit the *Yes* button on the delete confirmation.

EJECTING THE REQUEST INTERCEPTOR

Showing a spinner globally whenever any AJAX calls are made is very helpful and make our life easier. But in other hand it can have some pitfalls.

One example is that when you're deleting a question in *All Questions* page. Once a question successfully deleted you'll see the question for a fraction of a second and the questions list will also get reset.



If you can't see that, then you can add php `sleep` function in `destroy` method in the `QuestionsController`.

```
public function destroy(Question $question)
{
  $this->authorize("delete", $question);

  // $question->delete();
  if (env('APP_ENV') == 'local') sleep(2);

  return response()->json([
    'message' => "Your question has been deleted."
  ]);
}
```

1. Enable and disable the interceptor

To fix this issue, let's reopen the `app.js` file inside `resources/js` folder. In the `data` property of the vue instance let's add another property called `interceptor`. It should be `null` dy default.

```
data: {
  loading: false,
  interceptor: null
},
```

Next, let's cut everything inside the `created` hook. Then inside that we'll call `enableInterceptor`. This method basically will hold our previous code.

```
created () {
  this.enableInterceptor();
},
```

Now let's add `method` property, then define the `enableInterceptor` method inside, and the paste the code that we grabbed from the `created` hook.

In this method we'll also need to assign the `axios.interceptors.request.use` into `interceptor` property.

```
enableInterceptor () {
  // Add a request interceptor
  this.interceptor = axios.interceptors.request.use((config) => {
    this.loading = true
    return config;
  }, (error) => {
    this.loading = false
    return Promise.reject(error);
  });

  // Add a response interceptor
  axios.interceptors.response.use((response) => {
    this.loading = false
    return response;
  }, (error) => {
    this.loading = false
    return Promise.reject(error);
  });
},
```

Last, still in the `methods` property, let's also define another method called `disableInterceptor`. In this method we'll eject the interceptor

```
disableInterceptor () {
  axios.interceptors.request.eject(this.interceptor);
}
```

So here is the final code in the Vue instance that we have.

```
const app = new Vue({
    el: '#app',

    data: {
        loading: false,
        interceptor: null
    },

    created () {
        this.enableInterceptor();
    },

    methods: {
        enableInterceptor () {
            // Add a request interceptor
            this.interceptor = axios.interceptors.request.use((config)
=> {
                this.loading = true
                return config;
            }, (error) => {
                this.loading = false
                return Promise.reject(error);
            });

            // Add a response interceptor
            axios.interceptors.response.use((response) => {
                this.loading = false
                return response;
            }, (error) => {
                this.loading = false
                return Promise.reject(error);
            });
        },

        disableInterceptor () {
            axios.interceptors.request.eject(this.interceptor);
        }
    },

    router
});
```

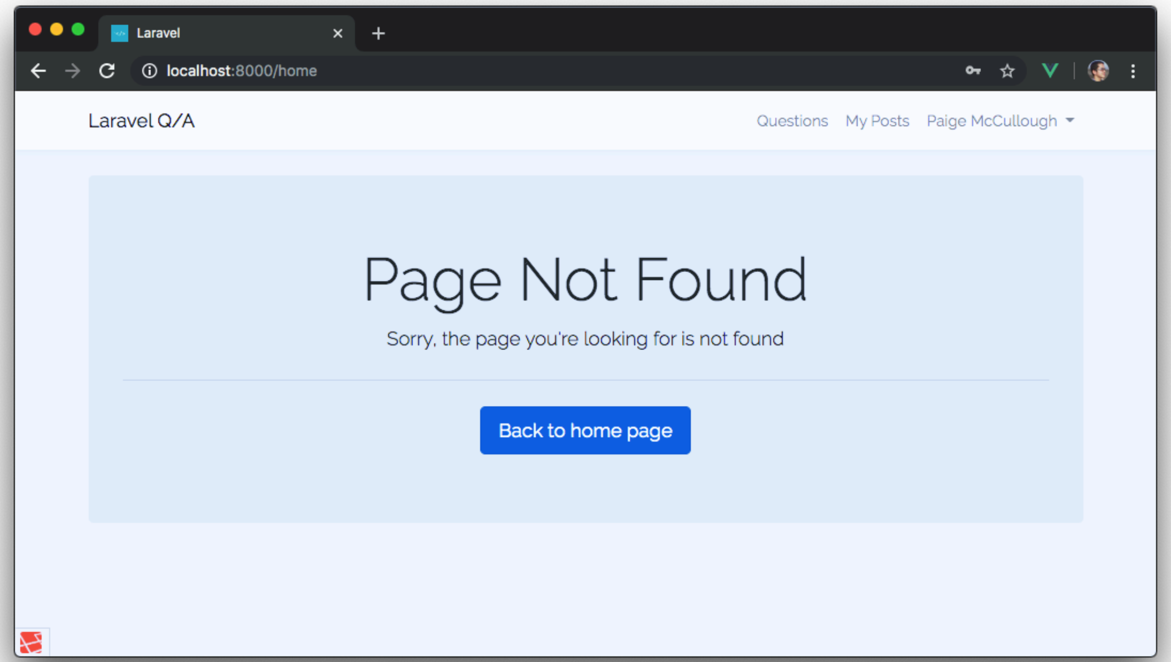
Now we can call the `disableInterceptor` in the ajax call where we don't need to show the spinner. In this case let's call it in `delete` method in the `QuestionExcerpt.vue`.

```
delete () {
  this.$root.disableInterceptor();

  axios.delete(`/questions/${this.question.id}`)
    .then(res => {
      this.$toast.success(res.data.message, "Success", { timeout: 2000
});
      eventBus.$emit('deleted', this.question.id)

      this.$root.enableInterceptor();
  });
}
```

Now if you try deleting any question in *All Questions* page you no longer see the spinner appear on the page once the question removed.

