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

import {

  View,

  Text,

  TextInput,

  TouchableOpacity,

  StyleSheet,

  SafeAreaView,

  Alert,

  ActivityIndicator,

  KeyboardAvoidingView,

  Platform,

  ScrollView,

  Image,

} from "react-native";

import { Ionicons } from "@expo/vector-icons";

import \* as Location from "expo-location";

import \* as ImagePicker from "expo-image-picker";

import MapView, { Marker } from "react-native-maps";

import { eventsService } from "../services/api";

import { authUtils } from "../utils/auth";

export default function CreateReportScreen({ navigation }) {

  const [formData, setFormData] = useState({

    title: "",

    description: "",

    type: "other",

    hasInjured: false,

    hasVictims: false,

    hasDanger: false,

  });

  const [location, setLocation] = useState(null);

  const [loading, setLoading] = useState(false);

  const [images, setImages] = useState([]);

  const [isAuthenticated, setIsAuthenticated] = useState(false);

  const [phoneVerified, setPhoneVerified] = useState(true); // Simula telefono verificato

  useEffect(() => {

    checkAuthStatus();

    requestLocationPermission();

  }, []);

  const checkAuthStatus = async () => {

    const authenticated = await authUtils.isAuthenticated();

    setIsAuthenticated(authenticated);

    if (!authenticated) {

      Alert.alert(

        "Accesso richiesto",

        "Per creare una segnalazione devi effettuare il login.",

        [

          { text: "Annulla", style: "cancel" },

          { text: "Vai al Login", onPress: () => navigation.navigate("Login") },

        ]

      );

    }

  };

  const requestLocationPermission = async () => {

    try {

      const { status } = await Location.requestForegroundPermissionsAsync();

      if (status === "granted") {

        const currentLocation = await Location.getCurrentPositionAsync({});

        // Verifica se l'utente è in Calabria

        if (

          isInCalabria(

            currentLocation.coords.latitude,

            currentLocation.coords.longitude

          )

        ) {

          setLocation({

            latitude: currentLocation.coords.latitude,

            longitude: currentLocation.coords.longitude,

            latitudeDelta: 0.01,

            longitudeDelta: 0.01,

          });

        } else {

          Alert.alert(

            "Posizione non valida",

            "Le segnalazioni possono essere create solo dalla Regione Calabria."

          );

        }

      } else {

        Alert.alert(

          "Geolocalizzazione richiesta",

          "Per creare una segnalazione è necessario attivare la geolocalizzazione."

        );

      }

    } catch (error) {

      console.error("Errore geolocalizzazione:", error);

      Alert.alert("Errore", "Impossibile ottenere la posizione corrente");

    }

  };

  const isInCalabria = (latitude, longitude) => {

    // Coordinate approssimative della Calabria

    return (

      latitude >= 37.9 &&

      latitude <= 40.2 &&

      longitude >= 15.6 &&

      longitude <= 17.2

    );

  };

  const updateFormData = (field, value) => {

    setFormData((prev) => ({ ...prev, [field]: value }));

  };

  const pickImage = async () => {

    const result = await ImagePicker.launchImageLibraryAsync({

      mediaTypes: ImagePicker.MediaTypeOptions.Images,

      allowsEditing: true,

      aspect: [4, 3],

      quality: 0.8,

    });

    if (!result.canceled) {

      setImages((prev) => [...prev, result.assets[0]]);

    }

  };

  const takePhoto = async () => {

    const result = await ImagePicker.launchCameraAsync({

      allowsEditing: true,

      aspect: [4, 3],

      quality: 0.8,

    });

    if (!result.canceled) {

      setImages((prev) => [...prev, result.assets[0]]);

    }

  };

  const removeImage = (index) => {

    setImages((prev) => prev.filter((\_, i) => i !== index));

  };

  const handleSubmit = async () => {

    if (!isAuthenticated) {

      Alert.alert(

        "Errore",

        "Devi effettuare il login per creare una segnalazione"

      );

      return;

    }

    if (!phoneVerified) {

      Alert.alert(

        "Errore",

        "Devi verificare il numero di telefono per creare segnalazioni"

      );

      return;

    }

    if (!location) {

      Alert.alert(

        "Errore",

        "Posizione non disponibile. Attiva la geolocalizzazione."

