WORKOUT BUDDY

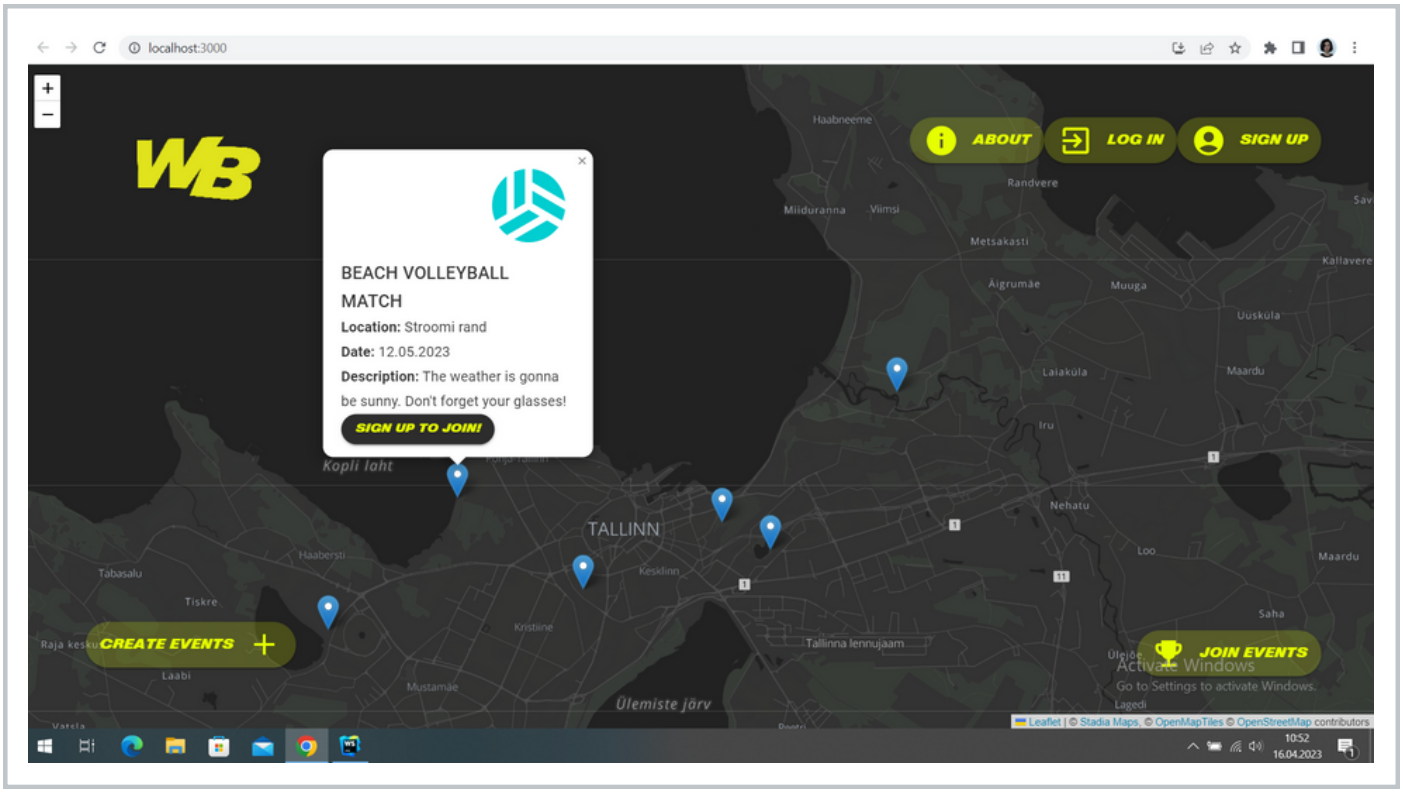
Roman-Sten, Kaisa and Arzu

April 2023

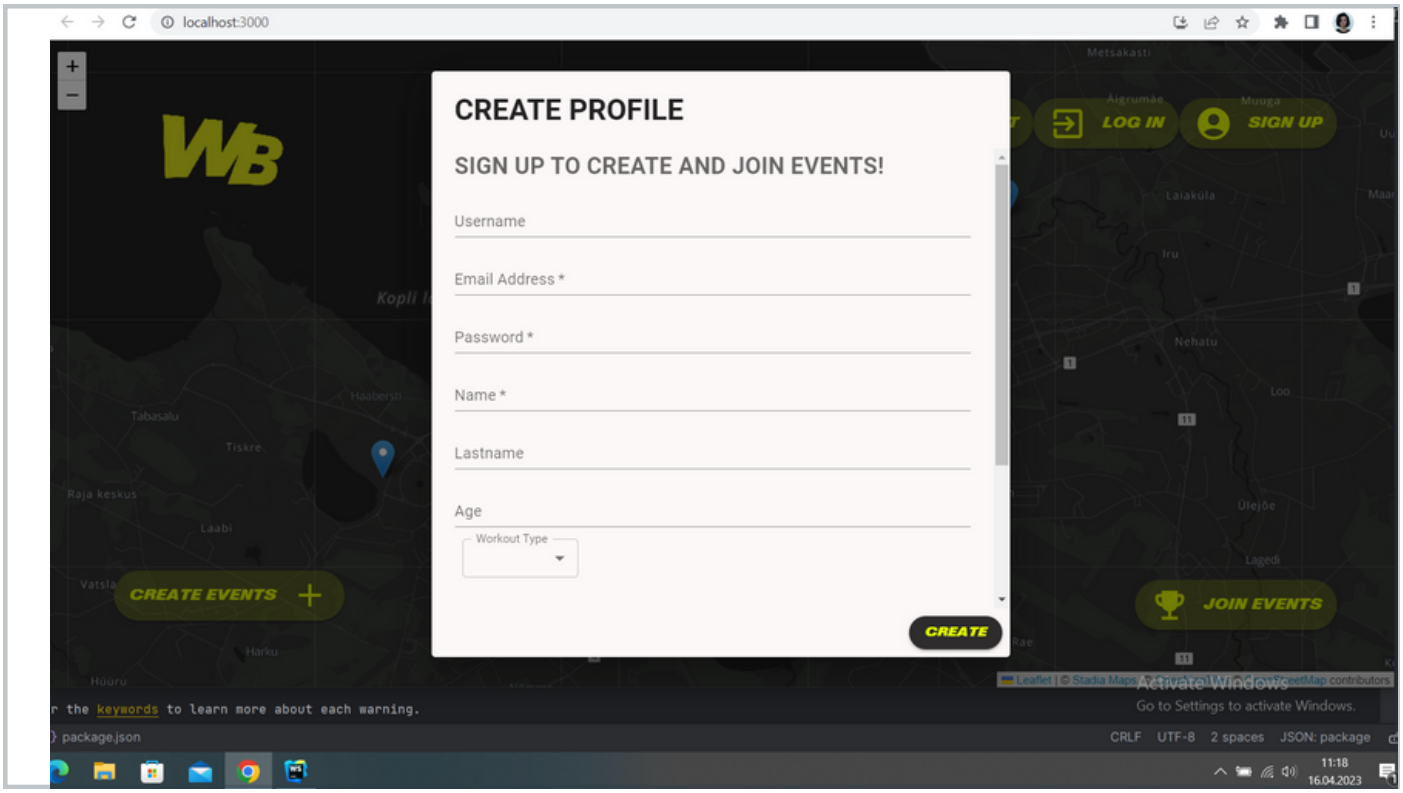
*Our website is designed to make it **easy for you to create and join sports events in your community**. Whether you are looking for a weekly running group, a game of basketball with friends, or a yoga class in the park, you can find it here.*

*Not only do we make it easy to find and join sports events, but we also provide a platform to help you **connect with workout buddies** who* share your interest*s and **fitness goals**. Working out with a buddy outside can be a fun and effective way to stay motivated and reach your fitness goals.*

So whether you're an experienced athlete or just starting your fitness journey, we invite you to explore our website, join an event, and connect with others who share your passion for sports and fitness. Let's create a healthier and happier world together!



