Firebase.js

import { initializeApp } from "firebase/app";

import { getAuth, GoogleAuthProvider } from "firebase/auth";

import { getFirestore } from "firebase/firestore";

import { getStorage } from "firebase/storage";

const firebaseConfig = {

  apiKey: "AIzaSyBH008FIAc1g5oVLSKP2dNDF-prMsKlYBg",

  authDomain: "women-safety-reporting.firebaseapp.com",

  projectId: "women-safety-reporting",

  storageBucket: "women-safety-reporting.appspot.com", // ✅ Fixed Storage Bucket

  messagingSenderId: "1068013256318",

  appId: "1:1068013256318:web:e040c86485e35f144e622f",

  measurementId: "G-BQD0Y6Z4EH"

};

const app = initializeApp(firebaseConfig);

export const auth = getAuth(app);

export const googleProvider = new GoogleAuthProvider();

export const db = getFirestore(app);

export const storage = getStorage(app);

Login.js

import React, { useState } from "react";

import { auth, googleProvider } from "../firebase";

import { signInWithPopup, signInAnonymously, signInWithEmailAndPassword, createUserWithEmailAndPassword } from "firebase/auth";

import { useAuthState } from "react-firebase-hooks/auth";

import { useNavigate } from "react-router-dom";

import "./Login.css";

import ReportIncident from "./ReportIncident";

import AuthorityDashboard from "./AuthorityDashboard";

//npm install react-router-dom

function Login() {

  const [isRegister, setIsRegister] = useState(false);

  const [isAuthorityLogin, setIsAuthorityLogin] = useState(false); // New state for authority login

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

  const [successMessage, setSuccessMessage] = useState("");

  const [user] = useAuthState(auth);

  const navigate = useNavigate();

  const handleGoogleSignIn = async () => {

    try {

      await signInWithPopup(auth, googleProvider);

    } catch (err) {

      setError("Google Sign-In failed. Try again.");

    }

  };

  const handleAnonymousSignIn = async () => {

    try {

      await signInAnonymously(auth);

    } catch (err) {

      setError("Anonymous Sign-In failed.");

    }

  };

  const handleSubmit = async (e) => {

    e.preventDefault();

    const email = e.target.email.value;

    const password = e.target.password.value;

    try {

      if (isRegister) {

        await createUserWithEmailAndPassword(auth, email, password);

        setSuccessMessage("Registration successful! You can now log in.");

        setError("");

      } else {

        const userCredential = await signInWithEmailAndPassword(auth, email, password);

        // Check if it's an authority login

        if (isAuthorityLogin) {

          const authorityEmails = ["authority@example.com", "admin@womensafety.com"]; // Update with actual authority emails

          if (authorityEmails.includes(userCredential.user.email)) {

            navigate("/dashboard"); // Redirect to authority dashboard

          } else {

            setError("Unauthorized access. Only authorities can log in here.");

          }

        }

      }

    } catch (err) {

      setError("Error: " + err.message);

      setSuccessMessage("");

    }

  };

  if (user && !isAuthorityLogin) {

    return <ReportIncident />;

  }

  return (

    <div className="full-screen-container">

      <header className="app-header">

        <h1>Women Safety Incident Reporting System</h1>

        <p>Your safety is our priority.</p>

      </header>

      <div className="login-box">

        <div className="login-form-container">

          <h2 className="form-title">

            {isAuthorityLogin

              ? "Authority Login"

              : isRegister

              ? "Create an Account"

              : "Login to Report an Incident"}

          </h2>

          {error && <p className="error-message">{error}</p>}

          {successMessage && <p className="success-message">{successMessage}</p>}

          <form onSubmit={handleSubmit} className="form">

            <input type="email" name="email" placeholder="Email" required className="input-field" />

            <input type="password" name="password" placeholder="Password" required className="input-field" />

