**Implement pagination in the frontend**

Let's implement pagination in the React frontend. First of all, lets go to /post/apiPost.js and update the list()

Here we will send page number as a query parameter. So we need to send page as an argument to this list method whenever we want to execute this method to make API call to get posts.

**post/apiPost.js**

export const list = page => {

return fetch(`${process.env.REACT\_APP\_API\_URL}/posts/?page=${page}`, {

method: "GET"

})

.then(response => {

return response.json();

})

.catch(err => console.log(err));

};

We are going to use this list() in /post/Posts.js . This Posts component is what being used in '/' homepage as well as in admin '/admin' page.

There are a couple of steps to follow to make this work. I will explain each steps and write the corresponding code. At the end of this lecture, there will be full code for the /post/Posts.js component.

**/post/Posts.js**

Add page to state with default value of 1

Also add noMmorePosts

this.state = {

posts: [],

page: 1

};

Extract the list method from componentDidMount and put inside a new method called loadPosts()

loadPosts = page => {

list(page).then(data => {

if (data.error) {

console.log(data.error);

} else {

this.setState({ posts: data });

}

});

};

Now loadPosts() is reusable. This will load posts based on page number given as argument.

Pass this.state.page to loadPosts() method inside componentDidMount

componentDidMount() {

this.loadPosts(this.state.page);

}

Create two methods called loadMore and loadLess. They will require number as argument. They will use that number to setState, so the state's page can be updated.

loadMore() and loadLess() methods will be executed, when **Previous** and **Next** buttons will be clicked. loadMore() and loadLess() methods will also update state.page

loadMore = number => {

this.setState({ page: this.state.page + number });

this.loadPosts(this.state.page + number);

};

loadLess = number => {

this.setState({ page: this.state.page - number });

this.loadPosts(this.state.page - number);

};

* Inside render method, show buttons - Next and Previous
* Based on state.page count show Previous button conditionally
* If state page is greater than 1 then show previous
* Show Next button conditionally.
* Put the following code inside render() - Just before the closing </div>

{page > 1 ? (

<button

className="btn btn-raised btn-warning mr-5 mt-5 mb-5"

onClick={() => this.loadLess(1)}

>

Previous ({this.state.page - 1})

</button>

) : (

""

)}

{posts.length ? (

<button

className="btn btn-raised btn-success mt-5 mb-5"

onClick={() => this.loadMore(1)}

>

Next ({page + 1})

</button>

) : (

""

)}

Here is the full code for /post/Posts.js

**/post/Posts.js**

import React, { Component } from "react";

import { list } from "./apiPost";

import DefaultPost from "../images/mountains.jpg";

import { Link } from "react-router-dom";

class Posts extends Component {

constructor() {

super();

this.state = {

posts: [],

page: 1,

};

}

loadPosts = page => {

list(page).then(data => {

if (data.error) {

console.log(data.error);

} else {

this.setState({ posts: data });

}

});

};

componentDidMount() {

this.loadPosts(this.state.page);

}

loadMore = number => {

this.setState({ page: this.state.page + number });

this.loadPosts(this.state.page + number);

};

loadLess = number => {

this.setState({ page: this.state.page - number });

this.loadPosts(this.state.page - number);

};

renderPosts = posts => {

return (

<div className="row">

{posts.map((post, i) => {

const posterId = post.postedBy

? `/user/${post.postedBy.\_id}`

: "";

const posterName = post.postedBy

? post.postedBy.name

: " Unknown";

return (

<div className="card col-md-4" key={i}>

<div className="card-body">

<img

src={`${

process.env.REACT\_APP\_API\_URL

}/post/photo/${post.\_id}`}

alt={post.title}

onError={i =>

(i.target.src = `${DefaultPost}`)

}

className="img-thunbnail mb-3"

style={‌{ height: "200px", width: "100%" }}

/>

<h5 className="card-title">{post.title}</h5>

<p className="card-text">

{post.body.substring(0, 100)}

</p>

<br />

<p className="font-italic mark">

Posted by{" "}

<Link to={`${posterId}`}>

{posterName}{" "}

</Link>

on {new Date(post.created).toDateString()}

</p>

<Link

to={`/post/${post.\_id}`}

className="btn btn-raised btn-primary btn-sm"

>

Read more

</Link>

</div>

</div>

);

})}

</div>

);

};

render() {

const { posts, page } = this.state;

return (

<div className="container">

<h2 className="mt-5 mb-5">

{!posts.length ? "No more posts!" : "Recent Posts"}

</h2>

{this.renderPosts(posts)}

{page > 1 ? (

<button

className="btn btn-raised btn-warning mr-5 mt-5 mb-5"

onClick={() => this.loadLess(1)}

>

Previous ({this.state.page - 1})

</button>

) : (

""

)}

{posts.length ? (

<button

className="btn btn-raised btn-success mt-5 mb-5"

onClick={() => this.loadMore(1)}

>

Next ({page + 1})

</button>

) : (

""