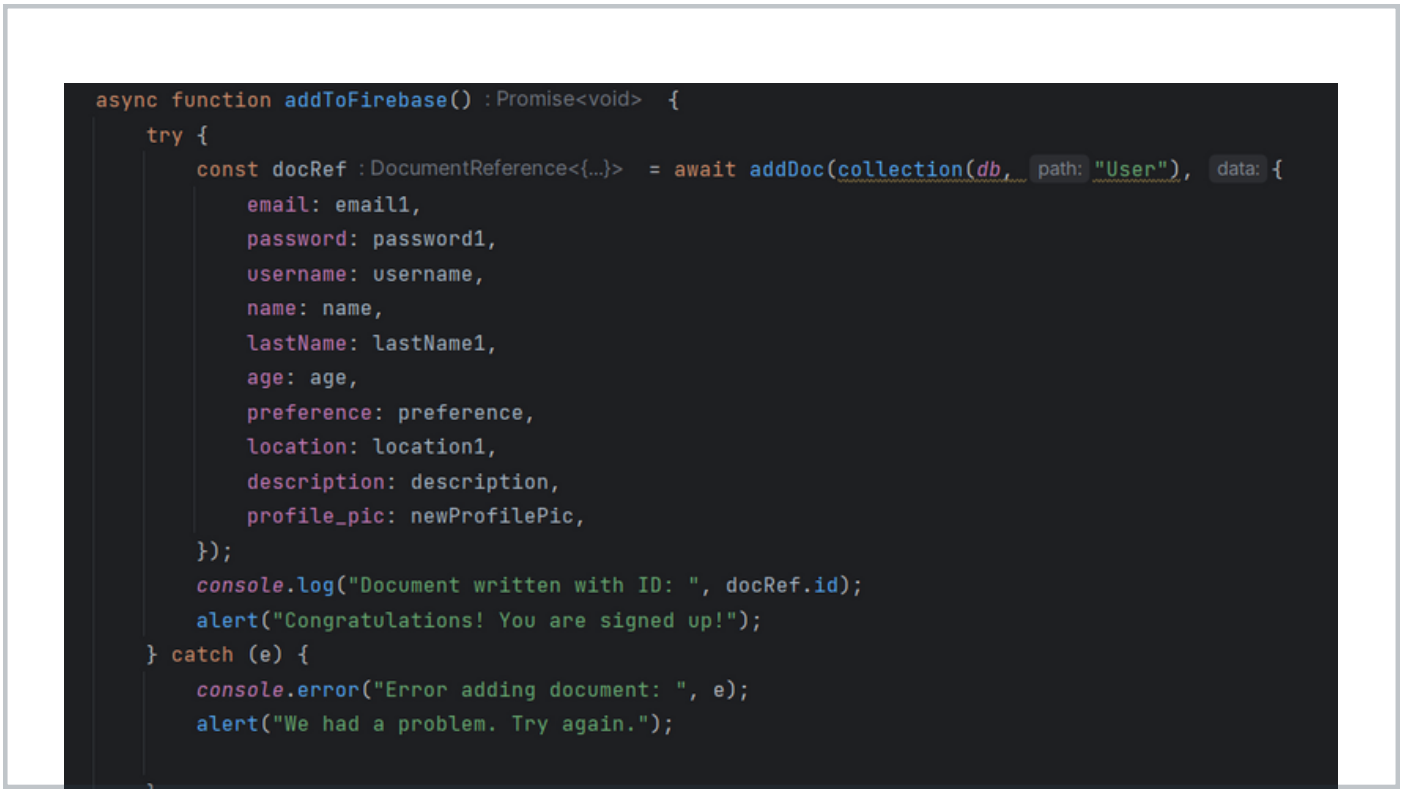
This is the look of our website, when you first enter. If you are NOT A USER, you are able to see:
* * Create event: the alert dialog will ask you to sign up.
*About - the description;
*Sign up button - leads you to profile creation;
* Log in button;
* View on the map with upcoming events that include date, location and the description, but NO TIME (only available for users, who have signed up).



If you click on a SIGN UP button, it will lead you to a dialog where you can create a profile. There are several IF conditions applied:
* email, password and name are obligatory - you cannot sign up without adding those;
* email address must contain "@";
* password must contain 8 letters + special signs.

