Final Year Project, Technical Document, Gym Personal Training & Analytics App.

Author – Daniel Murphy

Student No. – C00247818

Supervisor – Greg Doyle

Contents

Introduction

This manual provides a comprehensive reference guide for developers working on this React Native project. It includes documentation on important screens, components, methods, and other key aspects of the codebase.

The full code for this project can be accessed and downloaded at : https://github.com/Daniel-Murphy33/PT-Mobile-App

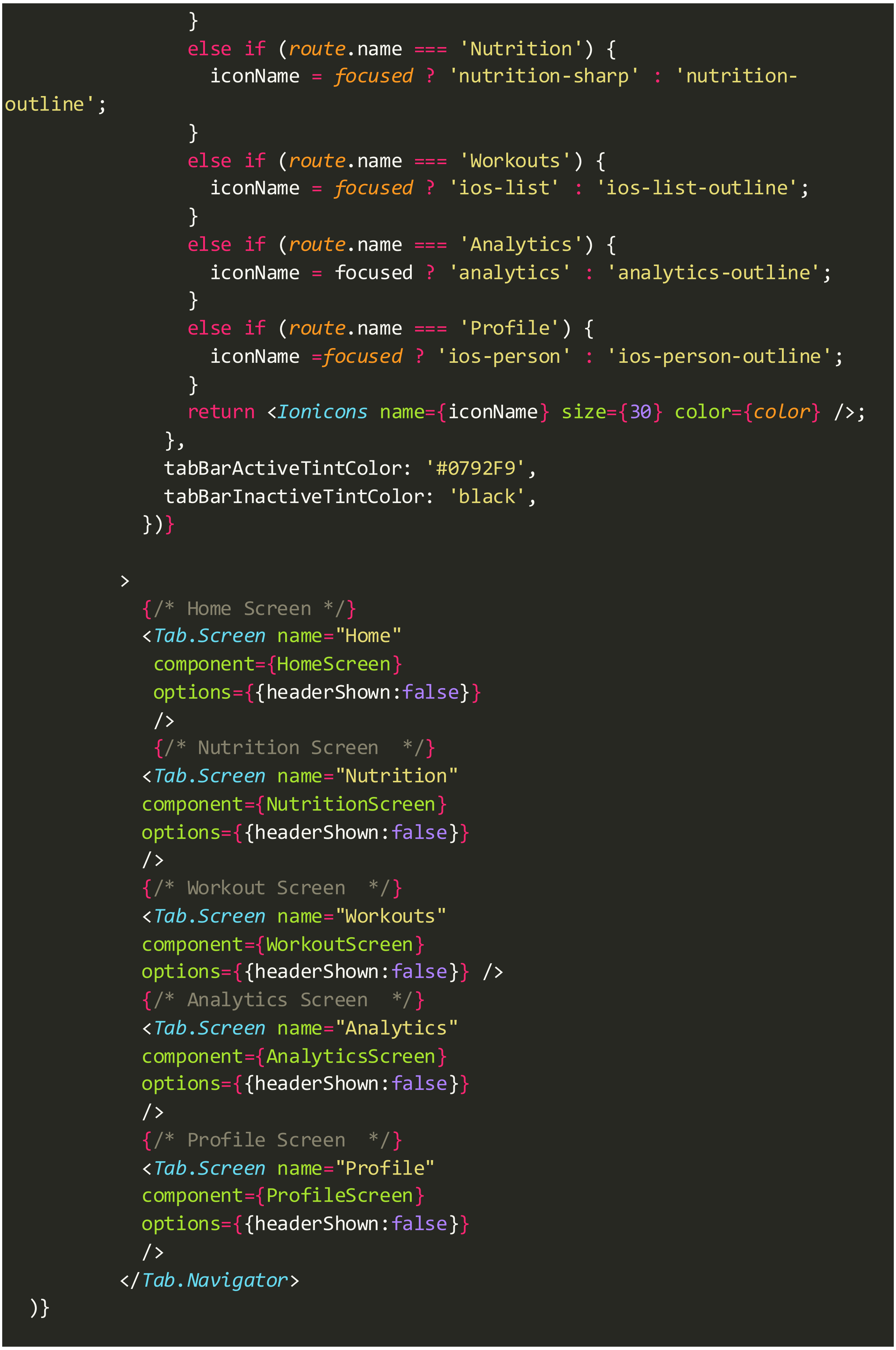
Application Code

Root Directory

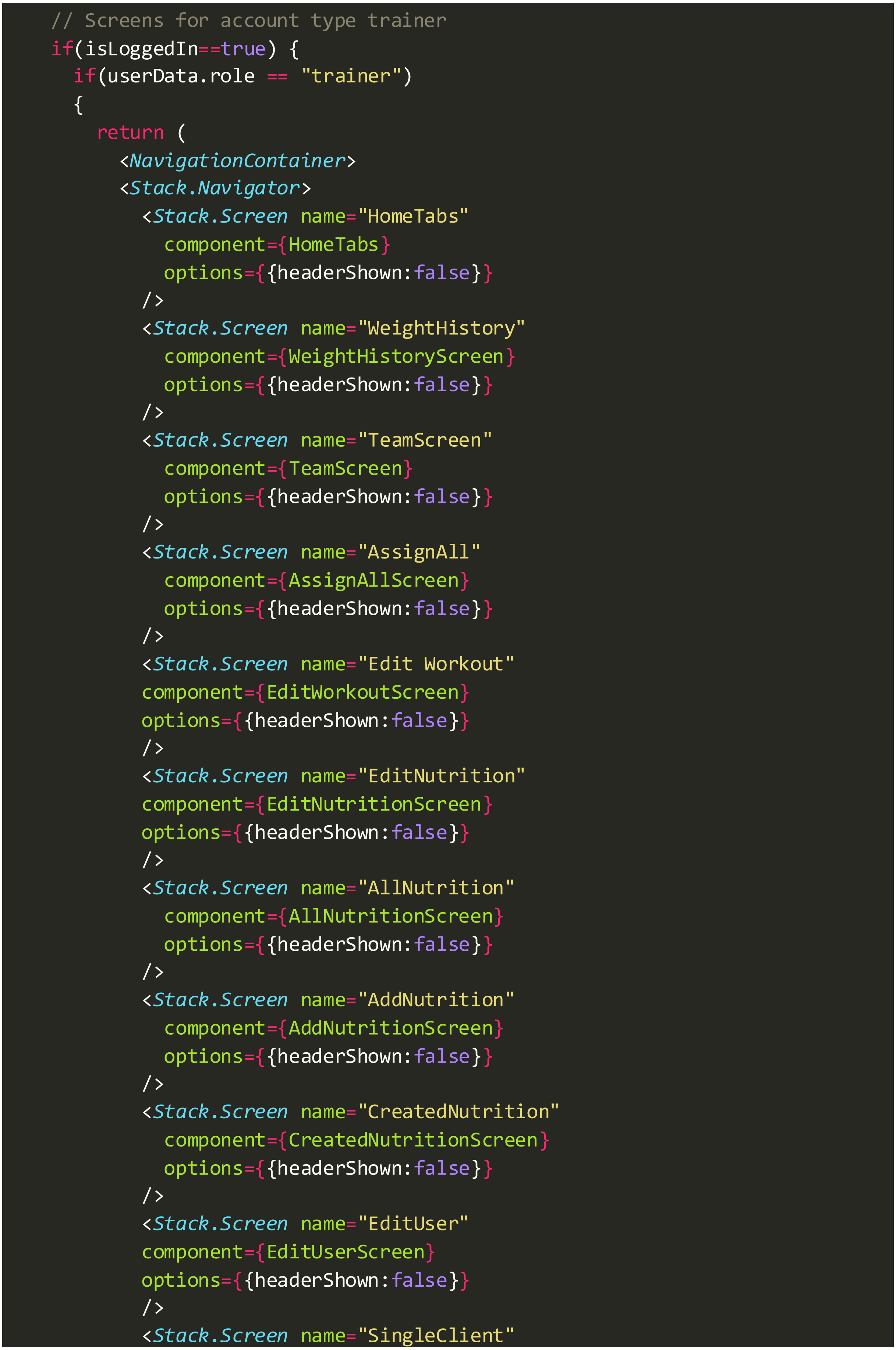
This section will display the code from all of the files in the root directory of the app. This is the entry point for the application containing the foundation for the application.

App.js :

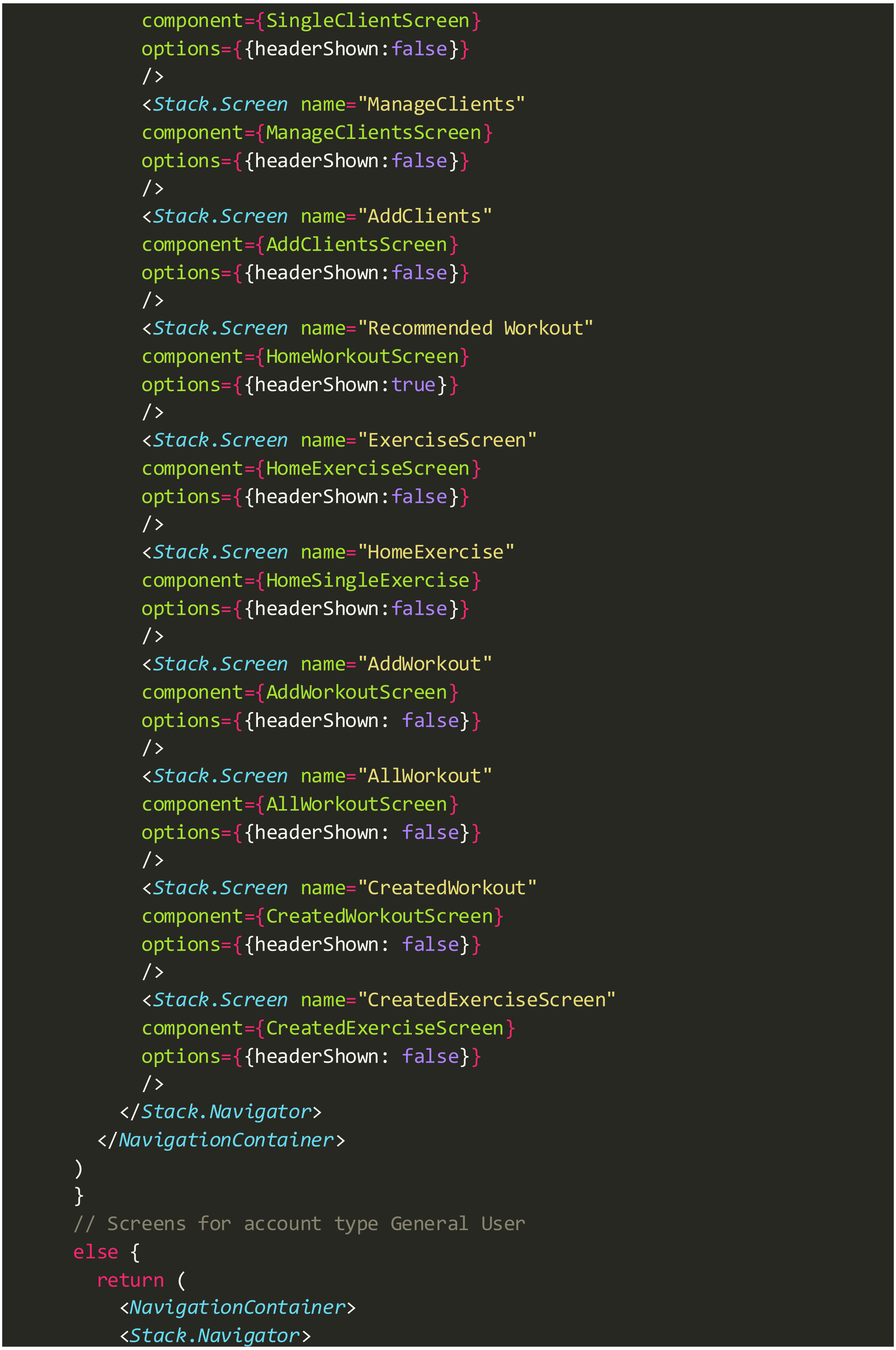
```
export default function App() {

  const Tab = createBottomTabNavigator();
  const Stack = createNativeStackNavigator();
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [userData, setUserData] = useState("");

  useEffect(() => {
    auth.onAuthStateChanged(user => {
      if (user) {
        const uid = getAuth().currentUser.uid;
        setIsLoggedIn(true);

        // fetch user profile so we can check if trainer or client
        const fetchUserProfile = async () => {
          const userRef = doc(db, 'users',uid);
          const userSnapshot = await getDoc(userRef);
          await setUserData(userSnapshot.data());
          console.log(userData);
      }
      fetchUserProfile();
      }
      else {
        setIsLoggedIn(false);
      }
    })
  }, [])


  // Function to create the bottom tabs with icons
  function HomeTabs () {
    return (
    <Tab.Navigator
            screenOptions={({ route }) => ({
              tabBarIcon: ({ focused, color }) => {
                let iconName;

                if (route.name === 'Home') {
                  iconName = focused? 'home' : 'home-outline';
```



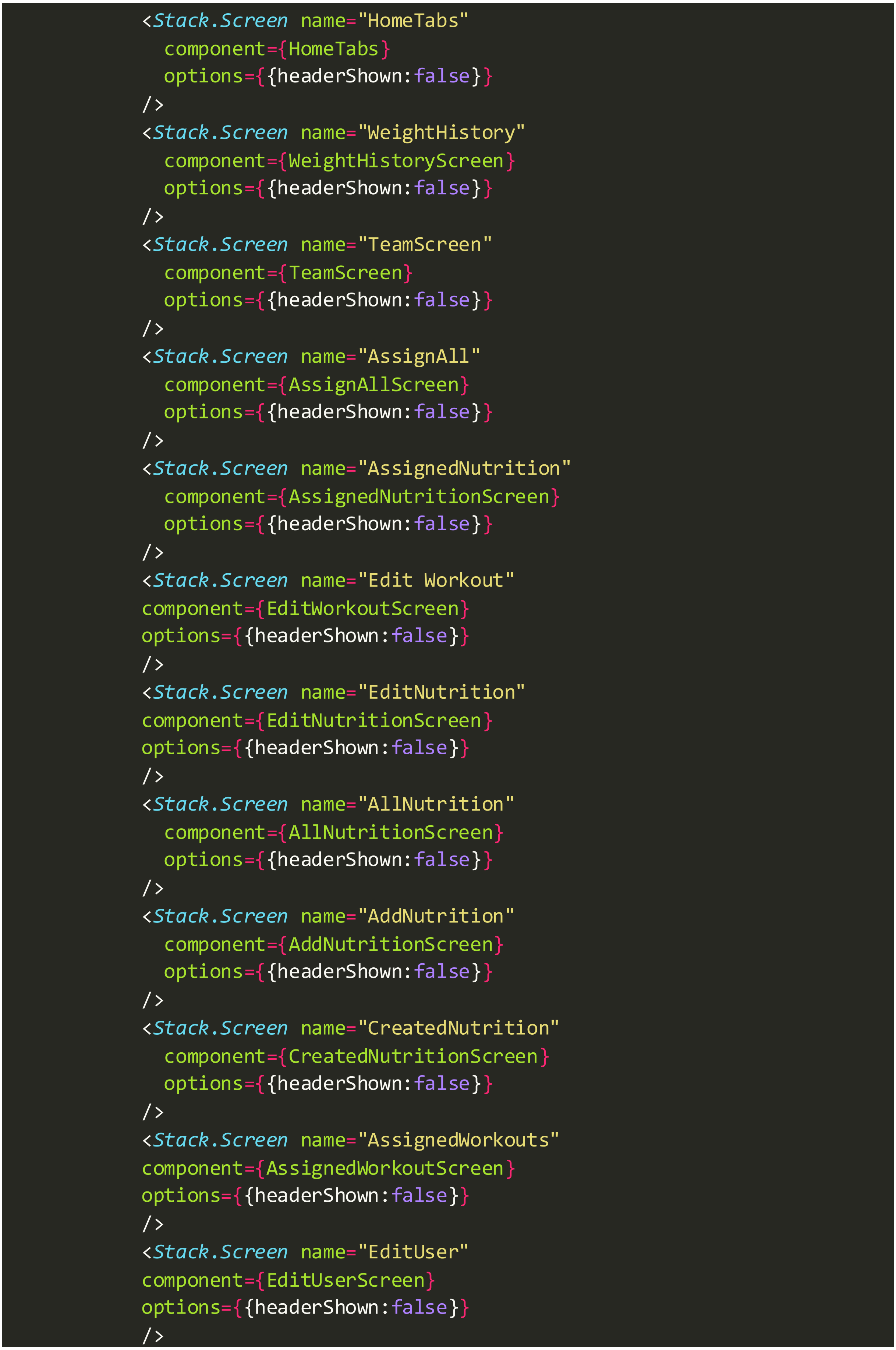
```
                }
                else if (route.name === 'Nutrition') {
                  iconName = focused ? 'nutrition-sharp' : 'nutrition-
outline';
                }
                else if (route.name === 'Workouts') {
                  iconName = focused ? 'ios-list' : 'ios-list-outline';
                }
                else if (route.name === 'Analytics') {
                  iconName = focused ? 'analytics' : 'analytics-outline';
                }
                else if (route.name === 'Profile') {
                  iconName =focused ? 'ios-person' : 'ios-person-outline';
                }
                return <Ionicons name={iconName} size={30} color={color} />;
              },
              tabBarActiveTintColor: '#0792F9',
              tabBarInactiveTintColor: 'black',
            })}

          >
            {/* Home Screen */}
            <Tab.Screen name="Home"
             component={HomeScreen}
             options={{headerShown:false}}
             />
             {/* Nutrition Screen  */}
            <Tab.Screen name="Nutrition"
            component={NutritionScreen}
            options={{headerShown:false}}
            />
            {/* Workout Screen  */}
            <Tab.Screen name="Workouts"
            component={WorkoutScreen}
            options={{headerShown:false}} />
            {/* Analytics Screen  */}
            <Tab.Screen name="Analytics"
            component={AnalyticsScreen}
            options={{headerShown:false}}
            />
            {/* Profile Screen  */}
            <Tab.Screen name="Profile"
            component={ProfileScreen}
            options={{headerShown:false}}
            />
          </Tab.Navigator>
  )}
```



