**Source Code**

FindMyMovies Local Events Website

Programmer: **Luis A. Silva Soto**

**Store.js**

import { legacy\_createStore , applyMiddleware} from "redux";

import thunk from 'redux-thunk';

import eventReducer from "./reducers/eventReducer";

const initialState = {};

const store = legacy\_createStore(eventReducer, applyMiddleware(thunk));

export default store;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import { Provider } from 'react-redux';

import store from './store';

ReactDOM.render(

<Provider store={store}>

<App />

</Provider>,

document.getElementById('root')

);

**App.js**

import './App.css';

import React, {Component} from "react";

import "./bootstrap.min.css";

import { BrowserRouter as Router, Routes, Route, Link} from 'react-router-dom';

import AddEvents from './components/add-events.component';

import EventsList from './components/event-list.component';

import Events from './components/events.component';

import HomePage from './components/home-page.component';

class App extends Component{

render() {

return (

<Router>

<nav className="navbar navbar-expand navbar-dark bg-dark">

<Link to={"/"} className="navbar-brand">

FindmyMovies Local Events App!

</Link>

<div className="navbar-nav mr-auto">

<li className ="nav-item">

<Link to={"/events"} className="nav-link">

All Events

</Link>

</li>

<li className ="nav-item">

<Link to={"/searchEvents"} className="nav-link">

Search Events

</Link>

</li>

<li className="nav-item">

<Link to={"/add"} className="nav-link">

Add Event

</Link>

</li>

</div>

</nav>

<div className="container mt-3">

<Routes>

<Route path="/" element={<HomePage />} />

<Route path={'/events'} element={<EventsList></EventsList>} />

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

<Route path="/searchEvents" element={<Events />} />

</Routes>

</div>

</Router>

);

}

}

export default App;

**eventReducer.js**

import {

CREATE\_EVENTS,

UPDATE\_EVENTS,

DELETE\_EVENT,

GET\_EVENTS

} from "../actions/type";

const initialState = {

events: [],

};

const eventReducer = (state = initialState, action) => {

switch (action.type) {

case GET\_EVENTS:

return {

...state,

events: action.payload,

};

case CREATE\_EVENTS:

return {

...state,

events: action.payload,

}

case DELETE\_EVENT:

const updatedEvents = state.events.filter((event)=> event.id !==

action.payload);

return {

...state,

events: updatedEvents,

}

case UPDATE\_EVENTS:

return{

...state,

events: action.payload,

}

default:

return state;

}

};

export default eventReducer;

**event.service.js**

import http from "./http.common.all";

import axios from 'axios';

const BASE\_URL = '<http://localhost:3000/>';

class eventsDataService{

get(id) {

return http.get(`/events/${id}`);

}

create(data) {

return http.post("/events",data);

}

update(id,data){

return http.put(`/events/${id}`,data);

}

delete(id){

return http.delete(`/events/${id}`);

}

findByName(name) {

return axios.get(`${BASE\_URL}events`, { params: { name } });

}

}

export default new eventsDataService;

**http.common.all.js**

import axios from "axios";

export default axios.create({

baseURL: "<http://localhost:3000/>",

headers: {

"Content-Type": "application/json"

}

})

**Action Folder**

**types.js**

export const CREATE\_EVENTS ="CREATE\_EVENTS";

export const UPDATE\_EVENTS="UPDATE\_EVENTS";

export const DELETE\_EVENT="DELETE\_EVENT";

export const GET\_EVENTS="GET\_EVENTS";

**events.js**

import {

CREATE\_EVENTS,

UPDATE\_EVENTS,

DELETE\_EVENT,

GET\_EVENTS

} from "./type";

import axios from 'axios';

import eventsDataService from "../events.service"

export const createEvent= (type,name,date,description,location) => async (dispatch) =>{

try {

const res = await eventsDataService.create({type,name,date,description,location});

dispatch({

type: CREATE\_EVENTS,

payload: res.data,

});

return Promise.resolve(res.data);

}

catch (error){

return Promise.reject(error);