      );

      return;

    }

    if (!formData.title.trim() || !formData.description.trim()) {

      Alert.alert("Errore", "Compila tutti i campi obbligatori");

      return;

    }

    setLoading(true);

    try {

      const reportData = {

        title: formData.title,

        description: formData.description,

        type: formData.type,

        latitude: location.latitude,

        longitude: location.longitude,

        hasInjured: formData.hasInjured,

        hasVictims: formData.hasVictims,

        hasDanger: formData.hasDanger,

        images: images.map((img) => img.uri),

        createdAt: new Date().toISOString(),

      };

      await eventsService.createEvent(reportData);

      Alert.alert(

        "Segnalazione inviata",

        "La tua segnalazione è stata inviata con successo e sarà verificata dal personale competente.",

        [

          {

            text: "OK",

            onPress: () => {

              // Reset form

              setFormData({

                title: "",

                description: "",

                type: "other",

                hasInjured: false,

                hasVictims: false,

                hasDanger: false,

              });

              setImages([]);

              navigation.goBack();

            },

          },

        ]

      );

    } catch (error) {

      console.error("Errore invio segnalazione:", error);

      Alert.alert(

        "Errore",

        "Impossibile inviare la segnalazione. Riprova più tardi."

      );

    } finally {

      setLoading(false);

    }

  };

  const eventTypes = [

    { value: "flood", label: "Allagamento", icon: "water" },

    { value: "landslide", label: "Frana", icon: "triangle" },

    { value: "fire", label: "Incendio", icon: "flame" },

    { value: "obstacle", label: "Ostacolo stradale", icon: "warning" },

    { value: "accident", label: "Incidente", icon: "car-crash" },

    { value: "other", label: "Altro", icon: "alert-circle" },

  ];

  if (!isAuthenticated) {

    return (

      <SafeAreaView style={styles.container}>

        <View style={styles.errorContainer}>

          <Ionicons name="lock-closed" size={60} color="#ccc" />

          <Text style={styles.errorTitle}>Accesso richiesto</Text>

          <Text style={styles.errorText}>

            Per creare una segnalazione devi effettuare il login

          </Text>

          <TouchableOpacity

            style={styles.loginButton}

            onPress={() => navigation.navigate("Login")}

          >

            <Text style={styles.loginButtonText}>Vai al Login</Text>

          </TouchableOpacity>

        </View>

      </SafeAreaView>

    );

  }

  return (

    <SafeAreaView style={styles.container}>

      <KeyboardAvoidingView

        behavior={Platform.OS === "ios" ? "padding" : "height"}

        style={styles.keyboardView}

      >

        <ScrollView contentContainerStyle={styles.scrollContent}>

          {/\* Header \*/}

          <View style={styles.header}>

            <Ionicons name="add-circle" size={60} color="#FF6B35" />

            <Text style={styles.title}>Nuova Segnalazione</Text>

            <Text style={styles.subtitle}>

              Descrivi la situazione di emergenza

            </Text>

          </View>

          {/\* Form \*/}

          <View style={styles.form}>

            {/\* Titolo \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Titolo della segnalazione \*</Text>

              <TextInput

                style={styles.input}

                value={formData.title}

                onChangeText={(value) => updateFormData("title", value)}

                placeholder="Es: Allagamento strada provinciale"

                maxLength={100}

              />

            </View>

            {/\* Tipo di evento \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Tipo di evento \*</Text>

              <View style={styles.typeContainer}>

                {eventTypes.map((type) => (

                  <TouchableOpacity

                    key={type.value}

                    style={[

                      styles.typeButton,

                      formData.type === type.value && styles.typeButtonSelected,

                    ]}

                    onPress={() => updateFormData("type", type.value)}

                  >

                    <Ionicons

                      name={type.icon}

                      size={20}

                      color={formData.type === type.value ? "#fff" : "#666"}

                    />

                    <Text

                      style={[

                        styles.typeText,

                        formData.type === type.value && styles.typeTextSelected,

                      ]}

                    >

                      {type.label}

                    </Text>

                  </TouchableOpacity>

                ))}

              </View>

            </View>

            {/\* Descrizione \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Descrizione dettagliata \*</Text>