            <button type="submit" className="submit-button">

              {isRegister ? "Register" : "Login"}

            </button>

          </form>

          {!isAuthorityLogin && (

            <div className="toggle-register-login">

              <p onClick={() => setIsRegister(!isRegister)} className="toggle-text">

                {isRegister ? "Already have an account? Login here" : "Don't have an account? Register here"}

              </p>

            </div>

          )}

          {!isAuthorityLogin && (

            <div className="alternative-sign-in">

              <button onClick={handleGoogleSignIn} className="google-sign-in">Sign in with Google</button>

              <button onClick={handleAnonymousSignIn} className="guest-sign-in">Continue as Guest</button>

            </div>

          )}

          <div className="authority-login-toggle">

            <p onClick={() => setIsAuthorityLogin(!isAuthorityLogin)} className="toggle-text">

              {isAuthorityLogin ? "Back to User Login" : "Login as Authority"}

            </p>

          </div>

        </div>

        <footer className="login-footer">&copy; 2025 Code Tact | Women Safety Project</footer>

      </div>

    </div>

  );

}

export default Login;

Login.css

/\* Full-screen container \*/

.full-screen-container {

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  height: 100vh;

  width: 100vw;

  background-color: #f4f4f9;

  font-family: 'Arial', sans-serif;

}

/\* Header Section \*/

.app-header {

  text-align: center;

  margin-bottom: 20px;

}

.app-header h1 {

  font-size: 2.5rem;

  color: #333;

  margin-bottom: 10px;

}

.app-header p {

  font-size: 1rem;

  color: #666;

}

/\* Login Box \*/

.login-box {

  background-color: white;

  width: 100%;

  max-width: 500px;

  padding: 30px;

  border-radius: 8px;

  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

/\* Form Container \*/

.login-form-container {

  display: flex;

  flex-direction: column;

  align-items: center;

}

.form-title {

  margin-bottom: 20px;

  font-size: 20px;

  font-weight: bold;

  text-align: center;

}

/\* Input fields \*/

.input-field {

  width: 100%;

  padding: 12px;

  margin: 10px 0;

  border-radius: 5px;

  border: 1px solid #ddd;

}

/\* Submit button \*/

.submit-button {

  width: 100%;

  padding: 12px;

  background-color: #4CAF50;

  color: white;

  border-radius: 5px;

  cursor: pointer;

  border: none;

  font-size: 16px;

}

.submit-button:hover {

  background-color: #45a049;

}

/\* Error Message \*/

.error-message {

  color: red;

  font-size: 14px;

  margin-bottom: 10px;

}

/\* Success Message \*/

.success-message {

  color: green;

  font-size: 14px;

  margin-bottom: 10px;

  text-align: center;

}

/\* Toggle between Login/Registration \*/

.toggle-register-login {

  margin-top: 20px;

}

.toggle-text {

  color: #4CAF50;

  cursor: pointer;

  font-size: 14px;

}

/\* Alternative Sign-In buttons \*/

.alternative-sign-in {

  margin-top: 20px;

}

.google-sign-in, .guest-sign-in {

  width: 100%;

  padding: 12px;

  border-radius: 5px;

  margin: 10px 0;

  border: none;

  cursor: pointer;

  font-size: 16px;

}

.google-sign-in {

  background-color: #DB4437;

  color: white;

}

.google-sign-in:hover {

  background-color: #c1351d;

}

.guest-sign-in {

  background-color: #9e9e9e;

  color: white;

}

.guest-sign-in:hover {

  background-color: #757575;

}

/\* Footer \*/

.login-footer {

  background-color: #333;

  color: white;

  text-align: center;

  padding: 10px 0;

  margin-top: 20px;

}

/\* Responsive Design \*/

@media (max-width: 768px) {

  .app-header h1 {

    font-size: 2rem;

  }

  .app-header p {

    font-size: 0.9rem;

  }

  .login-box {

    padding: 20px;