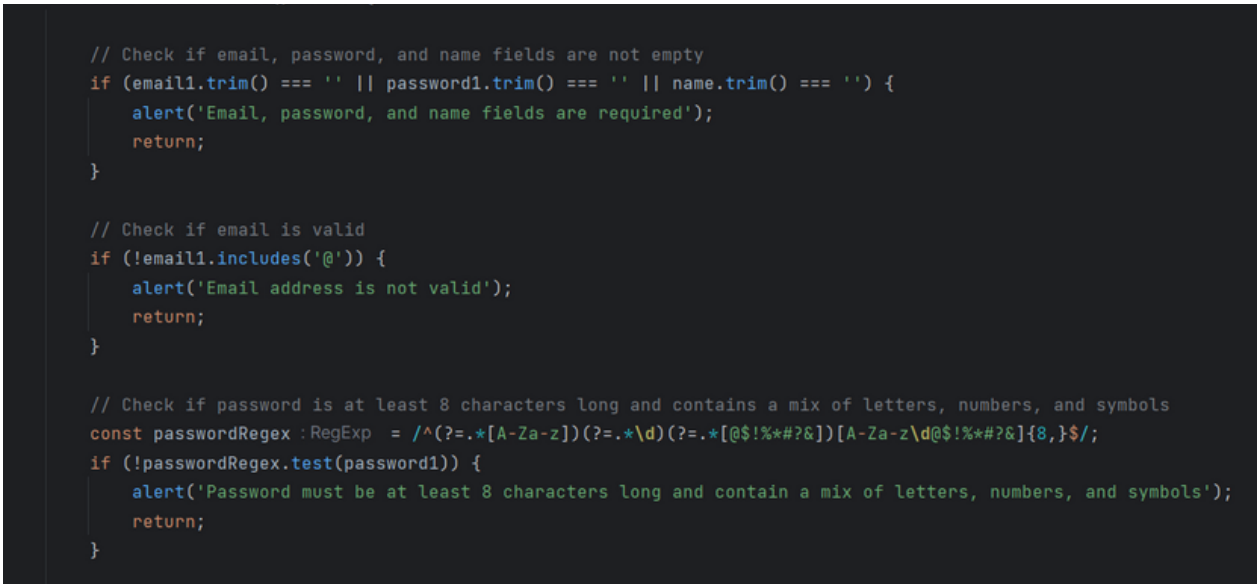
You can also upload your picture and choose a workout type you prefer.

Name, email, description, location and picture will be also linked and shown in your profile.



```
async function addToFirebase() : Promise<void>  {
    try {
        const docRef : DocumentReference<{...}>  = await addDoc(collection(db,  path: "User"),  data: {
            email: email1,
            password: password1,
            username: username,
            name: name,
            lastName: lastName1,
            age: age,
            preference: preference,
            location: location1,
            description: description,
            profile_pic: newProfilePic,
        });
        console.log("Document written with ID: ", docRef.id);
        alert("Congratulations! You are signed up!");
    } catch (e) {
        console.error("Error adding document: ", e);
        alert("We had a problem. Try again.");
```

This part checks that when You click on SIGN UP button, you are storing all the data to Firestore after filling all the "create profile" fields.



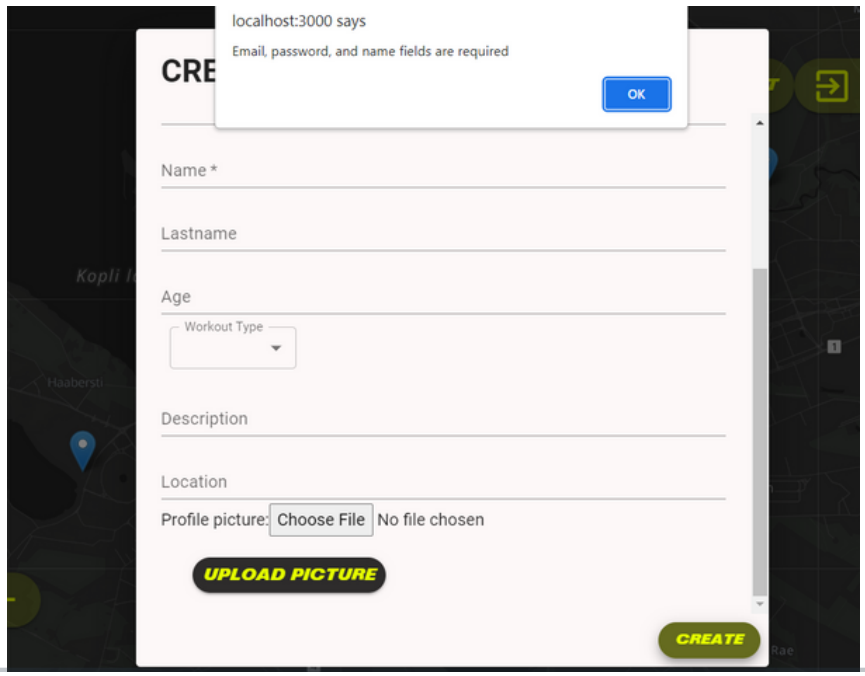
```
// Check if email, password, and name fields are not empty
if (email1.trim() === '' || password1.trim() === '' || name.trim() === '') {
    alert('Email, password, and name fields are required');
    return;
}

// Check if email is valid
if (!email1.includes('@')) {
    alert('Email address is not valid');
    return;
}

// Check if password is at least 8 characters long and contains a mix of letters, numbers, and symbols
const passwordRegex : RegExp  = /^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{8,}$/;
if (!passwordRegex.test(password1)) {
    alert('Password must be at least 8 characters long and contain a mix of letters, numbers, and symbols');
    return;
}
```

These are the IF conditions applied in profile creation dialog:

Check if email, password, and name fields are not empty.