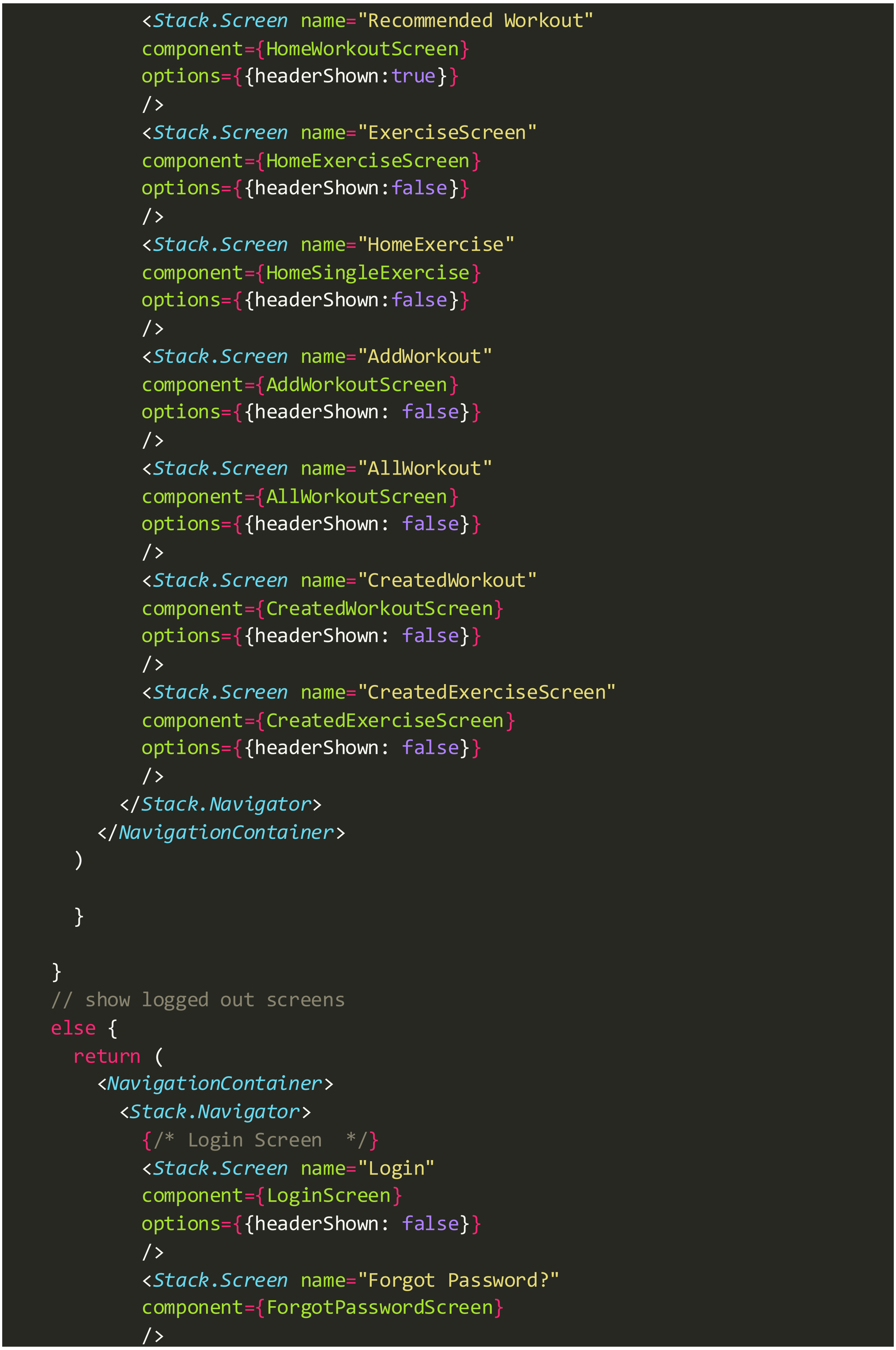
```
  // Screens for account type trainer
  if(isLoggedIn==true) {
   if(userData.role == "trainer")
   {
    return (
     <NavigationContainer>
     <Stack.Navigator>
      <Stack.Screen name="HomeTabs"
       component={HomeTabs}
       options={{headerShown:false}}
      />
      <Stack.Screen name="WeightHistory"
       component={WeightHistoryScreen}
       options={{headerShown:false}}
      />
      <Stack.Screen name="TeamScreen"
       component={TeamScreen}
       options={{headerShown:false}}
      />
      <Stack.Screen name="AssignAll"
       component={AssignAllScreen}
       options={{headerShown:false}}
      />
      <Stack.Screen name="Edit Workout"
      component={EditWorkoutScreen}
      options={{headerShown:false}}
      />
      <Stack.Screen name="EditNutrition"
      component={EditNutritionScreen}
      options={{headerShown:false}}
      />
      <Stack.Screen name="AllNutrition"
       component={AllNutritionScreen}
       options={{headerShown:false}}
      />
      <Stack.Screen name="AddNutrition"
       component={AddNutritionScreen}
       options={{headerShown:false}}
      />
      <Stack.Screen name="CreatedNutrition"
       component={CreatedNutritionScreen}
       options={{headerShown:false}}
      />
      <Stack.Screen name="EditUser"
      component={EditUserScreen}
      options={{headerShown:false}}
      />
      <Stack.Screen name="SingleClient"
```



```
          component={SingleClientScreen}
          options={{headerShown:false}}
          />
          <Stack.Screen name="ManageClients"
          component={ManageClientsScreen}
          options={{headerShown:false}}
          />
          <Stack.Screen name="AddClients"
          component={AddClientsScreen}
          options={{headerShown:false}}
          />
          <Stack.Screen name="Recommended Workout"
          component={HomeWorkoutScreen}
          options={{headerShown:true}}
          />
          <Stack.Screen name="ExerciseScreen"
          component={HomeExerciseScreen}
          options={{headerShown:false}}
          />
          <Stack.Screen name="HomeExercise"
          component={HomeSingleExercise}
          options={{headerShown:false}}
          />
          <Stack.Screen name="AddWorkout"
          component={AddWorkoutScreen}
          options={{headerShown: false}}
          />
          <Stack.Screen name="AllWorkout"
          component={AllWorkoutScreen}
          options={{headerShown: false}}
          />
          <Stack.Screen name="CreatedWorkout"
          component={CreatedWorkoutScreen}
          options={{headerShown: false}}
          />
          <Stack.Screen name="CreatedExerciseScreen"
          component={CreatedExerciseScreen}
          options={{headerShown: false}}
          />
        </Stack.Navigator>
      </NavigationContainer>
    )
    }
    // Screens for account type General User
    else {
      return (
        <NavigationContainer>
        <Stack.Navigator>
```



```
            <Stack.Screen name="HomeTabs"
              component={HomeTabs}
              options={{headerShown:false}}
            />
            <Stack.Screen name="WeightHistory"
              component={WeightHistoryScreen}
              options={{headerShown:false}}
            />
            <Stack.Screen name="TeamScreen"
              component={TeamScreen}
              options={{headerShown:false}}
            />
            <Stack.Screen name="AssignAll"
              component={AssignAllScreen}
              options={{headerShown:false}}
            />
            <Stack.Screen name="AssignedNutrition"
              component={AssignedNutritionScreen}
              options={{headerShown:false}}
            />
            <Stack.Screen name="Edit Workout"
            component={EditWorkoutScreen}
            options={{headerShown:false}}
            />
            <Stack.Screen name="EditNutrition"
            component={EditNutritionScreen}
            options={{headerShown:false}}
            />
            <Stack.Screen name="AllNutrition"
              component={AllNutritionScreen}
              options={{headerShown:false}}
            />
            <Stack.Screen name="AddNutrition"
              component={AddNutritionScreen}
              options={{headerShown:false}}
            />
            <Stack.Screen name="CreatedNutrition"
              component={CreatedNutritionScreen}
              options={{headerShown:false}}
            />
            <Stack.Screen name="AssignedWorkouts"
            component={AssignedWorkoutScreen}
            options={{headerShown:false}}
            />
            <Stack.Screen name="EditUser"
            component={EditUserScreen}
            options={{headerShown:false}}
            />
```



```
            <Stack.Screen name="Recommended Workout"
            component={HomeWorkoutScreen}
            options={{headerShown:true}}
            />
            <Stack.Screen name="ExerciseScreen"
            component={HomeExerciseScreen}
            options={{headerShown:false}}
            />
            <Stack.Screen name="HomeExercise"
            component={HomeSingleExercise}
            options={{headerShown:false}}
            />
            <Stack.Screen name="AddWorkout"
            component={AddWorkoutScreen}
            options={{headerShown: false}}
            />
            <Stack.Screen name="AllWorkout"
            component={AllWorkoutScreen}
            options={{headerShown: false}}
            />
            <Stack.Screen name="CreatedWorkout"
            component={CreatedWorkoutScreen}
            options={{headerShown: false}}
            />
            <Stack.Screen name="CreatedExerciseScreen"
            component={CreatedExerciseScreen}
            options={{headerShown: false}}
            />
          </Stack.Navigator>
        </NavigationContainer>
      )

      }

    }
    // show logged out screens
    else {
      return (
        <NavigationContainer>
          <Stack.Navigator>
            {/* Login Screen  */}
            <Stack.Screen name="Login"
            component={LoginScreen}
            options={{headerShown: false}}
            />
            <Stack.Screen name="Forgot Password?"
            component={ForgotPasswordScreen}
            />
            <Stack.Screen name="Register"
            component={RegisterScreen}
            options={{headerShown: false}}
            />
        </Stack.Navigator>
      </NavigationContainer>
      )

    }
}
```

package.json

```
{
  "name": "frontend",
  "version": "1.0.0",
  "main": "node_modules/expo/AppEntry.js",
  "scripts": {
    "start": "expo start",
    "android": "expo start --android",
    "ios": "expo start --ios",
    "web": "expo start --web"
  },
  "dependencies": {
    "@emotion/react": "^11.10.8",
    "@emotion/styled": "^11.10.8",
    "@expo/webpack-config": "^18.0.4",
    "@react-native-firebase/app": "^17.4.3",
    "@react-navigation/bottom-tabs": "^6.5.7",
    "@react-navigation/native": "^6.1.6",
    "@react-navigation/native-stack": "^6.9.12",
    "@react-navigation/stack": "^6.3.16",
    "@types/react-native-elements": "^0.18.0",
    "expo": "~48.0.15",
    "expo-image-picker": "~14.0.2",
    "expo-status-bar": "~1.4.4",
    "expo-updates": "~0.15.6",
    "firebase": "^9.21.0",
    "native-base": "^3.4.28",
    "react": "18.1.0",
    "react-dom": "18.1.0",
    "react-native": "0.70.8",
    "react-native-chart-kit": "^6.12.0",
    "react-native-dropdown-picker": "^5.4.6",
    "react-native-elements": "^3.4.3",
    "react-native-gesture-handler": "~2.8.0",
    "react-native-pager-view": "6.0.1",
    "react-native-reanimated": "~2.12.0",
    "react-native-safe-area-context": "4.4.1",
    "react-native-screens": "~3.18.0",
    "react-native-tab-view": "^3.5.1",
    "react-native-tooltip": "^5.2.0",
    "react-native-vector-icons": "^9.2.0",
    "react-native-web": "~0.18.9",
    "react-native-youtube-iframe": "^2.2.2",
    "styled-components": "^5.3.10",
    "upgrade": "^1.1.0"
  },
  "devDependencies": {
    "@babel/core": "^7.21.5",
    "@types/react-native": "~0.71.6"
  },
  "private": true
}
```

Firebase.js (firebase configuration file)

```
// Import the functions you need from the SDKs you need
import firebase from 'firebase/compat/app';
import 'firebase/compat/auth';
import { getFirestore, collection, addDoc, getDocs, onSnapshot,
  setDoc, doc, getDoc, } from "firebase/firestore";

// app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyCYo2-B4yQVU3I0YaT9NLsygoUKhg8GuzY",
  authDomain: "backend-fyp.firebaseapp.com",
  projectId: "backend-fyp",
  storageBucket: "backend-fyp.appspot.com",
  messagingSenderId: "926928198367",
  appId: "1:926928198367:web:af64cc798c832317a8e76a",
  measurementId: "G-4T1BF70577"
};

// Initialize Firebase
let app;
//if app not initialised
if (firebase.apps.length === 0) {
    app = firebase.initializeApp(firebaseConfig);
}
//app greater then 0 === app initialised
else {
    app = firebase.app()
}

//init services
const db = getFirestore(app);
//get authentication
const auth = firebase.auth()

export { auth, app, db, getFirestore, collection, addDoc, getDocs, getDoc,
onSnapshot,
setDoc, doc, };
```

Screens & Components

This folder contains the code for each screen in the application. There are two subfolders, loggedIn and loggedOut. loggedIn then has subfolders splitting the screens into the different tabs they belong keeping the code clean and easy to manage. The components folder is separate but the screens use the components so they will be grouped together when necessary.

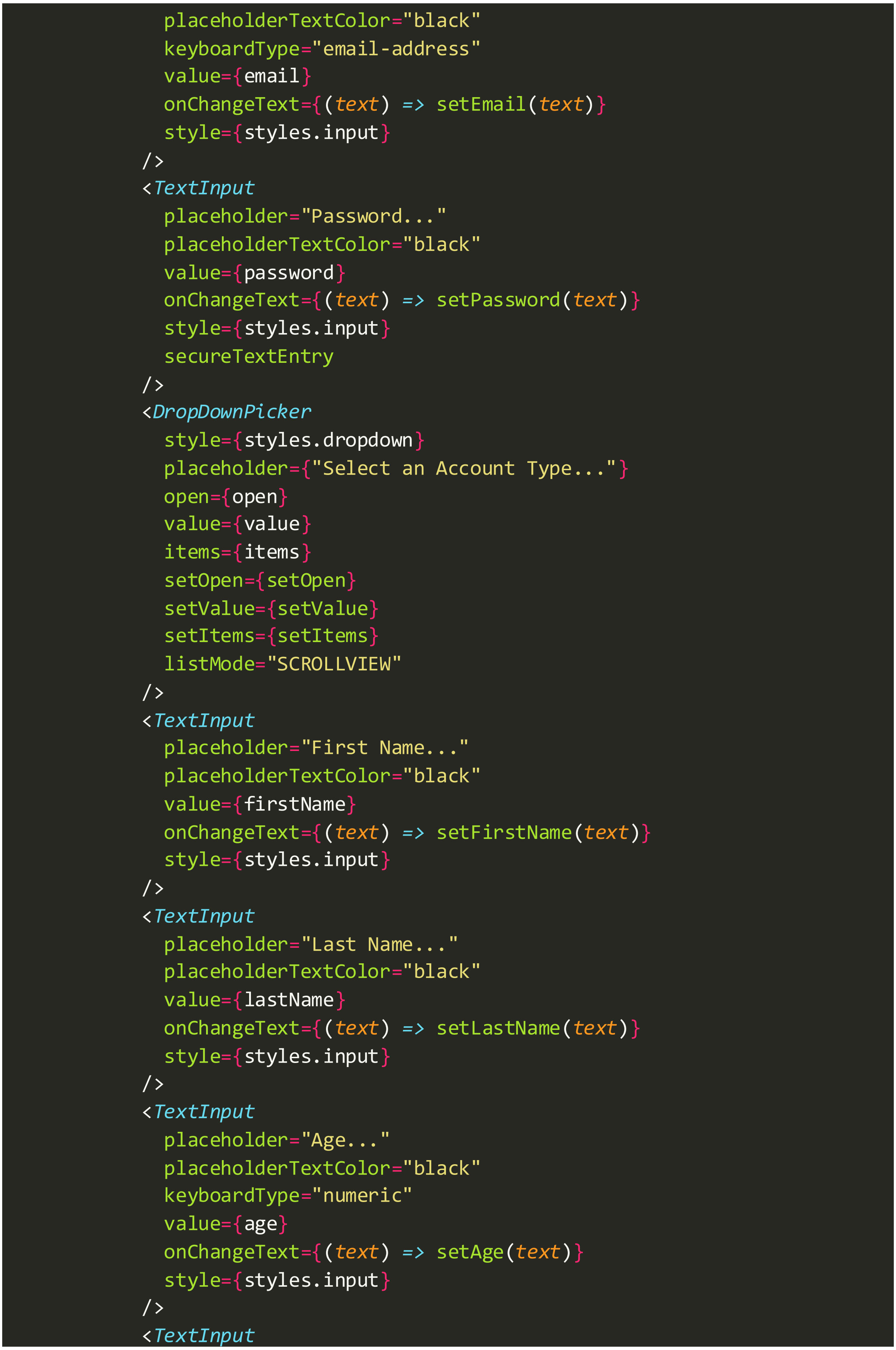
LoginScreen.js

```
const LoginScreen = () => {

  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const navigation = useNavigation()

  //navigating to screens
  const RegisterUserScreen = () => {
    navigation.navigate('Register');
  }

  //navigating through screens
  const ForgotPasswordScreen = () => {
    navigation.navigate("Forgot Password?")
  }

  //function for signing in
  const handleSignIn = () => {
    auth.
    signInWithEmailAndPassword (email, password)
    .then(UserCredentials => {
      const user = UserCredentials.user;
      console.log("Logged in with: ", user.email);
    })
    .catch(error => alert(error.message))
  }
  return (
    //allows for dismissing keyboard
    <TouchableWithoutFeedback onPress={() => {
      Keyboard.dismiss();
    }}>
      <View style={styles.container}>
      <Image source={require('../../assets/logo-no-bg.png')}
style={styles.logo} />
        <View style={styles.inputContainer}>
          <TextInput placeholder='Email'
          placeholderTextColor="black"
          keyboardType='email-address'
          value={email}
          autoCapitalize='none'
          required={true}
          onChangeText={text => setEmail(text)} style={styles.input} />
          <TextInput placeholder='Password'
          placeholderTextColor="black"
          value={password}
          required={true}
          onChangeText={text => setPassword(text)} style={styles.input}
secureTextEntry />
        </View>

        <View style={styles.buttonContainer}>
          <TouchableOpacity onPress={handleSignIn} style={styles.button} >
            <Text style={styles.buttonText}>Login</Text>
          </TouchableOpacity>
          <TouchableOpacity onPress={RegisterUserScreen}
style={[styles.button, styles.buttonOutline]} >
            <Text style={styles.buttonOutlineText}>Register</Text>
          </TouchableOpacity>
          <TouchableOpacity onPress={ForgotPasswordScreen}
style={[styles.button, styles.buttonOutline]} >
            <Text style={styles.buttonOutlineText}>Forgot Password ?</Text>
          </TouchableOpacity>
        </View>
      </View>
    </TouchableWithoutFeedback>
  )
}
```