}

};

export const getEvents = () => {

return async (dispatch) => {

try {

const response = await axios.get('http://localhost:3000/events');

dispatch({

type: GET\_EVENTS,

payload: response.data,

});

console.log(response.data)

} catch (error) {

console.error('Error while fetching events:', error);

}

};

};

export const updateEvents= (id,data) => async (dispatch) =>{

try {

const res = await eventsDataService.update(id,data);

dispatch({

type: UPDATE\_EVENTS,

payload: data,

});

return Promise.resolve(res.data);

}

catch (error){

return Promise.reject(error);

}

};

export const deleteEvents= (id) => async (dispatch) =>{

try {

const res = await eventsDataService.delete(id);

dispatch({

type: DELETE\_EVENT,

payload: { id },

});

return Promise.resolve(res.data);

}

catch (error){

return Promise.reject(error);

}

};

export const findEventsByName = (name) => {

return (dispatch) => {

return new Promise((resolve, reject) => {

eventsDataService.findByName(name)

.then((response) => {

const foundEvents = response.data;

console.log(foundEvents);

dispatch({ type: "RETRIEVE\_EVENTS", payload: foundEvents });

resolve(foundEvents);

})

.catch((error) => {

console.log(error);

reject(error);

});

});

};

};

**Components Folder**

**add-events.component.js**

import React, {Component} from "react";

import {connect} from "react-redux";

import {createEvent} from "../actions/events";

import image3 from "../Images/outdoor2.jpeg"

class AddEvents extends Component {

constructor(props) {

super(props);

this.onChangeType = this.onChangeType.bind(this);

this.onChangeName = this.onChangeName.bind(this);

this.onChangeDate = this.onChangeDate.bind(this);

this.onChangeDescription = this.onChangeDescription.bind(this);

this.onChangeLocation = this.onChangeLocation.bind(this);

this.saveEvents = this.saveEvents.bind(this);

this.newEvents = this.newEvents.bind(this);

this.state = {

id: null,

type: "",

name: "",

date: "",

description: "",

location: "",

submitted:false,

};

}

onChangeType(e){

this.setState({

type: e.target.value,

});

}

onChangeName(e){

this.setState({

name: e.target.value,

});

}

onChangeDate(e){

this.setState({

date: e.target.value,

});

}

onChangeDescription(e){

this.setState({

description: e.target.value,

});

}

onChangeLocation(e){

this.setState({

location: e.target.value,

});

}

saveEvents(){

const {type,name,date,description,location} = this.state;

this.props

.createEvent(type,name,date,description,location)

.then((data)=> {

this.setState({

id:data.id,

type: data.type,

name: data.name,

date: data.date,

description: data.description,

location: data.location,

submitted:true,

});

console.log(data);

})

.catch((e) =>{

console.log(e);

});

}

newEvents(){

this.setState({

id: null,

type: "",

name: "",

date: "",

description: "",

location: "",

submitted:false,

});

}

render() {

return (

<div className="bg-secondary">

<img src={image3} alt="Image 1" className="center"/>

<div className="submit-form" >

{this.state.submitted ? (

<div>

<h4>You have submitted successfully the Event!</h4>

<button className="btn btn-success" onClick={this.newEvents}>

Add another event

</button>

</div>

) : (

<div>

<div className="form-group">

<label htmlFor="type">Type of Event</label>

<input

type="text"

className="form-control"

id="type"

required

value={this.state.type}

onChange={this.onChangeType}

name="type"

/>

</div>

<div className="form-group">

<label htmlFor="name">Name of event</label>

<input

type="text"

className="form-control"

id="name"

required

value={this.state.name}

onChange={this.onChangeName}

name="name"

/>

</div>

<div className="form-group">

<label htmlFor="date">Date of the event</label>

<input

type="text"

className="form-control"

id="date"

required

value={this.state.date}

onChange={this.onChangeDate}

name="date"

/>

</div>

<div className="form-group">

<label htmlFor="description">Description of the event</label>

<input

type="text"

className="form-control"

id="description"

required

value={this.state.description}

onChange={this.onChangeDescription}

name="description"

