### FETCH API

#### Objectives and Outcomes

In this exercise, you will use FETCH API. At the end of this exercise, you will be able to:

* Using FETCH API to GET and POST requests.

#### Using FETCH to GET API

* Import and use useState and useEffect

import { useEffect, useState } from 'react';

…

const [APIData, setAPIData]= useState([])

const baseURL=`………`;

useEffect(() => {fetch(baseURL)

.then(response =>{

if(!response.ok){

throw new Error(`HTTP Status: ${response.status}`)

}

return response.json()

})

.then(data => {setAPIData(data)})

.catch(error => console.log(error.message));

},[]);

* Rendering the players in the view

{APIData.map((data)=>{

return (

<Grid item md={4}>

<Card>

<CardMedia

component="img"

height="240"

image={data.img}

alt={data.name}

/>

<CardContent>

<Typography gutterBottom variant="h5" component="div">

{data.name}

</Typography>

<Typography variant="body2" color="text.secondary">

{data.club}

</Typography>

</CardContent>

<CardActions>

<Button size="small">{data.nation}</Button>

<Button size="small">Detail</Button>

</CardActions>

</Card>

</Grid>

)})}

#### Using FETCH to POST API

* Create a add component

<Route path='/add' element={<Add/>}></Route>

* Declare the URL of your API endpoint

const baseUrl=`………..`;

* Create Add component
* Using Formik and Yup for validating data

const formik = useFormik({

initialValues:{

name:"",

nation:"",

club:"",

cost:0,

clip:"",

description:"",

img:"",

top:false

},

onSubmit: (values)=>{

},

validationSchema: Yup.object({

name: Yup.string().required("Required.").min(2, "Must be 2 characters or more"),

nation: Yup.string().required("Required.").min(2, "Must be 2 characters or more"),

club: Yup.string().required("Required.").min(2, "Must be 2 characters or more"),

program: Yup.number().integer().typeError("Please type a number."),

description: Yup.string().required("Required.").min(10, "Must be 10 characters or more"),

clip: Yup.string().required("Required.").min(10, "Must be 10 characters or more"),

img: Yup.string().required("Required.").min(10, "Must be 10 characters or more"),

}),

});

* Update the form

<form onSubmit={formik.handleSubmit}>

<TextField

autoFocus

margin="dense"

name="name"

label="Name"

type="text"

fullWidth

variant="standard"

value={formik.values.name}

onChange={formik.handleChange}

/>

{formik.errors.name && (<Typography variant="caption" color="red">{formik.errors.name}</Typography>)}

<TextField

margin="dense"

name="club"

label="Club"

type="text"

fullWidth

variant="standard"

value={formik.values.club}

onChange={formik.handleChange}

/>

{formik.errors.club && (<Typography variant="caption" color="red">{formik.errors.club}</Typography>)}

<TextField

margin="dense"

name="nation"

label="Nation"

type="text"

fullWidth

variant="standard"

value={formik.values.nation}

onChange={formik.handleChange}

/>

{formik.errors.nation && (<Typography variant="caption" color="red">{formik.errors.nation}</Typography>)}

<TextField

margin="dense"

name="img"

label="URL of image"

type="text"

fullWidth

variant="standard"

value={formik.values.img}

onChange={formik.handleChange}

/>

{formik.errors.img && (<Typography variant="caption" color="red">{formik.errors.img}</Typography>)}

<TextField

margin="dense"

name="cost"

label="Market value"

type="text"

fullWidth

variant="standard"

value={formik.values.cost}

onChange={formik.handleChange}

/>

{formik.errors.cost && (<Typography variant="caption" color="red">{formik.errors.cost}</Typography>)}

<TextField

margin="dense"

name="clip"

label="Intro video"

type="text"

fullWidth

variant="standard"

value={formik.values.clip}

onChange={formik.handleChange}

/>

{formik.errors.clip && (<Typography variant="caption" color="red">{formik.errors.clip}</Typography>)}

<TextField

multiline

rows={2}

margin="dense"

name="description"

label="Information"

type="text"

fullWidth

variant="standard"

value={formik.values.description}

onChange={formik.handleChange}

/>

{formik.errors.description && (<Typography variant="caption" color="red" display="block">{formik.errors.description}</Typography>)}

<FormControlLabel control={<Switch/>}

label="Top players" name='agree'

/>

<br />

<Button variant="contained" size="small" type='submit'>Add</Button>

</form>

Graphical user interface, text, application

Description automatically generated

* Using the POST method in onSubmit:

onSubmit: (values)=>{

fetch(baseUrl, { method: 'POST',

body: JSON.stringify(values), headers: {

'Content-Type': 'application/json'

},

credentials: 'same-origin'

}).then(response =>{

if(!response.ok){

throw new Error(`HTTP Status: ${response.status}`)

}

return response.json()

})

.then(data => setOpen(true))

.catch(error => console.log(error.message));

},

* Show the dialog when adding successful by using Dialog and state

const [open, setOpen] = useState(false);

const handleClose = () => {

setOpen(false);

};

<Dialog

open={open}

onClose={handleClose}

aria-labelledby="alert-dialog-title"

aria-describedby="alert-dialog-description"

>

<DialogTitle id="alert-dialog-title">

{"Congraturation"}

</DialogTitle>

<DialogContent>

<DialogContentText id="alert-dialog-description">

<Alert severity="success">

<AlertTitle>Adding successful!</AlertTitle>

</Alert>

</DialogContentText>

</DialogContent>

<DialogActions>

<Button><Link to='/dashboard' style={{textDecoration:"none"}}>Dashboard</Link></Button>

<Button autoFocus onClick={handleClose}>

Close

</Button>

</DialogActions>

</Dialog>

#### Conclusions

In this exercise, you learnt how to use FETCH to GET and POST API.