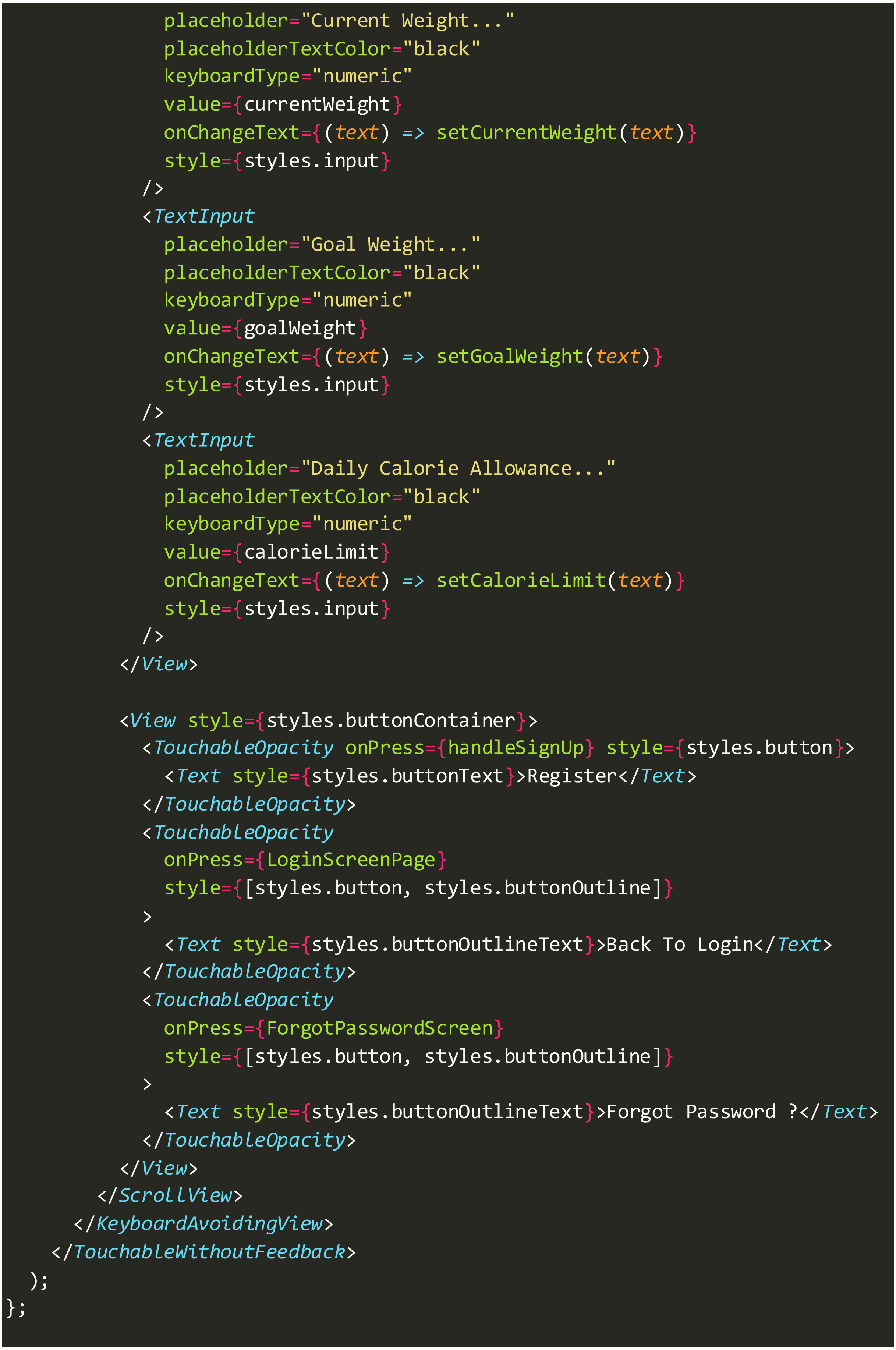
RegisterScreen.js

```
const RegisterScreen = () => {
  // for dropdown
  const [open, setOpen] = useState(false);
  const [value, setValue] = useState(null);
  const [items, setItems] = useState([
    { label: "Personal Trainer", value: "trainer" },
    { label: "General User", value: "client" },
  ]);

  // user info
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [firstName, setFirstName] = useState("");
  const [lastName, setLastName] = useState("");
  const [age, setAge] = useState("");
  const [currentWeight, setCurrentWeight] = useState("");
  const [goalWeight, setGoalWeight] = useState("");
  const [calorieLimit, setCalorieLimit] = useState("");
  const navigation = useNavigation();

  //navigating to screens
  const LoginScreenPage = () => {
    navigation.navigate("Login");
  };

  //navigating through screens
  const ForgotPasswordScreen = () => {
    navigation.navigate("Forgot Password?");
  };

  //function for signing up
const handleSignUp = async () => {

  if (!email || !password || !firstName || !lastName || !age || !currentWeight
|| !goalWeight || !value || !calorieLimit) {
    alert("Please fill in all fields");
    return;
  }

  auth
    .createUserWithEmailAndPassword(email, password)
    .then(async (UserCredentials) => {
      const user = UserCredentials.user;
      console.log("Registered with: ", user.email);
      try {
        const uidRef = doc(db, "users", user.uid);
        updateProfile(auth.currentUser, {
```

```
     displayName: firstName
    }).then(() => {
     console.log("Display name updated");
    }).catch((error) => {
     console.log(error)
    });
    await setDoc(uidRef, {
     role: value,
     firstName: firstName,
     lastName: lastName,
     age: age,
     currentWeight: currentWeight,
     goalWeight: goalWeight,
     calorieLimit: calorieLimit,
    });

    // Create a new weights collection for the user
    const initialWeight = {
     date: new Date(),
     weight: currentWeight,
    };
    await setDoc(doc(db, `users/${user.uid}/weights`,
initialWeight.date.toISOString()), initialWeight);

   } catch (e) {
    console.error("Error adding document: ", e);
   }
  })
  .catch((error) => alert(error.message));
};

 return (
  //allows for dismissing keyboard
  <TouchableWithoutFeedback onPress={Keyboard.dismiss}>
   <KeyboardAvoidingView
    behavior={Platform.OS === "ios" ? "padding" : "height"}
    style={styles.container}
   >
    <ScrollView contentContainerStyle={styles.container}
nestedScrollEnabled={true}>
     <Image
      source={require("../../assets/logo-no-bg.png")}
      style={styles.logo}
     />
     <View style={styles.inputContainer}>
      <TextInput
       placeholder="Email..."
       autoCapitalize='none'
```



```
          placeholderTextColor="black"
          keyboardType="email-address"
          value={email}
          onChangeText={(text) => setEmail(text)}
          style={styles.input}
        />
        <TextInput
          placeholder="Password..."
          placeholderTextColor="black"
          value={password}
          onChangeText={(text) => setPassword(text)}
          style={styles.input}
          secureTextEntry
        />
        <DropDownPicker
          style={styles.dropdown}
          placeholder={"Select an Account Type..."}
          open={open}
          value={value}
          items={items}
          setOpen={setOpen}
          setValue={setValue}
          setItems={setItems}
          listMode="SCROLLVIEW"
        />
        <TextInput
          placeholder="First Name..."
          placeholderTextColor="black"
          value={firstName}
          onChangeText={(text) => setFirstName(text)}
          style={styles.input}
        />
        <TextInput
          placeholder="Last Name..."
          placeholderTextColor="black"
          value={lastName}
          onChangeText={(text) => setLastName(text)}
          style={styles.input}
        />
        <TextInput
          placeholder="Age..."
          placeholderTextColor="black"
          keyboardType="numeric"
          value={age}
          onChangeText={(text) => setAge(text)}
          style={styles.input}
        />
        <TextInput
```



```
              placeholder="Current Weight..."
              placeholderTextColor="black"
              keyboardType="numeric"
              value={currentWeight}
              onChangeText={(text) => setCurrentWeight(text)}
              style={styles.input}
            />
            <TextInput
              placeholder="Goal Weight..."
              placeholderTextColor="black"
              keyboardType="numeric"
              value={goalWeight}
              onChangeText={(text) => setGoalWeight(text)}
              style={styles.input}
            />
            <TextInput
              placeholder="Daily Calorie Allowance..."
              placeholderTextColor="black"
              keyboardType="numeric"
              value={calorieLimit}
              onChangeText={(text) => setCalorieLimit(text)}
              style={styles.input}
            />
          </View>

          <View style={styles.buttonContainer}>
            <TouchableOpacity onPress={handleSignUp} style={styles.button}>
              <Text style={styles.buttonText}>Register</Text>
            </TouchableOpacity>
            <TouchableOpacity
              onPress={LoginScreenPage}
              style={[styles.button, styles.buttonOutline]}
            >
              <Text style={styles.buttonOutlineText}>Back To Login</Text>
            </TouchableOpacity>
            <TouchableOpacity
              onPress={ForgotPasswordScreen}
              style={[styles.button, styles.buttonOutline]}
            >
              <Text style={styles.buttonOutlineText}>Forgot Password ?</Text>
            </TouchableOpacity>
          </View>
        </ScrollView>
      </KeyboardAvoidingView>
    </TouchableWithoutFeedback>
  );
};
```

```
export default RegisterScreen;
```

ForgotPasswordScreen.js

```
const ForgotPasswordScreen = () => {

  const [email, setEmail] = useState('');
  const navigation = useNavigation();

  // for resetting password when user forgets
  const ForgotPassword = () => {

    console.log("reset email sent to " + email);
    auth.sendPasswordResetEmail(email)
      .then(() => {
        alert("Reset password sent to : " + email);
      })
      .catch(function (error) {
        alert(error);
      });
  };

  //navigates user to login page
  const LoginScreenPage = () => {
    navigation.navigate('Login')
  }

  return (
    <TouchableWithoutFeedback onPress={() => {
      Keyboard.dismiss();
    }}>
      <View style={styles.container}>
        <Image source={require('../../assets/logo-no-bg.png')}
style={styles.logo} />
        <Text style={styles.heading}>Forgot your password?</Text>
        <Text style={styles.heading}>Enter your email address and we will send
you a link to reset your password!</Text>
        <View style={styles.inputContainer}>
          <TextInput placeholder='Email'
            placeholderTextColor="black"
            required={true}
            autoCapitalize='none'
            keyboardType='email-address'
            value={email}
            onChangeText={text => setEmail(text)} style={styles.input} />
        </View>

        <View style={styles.buttonContainer}>
          <TouchableOpacity onPress={ForgotPassword} style={styles.button} >
```

```
            <Text style={styles.buttonText}>Send Reset Link To Email</Text>
          </TouchableOpacity>
        </View>
      </View>
    </TouchableWithoutFeedback>
  )
}

export default ForgotPasswordScreen
```

```
const HomeScreen = () => {

  const user = getAuth().currentUser;

  return (
    <ScrollView style={styles.container}>
      <View style={styles.headerBlockWrapper}>
        <View style={styles.headerBlock}>
          <View style={{ width: "50%" }}>
            <Text style={styles.headerText}>Welcome {user.displayName}</Text>
          </View>
          <View style={{ width: "50%", alignItems: "flex-end" }}>
            <Image source={require('../../../assets/logo-no-bg.jpg')} style={{
height: 60, width: 90 }} />
          </View>
        </View>
        <Image style={styles.headerImage} source={{
          uri: 'https://img.freepik.com/premium-photo/muscular-tattooed-
bearded-male-exercising_136403-9395.jpg?w=1380',
          // method: 'POST',
        }}
        />
        <FitnessCards style={{ marginTop: 20 }} />
      </View>
    </ScrollView>
  )
}
export default HomeScreen;
```

FitnessCards.js (Fitness data from a file with sample json data)

```
import Fitness from '../data/Fitness'
const FitnessCards = () => {

    const FitnessData = Fitness;
    const navigation = useNavigation();

  return (
    <View>
```

```
        {FitnessData.map((item, key) => (
            <Pressable style={styles.card} key={(key)} onPress={() =>
navigation.navigate("Recommended Workout", {
                image: item.image,
                excersises: item.excersises,
                id: item.id,
            })}>
                <Image style={styles.image}source={{uri:item.image}}/>
                <Text style={styles.cardText}>{item.name}</Text>
                <MaterialCommunityIcons style ={styles.icon} name="lightning-
bolt" size={24} color="black" />
            </Pressable>
        ))}
    </View>
  )
}

export default FitnessCards
```

HomeWorkoutScreen.js

```
const HomeWorkoutScreen = () => {

    const route = useRoute();
    const navigation = useNavigation();

    return (
        <>
        <ScrollView style={styles.container} >
            <Image style={styles.headerImage} source={{ uri:
route.params.image }} />
            <Ionicons style={styles.icon} name="fitness" size={32}
color="white" />

            {route.params.excersises.map((item, index) => (
                <TouchableOpacity style={styles.gif} key={index} onPress={()
=> navigation.navigate(item.screen, {
                    image: item.image,
                    name: item.name,
                    sets: item.sets,
                    reps: item.reps,
                    screen: item.screen,
                    id: item.id,
                })}>
                    <Image style={styles.exeImage} source={{uri:item.image}}
/>
                    <View style={{marginLeft: 6}}>
                        <Text style={styles.title}>{item.name}</Text>
                        <Text style={styles.sets}>x{item.sets} Sets</Text>
                        <Text style={styles.sets}>x{item.reps} Reps</Text>
                    </View>
                </TouchableOpacity>
            ))}
        </ScrollView>
        <TouchableOpacity onPress={() => navigation.navigate("ExerciseScreen",
{
            excersises:route.params.excersises,
        })} style={styles.startBtn}>
            <Text style={styles.btnText}>START</Text>
        </TouchableOpacity>
        </>
    )
}

export default HomeWorkoutScreen
```

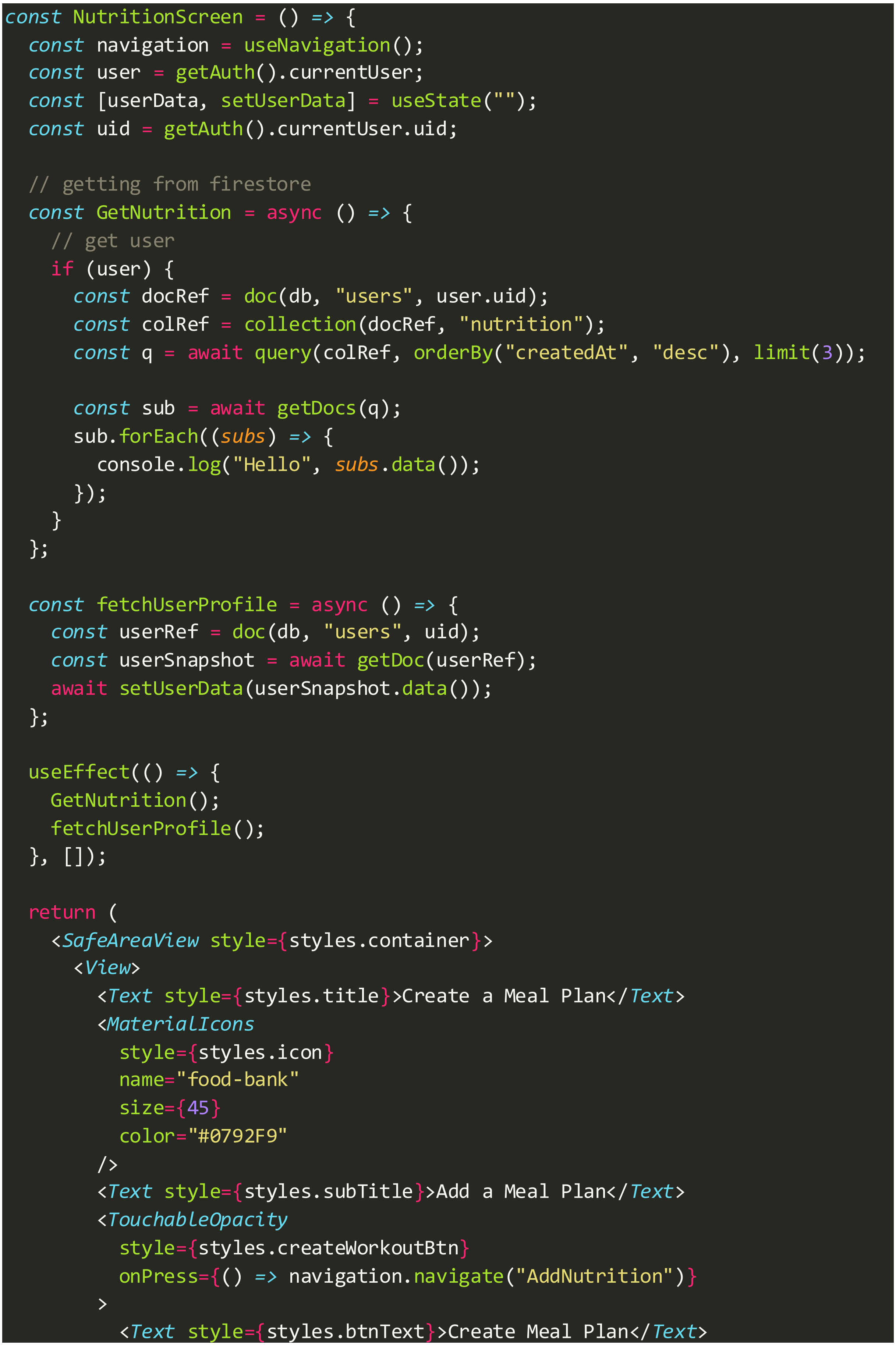
HomeSingleExerciseScreen

```
const HomeSignleExerciseScreen = () => {

  const navigation = useNavigation();
  const route = useRoute();
  const exercise = route.params
  console.log(exercise);

  return (
    <SafeAreaView>
      <Image source={{uri:exercise.image}} style={styles.image}/>
      <Text style={styles.title}>{exercise.name}</Text>
      <Text style={styles.sets}>x{exercise.sets} Sets</Text>
      <Text style={styles.sets}>x{exercise.reps} Reps</Text>
      <TouchableOpacity style={styles.doneBtn} onPress={() =>
navigation.goBack()}>
        <Text style={styles.btnText}>BACK</Text>
      </TouchableOpacity>
    </SafeAreaView>
  )
}

export default HomeSignleExerciseScreen
```

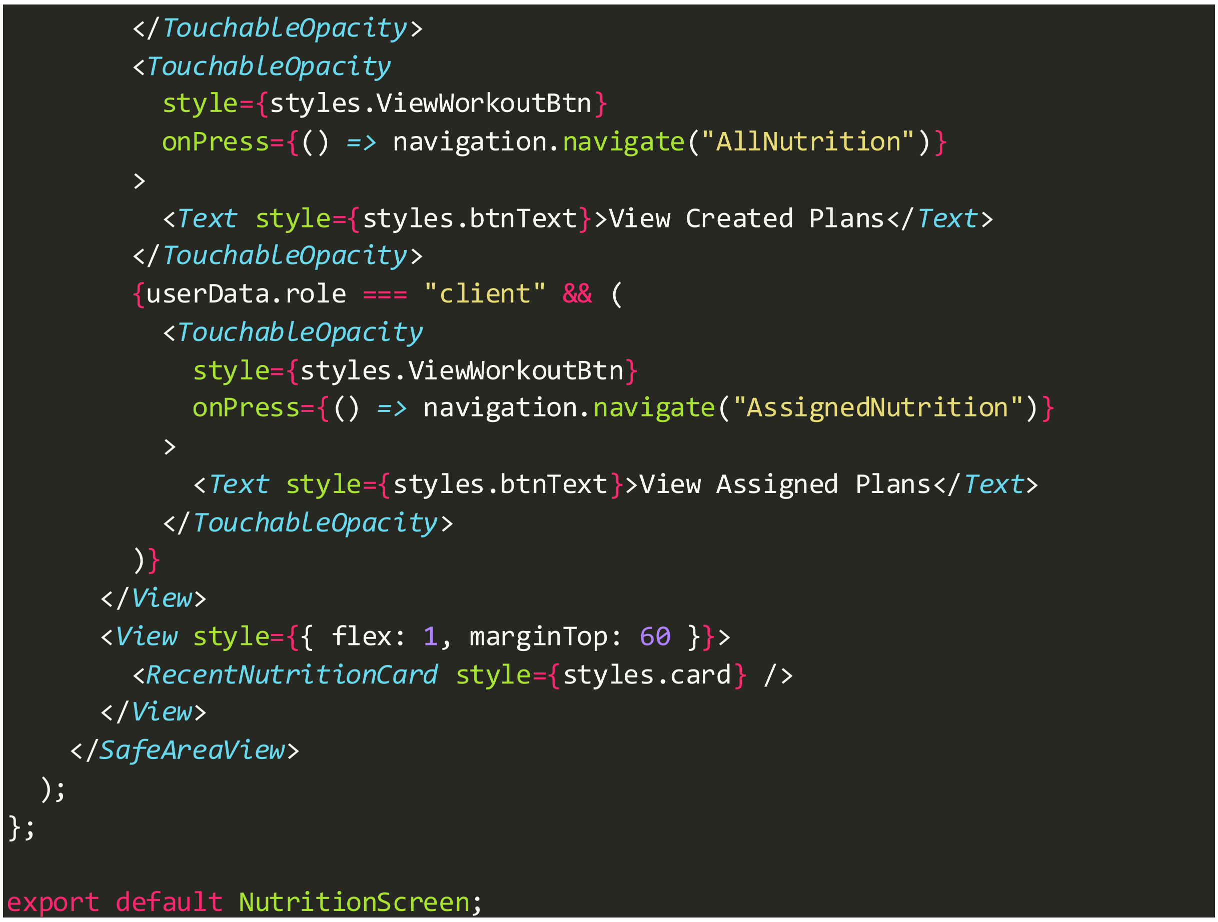
HomeExerciseScreen.js

```
const HomeExerciseScreen = () => {

  const navigation = useNavigation();
  const route = useRoute();
  const [index, setIndex] = useState(0);
  const exercise = route.params.excersises;
  const current = exercise[index];
  const isFirst = index === 0;
  const isLast = index === exercise.length - 1;

  const handleNext = () => {
      if (!isLast) {
          setIndex(index + 1);
      }
  };

  const handlePrev = () => {
      if (!isFirst) {
          setIndex(index - 1);
      }
  };

  return (
      <SafeAreaView>
          <Image source={{uri:current.image}} style={styles.image}/>
          <Text style={styles.title}>{current.name}</Text>
          <Text style={styles.sets}>x{current.sets} Sets</Text>
          <Text style={styles.sets}>x{current.reps} Reps</Text>
          <TouchableOpacity style={styles.doneBtn} onPress={() =>
navigation.navigate("HomeTabs")}>
              <Text style={styles.btnText}>FINISH WORKOUT</Text>
          </TouchableOpacity>
          <TouchableOpacity style={[styles.prevBtn, isFirst &&
styles.disabledBtn]} disabled={isFirst} onPress={handlePrev}>
              <Text style={styles.btnText}>PREVIOUS</Text>
          </TouchableOpacity>
          <TouchableOpacity style={[styles.nextBtn, isLast &&
styles.disabledBtn]} disabled={isLast} onPress={handleNext}>
              <Text style={styles.btnText}>NEXT</Text>
          </TouchableOpacity>
      </SafeAreaView>
  );
};

export default HomeExerciseScreen
```

