Explanation of React Native Code for Assignment 4

Overview

This document explains the implementation of the given assignment requirements in the provided React Native code.

The code consists of three main parts: replacing useState with Redux, storing user data in Firebase, and fetching/storing user location in AsyncStorage.

Part 1: Replace useState with Redux

Redux Setup

- Redux Installation: Redux is installed using the command:

```
npm install redux react-redux
```

- Global State Management:
  - A reducer is created to handle the state changes for username, password, email, and phone.
  - The store is created using createStore() and passed to the application through the Provider component from react-redux.

Using Redux in Components

- Access State: The useSelector hook is used to retrieve the current state values from the Redux store.
- Update State: The useDispatch hook is used to dispatch actions that update the state in the store.

Code Example:

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

```
<TextInput
  value={username}
  onChangeText={(text) => dispatch({ type: 'SET_USER_DATA', payload: { username: text } })}
/>
```

Benefits of Redux

- Centralized state management.
- Easier to manage complex state changes.
- Improves scalability of the application.

Part 2: Store User Data in Firebase

Firebase Integration

- Firebase Setup:
  - Firebase is configured with project details using the firebaseConfig object.
  - Firebase is initialized using firebase.initializeApp().

Storing User Data

- When a user signs up, their data (username, email, phone) is stored in the Firebase Firestore database.

Code Example:

```
firebase.firestore().collection('users').add({
  username,
  email,
  phone,
});
```

Error Handling

- If there is an error while storing the data, an alert is displayed to notify the user.

Part 3: Fetch User Location and Store in AsyncStorage

Fetching Location

- The react-native-geolocation-service library is used to fetch the user's current location when they log in.

Code Example:

```
Geolocation.getCurrentPosition(
  (position) => {
    const location = {
      latitude: position.coords.latitude,
      longitude: position.coords.longitude,
    };
  }
);
```

Storing Location in AsyncStorage

- The fetched location is stored in AsyncStorage as a JSON string.

Code Example:

```
await AsyncStorage.setItem('userLocation', JSON.stringify(location));
```

Displaying Location

- The stored location is retrieved from AsyncStorage and displayed in the Profile screen.

Code Example:

```
const storedLocation = await AsyncStorage.getItem('userLocation');
setLocation(JSON.parse(storedLocation));
```

Navigation Setup

- The app uses react-navigation to switch between screens (Signup, Login, and Profile).
- The Stack.Navigator is used to define the navigation flow.

Screens

1. Signup Screen: Validates input and stores user data in Firebase.
2. Login Screen: Fetches user location and navigates to the Profile screen.
3. Profile Screen: Displays a welcome message and the user's location.

Styling

- Basic styling is applied to the components using StyleSheet for consistency and alignment.

Example:

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    padding: 20,
  },
  input: {
    height: 40,
    borderColor: 'gray',
    borderWidth: 1,
    marginBottom: 10,
    paddingLeft: 10,
  },
});
```

Conclusion

This implementation fulfills all the requirements of the assignment:

1. Replacing useState with Redux ensures centralized state management.

2. Storing user data in Firebase provides a reliable database solution.

3. Fetching and storing user location in AsyncStorage enhances the user experience by displaying location data.

The application is modular, scalable, and adheres to best practices for React Native development.