  }

}

ReportIncident.js

import React, { useState } from "react";

import { db } from "../firebase"; // Importing Firebase Firestore

import { addDoc, collection } from "firebase/firestore";

import { useAuthState } from "react-firebase-hooks/auth";

import { auth } from "../firebase";

import { signOut } from "firebase/auth"; // Importing signOut for logging out functionality

import { MapContainer, TileLayer, Marker, useMapEvents } from "react-leaflet";

import "./ReportIncident.css"; // Importing the CSS file for styling

// This component listens for map clicks and sets the selected location.

function LocationSelector({ setLatitude, setLongitude, markerPosition }) {

  useMapEvents({

    click(e) {

      setLatitude(e.latlng.lat);

      setLongitude(e.latlng.lng);

    },

  });

  return markerPosition ? <Marker position={markerPosition} /> : null;

}

function ReportIncident() {

  const [incidentType, setIncidentType] = useState("");

  const [incidentTime, setIncidentTime] = useState("");

  const [incidentPlace, setIncidentPlace] = useState("");

  const [additionalDetails, setAdditionalDetails] = useState("");

  const [latitude, setLatitude] = useState(null);

  const [longitude, setLongitude] = useState(null);

  const [successMessage, setSuccessMessage] = useState("");

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

  const [user] = useAuthState(auth); // Get current logged-in user

  // Handle form submission

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (!user) {

      setError("You must be logged in to report an incident");

      return;

    }

    try {

      // Store incident data in Firestore

      await addDoc(collection(db, "incidents"), {

        userId: user.uid,           // User reporting the incident

        incidentType,               // Type of the incident

        incidentTime,               // Time of the incident

        incidentPlace,              // Location of the incident (as text)

        additionalDetails,          // Additional description

        latitude,                   // Latitude selected from the map

        longitude,                  // Longitude selected from the map

        timestamp: new Date(),      // Timestamp of report

      });

      setSuccessMessage("Incident reported successfully!");

      setIncidentType("");

      setIncidentTime("");

      setIncidentPlace("");

      setAdditionalDetails("");

      setLatitude(null);

      setLongitude(null);

    } catch (err) {

      setError("Failed to report incident. Please try again.");

    }

  };

  // Handle user logout

  const handleLogout = async () => {

    try {

      await signOut(auth);

      setSuccessMessage("Successfully logged out.");

    } catch (err) {

      setError("Failed to log out. Please try again.");

    }

  };

  return (

    <div className="report-incident-container">

      <div className="report-incident-box">

        <header className="report-incident-header">

          Women Safety Incident Reporting System

        </header>

        <div className="report-incident-form-container">

          <h2 className="form-title">Report an Incident</h2>

          {error && <p className="error-message">{error}</p>}

          {successMessage && <p className="success-message">{successMessage}</p>}

          <form onSubmit={handleSubmit} className="form">

            <div className="form-group">

              <label>Type of Incident:</label>

              <input

                type="text"

                value={incidentType}

                onChange={(e) => setIncidentType(e.target.value)}

                required

                className="input-field"

              />

            </div>

            <div className="form-group">

              <label>Time of Incident:</label>

              <input

                type="datetime-local"

                value={incidentTime}

                onChange={(e) => setIncidentTime(e.target.value)}

                required

                className="input-field"

              />

            </div>

            <div className="form-group">

              <label>Place of Incident:</label>

              <input

                type="text"

                value={incidentPlace}

                onChange={(e) => setIncidentPlace(e.target.value)}

                required

                className="input-field"

              />

            </div>

            <div className="form-group">

              <label>Additional Details:</label>

              <textarea

                value={additionalDetails}

                onChange={(e) => setAdditionalDetails(e.target.value)}

                className="input-field"