/>

</div>

<div className="form-group">

<label htmlFor="location">Location of the event</label>

<input

type="text"

className="form-control"

id="location"

required

value={this.state.location}

onChange={this.onChangeLocation}

name="location"

/>

</div>

<button onClick={this.saveEvents} className="btn btn-success">

Submit

</button>

</div>

)}

</div>

</div>

);

}

}

export default connect(null,{createEvent})(AddEvents);

**event-list.component.js**

import React, { useEffect, useState } from 'react';

import { useSelector, useDispatch} from 'react-redux';

import { getEvents, deleteEvents} from '../actions/events';

//import { useNavigate } from 'react-router-dom';

const EventList = () => {

const events = useSelector((state) => state.events);

const dispatch = useDispatch();

//const navigate = useNavigate();

useEffect(() => {

dispatch(getEvents());

}, [dispatch]);

const handleDeleteEvent = (eventId) => {

dispatch(deleteEvents(eventId))

.then(() => dispatch(getEvents()))

.catch((error) => {

console.log('Error deleting event:', error);

});

};

if (!Array.isArray(events)) {

return <p>Loading events...</p>;

}

if (events.length === 0 ) {

return <p>No events are available...</p>;

}

return (

<div>

<div className='bg-secondary'>

<div className='center'>

<h2>Event List</h2>

{events.map((event) => (

<div key={event.id} className="card mb-2">

<div className="card-body">

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

<p className="card-text">{event.date}</p>

<p className="card-text">{event.description}</p>

<p className="card-text">{event.location}</p>

<button className="badge btn-danger mr-1"

onClick={() => handleDeleteEvent(event.id)}>Delete</button>

</div>

</div>

))}

</div>

</div>

</div>

);

};

export default EventList;

**events.component.js**

import React, { Component } from "react";

import { connect } from "react-redux";

import { updateEvents,deleteEvents,findEventsByName } from "../actions/events";

const mapStateToProps = (state) => {

return {

events: state.events.events,

};

};

class Events extends Component {

constructor(props) {

super(props);

this.onChangeType = this.onChangeType.bind(this);

this.onChangeName = this.onChangeName.bind(this);

this.onChangeDate = this.onChangeDate.bind(this);

this.onChangeDescription = this.onChangeDescription.bind(this);

this.onChangeLocation= this.onChangeLocation.bind(this);

this.updateContent = this.updateContent.bind(this);

this.removeEvents = this.removeEvents.bind(this);

this.handleSearch = this.handleSearch.bind(this);

this.state = {

currentEvent: {

id: null,

type: "",

name: "",

date: "",

description: "",

location: "",

},

message: "",

searchQuery: "",

};

}

componentDidMount() {

const { match } = this.props;

if (match && match.params && match.params.name) {

this.props.findEventsByName(match.params.name);

}

}

onChangeType(e) {

const type = e.target.value;

this.setState(function (prevState) {

return {

currentEvent: {

...prevState.currentEvent,

type: type,

},

};

});

}

onChangeName (e) {

const name = e.target.value;

this.setState((prevState) => ({

currentEvent: {

...prevState.currentEvent,

name: name,

},

}));

}

onChangeDate (e) {

const date = e.target.value;

this.setState((prevState) => ({

currentEvent: {

...prevState.currentEvent,

date: date,

},

}));

}

onChangeDescription(e) {

const description = e.target.value;

this.setState((prevState) => ({

currentEvent: {

...prevState.currentEvent,

description: description,

},

}));

}

onChangeLocation(e) {

const location = e.target.value;

this.setState((prevState) => ({

currentEvent: {

...prevState.currentEvent,

location: location,

},

}));

}

updateContent() {

this.props

.updateEvents(this.state.currentEvent.id, this.state.currentEvent)

.then((reponse) => {

console.log(reponse);

this.setState({ message: "The Event was updated successfully!" });

})

.catch((e) => {

console.log(e);

});