Check if email is valid.

Check if password is at least 8 characters long and contains a mix of letters, numbers, and symbols.



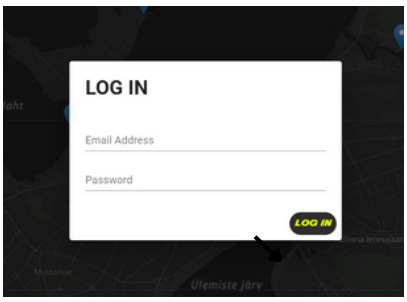
If conditions are not met, then create button triggers alert and information is not stored in Firestore.

```
//If all correct, sends info to database:
const foundObject1 : {age: number, description: string, email: string, lastName: string, location: string, name: string, passw
    email: email1,
    username: username,
    password: password1,
    name: name,
    lastName: lastName1,
    age: age,
    preference: preference,
    location: location1,
    description: description,
    profile_pic: newProfilePic,

};

//adds info to Firestore
addToFirebase();


//changes the states:
setUserIsLogged( value: true);
setLoggedUser(foundObject1);
setOpen1( value: false);
```

If all correct, sends info to database.

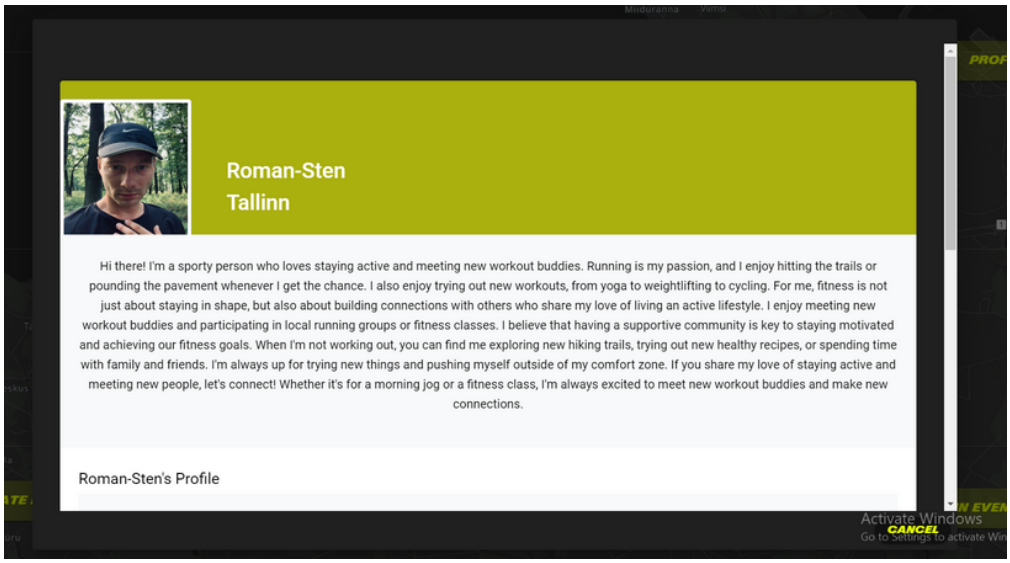
```
///Getting info from Firestore:
2 usages  ± ArZuStr +1
async function fetchUsers() : Promise<void>  {
    await getDocs(collection(db, path: 'User'))
        .then((querySnapshot : QuerySnapshot<DocumentData> ) : void  => {
            const newData : (...)[]  = querySnapshot.docs
                .map((doc : QueryDocumentSnapshot<DocumentData> ) : any & {...}  => ({...doc.data(), id: doc.id}));
            setUsers(newData);
            //console.log(newData);
        })
}

2 usages  ± Kaisa +1
async function fetchEvents() : Promise<void>  {
    await getDocs(collection(db, path: 'Events'))
        .then((querySnapshot : QuerySnapshot<DocumentData> ) : void  => {
            const newData : (...)[]  = querySnapshot.docs
                .map((doc : QueryDocumentSnapshot<DocumentData> ) : any & {...}  => ({...doc.data(), id: doc.id}));
            setEvents(newData);
            //console.log(newData);
        })
}


useEffect( effect: () : void  => {
    // Update the data (users. events) from Firestore
    fetchUsers();
    fetchEvents();
```

Gets info from a Firestore about the user and events.