              />

            </div>

            <div className="location-group">

              <h3>Select Incident Location</h3>

              <MapContainer

                center={[28.6139, 77.2090]}

                zoom={12}

                style={{ height: "300px", width: "100%" }}

              >

                <TileLayer

                  url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"

                  attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

                />

                <LocationSelector

                  setLatitude={setLatitude}

                  setLongitude={setLongitude}

                  markerPosition={latitude && longitude ? [latitude, longitude] : null}

                />

              </MapContainer>

              {latitude && longitude && (

                <p>Selected Location: {latitude}, {longitude}</p>

              )}

            </div>

            <button type="submit" className="submit-button">

              Submit Incident Report

            </button>

          </form>

          {/\* Logout Button \*/}

          <div className="logout-container">

            <button onClick={handleLogout} className="logout-button">

              Logout

            </button>

          </div>

        </div>

        <footer className="report-incident-footer">

          &copy; 2025 Code Tact | Women Safety Project

        </footer>

      </div>

    </div>

  );

}

export default ReportIncident;

ReportIncident.css

/\* General container styling \*/

.report-incident-container {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background-color: #f3f4f6;

  }

  /\* Box styling \*/

  .report-incident-box {

    width: 100%;

    max-width: 500px;

    background-color: #fff;

    border-radius: 10px;

    padding: 20px;

    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

    display: flex;

    flex-direction: column;

  }

  /\* Header styling \*/

  .report-incident-header {

    font-size: 24px;

    font-weight: bold;

    text-align: center;

    color: #4caf50; /\* Green color for project title \*/

    margin-bottom: 20px;

  }

  /\* Form container \*/

  .report-incident-form-container {

    margin-top: 20px;

  }

  /\* Title of the form \*/

  .form-title {

    font-size: 20px;

    color: #333;

    margin-bottom: 20px;

    text-align: center;

  }

  /\* Input fields styling \*/

  .input-field {

    width: 100%;

    padding: 10px;

    margin: 10px 0;

    border: 1px solid #ccc;

    border-radius: 5px;

    box-sizing: border-box;

  }

  /\* Form group (labels and inputs) \*/

  .form-group {

    display: flex;

    flex-direction: column;

    margin-bottom: 15px;

  }

  .form-group label {

    font-size: 14px;

    color: #333;

    margin-bottom: 5px;

  }

  /\* Location button \*/

  .location-group {

    margin-top: 10px;

  }

  .location-button {

    background-color: #4caf50;

    color: white;

    padding: 10px;

    border-radius: 5px;

    border: none;

    cursor: pointer;

  }

  .location-button:hover {

    background-color: #45a049;

  }

  /\* Submit button \*/

  .submit-button {

    background-color: #4caf50;

    color: white;

    padding: 12px 20px;

    border-radius: 5px;

    border: none;

    cursor: pointer;

    margin-top: 20px;

  }

  .submit-button:hover {

    background-color: #45a049;

  }

  /\* Logout Button \*/

  .logout-container {

    margin-top: 20px;

    text-align: center;

  }

  .logout-button {

    background-color: #f44336; /\* Red color for logout \*/

    color: white;

    padding: 10px 20px;

    border-radius: 5px;

    border: none;

    cursor: pointer;

    width: 100%;

  }

  .logout-button:hover {

    background-color: #e53935;

  }

  /\* Success message \*/

  .success-message {

    color: #4caf50;

    font-size: 16px;

    margin-top: 10px;

    text-align: center;

  }

  /\* Error message \*/

  .error-message {

    color: #f44336;

    font-size: 16px;

    margin-top: 10px;

    text-align: center;

  }

  /\* Footer styling \*/

  .report-incident-footer {

    text-align: center;

    margin-top: 20px;

    font-size: 14px;

    color: #777;

  }

App.js

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

import { MapContainer, TileLayer, Marker, Popup } from "react-leaflet";

import "leaflet/dist/leaflet.css";

import { db } from "../firebase"; // Make sure this is correct

import { collection, getDocs } from "firebase/firestore";

function AuthorityDashboard() {

  const [incidents, setIncidents] = useState([]); // ✅ Ensures useState is inside a function

  useEffect(() => {

    const fetchIncidents = async () => {

      try {

        const querySnapshot = await getDocs(collection(db, "incidents"));

        const reports = querySnapshot.docs.map((doc) => ({

          id: doc.id,

          ...doc.data(),

        }));

        setIncidents(reports);