NutritionScreen.js

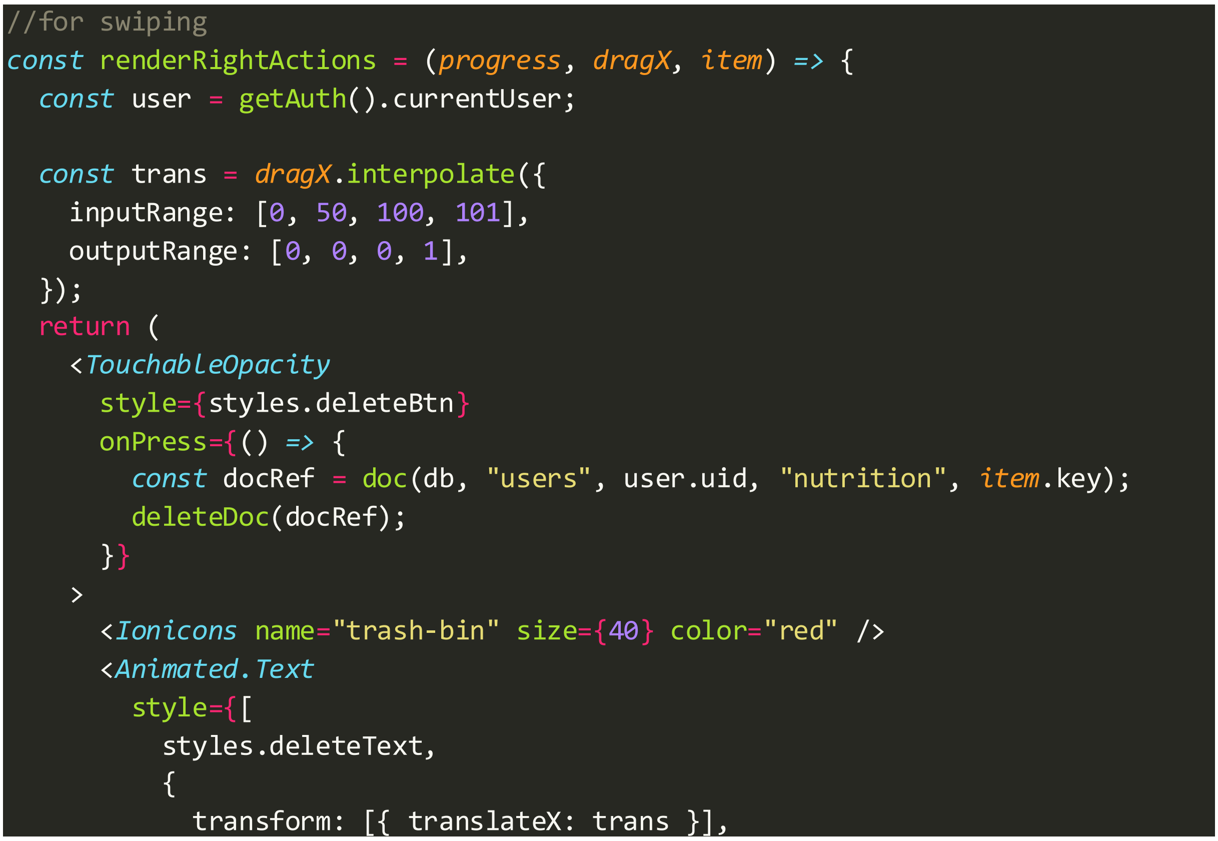


```
const NutritionScreen = () => {
  const navigation = useNavigation();
  const user = getAuth().currentUser;
  const [userData, setUserData] = useState("");
  const uid = getAuth().currentUser.uid;

  // getting from firestore
  const GetNutrition = async () => {
    // get user
    if (user) {
      const docRef = doc(db, "users", user.uid);
      const colRef = collection(docRef, "nutrition");
      const q = await query(colRef, orderBy("createdAt", "desc"), limit(3));

      const sub = await getDocs(q);
      sub.forEach((subs) => {
        console.log("Hello", subs.data());
      });
    }
  };

  const fetchUserProfile = async () => {
    const userRef = doc(db, "users", uid);
    const userSnapshot = await getDoc(userRef);
    await setUserData(userSnapshot.data());
  };

  useEffect(() => {
    GetNutrition();
    fetchUserProfile();
  }, []);

  return (
    <SafeAreaView style={styles.container}>
      <View>
        <Text style={styles.title}>Create a Meal Plan</Text>
        <MaterialIcons
          style={styles.icon}
          name="food-bank"
          size={45}
          color="#0792F9"
        />
        <Text style={styles.subTitle}>Add a Meal Plan</Text>
        <TouchableOpacity
          style={styles.createWorkoutBtn}
          onPress={() => navigation.navigate("AddNutrition")}
        >
          <Text style={styles.btnText}>Create Meal Plan</Text>
```

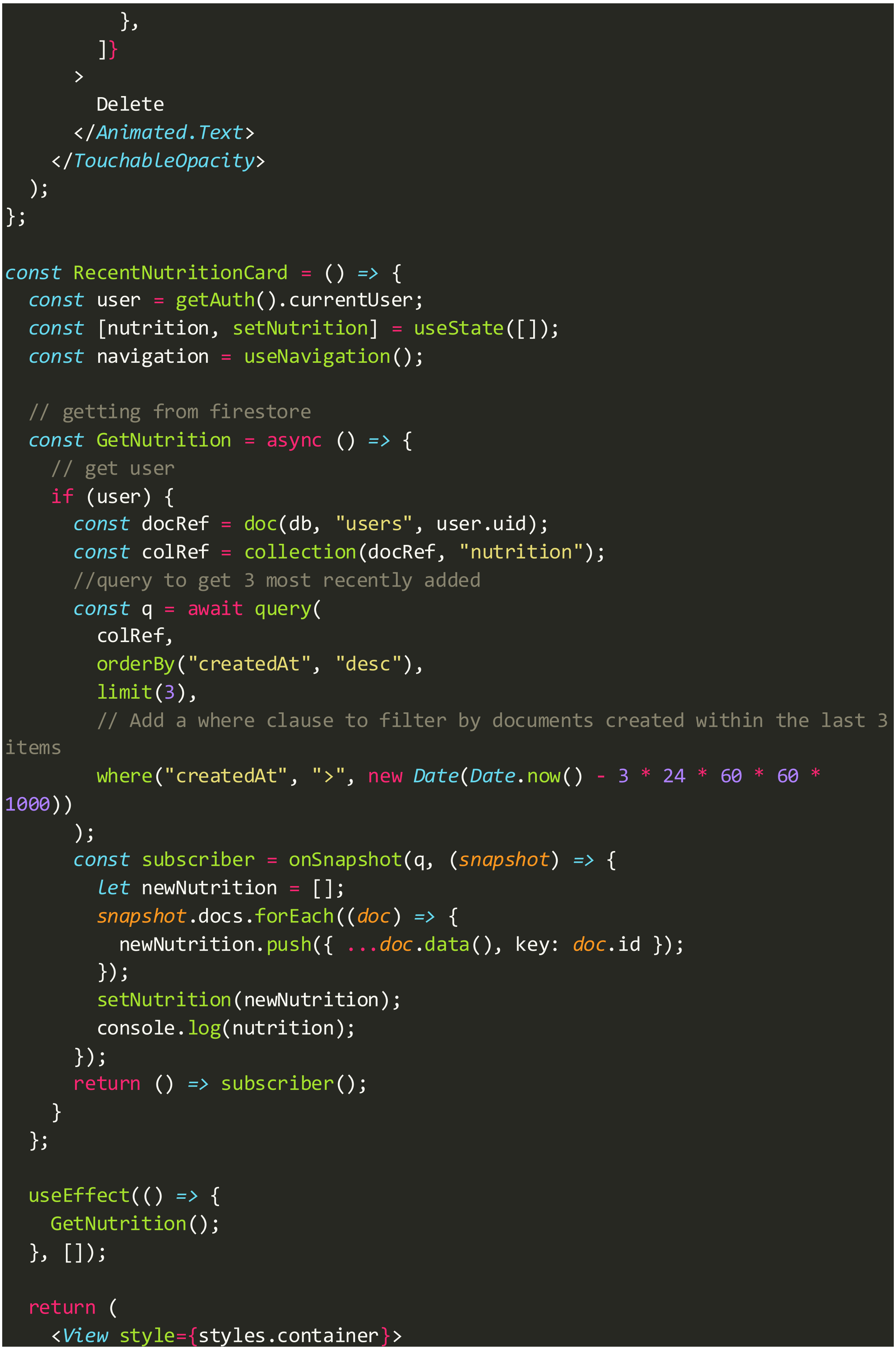


```
        </TouchableOpacity>
        <TouchableOpacity
          style={styles.ViewWorkoutBtn}
          onPress={() => navigation.navigate("AllNutrition")}
        >
          <Text style={styles.btnText}>View Created Plans</Text>
        </TouchableOpacity>
        {userData.role === "client" && (
          <TouchableOpacity
            style={styles.ViewWorkoutBtn}
            onPress={() => navigation.navigate("AssignedNutrition")}
          >
            <Text style={styles.btnText}>View Assigned Plans</Text>
          </TouchableOpacity>
        )}
      </View>
      <View style={{ flex: 1, marginTop: 60 }}>
        <RecentNutritionCard style={styles.card} />
      </View>
    </SafeAreaView>
  );
};

export default NutritionScreen;
```

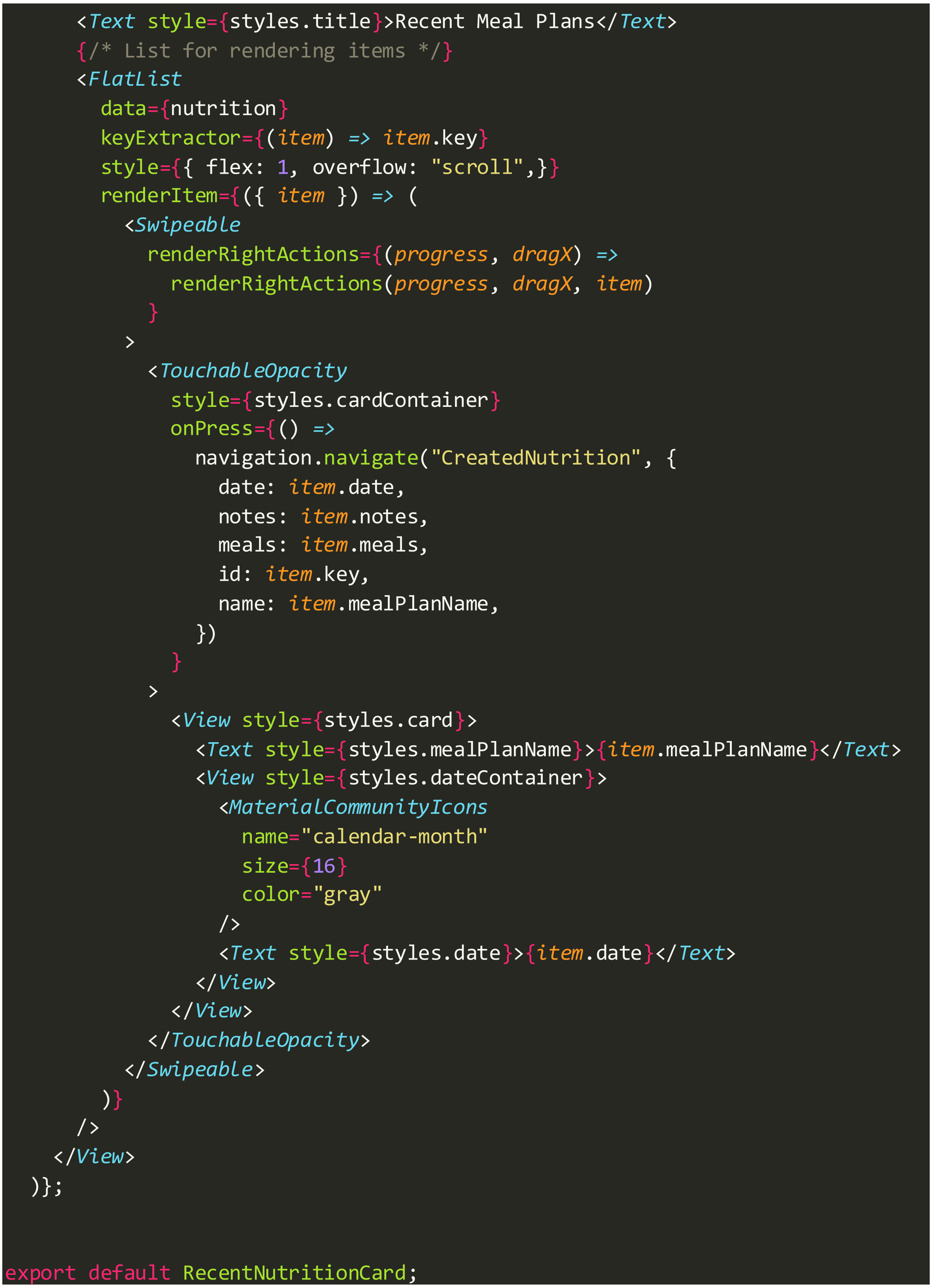
RecentNutritionCards.js



```
//for swiping
const renderRightActions = (progress, dragX, item) => {
  const user = getAuth().currentUser;

  const trans = dragX.interpolate({
    inputRange: [0, 50, 100, 101],
    outputRange: [0, 0, 0, 1],
  });
  return (
    <TouchableOpacity
      style={styles.deleteBtn}
      onPress={() => {
        const docRef = doc(db, "users", user.uid, "nutrition", item.key);
        deleteDoc(docRef);
      }}
    >
      <Ionicons name="trash-bin" size={40} color="red" />
      <Animated.Text
        style={[
          styles.deleteText,
          {
            transform: [{ translateX: trans }],
```



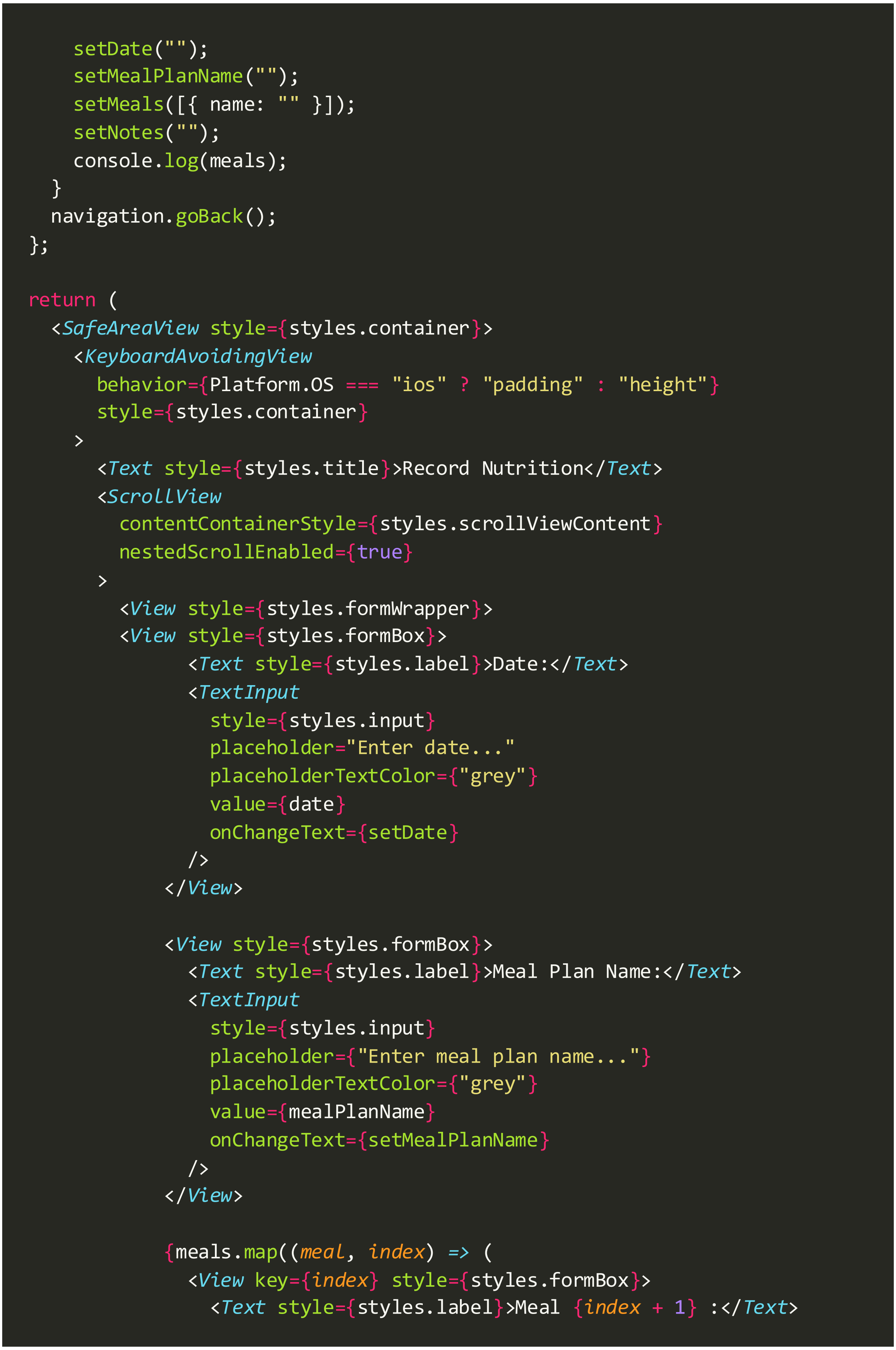
```
          },
        ]}
      >
        Delete
      </Animated.Text>
    </TouchableOpacity>
  );
};

const RecentNutritionCard = () => {
  const user = getAuth().currentUser;
  const [nutrition, setNutrition] = useState([]);
  const navigation = useNavigation();

  // getting from firestore
  const GetNutrition = async () => {
    // get user
    if (user) {
      const docRef = doc(db, "users", user.uid);
      const colRef = collection(docRef, "nutrition");
      //query to get 3 most recently added
      const q = await query(
        colRef,
        orderBy("createdAt", "desc"),
        limit(3),
        // Add a where clause to filter by documents created within the last 3
items
        where("createdAt", ">", new Date(Date.now() - 3 * 24 * 60 * 60 *
1000))
      );
      const subscriber = onSnapshot(q, (snapshot) => {
        let newNutrition = [];
        snapshot.docs.forEach((doc) => {
          newNutrition.push({ ...doc.data(), key: doc.id });
        });
        setNutrition(newNutrition);
        console.log(nutrition);
      });
      return () => subscriber();
    }
  };

  useEffect(() => {
    GetNutrition();
  }, []);

  return (
    <View style={styles.container}>
```



