1. **تسجيل الدخول للموقع**

loginHandler(e) {

e.preventDefault();

let loginData = {

email: this.state.email,

password: this.state.password,

};

axios

.post(serverURL + "/login", loginData)

.then((res) => {

if (res.status === 200) {

let in1Hour = 1 / 24;

Cookies.set(

"session",

{

token: res.data[0].original.token,

email: res.data[1].email,

img: res.data[1].img,

NID: res.data[1].NID,

tags: res.data[1].tags,

name: res.data[1].name,

mobile2: res.data[1].mobile2,

mobile: res.data[1].mobile,

type: res.data[1].type,

},

{

expires: in1Hour,

}

);

window.location.href = "/";

} else {

console.log(res);

}

})

.catch((err) => {

this.props.alert.error(JsonData.Login.loginAlertError);

});

}

1. **تسجيل الخروج من الموقع**

logoutHandler = async (e) =>{

console.log(e);

Cookies.remove('session');

window.location.href="/"}

1. **اضافه عمل جديد من مبادره**

const handleNewEvent = () => {

const token = Cookies.getJSON("session").token;

const config = {

headers: { Authorization: `bearer ${token}` },

};

const bodyParameters = {

name: title,

description: description,

end\_date: endDate,

img: img,

city\_id: usedCity,

tag\_id: usedTag,

registration\_date: endDateReg,

};

//console.log(bodyParameters);

axios

.post(serverURL + "/jobs/store", bodyParameters, config)

.then(function (response) {

alert.show(JsonData.NewEvent.successAlert);

setTimeout(() => {

window.location.href = "/";

}, 2000);

})

.catch((err) => {

alert.error(JsonData.NewEvent.failAlert);

});

//console.log(bodyParameters);

//console.log(token);

};

1. **قبول متطوع في عمل**

const handleAccept = (data) => {

//console.log(data);

//console.log(eventDetails);

const token = Cookies.getJSON("session").token;

const config = {

headers: { Authorization: `bearer ${token}` },

};

const bodyParameters = {

job\_id: eventDetails.id,

volunteer\_id: data.id,

value: "accepted",

};

console.log(bodyParameters);

axios

.post(serverURL + "/organization/updaterequest", bodyParameters, config)

.then(function (response) {

window.location.href = "EventVolunteer";

})

.catch(console.log);

};

1. **رفض متطوع في عمل**

const handleReject = (data) => {

//console.log(data);

//console.log(eventDetails);

const token = Cookies.getJSON("session").token;

const config = {

headers: { Authorization: `bearer ${token}` },

};

const bodyParameters = {

job\_id: eventDetails.id,

volunteer\_id: data.id,

value: "rejected",

};

console.log(bodyParameters);

axios

.post(serverURL + "/organization/updaterequest", bodyParameters, config)

.then(function (response) {

window.location.href = "EventVolunteer";

})

.catch(console.log);

};

1. **عرض اعمال المتطوع**

const transformData = (data) => {

//console.log(myEvents);

// const rows = [];

data.forEach((dataRow) => {

let tempObject = {

eventName: dataRow.name,

endDate: dataRow.end\_date,

endDateReg: dataRow.registration\_date,

active: (

<>

{dataRow.can\_register === "working" ? (

<MDBBadge color="warning">{JsonData.MyEvents.eventRegOpen}</MDBBadge>

) : dataRow.is\_ended === "working" ? (

<MDBBadge color="success">{JsonData.MyEvents.eventInWork}</MDBBadge>

) : (

<MDBBadge color="danger">{JsonData.MyEvents.eventEnded}</MDBBadge>

)}

</>

),

count: dataRow.volunteers\_count,

moreInfo: (

<>

<MDBNavLink

className="waves-effect waves-light"

to={{ pathname: `/EventVolunteer`, params: { dataRow } }}>

<MDBBtn color="success" size="sm" className="My-events-font">

{JsonData.MyEvents.eventMoreBtn}

</MDBBtn>

</MDBNavLink>

</>

),

};

setMyEventsRows((oldArray) => [...oldArray, tempObject]);

});

};

const queryMyEvents = () => {

const token = Cookies.getJSON("session").token;

const config = {

headers: { Authorization: `bearer ${token}` },

};

axios

.get(serverURL + "/organization/alljobs", config)

.then(function (response) {

//console.log(response.data.data);

transformData(response.data.data);

})

.catch(console.log);

};

1. **اضافه عمل جديد**

const handleNewEvent = () => {

const token = Cookies.getJSON("session").token;

const config = {

headers: { Authorization: `bearer ${token}` },

};

const bodyParameters = {

name: title,

description: description,

end\_date: endDate,

img: img,

city\_id: usedCity,

tag\_id: usedTag,

registration\_date: endDateReg,

};

//console.log(bodyParameters);

axios

.post(serverURL + "/jobs/store", bodyParameters, config)

.then(function (response) {

alert.show(JsonData.NewEvent.successAlert);

setTimeout(() => {

window.location.href = "/";

}, 2000);

})

.catch((err) => {

alert.error(JsonData.NewEvent.failAlert);

});

//console.log(bodyParameters);

//console.log(token);

};

1. **تسجيل متطوع او مبادره جديده**

const registerNewUser = async (e) => {

e.preventDefault();

console.log("Tags should be here");

console.log(usedTags);

const newEntity = {

name: fullName,

email: email,

password: password,

mobile: mobile,

mobile2: mobile1,

NID: NID,

img: img,

tags: usedTags,

};

//console.log(newEntity);

let requestURL;

volunteer

? (requestURL = serverURL + "/volunteer/store")

: (requestURL = serverURL + "/organization/store");

if (fullName && email && password && mobile && img && usedTags && (NID || mobile1)) {

axios

.post(requestURL, newEntity)

.then(function (response) {

alert.show(JsonData.Register.successAlert);

setTimeout(() => {

console.log("");

window.location.href = "/";

}, 2000);

})

.catch(function (error) {

alert.error(JsonData.Register.failAlert);

console.log(error);

});

} else {

alert.error(JsonData.Register.correctInfoAlert);

}

};

1. **اضافه مجال تطوع لمدير الموقع**

onst handleNewTag = () => {

const token = Cookies.getJSON("session").token;

const config = {

headers: { Authorization: `bearer ${token}` },

};

const bodyParameters = {

name: newTag,

};

if (newTag) {

// console.log(bodyParameters);

axios

.post(serverURL + "/tags/store", bodyParameters, config)

.then(function (response) {

if (response.status == 200) {

console.log(response);

alert.show(JsonData.AddTag.addTagSuccess);

} else {

alert.error(JsonData.AddTag.AddTagFail);

}

})

.catch(console.log);

} else {

alert.error(JsonData.AddTag.addTagError);

}

//console.log(bodyParameters);

//console.log(token);

};