**Mian hamza alam**

**sp21-BSE-037**

import React from 'react';

import {

  View,

  Text,

  TextInput,

  TouchableOpacity,

  StyleSheet,

  Alert,

  ScrollView,

} from 'react-native';

import { Provider, useDispatch, useSelector } from 'react-redux';

import { createStore } from 'redux';

import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from '@react-navigation/stack';

import AsyncStorage from '@react-native-async-storage/async-storage';

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

*// Firebase API Endpoint*

const FIREBASE\_API\_ENDPOINT =

  'https://reactnativeapp-e9632-default-rtdb.firebaseio.com';

*// Redux initial state*

const initialState = {

  email: '',

  username: '',

  password: '',

  phone: '',

  location: '',

};

*// Redux actions*

const UPDATE\_SIGNUP = 'UPDATE\_SIGNUP';

const SET\_LOCATION = 'SET\_LOCATION';

const updateSignup = (payload) => ({ type: UPDATE\_SIGNUP, payload });

const setLocation = (location) => ({ type: SET\_LOCATION, location });

*// Redux reducer*

const reducer = (state = initialState, action) => {

  switch (action.type) {

    case UPDATE\_SIGNUP:

      return { ...state, ...action.payload };

    case SET\_LOCATION:

      return { ...state, location: action.location };

    default:

      return state;

  }

};

*// Redux store*

const store = createStore(reducer);

*// Stack Navigator*

const Stack = createStackNavigator();

*// Signup Screen*

const SignupScreen = ({ navigation }) => {

  const dispatch = useDispatch();

  const { email, username, password, phone } = useSelector((state) => state);

  const validateSignup = async () => {

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    const usernameRegex = /^[a-zA-Z]+$/;

    const phoneRegex = /^\+92-3\d{2}-\d{7}$/;

    if (!emailRegex.test(email)) {

      Alert.alert('Invalid Email', 'Please enter a valid email address.');

      return;

    }

    if (!usernameRegex.test(username)) {

      Alert.alert('Invalid Username', 'Username should only contain letters.');

      return;

    }

    if (password.length < 6) {

      Alert.alert('Invalid Password', 'Password must be at least 6 characters.');

      return;

    }

    if (!phoneRegex.test(phone)) {

      Alert.alert('Invalid Phone', 'Phone number must follow +92-3xx-xxxxxxx format.');

      return;

    }

    const userData = { email, username, password, phone };

    try {

      await fetch(`${FIREBASE\_API\_ENDPOINT}/users.json`, {

        method: 'POST',

        body: JSON.stringify(userData),

        headers: { 'Content-Type': 'application/json' },

      });

      Alert.alert('Success', 'User registered successfully!');

      navigation.navigate('Login');

    } catch (error) {

      Alert.alert('Error', 'Could not store user data. Please try again later.');

    }

  };

  return (

    <View style={styles.container}>

      <Text style={styles.title}>Signup</Text>

      <TextInput

        placeholder="Email"

        style={styles.input}

        value={email}

        onChangeText={(value) => dispatch(updateSignup({ email: value }))}

      />

      <TextInput

        placeholder="Username"

        style={styles.input}

        value={username}

        onChangeText={(value) => dispatch(updateSignup({ username: value }))}

      />

      <TextInput

        placeholder="Password"

        secureTextEntry

        style={styles.input}

        value={password}

        onChangeText={(value) => dispatch(updateSignup({ password: value }))}

      />

      <TextInput

        placeholder="Phone (+92-3xx-xxxxxxx)"

        style={styles.input}

        value={phone}

        onChangeText={(value) => dispatch(updateSignup({ phone: value }))}

      />

      <TouchableOpacity style={styles.button} onPress={validateSignup}>

        <Text style={styles.buttonText}>Signup</Text>

      </TouchableOpacity>

    </View>

  );

};

*// Login Screen*

const LoginScreen = ({ navigation }) => {

  const { email, password } = useSelector((state) => state);

  const dispatch = useDispatch();

  const validateLogin = async () => {

    if (!email || !password) {

      Alert.alert('Error', 'Please fill in all fields.');

      return;

    }

    try {

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

      if (status !== 'granted') {

        Alert.alert('Permission Denied', 'Location access is required.');

        return;

      }

      const location = await Location.getCurrentPositionAsync();

      const locationString = `${location.coords.latitude}, ${location.coords.longitude}`;

      dispatch(setLocation(locationString));

      await AsyncStorage.setItem('userLocation', locationString);

      navigation.navigate('Profile');

    } catch (error) {

      Alert.alert('Error', 'Could not fetch location. Please try again.');

    }

  };

  return (

    <View style={styles.container}>

      <Text style={styles.title}>Login</Text>

      <TextInput

        placeholder="Email"

        style={styles.input}

        value={email}

        onChangeText={(value) => dispatch(updateSignup({ email: value }))}

      />

      <TextInput

        placeholder="Password"

        secureTextEntry

        style={styles.input}

        value={password}

        onChangeText={(value) => dispatch(updateSignup({ password: value }))}

      />

      <TouchableOpacity style={styles.button} onPress={validateLogin}>

        <Text style={styles.buttonText}>Login</Text>

      </TouchableOpacity>

    </View>

  );

};

*// Profile Screen*

const ProfileScreen = () => {

  const { email, username, phone, location } = useSelector((state) => state);

  return (

    <ScrollView style={styles.container}>

      <Text style={styles.title}>My Profile</Text>

      <View style={styles.profileCard}>

        <Text>Email: {email}</Text>

        <Text>Username: {username}</Text>

        <Text>Phone: {phone}</Text>

        <Text>Location: {location}</Text>

      </View>

    </ScrollView>

  );

};

*// Main App Component*

const App = () => {

  return (

    <Provider store={store}>

      <NavigationContainer>

        <Stack.Navigator initialRouteName="Signup">

          <Stack.Screen name="Signup" component={SignupScreen} />

          <Stack.Screen name="Login" component={LoginScreen} />

          <Stack.Screen name="Profile" component={ProfileScreen} />

        </Stack.Navigator>

      </NavigationContainer>

    </Provider>

  );

};

*// Styles*

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#f5f5f5',

    paddingHorizontal: 20,

    paddingVertical: 10,

  },

  title: {

    fontSize: 24,

    fontWeight: 'bold',

    marginBottom: 20,

    textAlign: 'center',

  },

  input: {

    borderWidth: 1,

    borderColor: '#ddd',

    padding: 10,

    marginBottom: 15,

    borderRadius: 5,

    backgroundColor: '#fff',

  },

  button: {

    backgroundColor: '#ff6347',

    padding: 15,

    borderRadius: 5,

  },

  buttonText: {

    color: '#fff',

    fontSize: 16,

    textAlign: 'center',

  },

  profileCard: {

    padding: 20,

    backgroundColor: '#fff',

    borderRadius: 10,

    elevation: 3,

  },

});

export default App;

