User app, old code

Regisger

*import* React, { useState, useContext  } *from* 'react';

*import* 'bootstrap/dist/css/bootstrap.min.css';

*import* { LanguageContext } *from* '../context/languageContext';

*//import '../styles/signup.css'; // Optional custom CSS*

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

*//import Header from './header';*

*//import NavBar from './navbar';*

const *SignUp* =()=>{

*// State for input fields and password visibility*

const[*name*, *setName*]= *useState*('');

const[*email*, *setEmail*]= *useState*('');

const[*password*, *setPassword*]= *useState*('');

const[*confirmPassword*, *setConfirmPassword*]= *useState*('');

const[*isPasswordVisible*, *setIsPasswordVisible*]= *useState*(*false*);

const[*error*, *setError*]= *useState*("");

*//for multi language*

const{ *multiLang* }= *useContext*(LanguageContext);

const *navigate* = *useNavigate*();

const *text* ={

*FI* :{

*h1txt*:"Rekisteröidy",

*name*:"Nimi",

*email*:"Säköposti",

*createpwd*:"Sinun Salasana",

*confirmpwd*:"Vahvista Salasana",

*haveacc*:"Onko sinulla jo tili?",

*login*:"Kirjaudu",

*SignUp*:"Rekisteröidy"

},

*EN* :{

*h1txt*:"Sign Up",

*name*:"Name",

*email*:"Email",

*createpwd*:"Create Password",

*confirmpwd*:"Confirm Password",

*haveacc*:"Already have an account?",

*login*:"Kirjaudu",

*signUp*:"Sign up"

}

}

*// Handler for form submission*

const *handleSubmit* =(*e*)=>{

*e.preventDefault*();

if(password!==confirmPassword){

*setError*("Passwords do not match");

*return*;

}

*setError*("");

*console.log*("Email:",email,"Password:",password);

*// Redirect after successful signup*

*navigate*('/signin');

};

*return* (

<>

{*/\* Include Header and NavBar*

*<Header />*

*<NavBar />*

*\*/*}

<div *className*="min-h-screen flex items-center justify-center bg-gray-200">

        <div *className*="bg-white rounded-lg shadow-lg p-8 w-full max-w-md mx-auto md:max-w-lg">

          <h1 *className*="text-center text-2xl md:text-3xl font-bold mb-6">

          {text[multiLang]*.*h1txt}

          </h1>

          {error && <p *className*="text-red-500 text-center">{error}</p>}

          <form *onSubmit*={handleSubmit} *className*="space-y-4">

            <div *className*="form-group">

              <label *htmlFor*="name" *className*="block text-sm font-medium">

              {text[multiLang]*.*name}

              </label>

              <input

*type*="name"

*id*="name"

*name*="name"

*className*="form-control block w-full px-3 py-2 border border-gray-300 rounded focus:outline-none focus:ring focus:border-green-300"

*value*={name}

*onChange*={(e) => *setName*(*e.target.*value)}

*required*

              />

               </div>

            <div *className*="form-group">

              <label *htmlFor*="email" *className*="block text-sm font-medium">

              {text[multiLang]*.*email}

              </label>

              <input

*type*="email"

*id*="email"

*name*="email"

*className*="form-control block w-full px-3 py-2 border border-gray-300 rounded focus:outline-none focus:ring focus:border-green-300"

*value*={email}

*onChange*={(e) => *setEmail*(*e.target.*value)}

*required*

              />

            </div>

            <div *className*="form-group">

              <label *htmlFor*="password" *className*="block text-sm font-medium">

              {text[multiLang]*.*createpwd}

              </label>

              <div *className*="relative">

                <input

*type*={isPasswordVisible ? "text" : "password"}

*id*="password"

*name*="password"

*className*="form-control block w-full px-3 py-2 border border-gray-300 rounded focus:outline-none focus:ring focus:border-green-300"

*value*={password}

*onChange*={(e) => *setPassword*(*e.target.*value)}

*required*

                />

                <button

*type*="button"

*onClick*={() => *setIsPasswordVisible*(!isPasswordVisible)}

*className*="absolute right-3 top-1/2 transform -translate-y-1/2 text-gray-600 focus:outline-none"

                >

                  {isPasswordVisible ? "Hide" : "Show"}

                </button>

              </div>

            </div>

            <div *className*="form-group">

              <label *htmlFor*="confirm-password" *className*="block text-sm font-medium">

              {text[multiLang]*.*confirmpwd}

              </label>

              <input

*type*="password"

*id*="confirm-password"

*name*="confirmPassword"

*className*="form-control block w-full px-3 py-2 border border-gray-300 rounded focus:outline-none focus:ring focus:border-green-300"

*value*={confirmPassword}

*onChange*={(e) => *setConfirmPassword*(*e.target.*value)}

*required*

              />

            </div>

            <div *className*="text-center">

              <button

*type*="submit"

*className*="btn btn-primary w-full bg-purple-500 text-white rounded py-3 hover:bg-purple-600 focus:outline-none text-lg"

              >

                 {text[multiLang]*.*signUp}

              </button>

            </div>

          </form>

          <div *className*="text-center mt-6">

            <p *className*="text-sm">

            {text[multiLang]*.*haveacc}{" "}

              <a

*href*="#signin"

*className*="text-purple-500 hover:text-purple-700"

*onClick*={() => *navigate*('/signin')}

              >

                 {text[multiLang]*.*login}

              </a>

            </p>

          </div>

        </div>

      </div>

</>

);

};

*export* *default* SignUp;

old code at userdata the fetchdata

*const fetchMediaFiles = async (currentUser) => {*

*try {*

*// Fetch user's first and last name from Firestore*

*const userDoc = doc(db, "users", currentUser.uid);*

*const userSnap = await getDoc(userDoc);*

*if (userSnap.exists()) {*

*const { fm, lm } = userSnap.data(); // Extract first and last names*

*const folderName = `${fm} ${lm}`; // Combine them as folder name*

*console.log("Fetching media files:", folderName);*

*// Reference the folder in Storage*

*const userFolderRef = ref(storage, `drone\_media/${folderName}`);*

*const files = await listAll(userFolderRef);*

*if (!files ||files.items.length === 0) {*

*console.log("No files found in storage for this user.");*

*setMediaFiles([]);*

*return;*

*}*

*const filesUrls = await Promise(*

*files.items.map(async (fileRef) => {*

*//if part is new code*

*if(!fileRef){*

*console.error("fileRef is null");*

*return null;*

*}*

*return{*

*name: fileRef.name,*

*url : await getDownloadURL(fileRef),*

*//const url = await getDownloadURL(fileRef);*

*//return url;*

*};})*

*);*

*setMediaFiles(filesUrls); // Store URLs for rendering*

*} else {*

*console.error("User document does not exist in database.");*

*}*

*} catch (error) {*

*console.error("Error fetching media files:", error);*

*}*

*setLoading(false);*

*};*