FIXING AUTHENTICATION REDIRECTION ISSUE

The last issue that we need to fix is the authentication redirection. If you login with your account you'll see **Page Not Found** message once you've successfully logged in.



Page Not Found

Sorry, the page you're looking for is not found

This because by default *built-in* Laravel authentication will redirect the page to `home` route. But now we don't have that route both in our frontend or backend routes.

So to fix this issue you have two possible solutions:

1. Change the Authentication redirection (in the backend)
2. Replace the `/my-posts` path (in the frontend)

1st Option - Change the Authentication redirection

The first option that you can choose is changing the Laravel authentiation redirection. You can open `Http/Controllers/Auth/LoginController.php` file. Then change the `redirectTo` property from `/home` to `/my-posts`.

```
protected $redirectTo = '/my-posts';
```

You can do the similar thing to other files such as `RegistrerController.php`

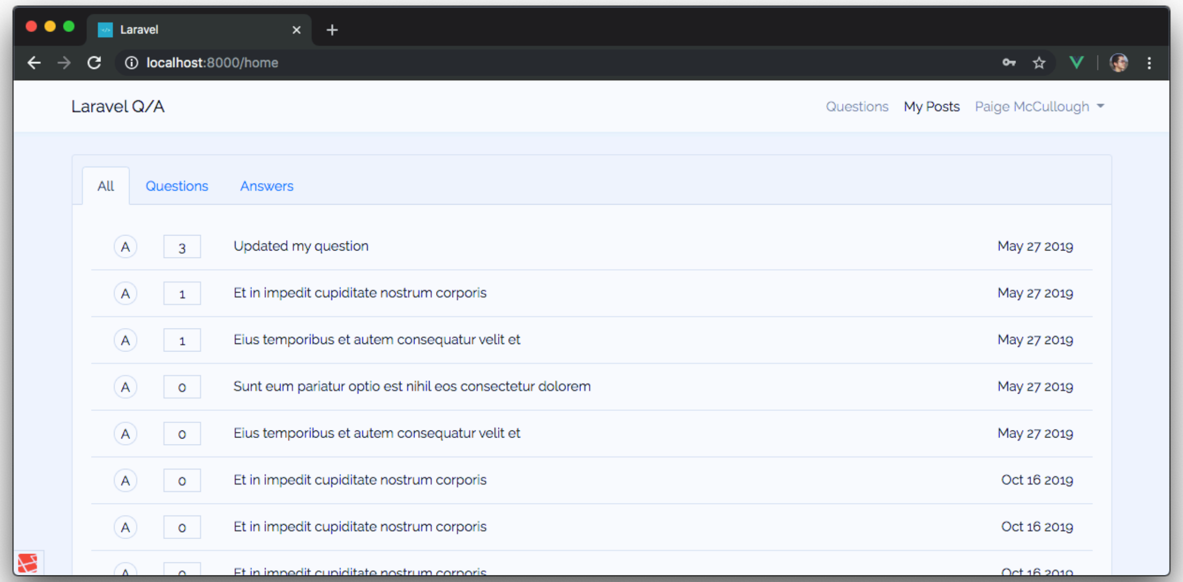
2nd Option - Replace the `/my-posts` path

The second option that you can choose is by replace the the path of `my-posts` route in your route record from `/my-posts` to `/home`.

```
{
  path: '/home',
  component: MyPostsPage,
  name: 'my-posts',
  meta: {
    requiresAuth: true
  }
},
```

By doing this way you don't need to make change in other places unless you call your route by path like this: `<router-link to="/my-posts">Link</router-link>`.

Now if you try to re-login to your app now you no longer have Page not found message.



A	3	Updated my question	May 27 2019
A	1	Et in impedit cupiditate nostrum corporis	May 27 2019
A	1	Eius temporibus et autem consequatur velit et	May 27 2019
A	0	Sunt eum pariatur optio est nihil eos consectetur dolorem	May 27 2019
A	0	Eius temporibus et autem consequatur velit et	May 27 2019
A	0	Et in impedit cupiditate nostrum corporis	Oct 16 2019
A	0	Et in impedit cupiditate nostrum corporis	Oct 16 2019
A	0	Et in impedit cupiditate nostrum corporis	Oct 16 2019

SUMMARY

In this guide we've fixed some issues such as:

- Fixed the *Page undefined of undefined* message on Pagination component
- Fixed the delete question issue on All Questions page
- Fixed Page not found error after successfully login or register
- Disabled axios interceptor for a certain Ajax call

With that being said, let's commit all changes that we made today into our git repo. Let's also merge our branc to the master branch.

```
git add .
git commit -m "Bugfix pagination, delete question & auth redirection
issue"
git checkout master
git merge lesson-65
git push origin master
```