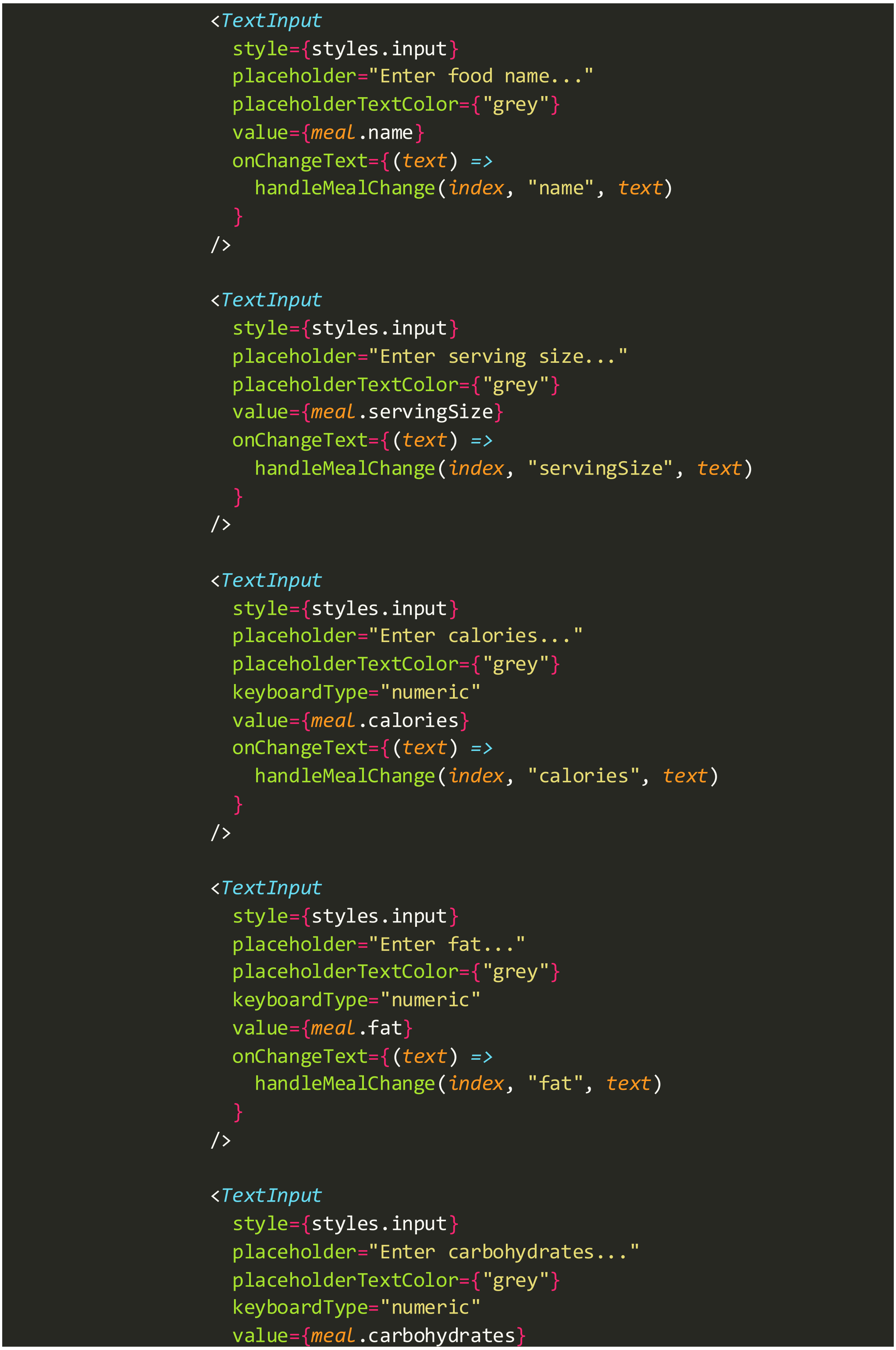
```
      <Text style={styles.title}>Recent Meal Plans</Text>
      {/* List for rendering items */}
      <FlatList
        data={nutrition}
        keyExtractor={(item) => item.key}
        style={{ flex: 1, overflow: "scroll",}}
        renderItem={({ item }) => (
          <Swipeable
            renderRightActions={(progress, dragX) =>
              renderRightActions(progress, dragX, item)
            }
          >
            <TouchableOpacity
              style={styles.cardContainer}
              onPress={() =>
                navigation.navigate("CreatedNutrition", {
                  date: item.date,
                  notes: item.notes,
                  meals: item.meals,
                  id: item.key,
                  name: item.mealPlanName,
                })
              }
            >
              <View style={styles.card}>
                <Text style={styles.mealPlanName}>{item.mealPlanName}</Text>
                <View style={styles.dateContainer}>
                  <MaterialCommunityIcons
                    name="calendar-month"
                    size={16}
                    color="gray"
                  />
                  <Text style={styles.date}>{item.date}</Text>
                </View>
              </View>
            </TouchableOpacity>
          </Swipeable>
        )}
      />
    </View>
  )};


export default RecentNutritionCard;
```

AddNutritionScreen.js

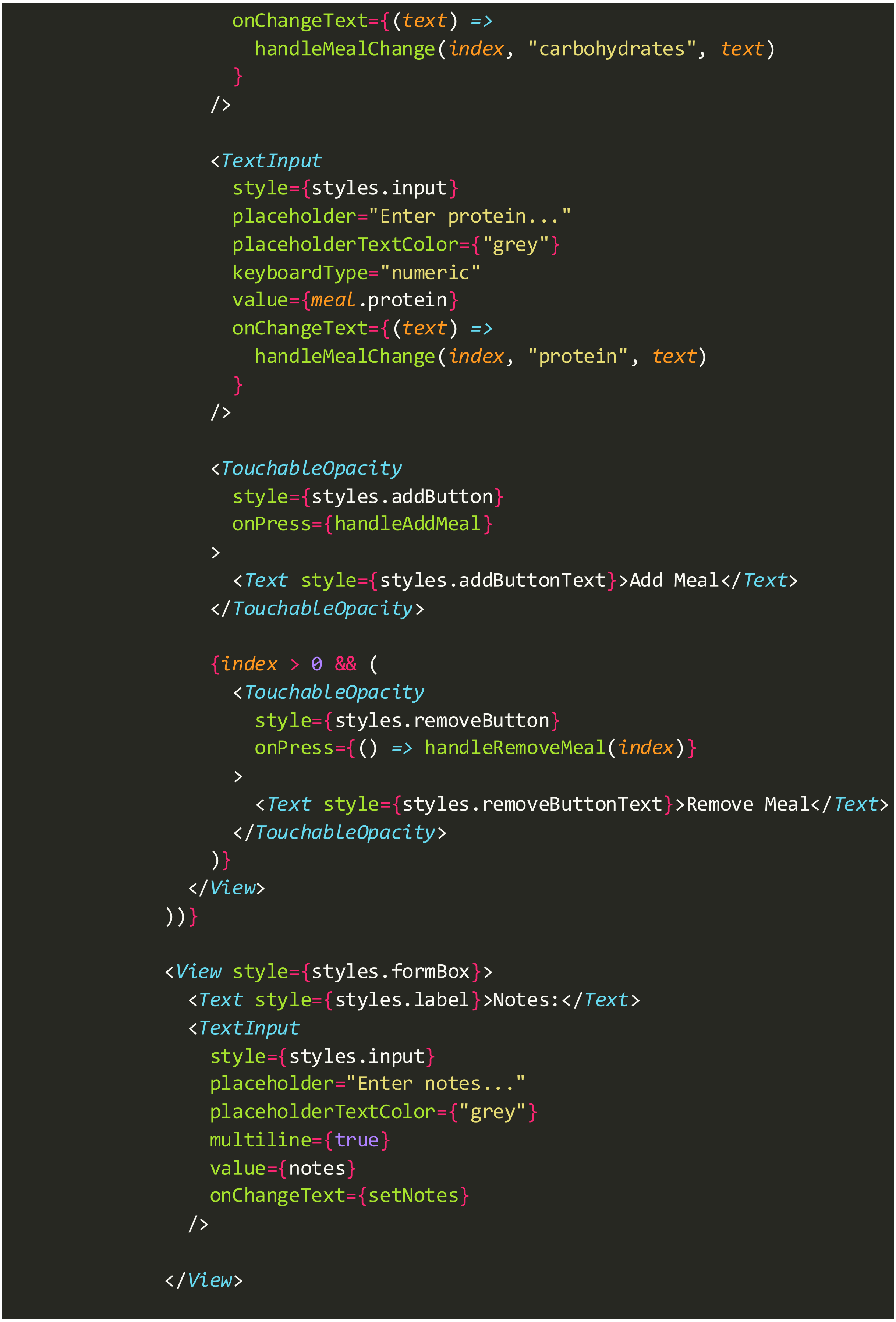
```
const AddNutritionScreen = () => {
  const navigation = useNavigation();
  const [date, setDate] = useState("");
  const [mealPlanName, setMealPlanName] = useState("");
  const [notes, setNotes] = useState("");
  const [meals, setMeals] = useState([
    { name: "", servingSize: "", calories: "", fat: "", carbohydrates: "",
protein: "" },
  ]);

  const handleAddMeal = () => {
    setMeals([
      ...meals,
      { name: "", servingSize: "", calories: "", fat: "", carbohydrates: "",
protein: "" },
    ]);
  };

  const handleRemoveMeal = (index) => {
    const newMeals = [...meals];
    newMeals.splice(index, 1);
    setMeals(newMeals);
  };

  const handleMealChange = (index, field, value) => {
    const newMeals = [...meals];
    newMeals[index][field] = value;
    setMeals(newMeals);
  };

  //Create in Firesotre
  const addNutrition = async () => {
    const user = getAuth().currentUser;
    if (user) {
      try {
        const docRef = doc(db, "users", user.uid);
        const colRef = collection(docRef, "nutrition");
        addDoc(colRef, {
          date: date,
          notes: notes,
          mealPlanName: mealPlanName,
          meals: meals,
          createdAt: serverTimestamp(),
        });
      } catch (e) {
        console.log(e);
      }
```



```
    setDate("");
    setMealPlanName("");
    setMeals([{ name: "" }]);
    setNotes("");
    console.log(meals);
  }
  navigation.goBack();
};

return (
  <SafeAreaView style={styles.container}>
    <KeyboardAvoidingView
      behavior={Platform.OS === "ios" ? "padding" : "height"}
      style={styles.container}
    >
      <Text style={styles.title}>Record Nutrition</Text>
      <ScrollView
        contentContainerStyle={styles.scrollViewContent}
        nestedScrollEnabled={true}
      >
        <View style={styles.formWrapper}>
        <View style={styles.formBox}>
              <Text style={styles.label}>Date:</Text>
              <TextInput
                style={styles.input}
                placeholder="Enter date..."
                placeholderTextColor={"grey"}
                value={date}
                onChangeText={setDate}
              />
            </View>

            <View style={styles.formBox}>
              <Text style={styles.label}>Meal Plan Name:</Text>
              <TextInput
                style={styles.input}
                placeholder={"Enter meal plan name..."}
                placeholderTextColor={"grey"}
                value={mealPlanName}
                onChangeText={setMealPlanName}
              />
            </View>

            {meals.map((meal, index) => (
              <View key={index} style={styles.formBox}>
                <Text style={styles.label}>Meal {index + 1} :</Text>
```



```
<TextInput
  style={styles.input}
  placeholder="Enter food name..."
  placeholderTextColor={"grey"}
  value={meal.name}
  onChangeText={(text) =>
    handleMealChange(index, "name", text)
  }
/>

<TextInput
  style={styles.input}
  placeholder="Enter serving size..."
  placeholderTextColor={"grey"}
  value={meal.servingSize}
  onChangeText={(text) =>
    handleMealChange(index, "servingSize", text)
  }
/>

<TextInput
  style={styles.input}
  placeholder="Enter calories..."
  placeholderTextColor={"grey"}
  keyboardType="numeric"
  value={meal.calories}
  onChangeText={(text) =>
    handleMealChange(index, "calories", text)
  }
/>

<TextInput
  style={styles.input}
  placeholder="Enter fat..."
  placeholderTextColor={"grey"}
  keyboardType="numeric"
  value={meal.fat}
  onChangeText={(text) =>
    handleMealChange(index, "fat", text)
  }
/>

<TextInput
  style={styles.input}
  placeholder="Enter carbohydrates..."
  placeholderTextColor={"grey"}
  keyboardType="numeric"
  value={meal.carbohydrates}
```



```
      onChangeText={(text) =>
        handleMealChange(index, "carbohydrates", text)
      }
    />

    <TextInput
      style={styles.input}
      placeholder="Enter protein..."
      placeholderTextColor={"grey"}
      keyboardType="numeric"
      value={meal.protein}
      onChangeText={(text) =>
        handleMealChange(index, "protein", text)
      }
    />

    <TouchableOpacity
      style={styles.addButton}
      onPress={handleAddMeal}
    >
      <Text style={styles.addButtonText}>Add Meal</Text>
    </TouchableOpacity>

    {index > 0 && (
      <TouchableOpacity
        style={styles.removeButton}
        onPress={() => handleRemoveMeal(index)}
      >
        <Text style={styles.removeButtonText}>Remove Meal</Text>
      </TouchableOpacity>
    )}
  </View>
))}

<View style={styles.formBox}>
  <Text style={styles.label}>Notes:</Text>
  <TextInput
    style={styles.input}
    placeholder="Enter notes..."
    placeholderTextColor={"grey"}
    multiline={true}
    value={notes}
    onChangeText={setNotes}
  />

</View>
```

```
              <TouchableOpacity style={styles.addButton}
onPress={addNutrition}>
                <Text style={styles.addButtonText}>Submit</Text>
              </TouchableOpacity>
              <TouchableOpacity style={styles.cancelButton} onPress={() =>
navigation.goBack()}>
                <Text style={styles.addButtonText}>Cancel</Text>
              </TouchableOpacity>
            </View>
          </ScrollView>
        </KeyboardAvoidingView>
      </SafeAreaView>
    );
  };

export default AddNutritionScreen;
```

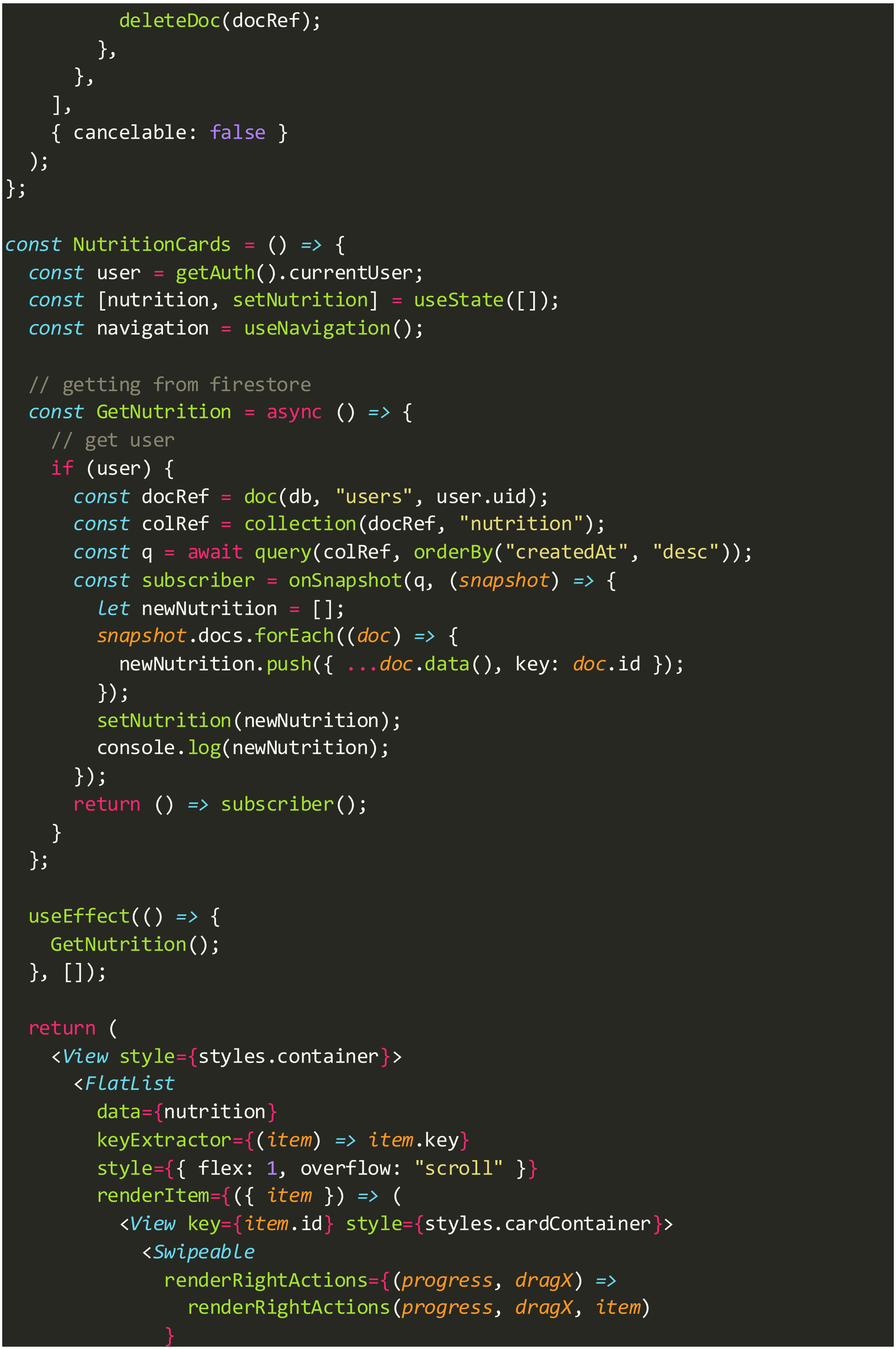
AllNutritionScreen.js

```
const AllNutritionScreen = () => {
  //getting the user data
  const user = getAuth().currentUser;

  //setting the state
  const [plams, setPlans] = useState([]);
  const navigation = useNavigation();

  return (
    <SafeAreaView style={styles.heading}>
      <View style ={{zIndex: 1}}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Ionicons
            name="arrow-back"
            size={30}
            color="#0792F9"
            style={styles.arrow}
          />
        </TouchableOpacity>
        </View>
      <View style={styles.header}>
        <Text style={styles.heading}>Nutrition Plans</Text>
        <Pressable>
          <Entypo name="menu" size={30} color="black" />
        </Pressable>
      </View>
      <NutritionCards />
    </SafeAreaView>
  );
};
```