      } catch (error) {

        console.error("Error fetching incidents:", error);

      }

    };

    fetchIncidents();

  }, []);

  return (

    <div>

      <h1>Authority Dashboard</h1>

      <p>View reported incidents and analyze high-risk areas.</p>

      <MapContainer center={[28.6139, 77.2090]} zoom={12} style={{ height: "500px", width: "100%" }}>

        <TileLayer

          url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"

          attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

        />

        {incidents.map((incident) => (

          <Marker key={incident.id} position={[incident.latitude, incident.longitude]}>

            <Popup>

              <b>Type:</b> {incident.type} <br />

              <b>Description:</b> {incident.description} <br />

              <b>Location:</b> {incident.location} <br />

              <b>Time:</b> {incident.time} <br />

            </Popup>

          </Marker>

        ))}

      </MapContainer>

    </div>

  );

}

export default AuthorityDashboard;

AuthorityDashboard.js

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

import { MapContainer, TileLayer, Marker, Popup } from "react-leaflet";

import "leaflet/dist/leaflet.css";

import { db } from "../firebase"; // Import Firebase Firestore

import { collection, getDocs } from "firebase/firestore";

function AuthorityDashboard() {

  const [incidents, setIncidents] = useState([]);

  const [highRiskAreas, setHighRiskAreas] = useState([]);

  useEffect(() => {

    // Fetch incident reports from Firestore and analyze high-risk areas

    const fetchIncidents = async () => {

      try {

        const querySnapshot = await getDocs(collection(db, "incidents"));

        const reports = querySnapshot.docs.map((doc) => ({

          id: doc.id,

          ...doc.data(),

        }));

        setIncidents(reports);

        // Analyze high-risk areas by counting incidents per location.

        // Here, we use incidentPlace if available, otherwise fall back to location.

        const counts = {};

        reports.forEach((incident) => {

          const place = incident.incidentPlace || incident.location || "Unknown";

          counts[place] = (counts[place] || 0) + 1;

        });

        // Define a high-risk area as one with more than 3 incidents.

        const highRisk = Object.entries(counts)

          .filter(([place, count]) => count > 3)

          .map(([place, count]) => ({ place, count }));

        setHighRiskAreas(highRisk);

      } catch (error) {

        console.error("Error fetching incidents:", error);

      }

    };

    fetchIncidents();

  }, []);

  return (

    <div>

      <h1>Authority Dashboard</h1>

      <p>View reported incidents and analyze high-risk areas.</p>

      {highRiskAreas.length > 0 && (

        <div style={{ marginBottom: "20px" }}>

          <h2>High-Risk Areas</h2>

          <ul>

            {highRiskAreas.map((area, index) => (

              <li key={index}>

                {area.place} ({area.count} incidents)

              </li>

            ))}

          </ul>

        </div>

      )}

      <MapContainer

        center={[28.6139, 77.2090]}

        zoom={12}

        style={{ height: "500px", width: "100%" }}

      >

        <TileLayer

          url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"

          attribution='&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

        />

        {incidents.map((incident) => (

          <Marker

            key={incident.id}

            position={[incident.latitude, incident.longitude]}

          >

            <Popup>

              <b>Type:</b> {incident.incidentType || incident.type} <br />

              <b>Description:</b> {incident.additionalDetails || incident.description} <br />

              <b>Location:</b> {incident.incidentPlace || incident.location} <br />

              <b>Time:</b> {incident.incidentTime || incident.time} <br />

            </Popup>

          </Marker>

        ))}

      </MapContainer>

    </div>

  );

}

export default AuthorityDashboard;

**npm install react-leaflet@4.2.1 leaflet**