}

removeEvents() {

this.props

.deleteEvents(this.state.currentEvent.id)

.then(() => {

this.setState({

currentEvent: {

id: null,

type: "",

name: "",

date: "",

description: "",

location: ""

},

message: "Event was Removed successfully"

});

})

.catch((e) => {

console.log(e);

});

}

handleSearch(event) {

event.preventDefault();

this.props.findEventsByName(this.state.searchQuery)

.then((response) => {

console.log(response);

if (response) {

let foundEvents;

if (Array.isArray(response)) {

foundEvents = response;

} else if (response.data && Array.isArray(response.data)) {

foundEvents = response.data;

} else {

console.log('Invalid response or missing data');

return;

}

console.log(foundEvents);

if (foundEvents.length > 0) {

const firstEvent = foundEvents[0];

this.setState({

currentEvent: {

id: firstEvent.id,

type: firstEvent.type,

name: firstEvent.name,

date: firstEvent.date,

description: firstEvent.description,

location: firstEvent.location

}

});

} else {

this.setState({ currentEvent: null });

}

} else {

console.log('Invalid response or missing data');

}

})

.catch((error) => {

console.log(error);

});

}

render() {

const { currentEvent } = this.state;

return (

<div>

<div>

<form onSubmit={this.handleSearch}>

<input type="text" placeholder="Search Events by Name"

value={this.state.searchQuery}

onChange={(e) => this.setState({ searchQuery: e.target.value })}/>

<button type="submit">Search</button>

</form>

</div>

{currentEvent ? (

<div className="edit-form">

<h4>Event</h4>

<form>

<div className="form-group">

<label htmlFor="type">Type of Event</label>

<input type="text" className="form-control" id="type"

value={currentEvent.type} onChange={this.onChangeType}/>

</div>

<div className="form-group">

<label htmlFor="name">Name of Event</label>

<input type="text"className="form-control"id="name" value={currentEvent.name}

onChange={this.onChangeName}

/>

</div>

<div className="form-group">

<label htmlFor="date">Date of Event</label><input type="text"

className="form-control" id="date" value={currentEvent.date}

onChange={this.onChangeDate}

/>

</div>

<div className="form-group">

<label htmlFor="description">Description</label>

<input type="text" className="form-control" id="description"

value={currentEvent.description}

onChange={this.onChangeDescription}

/>

</div>

<div className="form-group">

<label htmlFor="location">Location</label>

<input type="text" className="form-control" id="location"

value={currentEvent.location} onChange={this.onChangeLocation}

/>

</div>

</form>

<button className="badge badge-danger mr-1" onClick={this.removeEvents}>

Delete

</button>

<button type="submit" className="badge badge-success" onClick={this.updateContent}>

Update

</button>

<p>{this.state.message}</p>

</div>

) : (

<div>

<br />

<p>No Event Found...</p>

</div>

)}

</div>

);

}

}

export default connect(mapStateToProps, { updateEvents, deleteEvents,findEventsByName })(Events);

**home-page.component.js**

import React from "react";

import image from "../Images/films.jpeg"

import image2 from "../Images/OutdoorActivities.jpeg"

const HomePage = () => {

return(

<div>

<h2>Welcome to FindmyMovies Local Events website</h2>

<div className="images">

<img src={image} alt="Image 1" className="responsiveImage"/>

<img src={image2} alt="Image 2" className="responsiveImage" />

</div>

<p></p>

<p>This website has been setup to help you find Local Events in your area!</p>

<p>This can range from Films, Days out in the park or anything you can think off!,

please choose an option above for checking all the available events , adding an

event or searching events! , you can also update any event if you feel something is

not correct!

</p>

</div>

)

}

export default HomePage;

**events.json** file used as a backend for testing

{

"events": [

{

"id": 4,

"type": "Day Out",

"name": "Cinema",

"date": "3/8/2023",

"description": "Film in the Open park",

"location": "Hyde Park London"

},

{

"type": "test2",

"name": "test2",

"date": "test2",

"description": "test2",

"location": "test2",

"id": 6

}

]

}