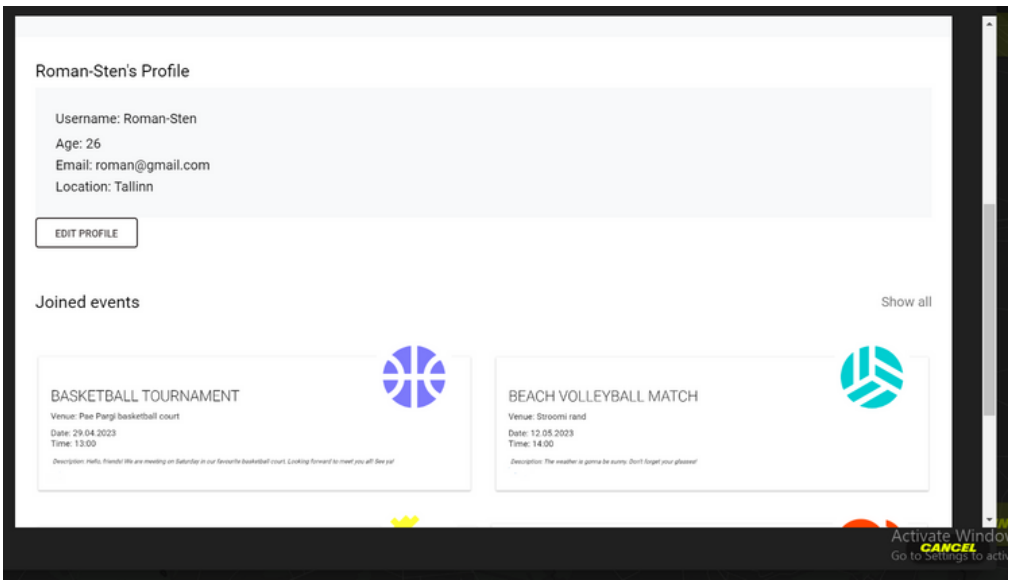



```
///Handles the "LOG IN" button, where it checks if email and password is filled and existing in Firestore:
    1 usage   opilane
    function handleLogIn() : void  {
        //console.log(email)
        //console.log(password)

        const foundObject = users.find(obj => {
            return obj.email === email && obj.password === password;
        });

        // if (Object.keys(foundObject).length > 0 ) {
        if (typeof foundObject === 'undefined') {
            setUserIsLogged( value: false);
            alert("email or password wrong");
            //setOpen(false);

        } else {
            setUserIsLogged( value: true);
            setOpen( value: false) ; //closes the login dialog
            setLoggedUser(foundObject);
        }

       //  console.log(foundObject)
```

the function HandleLogIn handles the "LOG IN" button, where it checks if email and password is filled and existing in Firestore:



This is a profile view of a user, who signs up and then clicks on Profile button.



This is a profile view of a user, who signs up and then clicks on Profile button.

```
    const [newProfilePic : string , setNewProfilePic] = useState( initialState: "")
    const [ProfilePicUpload, setProfilePicUpload] = useState( initialState: null)

    //Returns url of the picture and gives the value to useState setNewProfilePic:
    1 usage   opilane
    const upLoadProfilePic = () : void  => {
        if (ProfilePicUpload == null)
            return;
        const imageRef : StorageReference  = ref(storage,  url: `images/${ProfilePicUpload.name + v4() }`);
        uploadBytes(imageRef, ProfilePicUpload).then((snapshot : UploadResult ) : void  => {
                getDownloadURL(snapshot.ref).then((url : string ) : void  => {
                    setNewProfilePic(url);
                   // setIsProfilePicUploaded(true);
})
            }
        )
    }
```

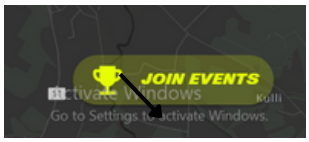
RETURN PART:

