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",

  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, db } from "../firebase";

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

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

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

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

import "./Login.css";

import ReportIncident from "./ReportIncident";

import AuthorityDashboard from "./AuthorityDashboard";

function Login() {

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

  const [isAuthorityLogin, setIsAuthorityLogin] = useState(false);

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

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

  const [user] = useAuthState(auth);

  const navigate = useNavigate();

  // Emergency Button Functionality

  const handleEmergencyAlert = async () => {

    if (navigator.geolocation) {

      navigator.geolocation.getCurrentPosition(

        async (position) => {

          const { latitude, longitude } = position.coords;

          alert(`🚨 Emergency Alert Sent!\nLocation: ${latitude}, ${longitude}`);

          // Save emergency alert to Firebase

          try {

            await addDoc(collection(db, "emergency\_alerts"), {

              userId: user?.uid || "Anonymous",

              timestamp: new Date(),

              latitude,

              longitude,

            });

            console.log("Emergency alert saved!");

          } catch (error) {

            console.error("Error saving emergency alert:", error);

          }

        },

        (error) => {

          alert("⚠️ Location access denied. Please enable GPS.");

        }

      );

    } else {

      alert("❌ Geolocation is not supported by this browser.");

    }

  };

  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

        );

        if (isAuthorityLogin) {

          const authorityEmails = ["authority@example.com", "admin@abhaya.com"];

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

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

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

              id: doc.id,

              ...doc.data(),

            }));

            const counts = {};

            reports.forEach((incident) => {

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

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

            });

            const highRiskAreas = Object.entries(counts)

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

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

            navigate("/dashboard", { state: { highRiskAreas } });

          } else {

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

          }

        } else {

          navigate("/report");

        }

      }

    } catch (err) {

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

      setSuccessMessage("");

    }

  };

  if (user && !isAuthorityLogin) {

    return <ReportIncident />;

  }

  return (

    <div className="login-container">

      <div className="login-box">

        <div className="login-left">

          <h1 className="app-title">ABHAYA</h1>

          <h2>{isAuthorityLogin ? "Authority Login" : isRegister ? "Sign Up" : "Login"}</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="login-button">

              {isRegister ? "Register" : "Login"}

            </button>

          </form>

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

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

          </p>

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

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

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

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

          </p>

          <button onClick={handleEmergencyAlert} className="emergency-button">🚨 Emergency Alert</button>

        </div>

        <div className="login-right">

          <img src="/images/womenSafety.png" alt="Women Safety" className="login-image" />

        </div>

      </div>

    </div>

  );

}

export default Login;

Login.css

body, html {

  height: 100%;

  margin: 0;

  font-family: Arial, sans-serif;

  background: #ede7f6;

  display: flex;

  justify-content: center;

  align-items: center;

}

.login-container {

  width: 100%;

  height: 100vh;

  display: flex;

  justify-content: center;

  align-items: center;

  background: #ede7f6;

}

.login-box {

  display: flex;

  width: 90%;

  max-width: 1000px;

  height: 80vh;

  background: white;

  border-radius: 15px;

  overflow: hidden;

  box-shadow: 0px 4px 20px rgba(0, 0, 0, 0.1);

}

.login-left, .login-right {

  flex: 1;

  display: flex;

  flex-direction: column;

  justify-content: center;

  align-items: center;

  padding: 40px;

}

.app-title {

  font-size: 32px;

  font-weight: bold;

  color: #7c4dff;

  text-align: center;

}

.input-field {

  width: 100%;

  padding: 12px;

  margin: 10px 0;

  border: 1px solid #ddd;

  border-radius: 8px;

}

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

  width: 100%;

  padding: 12px;

  margin-top: 10px;

  border: none;

  border-radius: 8px;

  cursor: pointer;

  transition: 0.3s;

}

.login-button {

  background: #7c4dff;

  color: white;

}

.login-button:hover {

  background: #651fff;

}

.google-sign-in {

  background: #db4437;

  color: white;

}

.google-sign-in:hover {

  background: #c1351d;

}

.guest-sign-in {

  background: #555;

  color: white;

}

.guest-sign-in:hover {

  background: #333;

}

.login-right {

  background: #7c4dff;

  display: flex;

  justify-content: center;

  align-items: center;

}

.login-image {

  max-width: 100%;

  height: auto;

}

@media (max-width: 768px) {

  .login-box {

    flex-direction: column;

    height: auto;

  }

  .login-right {

    display: none;

  }

}

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, useMap, useMapEvents } from "react-leaflet";

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

// Component to search for a location using Nominatim API.

function LocationSearch({ setLatitude, setLongitude, setMapCenter }) {

  const [query, setQuery] = useState("");

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

  const handleSearch = async () => {

    console.log("Search initiated with query:", query);

    try {

      const response = await fetch(

        `https://nominatim.openstreetmap.org/search?format=json&q=${encodeURIComponent(query)}`

      );

      const data = await response.json();

      console.log("Search results:", data);

      if (data && data.length > 0) {

        const location = data[0]; // Use the first result

        const lat = parseFloat(location.lat);

        const lon = parseFloat(location.lon);

        setLatitude(lat);

        setLongitude(lon);

        setMapCenter([lat, lon]);

        setError(null);

      } else {

        setError("Location not found");

      }

    } catch (err) {

      console.error("Error searching location:", err);

      setError("Error searching location");

    }

  };

  return (

    <div className="location-search">

      <input

        type="text"

        value={query}

        onChange={(e) => {

          setQuery(e.target.value);

          setError(null);

        }}

        placeholder="Search for a location"

      />

      <button onClick={handleSearch}>Search</button>

      {error && <p style={{ color: "red" }}>{error}</p>}

    </div>

  );

}

// Component to update the map's center when the mapCenter state changes.

function ChangeMapView({ center }) {

  const map = useMap();

  map.setView(center, map.getZoom());

  return null;

}

// Component to allow users to select a location by clicking on the map.