              <TextInput

                style={styles.textArea}

                value={formData.description}

                onChangeText={(value) => updateFormData("description", value)}

                placeholder="Descrivi la situazione nel dettaglio..."

                multiline

                numberOfLines={4}

                maxLength={500}

              />

            </View>

            {/\* Situazione di pericolo \*/}

            <View style={styles.checkboxContainer}>

              <Text style={styles.label}>Situazione di pericolo</Text>

              <TouchableOpacity

                style={styles.checkbox}

                onPress={() => updateFormData("hasDanger", !formData.hasDanger)}

              >

                <Ionicons

                  name={formData.hasDanger ? "checkbox" : "square-outline"}

                  size={24}

                  color="#FF6B35"

                />

                <Text style={styles.checkboxText}>Persone in pericolo</Text>

              </TouchableOpacity>

              <TouchableOpacity

                style={styles.checkbox}

                onPress={() =>

                  updateFormData("hasInjured", !formData.hasInjured)

                }

              >

                <Ionicons

                  name={formData.hasInjured ? "checkbox" : "square-outline"}

                  size={24}

                  color="#FF6B35"

                />

                <Text style={styles.checkboxText}>Presenza di feriti</Text>

              </TouchableOpacity>

              <TouchableOpacity

                style={styles.checkbox}

                onPress={() =>

                  updateFormData("hasVictims", !formData.hasVictims)

                }

              >

                <Ionicons

                  name={formData.hasVictims ? "checkbox" : "square-outline"}

                  size={24}

                  color="#FF6B35"

                />

                <Text style={styles.checkboxText}>Presenza di vittime</Text>

              </TouchableOpacity>

            </View>

            {/\* Posizione \*/}

            {location && (

              <View style={styles.inputContainer}>

                <Text style={styles.label}>Posizione</Text>

                <View style={styles.mapContainer}>

                  <MapView

                    style={styles.map}

                    region={location}

                    scrollEnabled={false}

                    zoomEnabled={false}

                  >

                    <Marker coordinate={location} />

                  </MapView>

                </View>

              </View>

            )}

            {/\* Foto \*/}

            <View style={styles.inputContainer}>

              <Text style={styles.label}>Foto (opzionale)</Text>

              <View style={styles.imageContainer}>

                <TouchableOpacity

                  style={styles.imageButton}

                  onPress={takePhoto}

                >

                  <Ionicons name="camera" size={24} color="#666" />

                  <Text style={styles.imageButtonText}>Scatta foto</Text>

                </TouchableOpacity>

                <TouchableOpacity

                  style={styles.imageButton}

                  onPress={pickImage}

                >

                  <Ionicons name="image" size={24} color="#666" />

                  <Text style={styles.imageButtonText}>Galleria</Text>

                </TouchableOpacity>

              </View>

              {images.length > 0 && (

                <View style={styles.imagePreview}>

                  {images.map((image, index) => (

                    <View key={index} style={styles.imageItem}>

                      <Image source={{ uri: image.uri }} style={styles.image} />

                      <TouchableOpacity

                        style={styles.removeButton}

                        onPress={() => removeImage(index)}

                      >

                        <Ionicons

                          name="close-circle"

                          size={24}

                          color="#FF3B30"