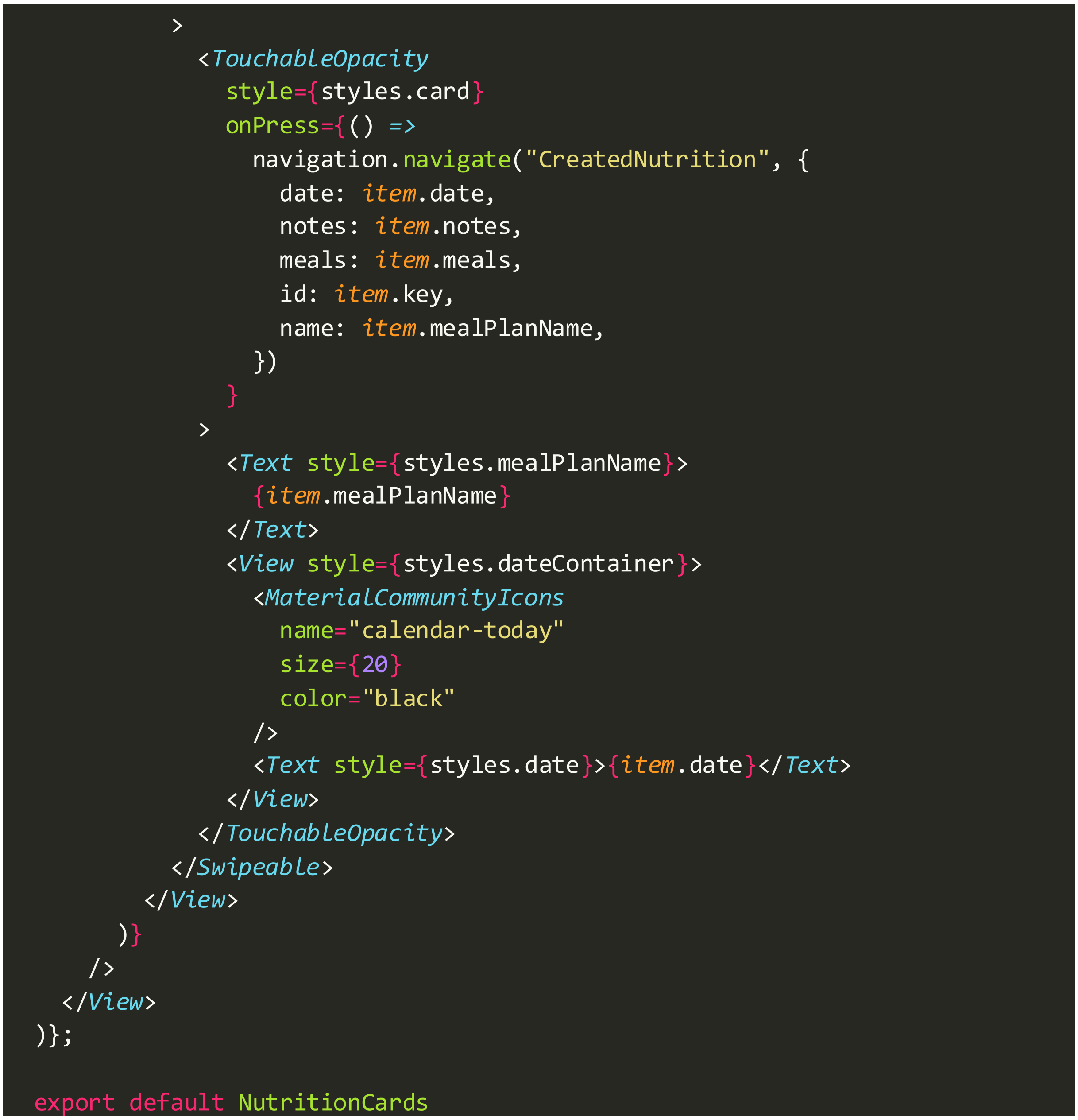
```
export default AllNutritionScreen;
```

NutritionCards.js

```
//for swiping
const renderRightActions = (progress, dragX, item) => {
  const user = getAuth().currentUser;

  const trans = dragX.interpolate({
    inputRange: [0, 50, 100, 101],
    outputRange: [0, 0, 0, 1],
  });
  return (
    <TouchableOpacity
      style={styles.deleteBtn}
      onPress={() => DeleteNutrition(item)}
    >
      <Ionicons name="trash-bin" size={40} color="red" />
      <Animated.Text
        style={[
          styles.deleteText,
          {
            transform: [{ translateX: trans }],
          },
        ]}
      >
        Delete
      </Animated.Text>
    </TouchableOpacity>
  );
};

const DeleteNutrition = (item) => {
  const user = getAuth().currentUser;

  Alert.alert(
    "Delete Account",
    "Are you sure you want to delete this meal plan?",
    [
      {
        text: "Cancel",
        style: "cancel",
      },
      {
        text: "Delete",
        onPress: () => {
          const docRef = doc(db, "users", user.uid, "nutrition", item.key);
```

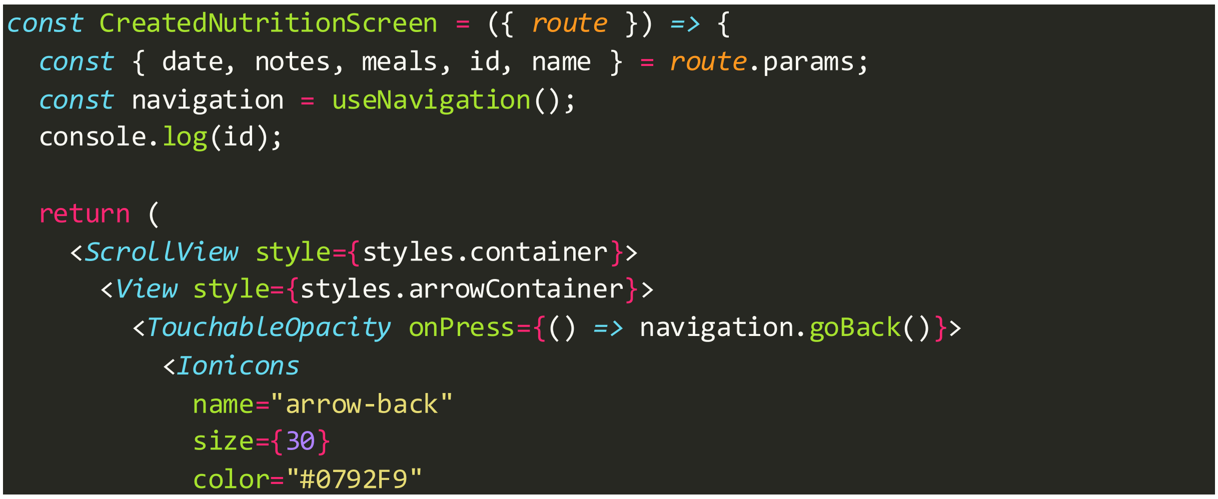


```
          deleteDoc(docRef);
        },
      },
    ],
    { cancelable: false }
  );
};

const NutritionCards = () => {
  const user = getAuth().currentUser;
  const [nutrition, setNutrition] = useState([]);
  const navigation = useNavigation();

  // getting from firestore
  const GetNutrition = async () => {
    // get user
    if (user) {
      const docRef = doc(db, "users", user.uid);
      const colRef = collection(docRef, "nutrition");
      const q = await query(colRef, orderBy("createdAt", "desc"));
      const subscriber = onSnapshot(q, (snapshot) => {
        let newNutrition = [];
        snapshot.docs.forEach((doc) => {
          newNutrition.push({ ...doc.data(), key: doc.id });
        });
        setNutrition(newNutrition);
        console.log(newNutrition);
      });
      return () => subscriber();
    }
  };

  useEffect(() => {
    GetNutrition();
  }, []);

  return (
    <View style={styles.container}>
      <FlatList
        data={nutrition}
        keyExtractor={(item) => item.key}
        style={{ flex: 1, overflow: "scroll" }}
        renderItem={({ item }) => (
          <View key={item.id} style={styles.cardContainer}>
            <Swipeable
              renderRightActions={(progress, dragX) =>
                renderRightActions(progress, dragX, item)
              }
```

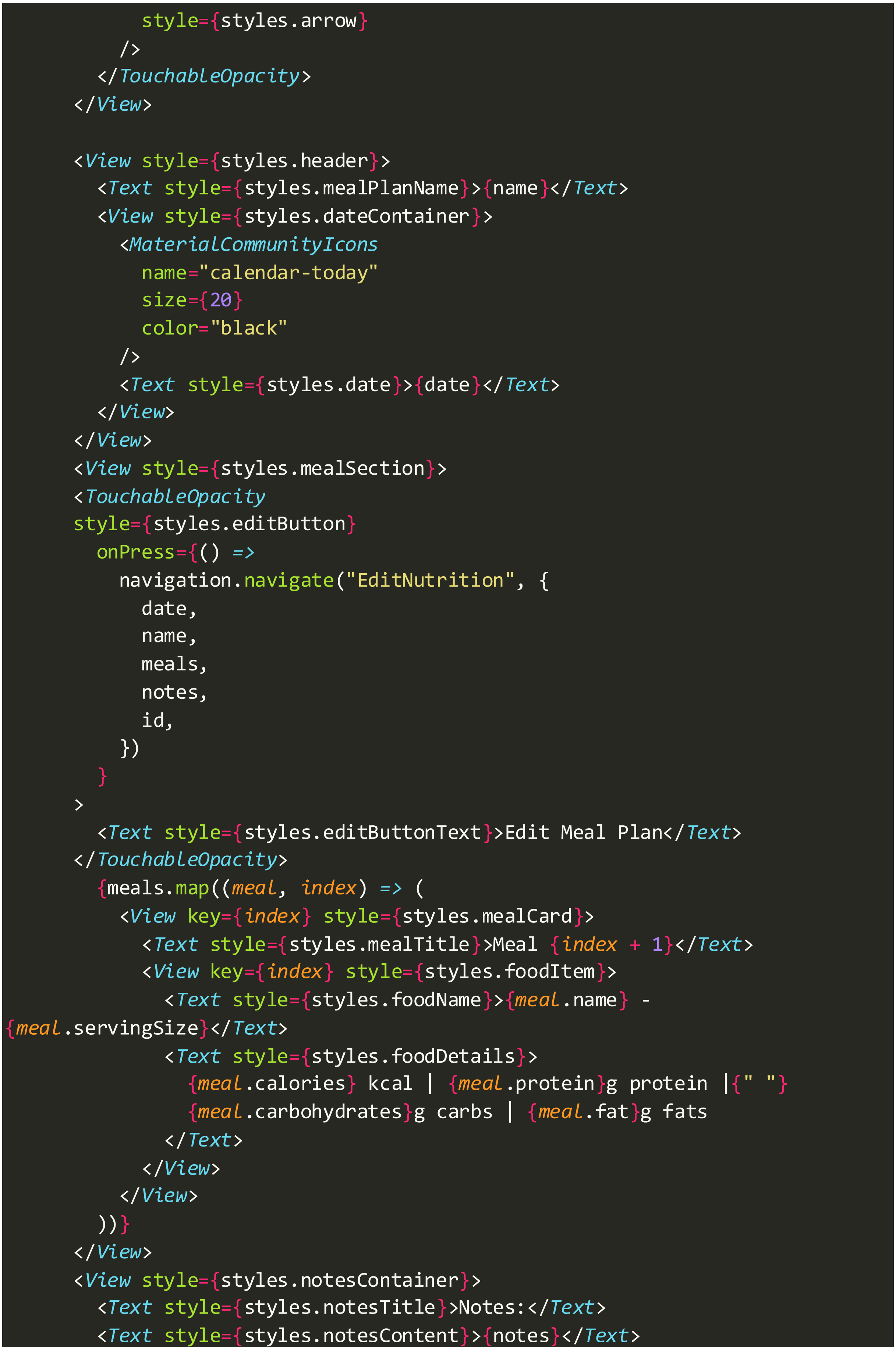


```
          >
            <TouchableOpacity
              style={styles.card}
              onPress={() =>
                navigation.navigate("CreatedNutrition", {
                  date: item.date,
                  notes: item.notes,
                  meals: item.meals,
                  id: item.key,
                  name: item.mealPlanName,
                })
              }
            >
              <Text style={styles.mealPlanName}>
                {item.mealPlanName}
              </Text>
              <View style={styles.dateContainer}>
                <MaterialCommunityIcons
                  name="calendar-today"
                  size={20}
                  color="black"
                />
                <Text style={styles.date}>{item.date}</Text>
              </View>
            </TouchableOpacity>
          </Swipeable>
        </View>
      )}
    />
  </View>
)};

export default NutritionCards
```

CreatedNutritionScreen.js



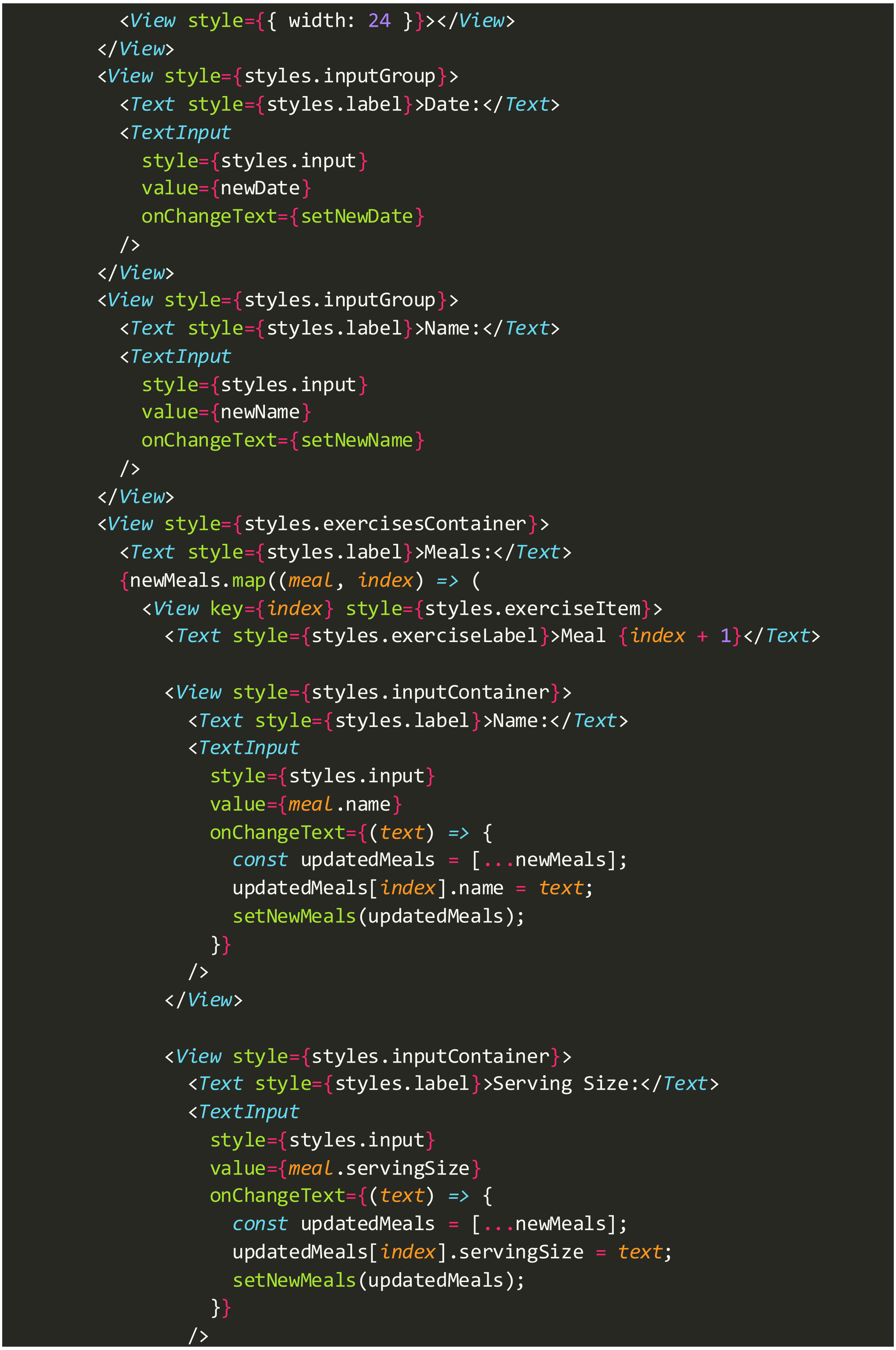
```
const CreatedNutritionScreen = ({ route }) => {
  const { date, notes, meals, id, name } = route.params;
  const navigation = useNavigation();
  console.log(id);

  return (
    <ScrollView style={styles.container}>
      <View style={styles.arrowContainer}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Ionicons
            name="arrow-back"
            size={30}
            color="#0792F9"
```



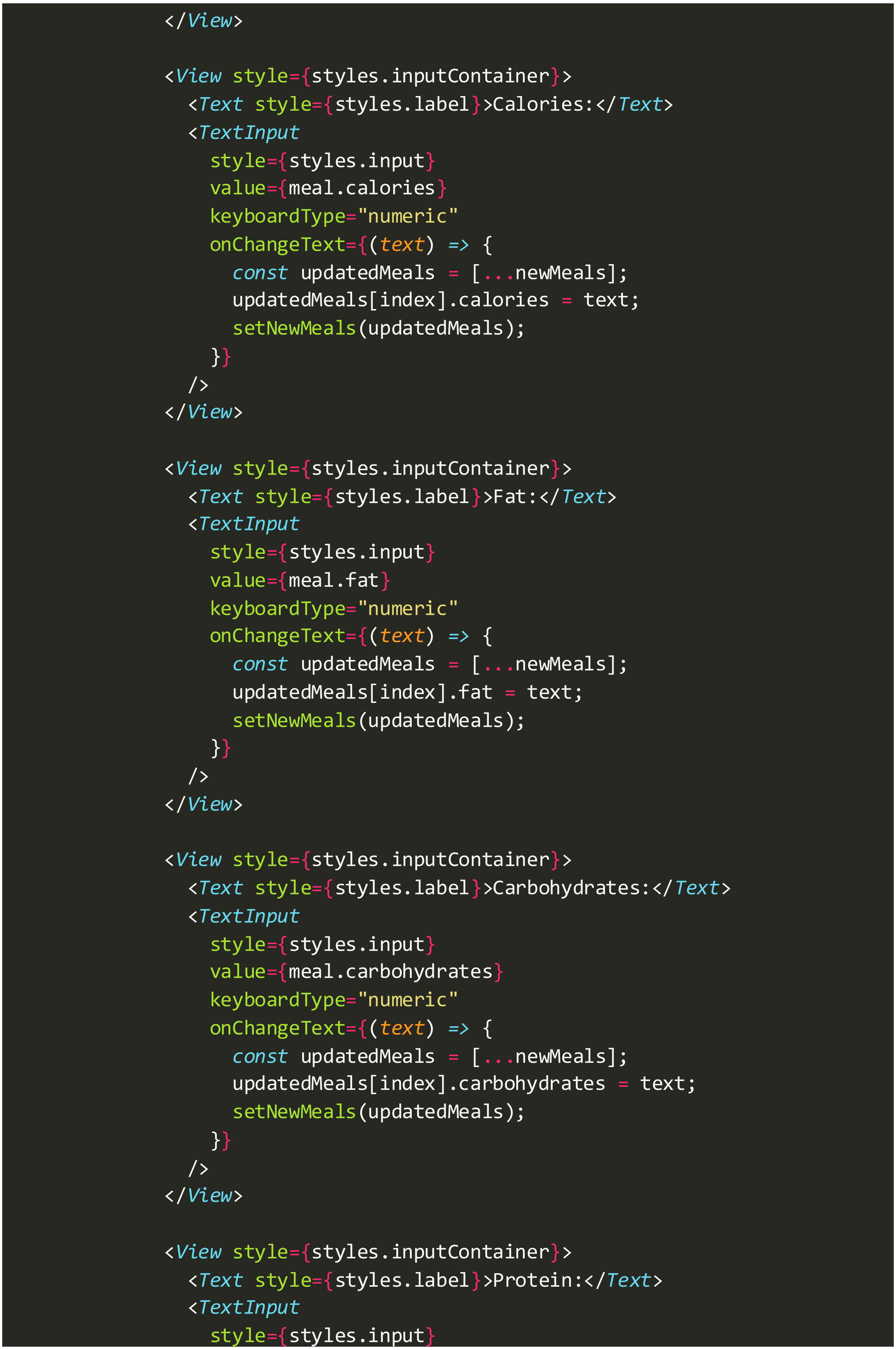
```
            style={styles.arrow}
          />
        </TouchableOpacity>
      </View>

      <View style={styles.header}>
        <Text style={styles.mealPlanName}>{name}</Text>
        <View style={styles.dateContainer}>
          <MaterialCommunityIcons
            name="calendar-today"
            size={20}
            color="black"
          />
          <Text style={styles.date}>{date}</Text>
        </View>
      </View>
      <View style={styles.mealSection}>
      <TouchableOpacity
      style={styles.editButton}
        onPress={() =>
          navigation.navigate("EditNutrition", {
            date,
            name,
            meals,
            notes,
            id,
          })
        }
      >
        <Text style={styles.editButtonText}>Edit Meal Plan</Text>
      </TouchableOpacity>
        {meals.map((meal, index) => (
          <View key={index} style={styles.mealCard}>
            <Text style={styles.mealTitle}>Meal {index + 1}</Text>
            <View key={index} style={styles.foodItem}>
              <Text style={styles.foodName}>{meal.name} -
{meal.servingSize}</Text>
              <Text style={styles.foodDetails}>
                {meal.calories} kcal | {meal.protein}g protein |{" "}
                {meal.carbohydrates}g carbs | {meal.fat}g fats
              </Text>
            </View>
          </View>
        ))}
      </View>
      <View style={styles.notesContainer}>
        <Text style={styles.notesTitle}>Notes:</Text>
        <Text style={styles.notesContent}>{notes}</Text>
      </View>
    </ScrollView>
  );
};

export default CreatedNutritionScreen;
```