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);

  // Set default map center (example: New Delhi)

  const [mapCenter, setMapCenter] = useState([28.6139, 77.2090]);

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

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

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

  // Handle form submission to add incident data to Firestore.

  const handleSubmit = async (e) => {

    e.preventDefault();

    if (!user) {

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

      return;

    }

    try {

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

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

        incidentType,               // Type of the incident

        incidentTime,               // Time of the incident

        incidentPlace,              // Place description entered by the user

        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!");

      // Reset form fields

      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>

              {/\* Location search component \*/}

              <LocationSearch

                setLatitude={setLatitude}

                setLongitude={setLongitude}

                setMapCenter={setMapCenter}

              />

              <MapContainer

                center={mapCenter}

                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'

                />

                {/\* Update map view when mapCenter changes \*/}

                <ChangeMapView center={mapCenter} />

                {/\* Allow user to click on the map to select a location \*/}

                <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 from "react";

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

import Login from "./components/Login";

import ReportIncident from "./components/ReportIncident";

import AuthorityDashboard from "./components/AuthorityDashboard";

function App() {

  return (

    <Router>

      <Routes>

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

        <Route path="/report" element={<ReportIncident />} />

        <Route path="/dashboard" element={<AuthorityDashboard />} />

      </Routes>

    </Router>

  );

}

export default App;

AuthorityDashboard.js

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

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

import "leaflet/dist/leaflet.css";

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

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

function AuthorityDashboard() {

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

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

  const [emergencyAlerts, setEmergencyAlerts] = useState([]);

  useEffect(() => {

    const fetchData = async () => {

      try {

        // Fetch Incidents

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

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

          id: doc.id,

          ...doc.data(),

        }));

        setIncidents(reports);

        identifyHighRiskZones(reports);

        // Fetch Emergency Alerts

        const alertSnapshot = await getDocs(collection(db, "emergency\_alerts"));

        const alerts = alertSnapshot.docs.map((doc) => ({

          id: doc.id,

          ...doc.data(),

        }));

        setEmergencyAlerts(alerts);

      } catch (error) {

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

      }

    };

    fetchData();

  }, []);

  const identifyHighRiskZones = (reports) => {

    let clusters = [];

    reports.forEach((incident) => {

      if (!incident.latitude || !incident.longitude) return;

      let foundCluster = false;

      for (let cluster of clusters) {

        const distance = getDistance(

          cluster.lat,

          cluster.lng,

          incident.latitude,

          incident.longitude

        );

        if (distance < 500) {

          cluster.count++;

          foundCluster = true;

          break;

        }

      }

      if (!foundCluster) {

        clusters.push({

          lat: incident.latitude,

          lng: incident.longitude,

          count: 1,

        });

      }

    });

    const highRisk = clusters.filter((cluster) => cluster.count >= 3);

    setHighRiskAreas(highRisk);

  };

  const getDistance = (lat1, lng1, lat2, lng2) => {

    const R = 6371000;

    const toRad = (deg) => (deg \* Math.PI) / 180;

    const dLat = toRad(lat2 - lat1);

    const dLng = toRad(lng2 - lng1);

    const a =

      Math.sin(dLat / 2) \* Math.sin(dLat / 2) +

      Math.cos(toRad(lat1)) \* Math.cos(toRad(lat2)) \*

      Math.sin(dLng / 2) \* Math.sin(dLng / 2);

    const c = 2 \* Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

    return R \* c;

  };

  return (

    <div>

      <h1>Authority Dashboard</h1>

      <p>View reported incidents, emergency alerts, and high-risk areas.</p>

      <MapContainer center={[17.3616, 78.4746]} zoom={12} style={{ height: "500px", width: "100%" }}>

        <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" attribution="&copy; OpenStreetMap contributors" />

        {/\* Incident Reports \*/}

        {incidents.map((incident) =>

          incident.latitude && incident.longitude ? (

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

              <Popup>

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

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

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

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

              </Popup>

            </Marker>

          ) : null

        )}

        {/\* High-Risk Areas \*/}

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

          area.lat && area.lng ? (

            <Circle

              key={index}

              center={[area.lat, area.lng]}

              radius={500}

              pathOptions={{ color: "red", fillColor: "red", fillOpacity: 0.4 }}

            />

          ) : null

        )}

        {/\* Emergency Alerts \*/}

        {emergencyAlerts.map((alert) =>

          alert.latitude && alert.longitude ? (

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

              <Popup>

                <b>🚨 Emergency Alert</b> <br />

                <b>User:</b> {alert.userId} <br />

                <b>Time:</b> {new Date(alert.timestamp.toDate()).toLocaleString()} <br />

                <b>Location:</b> {alert.latitude}, {alert.longitude} <br />

              </Popup>

            </Marker>

          ) : null

        )}

      </MapContainer>

      {/\* Emergency Alerts Table \*/}

      <h2>🚨 Emergency Alerts</h2>

      <table border="1">

        <thead>

          <tr>

            <th>User</th>

            <th>Time</th>

            <th>Location</th>

          </tr>

        </thead>

        <tbody>

          {emergencyAlerts.map((alert) => (

            <tr key={alert.id}>

              <td>{alert.userId}</td>

              <td>{new Date(alert.timestamp.toDate()).toLocaleString()}</td>

              <td>

                {alert.latitude}, {alert.longitude}

              </td>

            </tr>

          ))}

        </tbody>

      </table>

    </div>

  );

}

export default AuthorityDashboard;