                        />

                      </TouchableOpacity>

                    </View>

                  ))}

                </View>

              )}

            </View>

            {/\* Submit Button \*/}

            <TouchableOpacity

              style={[

                styles.submitButton,

                loading && styles.submitButtonDisabled,

              ]}

              onPress={handleSubmit}

              disabled={loading}

            >

              {loading ? (

                <ActivityIndicator color="#fff" />

              ) : (

                <>

                  <Ionicons name="send" size={20} color="#fff" />

                  <Text style={styles.submitButtonText}>

                    Invia Segnalazione

                  </Text>

                </>

              )}

            </TouchableOpacity>

          </View>

        </ScrollView>

      </KeyboardAvoidingView>

    </SafeAreaView>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: "#f8f9fa",

  },

  keyboardView: {

    flex: 1,

  },

  scrollContent: {

    flexGrow: 1,

    paddingHorizontal: 20,

    paddingVertical: 20,

  },

  header: {

    alignItems: "center",

    marginBottom: 30,

  },

  title: {

    fontSize: 28,

    fontWeight: "bold",

    color: "#333",

    marginTop: 15,

    marginBottom: 10,

  },

  subtitle: {

    fontSize: 16,

    color: "#666",

    textAlign: "center",

  },

  form: {

    flex: 1,

  },

  inputContainer: {

    marginBottom: 20,

  },

  label: {

    fontSize: 16,

    fontWeight: "600",

    color: "#333",

    marginBottom: 8,

  },

  input: {

    backgroundColor: "#fff",

    borderRadius: 12,

    borderWidth: 1,

    borderColor: "#ddd",

    paddingHorizontal: 15,

    paddingVertical: 12,

    fontSize: 16,

    color: "#333",

  },

  textArea: {

    backgroundColor: "#fff",

    borderRadius: 12,

    borderWidth: 1,

    borderColor: "#ddd",

    paddingHorizontal: 15,

    paddingVertical: 12,

    fontSize: 16,

    color: "#333",

    height: 100,

    textAlignVertical: "top",

  },

  typeContainer: {

    flexDirection: "row",

    flexWrap: "wrap",

    gap: 10,

  },

  typeButton: {

    flexDirection: "row",

    alignItems: "center",

    backgroundColor: "#fff",

    borderRadius: 20,

    borderWidth: 1,

    borderColor: "#ddd",

    paddingHorizontal: 12,

    paddingVertical: 8,

    gap: 6,

  },

  typeButtonSelected: {

    backgroundColor: "#FF6B35",

    borderColor: "#FF6B35",

  },

  typeText: {

    fontSize: 14,

    color: "#666",

  },

  typeTextSelected: {

    color: "#fff",

  },

  checkboxContainer: {

    marginBottom: 20,

  },

  checkbox: {

    flexDirection: "row",

    alignItems: "center",

    marginBottom: 10,

    gap: 10,

  },

  checkboxText: {

    fontSize: 16,

    color: "#333",

  },

  mapContainer: {

    height: 200,

    borderRadius: 12,

    overflow: "hidden",

    borderWidth: 1,

    borderColor: "#ddd",

  },

  map: {

    flex: 1,

  },

  imageContainer: {

    flexDirection: "row",

    gap: 10,

    marginBottom: 10,

  },

  imageButton: {

    flex: 1,

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "center",

    backgroundColor: "#fff",

    borderRadius: 12,

    borderWidth: 1,

    borderColor: "#ddd",

    paddingVertical: 12,

    gap: 8,

  },

  imageButtonText: {

    fontSize: 14,

    color: "#666",

  },

  imagePreview: {

    flexDirection: "row",

    flexWrap: "wrap",

    gap: 10,

  },

  imageItem: {

    position: "relative",

  },

  image: {

    width: 80,

    height: 80,

    borderRadius: 8,

  },

  removeButton: {

    position: "absolute",

    top: -8,

    right: -8,

  },

  submitButton: {

    backgroundColor: "#FF6B35",

    paddingVertical: 15,

    borderRadius: 12,

    flexDirection: "row",

    alignItems: "center",

    justifyContent: "center",

    marginTop: 20,

    elevation: 3,

    shadowColor: "#000",

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.25,

    shadowRadius: 3.84,

  },

  submitButtonDisabled: {

    backgroundColor: "#ccc",

  },

  submitButtonText: {

    color: "#fff",

    fontSize: 16,

    fontWeight: "bold",

    marginLeft: 8,

  },

  errorContainer: {

    flex: 1,

    justifyContent: "center",

    alignItems: "center",

    paddingHorizontal: 40,

  },

  errorTitle: {

    fontSize: 24,

    fontWeight: "bold",

    color: "#333",

    marginTop: 20,

    marginBottom: 10,

  },

  errorText: {

    fontSize: 16,

    color: "#666",

    textAlign: "center",

    marginBottom: 30,

  },

  loginButton: {

    backgroundColor: "#FF6B35",

    paddingHorizontal: 30,

    paddingVertical: 12,

    borderRadius: 25,

  },

  loginButtonText: {

    color: "#fff",

    fontSize: 16,

    fontWeight: "bold",

  },

});