EditNutritionScreen.js

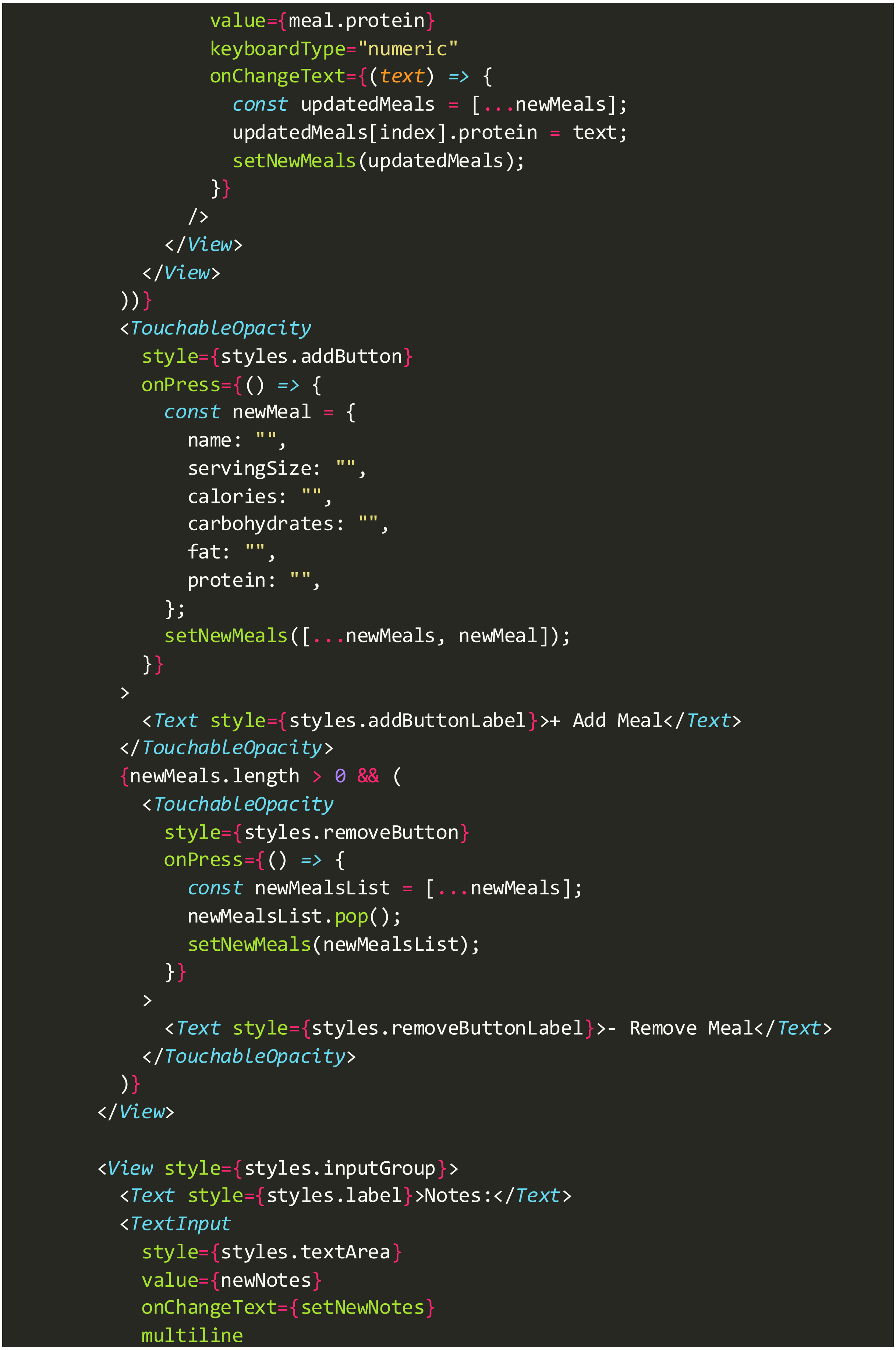
```
const EditNutritionScreen = ({ route, navigation }) => {
  const { date, meals, name, notes, id } = route.params;

  const user = getAuth().currentUser;
  const [newDate, setNewDate] = useState(date);
  const [newName, setNewName] = useState(name);
  const [newMeals, setNewMeals] = useState([...meals]);
  const [newNotes, setNewNotes] = useState(notes);

  const handleSave = async () => {
    try {
      if (user) {
        const docRef = doc(db, `users/${user.uid}/nutrition/${id}`);

        await setDoc(docRef, {
          mealPlanName: newName,
          date: newDate,
          notes: newNotes,
          meals: newMeals,
          createdAt: serverTimestamp(),
        });
        console.log("Updated successfully");
        navigation.goBack();
      }
    } catch (error) {
      console.error("Error updating workout: ", error);
    }
  };

  return (
    <KeyboardAvoidingView
      behavior={Platform.OS === "ios" ? "padding" : "height"}
      style={styles.container}
    >
      <ScrollView>
        <View style={styles.headerContainer}>
          <TouchableOpacity onPress={() => navigation.goBack()}>
            <Ionicons name="arrow-back" size={30} color="#0792F9" />
          </TouchableOpacity>
          <Text style={styles.header}>Edit Meal Plan</Text>
```



```
          <View style={{ width: 24 }}></View>
        </View>
        <View style={styles.inputGroup}>
          <Text style={styles.label}>Date:</Text>
          <TextInput
            style={styles.input}
            value={newDate}
            onChangeText={setNewDate}
          />
        </View>
        <View style={styles.inputGroup}>
          <Text style={styles.label}>Name:</Text>
          <TextInput
            style={styles.input}
            value={newName}
            onChangeText={setNewName}
          />
        </View>
        <View style={styles.exercisesContainer}>
          <Text style={styles.label}>Meals:</Text>
          {newMeals.map((meal, index) => (
            <View key={index} style={styles.exerciseItem}>
              <Text style={styles.exerciseLabel}>Meal {index + 1}</Text>

              <View style={styles.inputContainer}>
                <Text style={styles.label}>Name:</Text>
                <TextInput
                  style={styles.input}
                  value={meal.name}
                  onChangeText={(text) => {
                    const updatedMeals = [...newMeals];
                    updatedMeals[index].name = text;
                    setNewMeals(updatedMeals);
                  }}
                />
              </View>

              <View style={styles.inputContainer}>
                <Text style={styles.label}>Serving Size:</Text>
                <TextInput
                  style={styles.input}
                  value={meal.servingSize}
                  onChangeText={(text) => {
                    const updatedMeals = [...newMeals];
                    updatedMeals[index].servingSize = text;
                    setNewMeals(updatedMeals);
                  }}
                />
```



```
            </View>

            <View style={styles.inputContainer}>
              <Text style={styles.label}>Calories:</Text>
              <TextInput
                style={styles.input}
                value={meal.calories}
                keyboardType="numeric"
                onChangeText={(text) => {
                  const updatedMeals = [...newMeals];
                  updatedMeals[index].calories = text;
                  setNewMeals(updatedMeals);
                }}
              />
            </View>

            <View style={styles.inputContainer}>
              <Text style={styles.label}>Fat:</Text>
              <TextInput
                style={styles.input}
                value={meal.fat}
                keyboardType="numeric"
                onChangeText={(text) => {
                  const updatedMeals = [...newMeals];
                  updatedMeals[index].fat = text;
                  setNewMeals(updatedMeals);
                }}
              />
            </View>

            <View style={styles.inputContainer}>
              <Text style={styles.label}>Carbohydrates:</Text>
              <TextInput
                style={styles.input}
                value={meal.carbohydrates}
                keyboardType="numeric"
                onChangeText={(text) => {
                  const updatedMeals = [...newMeals];
                  updatedMeals[index].carbohydrates = text;
                  setNewMeals(updatedMeals);
                }}
              />
            </View>

            <View style={styles.inputContainer}>
              <Text style={styles.label}>Protein:</Text>
              <TextInput
                style={styles.input}
```



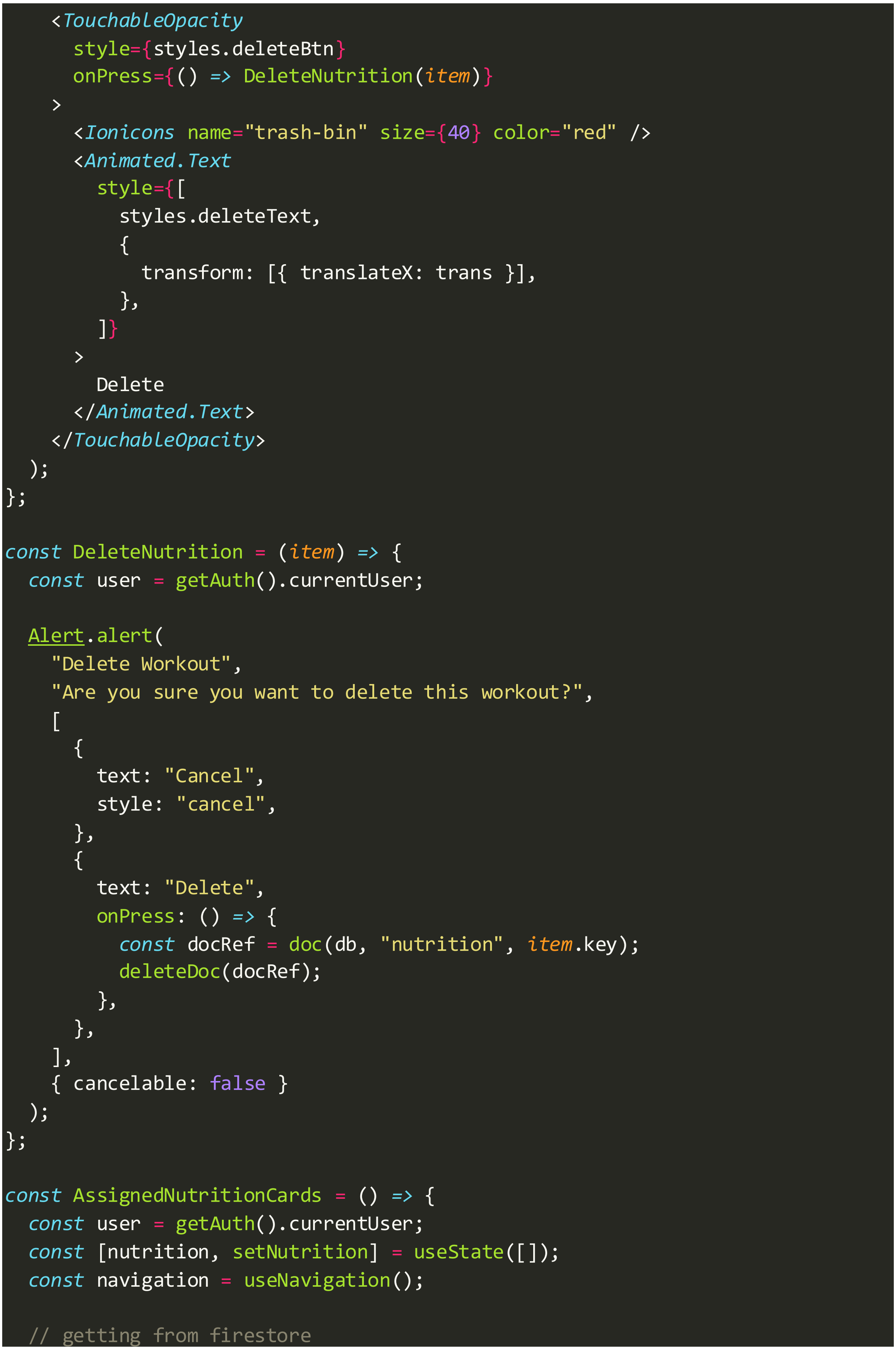
```
                  value={meal.protein}
                  keyboardType="numeric"
                  onChangeText={(text) => {
                    const updatedMeals = [...newMeals];
                    updatedMeals[index].protein = text;
                    setNewMeals(updatedMeals);
                  }}
                />
              </View>
            </View>
          ))}
          <TouchableOpacity
            style={styles.addButton}
            onPress={() => {
              const newMeal = {
                name: "",
                servingSize: "",
                calories: "",
                carbohydrates: "",
                fat: "",
                protein: "",
              };
              setNewMeals([...newMeals, newMeal]);
            }}
          >
            <Text style={styles.addButtonLabel}>+ Add Meal</Text>
          </TouchableOpacity>
          {newMeals.length > 0 && (
            <TouchableOpacity
              style={styles.removeButton}
              onPress={() => {
                const newMealsList = [...newMeals];
                newMealsList.pop();
                setNewMeals(newMealsList);
              }}
            >
              <Text style={styles.removeButtonLabel}>- Remove Meal</Text>
            </TouchableOpacity>
          )}
        </View>

        <View style={styles.inputGroup}>
          <Text style={styles.label}>Notes:</Text>
          <TextInput
            style={styles.textArea}
            value={newNotes}
            onChangeText={setNewNotes}
            multiline
          />
        </View>
        <TouchableOpacity style={styles.saveButton} onPress={handleSave}>
          <Text style={styles.saveButtonText}>Save</Text>
        </TouchableOpacity>
      </ScrollView>
    </KeyboardAvoidingView>
  );
};

export default EditNutritionScreen;
```

AssignedNutrition.js

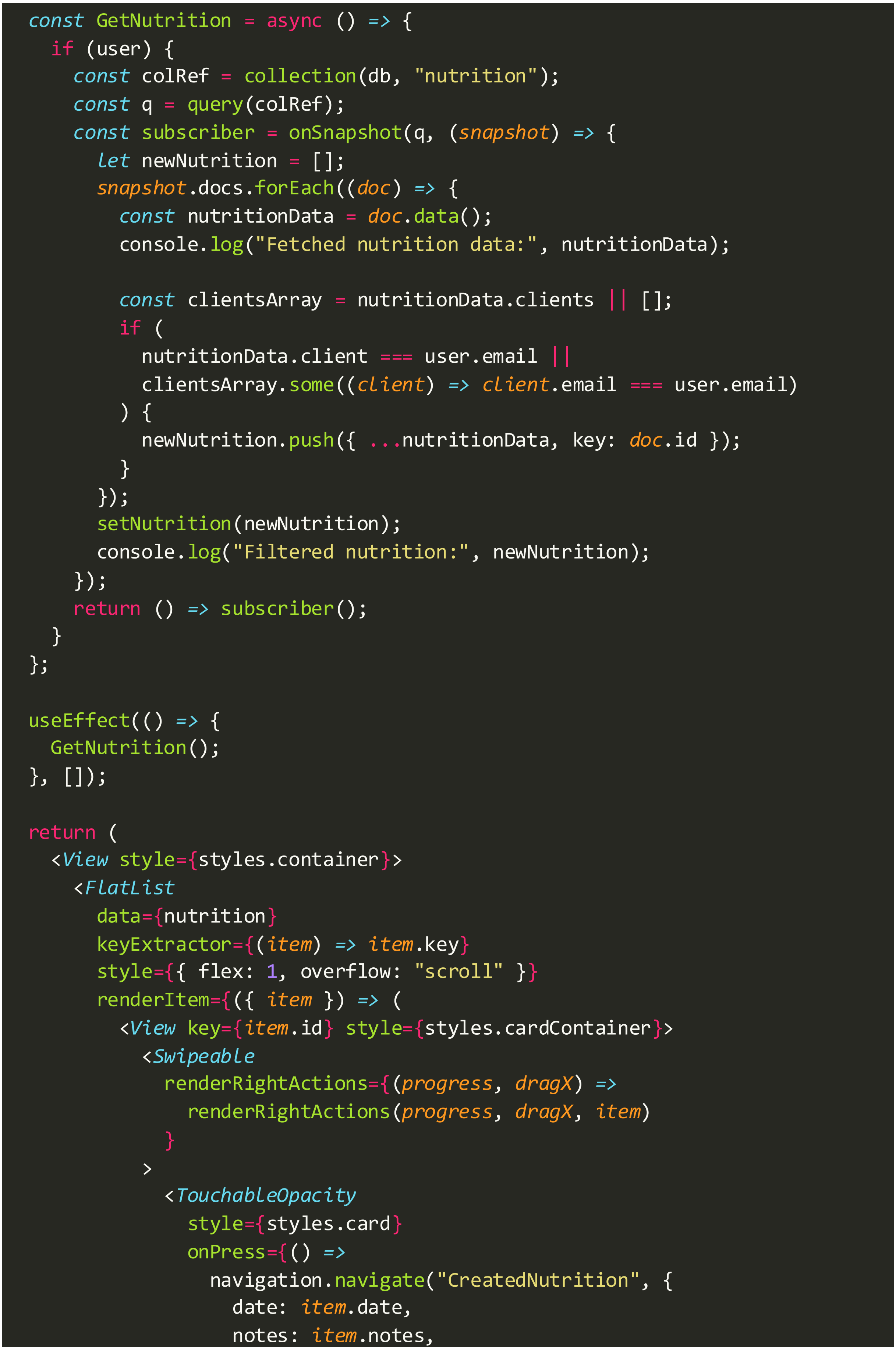
```
const AllWorkoutScreen = () => {
  const navigation = useNavigation();
  return (
    <SafeAreaView style={styles.heading}>
      <View style={styles.header}>
      <TouchableOpacity onPress={() => navigation.goBack()}
style={{right:15}}>
          <Ionicons name="arrow-back" size={30} color="#0792F9" />
        </TouchableOpacity>
        <Text style={styles.heading}>Nutrition Plans</Text>
        <Pressable>
          <Entypo name="menu" size={30} color="black" />
        </Pressable>
      </View>

      <AssignedNutritionCards/>

    </SafeAreaView>
  );
};

export default AllWorkoutScreen;
```