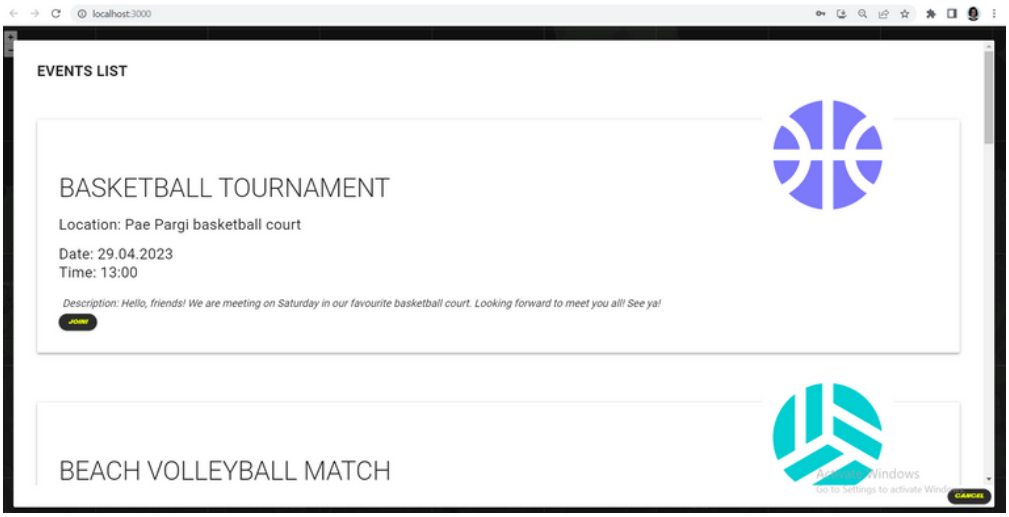
```
    />
    <label htmlFor="profilePic">Profile picture:</label>
    <input placeholder="Profile picture" type="file" id="profilePic"
           onChange={(e : ChangeEvent<HTMLInputElement> ) : void  => setProfilePicUpload(e.target.files[0])} />
    <br />
```

PROFILE PICTURE FUNCTION:

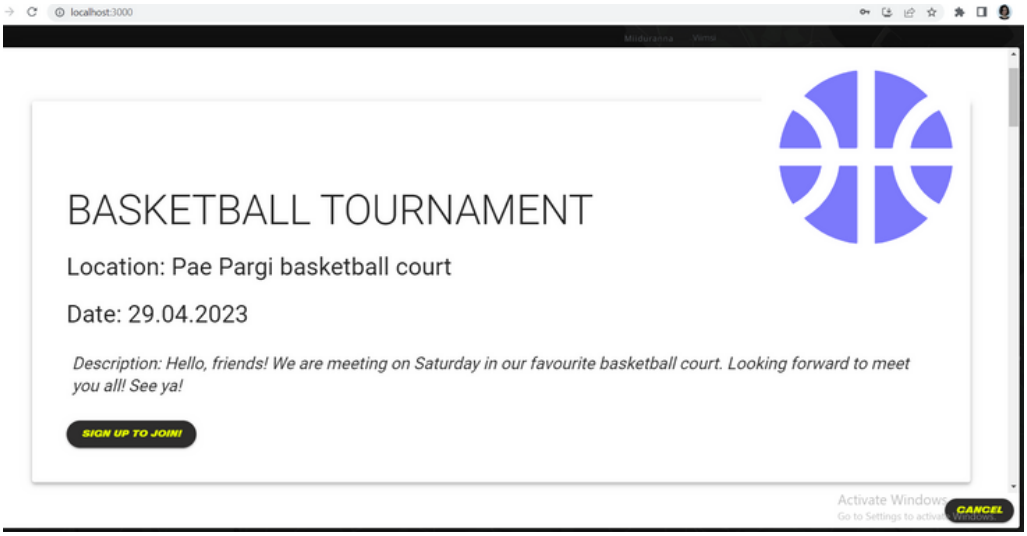
1. Returns url of the picture and gives the value to useState setNewProfilePic.

The return code, that allows the signed in user see in the Profile dialog personal information.

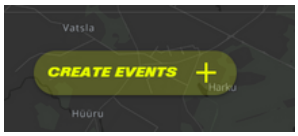


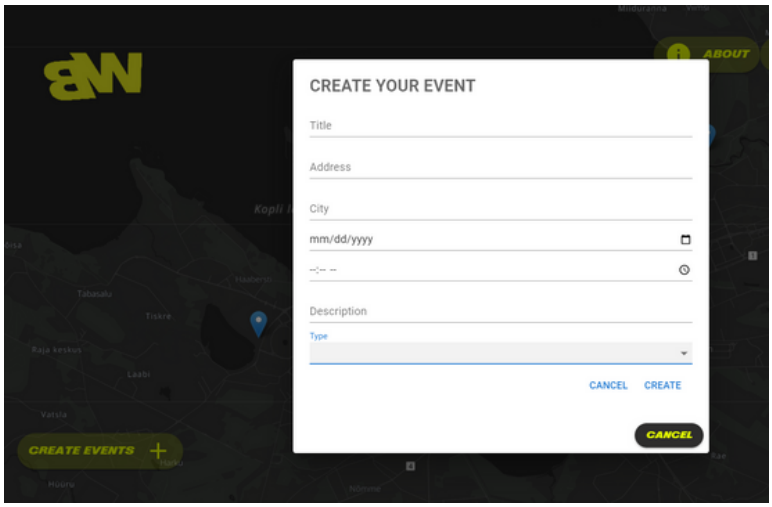


If you are logged in, you can see the full events list of upcoming events both when clicking on join events, where you can also register and join events, and on the map.