)}

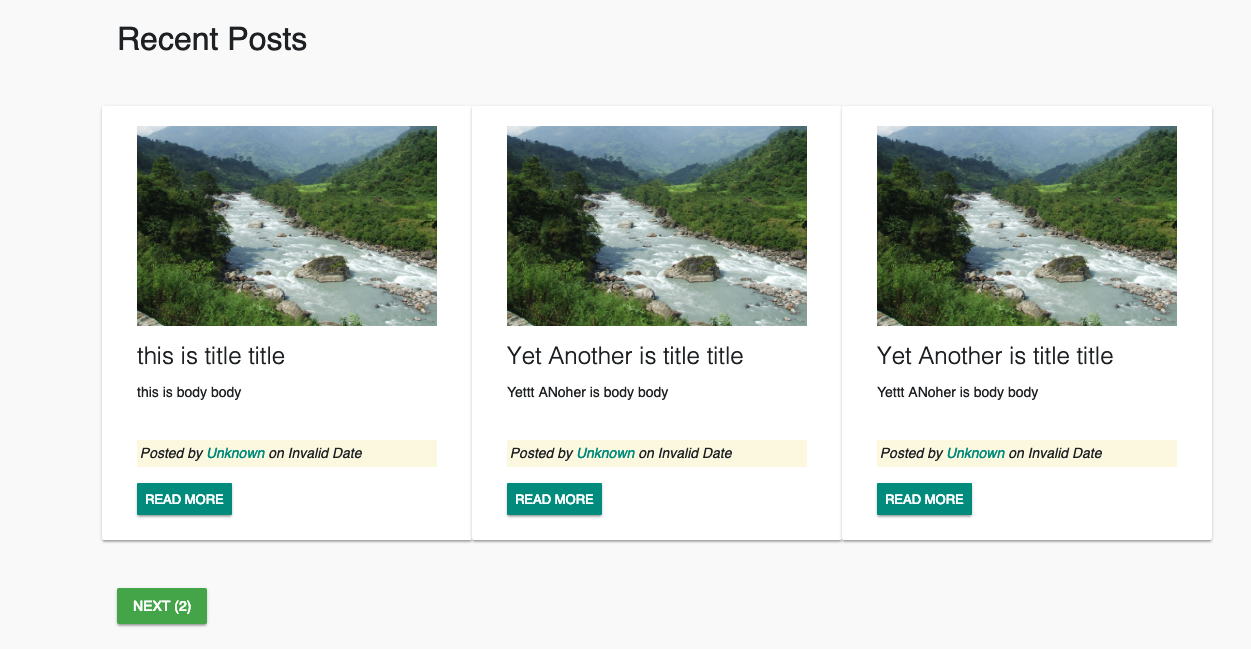
</div>

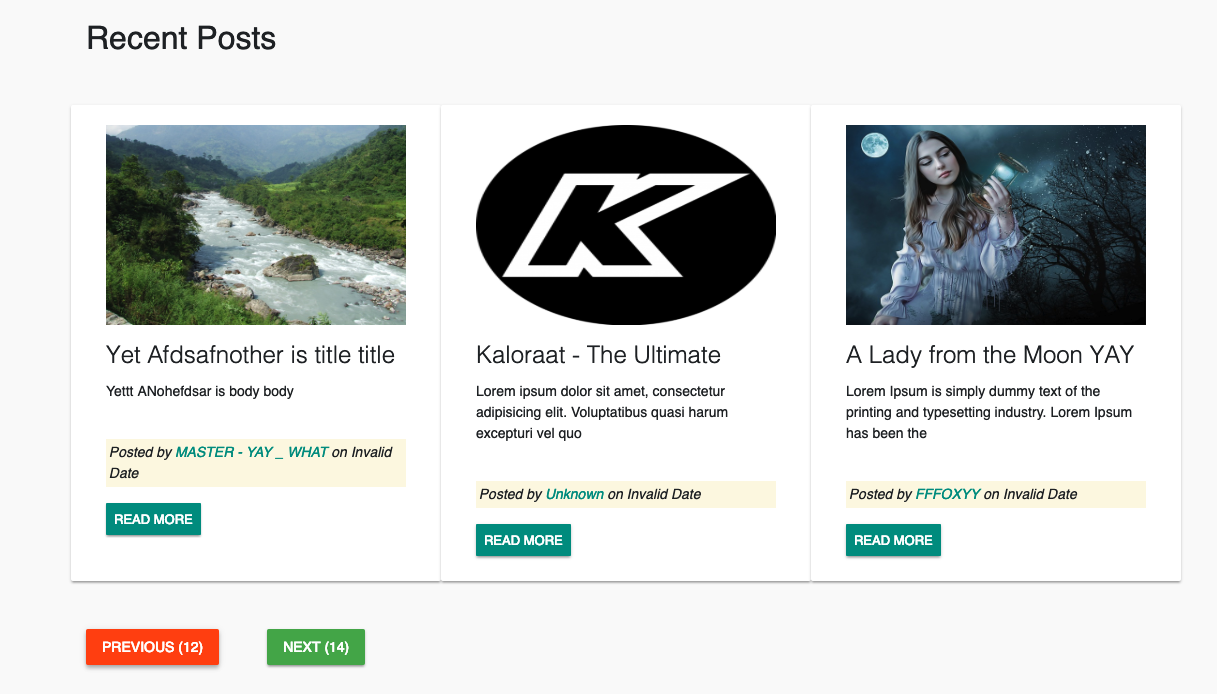
);

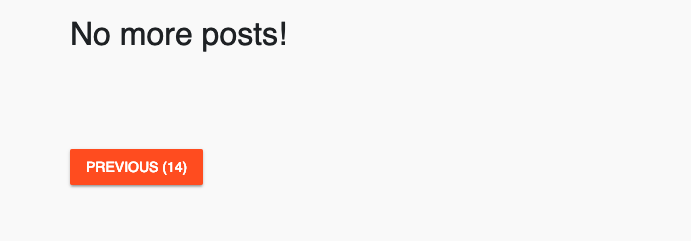
}

}

export default Posts;







This is how you can implement basic Pagination in react frontend.