Mobile Application Development

NAME : **ADEEL KHAN**

REG NO: **SP22-BSE-047**

SUBMITTED TO: **SIR KAMRAN**

ASSINGMENT # 4

```
// Install the required packages if not already done:
// npm install @reduxjs/toolkit react-redux
// npm install @react-native-async-storage/async-storage
// npm install @react-native-firebase/app @react-native-firebase/auth
// npm install react-native-geolocation-service

import React, { useEffect } from 'react';
import { View, Text, TextInput, Button, StyleSheet, Alert } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import { Provider, useDispatch, useSelector } from 'react-redux';
import { configureStore, createSlice } from '@reduxjs/toolkit';
import AsyncStorage from '@react-native-async-storage/async-storage';
import auth from '@react-native-firebase/auth';
import Geolocation from 'react-native-geolocation-service';

// Redux Slice for User
const userSlice = createSlice({
  name: 'user',
  initialState: {
    username: '',
    email: '',
    phone: '',
    location: '',
  },
  reducers: {
    setUser(state, action) {
      return { ...state, ...action.payload };
    },
```

```
    setLocation(state, action) {
      state.location = action.payload;
    },
  },
});

const { setUser, setLocation } = userSlice.actions;
const store = configureStore({ reducer: { user: userSlice.reducer } });

// Signup Screen
const SignupScreen = ({ navigation }) => {
  const dispatch = useDispatch();
  const [username, setUsername] = React.useState('');
  const [email, setEmail] = React.useState('');
  const [password, setPassword] = React.useState('');
  const [phone, setPhone] = React.useState('');

  const validateSignup = async () => {
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    const phoneRegex = /^\+92-3\d{2}-\d{7}$/;

    if (username === '' || !/^[a-zA-Z]+$/.test(username)) {
      Alert.alert('Error', 'Username must contain alphabets only.');
      return;
    }
    if (!emailRegex.test(email)) {
      Alert.alert('Error', 'Enter a valid email address.');
      return;
    }
```

```
  if (password.length < 6) {
    Alert.alert('Error', 'Password must be at least 6 characters long.');
    return;
  }
  if (!phoneRegex.test(phone)) {
    Alert.alert('Error', 'Phone number must follow the format +92-3xx-xxxxxxx.');
    return;
  }

  try {
    await auth().createUserWithEmailAndPassword(email, password);
    dispatch(setUser({ username, email, phone }));
    Alert.alert('Success', 'Signup Successful!');
    navigation.navigate('Login');
  } catch (error) {
    Alert.alert('Error', error.message);
  }
};

return (
  <View style={styles.container}>
    <Text style={styles.header}>Signup</Text>
    <TextInput
      placeholder="Username"
      value={username}
      onChangeText={setUsername}
      style={styles.input}
    />
    <TextInput
```

```
        placeholder="Email"
        value={email}
        onChangeText={setEmail}
        style={styles.input}
      />
      <TextInput
        placeholder="Password"
        value={password}
        onChangeText={setPassword}
        secureTextEntry
        style={styles.input}
      />
      <TextInput
        placeholder="Phone (+92-3xx-xxxxxxx)"
        value={phone}
        onChangeText={setPhone}
        style={styles.input}
      />
      <Button title="Signup" onPress={validateSignup} />
    </View>
  );
};

// Login Screen
const LoginScreen = ({ navigation }) => {
  const dispatch = useDispatch();
  const [email, setEmail] = React.useState('');
  const [password, setPassword] = React.useState('');
```

```
const validateLogin = async () => {
  if (email === '' || password === '') {
    Alert.alert('Error', 'Email and password cannot be empty.');
    return;
  }
  try {
    await auth().signInWithEmailAndPassword(email, password);

    Geolocation.getCurrentPosition(
      async (position) => {
        const location = `${position.coords.latitude}, ${position.coords.longitude}`;
        dispatch(setLocation(location));
        await AsyncStorage.setItem('userLocation', location);
        Alert.alert('Success', 'Login Successful!');
        navigation.navigate('Profile');
      },
      (error) => {
        Alert.alert('Error', `Unable to fetch location: ${error.message}`);
      },
      { enableHighAccuracy: true, timeout: 15000, maximumAge: 10000 }
    );
  } catch (error) {
    Alert.alert('Error', error.message);
  }
};

return (
  <View style={styles.container}>
    <Text style={styles.header}>Login</Text>
```

```
      <TextInput
        placeholder="Email"
        value={email}
        onChangeText={setEmail}
        style={styles.input}
      />
      <TextInput
        placeholder="Password"
        value={password}
        onChangeText={setPassword}
        secureTextEntry
        style={styles.input}
      />
      <Button title="Login" onPress={validateLogin} />
    </View>
  );
};

// Profile Screen
const ProfileScreen = () => {
  const user = useSelector((state) => state.user);

  return (
    <View style={styles.container}>
      <Text style={styles.header}>My Profile</Text>
      <Text>Username: {user.username}</Text>
      <Text>Email: {user.email}</Text>
      <Text>Phone: {user.phone}</Text>
      <Text>Location: {user.location}</Text>
```

```
    </View>
  );
};

const Stack = createStackNavigator();

// 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>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    justifyContent: 'center',
    backgroundColor: '#FFFBEA',
  },
  header: {
```

```
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
    textAlign: 'center',
  },
  input: {
    borderWidth: 1,
    borderColor: '#ccc',
    borderRadius: 5,
    padding: 10,
    marginBottom: 15,
  },
});

export default App;
```