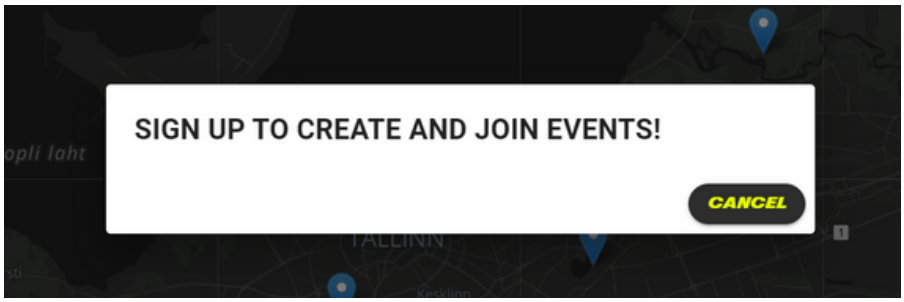


the non-user can still see the events when clicking on JOIN EVENTS button, but no information about the time will be provided. To see the full info and join, it will be necessary to click on SIGN UP TO JOIN button, which will lead to a sign up dialog.

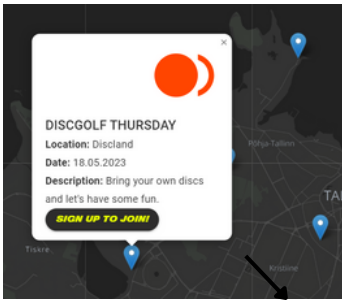




If a user will clicks on a CREATE EVENTS button, it will be possible to create events, which will be after successful creation visible both on a map and under the events list.



If not logged, when clicking on a CREATE EVENT button, it remind you to sign up (dialog) to create event.




```
App.js   package.json   Map.js

import React, {useState} from "react";
import './App.css';
import 'leaflet/dist/leaflet.css'
import {MapContainer, Marker, Popup, TileLayer} from 'react-leaflet'
import L from 'leaflet'
import NonUserEventComponent from "./Components/NonUserEventComponent";
import UserEventComponent from "./Components/UserEventComponent";

delete L.Icon.Default.prototype._getIconUrl;

L.Icon.Default.mergeOptions({
    iconRetinaUrl: require('leaflet/dist/images/marker-icon-2x.png'),
    iconUrl: require('leaflet/dist/images/marker-icon.png'),
    shadowUrl: require('leaflet/dist/images/marker-shadow.png')
});

//kaisa comment
3 usages   Kaisa
function Map(props) {

    const [userIsLogged :boolean , setUserIsLogged] = useState( initialState: true);
console.log(props.events)
    return (
        <div style={{ position: "relative", height: "100vh" }}>
        <MapContainer
            className="full-screen-map"
            center={[59.436962, 24.753574]}
            zoom={12}
            minZoom={3}
            maxZoom={19}
```

MAP component that allows users and non user to see the map on our web and later link events dialogs with the map.

```
            zoom={12}
            minZoom={3}
            maxZoom={19}
            maxBounds={[[-85.06, -180], [85.06, 180]]}
            scrollWheelZoom={true}>
            <TileLayer
                attribution='&copy; <a href="https://stadiamaps.com/">Stadia Maps</a>, &copy; <a href="https://openmaptiles.org/">OpenMapTiles</a> &copy; <a h
                url="https://tiles.stadiamaps.com/tiles/alidade_smooth_dark/{z}/{x}/{y}{r}.png" />

            {props.events.length > 0 && props.events.map((event, index) => (
                <Marker
                    key={index}
                    position={[event.lat, event.long]}>
                    <Popup className="kaisa-popup">

                        {props.userStatus === false &&
                            <NonUserEventComponent event={event} />
                        }
                        {props.userStatus === true &&
                            <UserEventComponent event={event} />
                        }

                    </Popup>
                </Marker>
            ))}
        </MapContainer>
        </div>
    );
}
```

RETURN PART:

```
        </header>

        <main>
            <Map events={events} userStatus={userIsLogged}/>
        </main>
```

MAP component that allows users and non user to see the map on our web and later link events dialogs with the map.

THANK YOU