AssignedNutritionCards.js

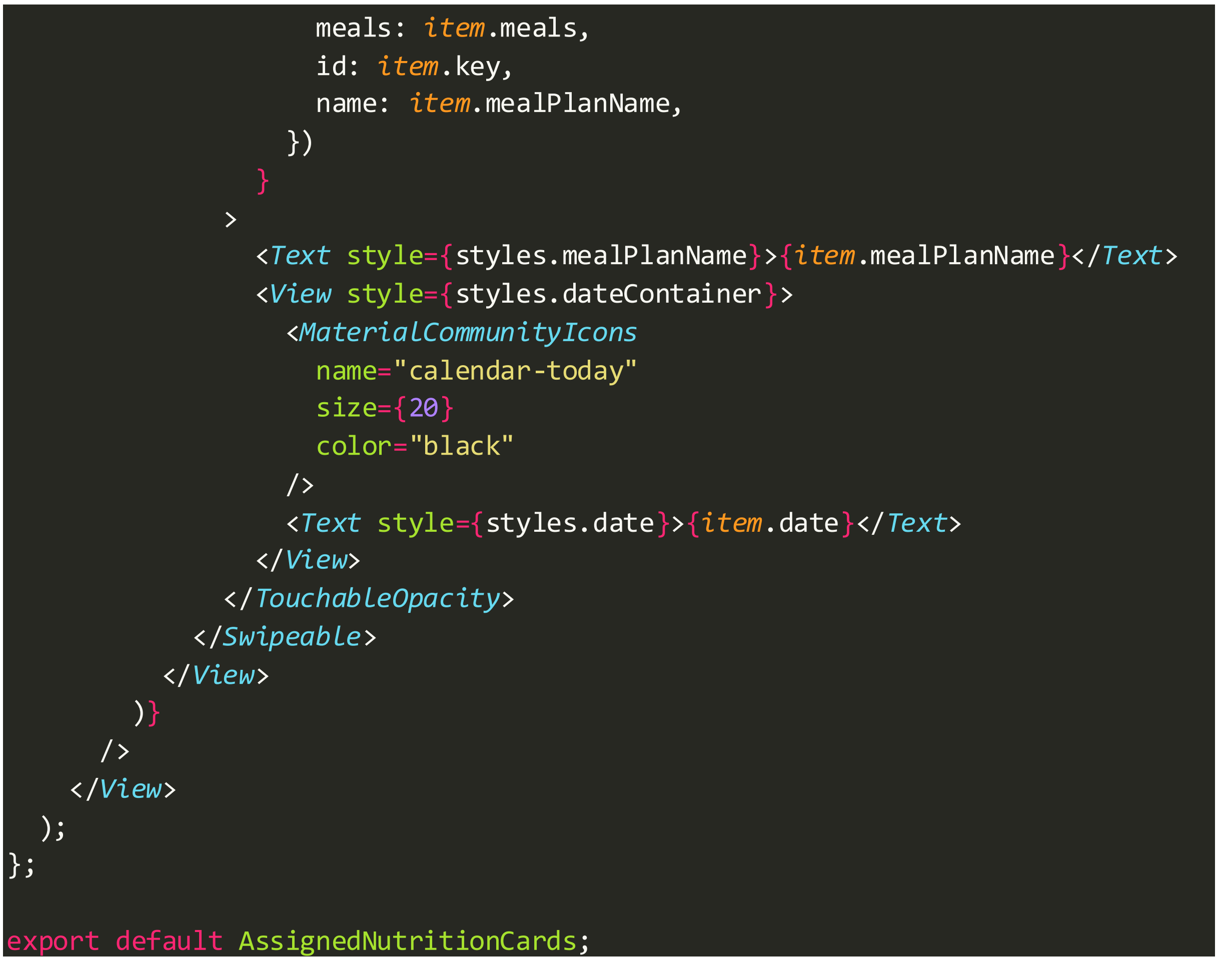
```
//for swiping
const renderRightActions = (progress, dragX, item) => {
  const user = getAuth().currentUser;

  const trans = dragX.interpolate({
    inputRange: [0, 50, 100, 101],
    outputRange: [0, 0, 0, 1],
  });
  return (
```



```
    <TouchableOpacity
      style={styles.deleteBtn}
      onPress={() => DeleteNutrition(item)}
    >
      <Ionicons name="trash-bin" size={40} color="red" />
      <Animated.Text
        style={[
          styles.deleteText,
          {
            transform: [{ translateX: trans }],
          },
        ]}
      >
        Delete
      </Animated.Text>
    </TouchableOpacity>
  );
};

const DeleteNutrition = (item) => {
  const user = getAuth().currentUser;

  Alert.alert(
    "Delete Workout",
    "Are you sure you want to delete this workout?",
    [
      {
        text: "Cancel",
        style: "cancel",
      },
      {
        text: "Delete",
        onPress: () => {
          const docRef = doc(db, "nutrition", item.key);
          deleteDoc(docRef);
        },
      },
    ],
    { cancelable: false }
  );
};

const AssignedNutritionCards = () => {
  const user = getAuth().currentUser;
  const [nutrition, setNutrition] = useState([]);
  const navigation = useNavigation();

  // getting from firestore
```

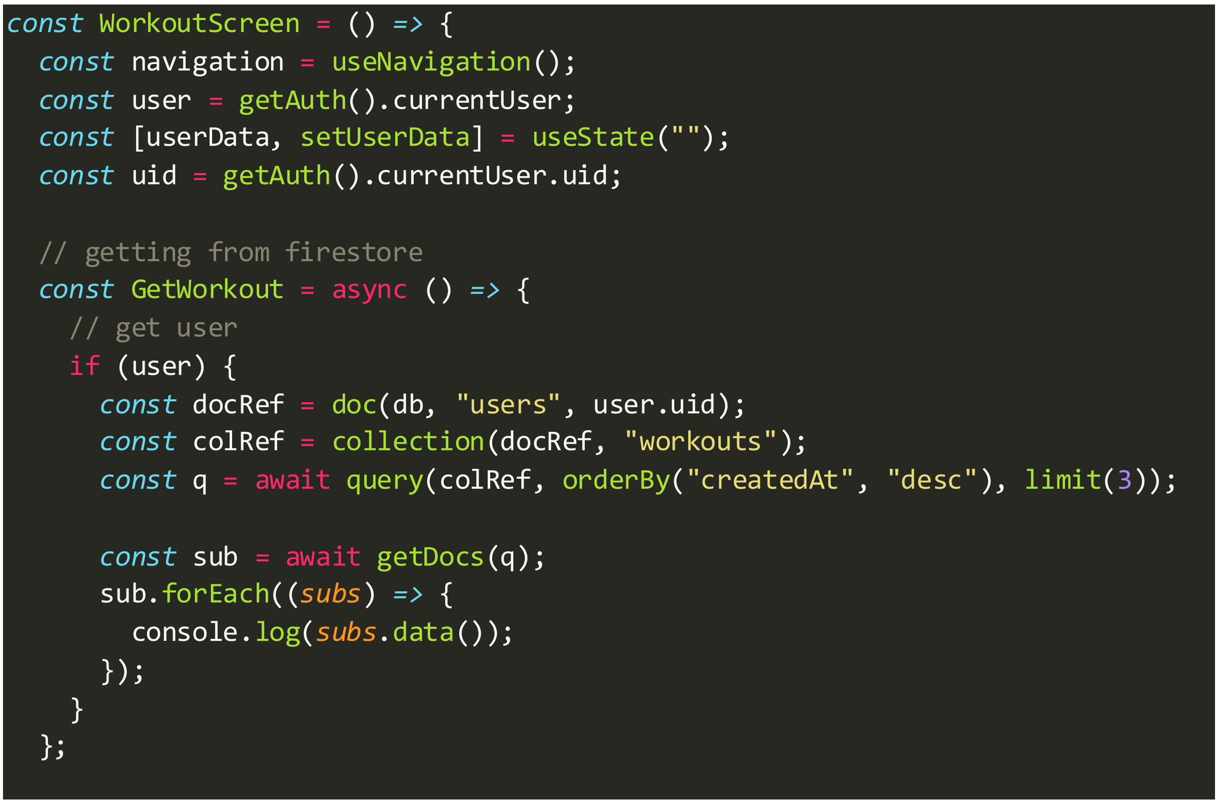


```
const GetNutrition = async () => {
  if (user) {
    const colRef = collection(db, "nutrition");
    const q = query(colRef);
    const subscriber = onSnapshot(q, (snapshot) => {
      let newNutrition = [];
      snapshot.docs.forEach((doc) => {
        const nutritionData = doc.data();
        console.log("Fetched nutrition data:", nutritionData);

        const clientsArray = nutritionData.clients || [];
        if (
          nutritionData.client === user.email ||
          clientsArray.some((client) => client.email === user.email)
        ) {
          newNutrition.push({ ...nutritionData, key: doc.id });
        }
      });
      setNutrition(newNutrition);
      console.log("Filtered nutrition:", newNutrition);
    });
    return () => subscriber();
  }
};

useEffect(() => {
  GetNutrition();
}, []);

return (
  <View style={styles.container}>
    <FlatList
      data={nutrition}
      keyExtractor={(item) => item.key}
      style={{ flex: 1, overflow: "scroll" }}
      renderItem={({ item }) => (
        <View key={item.id} style={styles.cardContainer}>
          <Swipeable
            renderRightActions={(progress, dragX) =>
              renderRightActions(progress, dragX, item)
            }
          >
            <TouchableOpacity
              style={styles.card}
              onPress={() =>
                navigation.navigate("CreatedNutrition", {
                  date: item.date,
                  notes: item.notes,
```

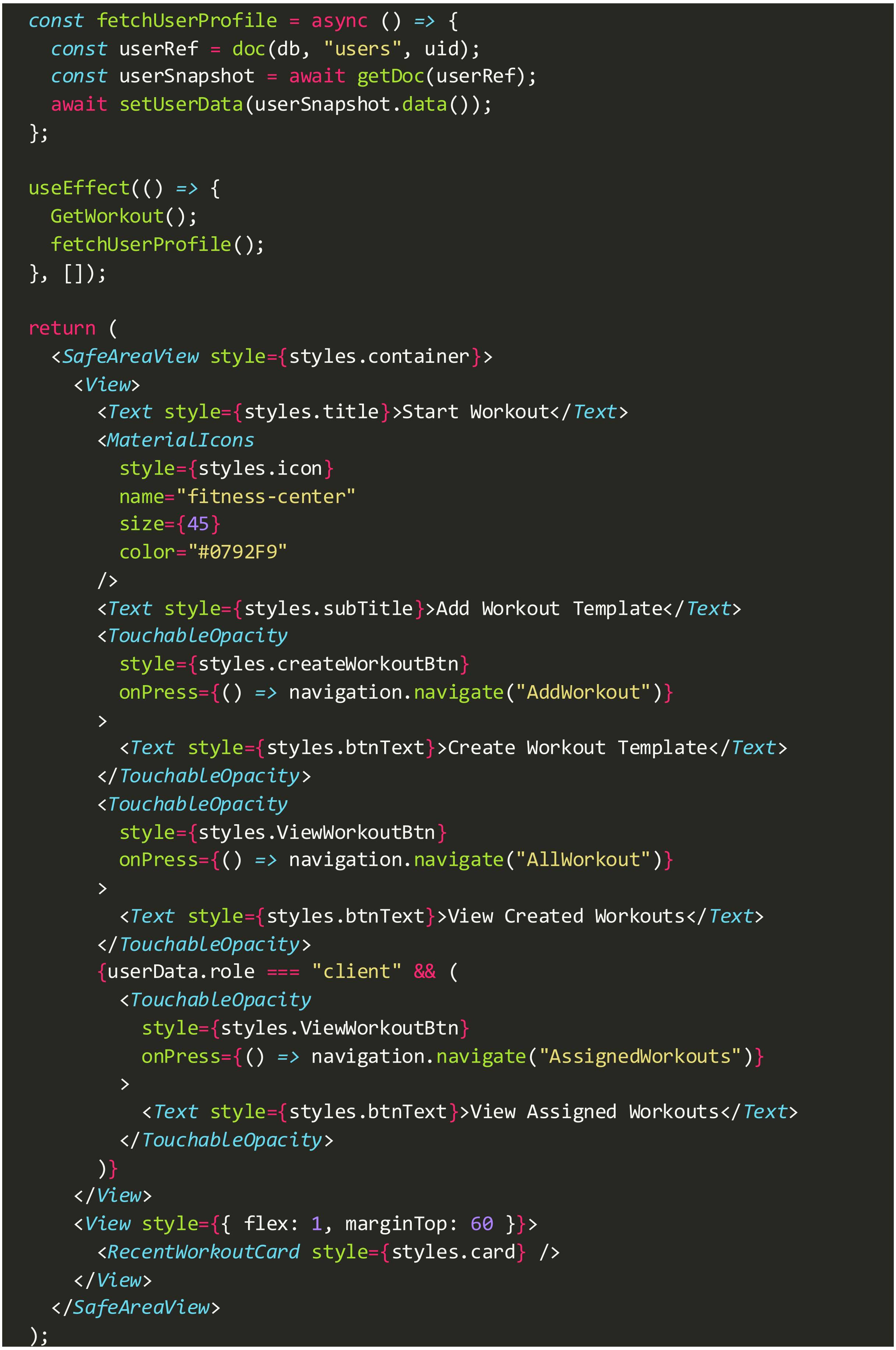


```
                    meals: item.meals,
                    id: item.key,
                    name: item.mealPlanName,
                  })
                }
              >
                <Text style={styles.mealPlanName}>{item.mealPlanName}</Text>
                <View style={styles.dateContainer}>
                  <MaterialCommunityIcons
                    name="calendar-today"
                    size={20}
                    color="black"
                  />
                  <Text style={styles.date}>{item.date}</Text>
                </View>
              </TouchableOpacity>
            </Swipeable>
          </View>
        )}
      />
    </View>
  );
};

export default AssignedNutritionCards;
```

WorkoutScreen.js



```
const WorkoutScreen = () => {
  const navigation = useNavigation();
  const user = getAuth().currentUser;
  const [userData, setUserData] = useState("");
  const uid = getAuth().currentUser.uid;

  // getting from firestore
  const GetWorkout = async () => {
    // get user
    if (user) {
      const docRef = doc(db, "users", user.uid);
      const colRef = collection(docRef, "workouts");
      const q = await query(colRef, orderBy("createdAt", "desc"), limit(3));

      const sub = await getDocs(q);
      sub.forEach((subs) => {
        console.log(subs.data());
      });
    }
  };
```

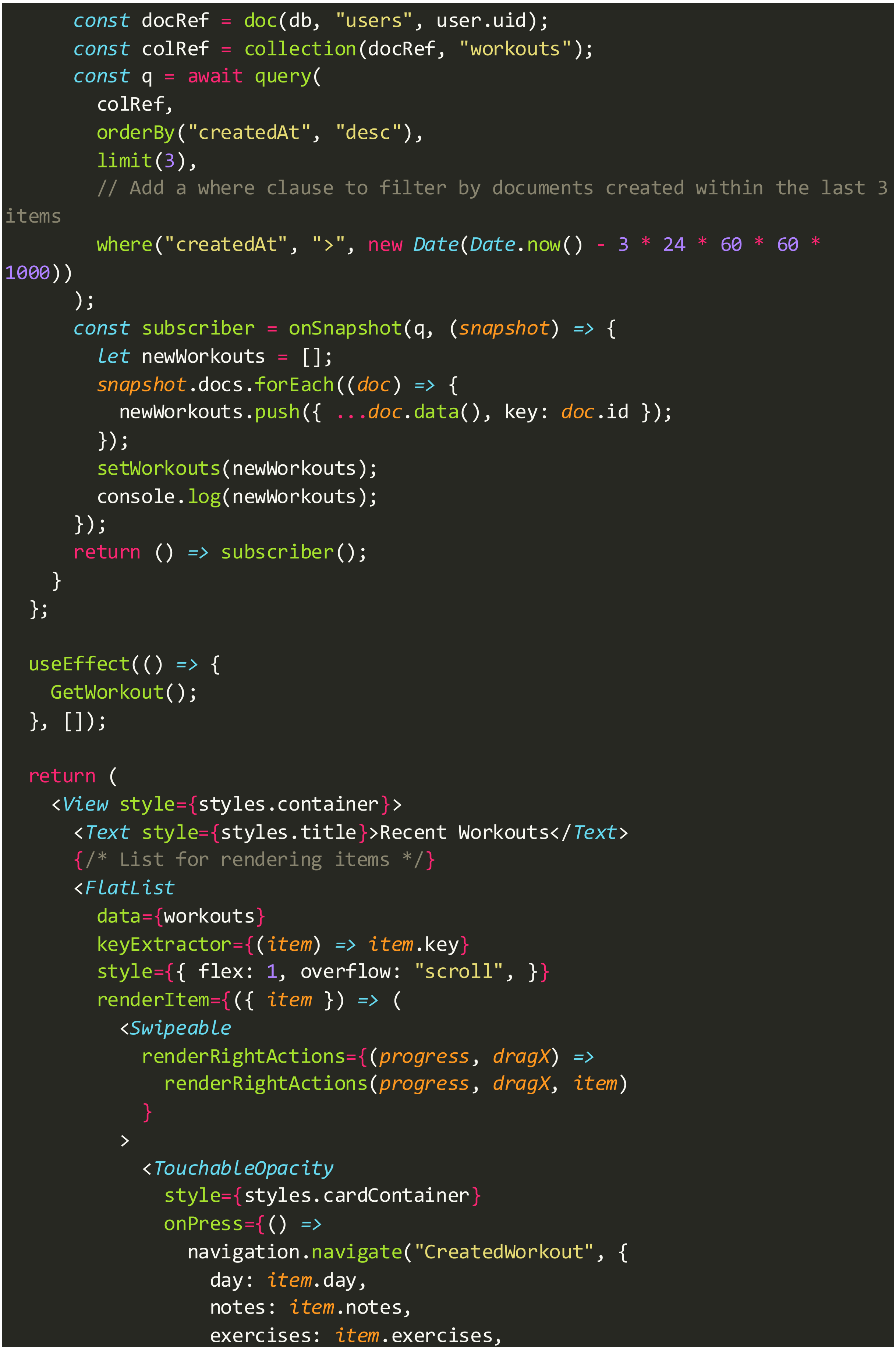


```
const fetchUserProfile = async () => {
  const userRef = doc(db, "users", uid);
  const userSnapshot = await getDoc(userRef);
  await setUserData(userSnapshot.data());
};

useEffect(() => {
  GetWorkout();
  fetchUserProfile();
}, []);

return (
  <SafeAreaView style={styles.container}>
    <View>
      <Text style={styles.title}>Start Workout</Text>
      <MaterialIcons
        style={styles.icon}
        name="fitness-center"
        size={45}
        color="#0792F9"
      />
      <Text style={styles.subTitle}>Add Workout Template</Text>
      <TouchableOpacity
        style={styles.createWorkoutBtn}
        onPress={() => navigation.navigate("AddWorkout")}
      >
        <Text style={styles.btnText}>Create Workout Template</Text>
      </TouchableOpacity>
      <TouchableOpacity
        style={styles.ViewWorkoutBtn}
        onPress={() => navigation.navigate("AllWorkout")}
      >
        <Text style={styles.btnText}>View Created Workouts</Text>
      </TouchableOpacity>
      {userData.role === "client" && (
        <TouchableOpacity
          style={styles.ViewWorkoutBtn}
          onPress={() => navigation.navigate("AssignedWorkouts")}
        >
          <Text style={styles.btnText}>View Assigned Workouts</Text>
        </TouchableOpacity>
      )}
    </View>
    <View style={{ flex: 1, marginTop: 60 }}>
      <RecentWorkoutCard style={styles.card} />
    </View>
  </SafeAreaView>
);
```

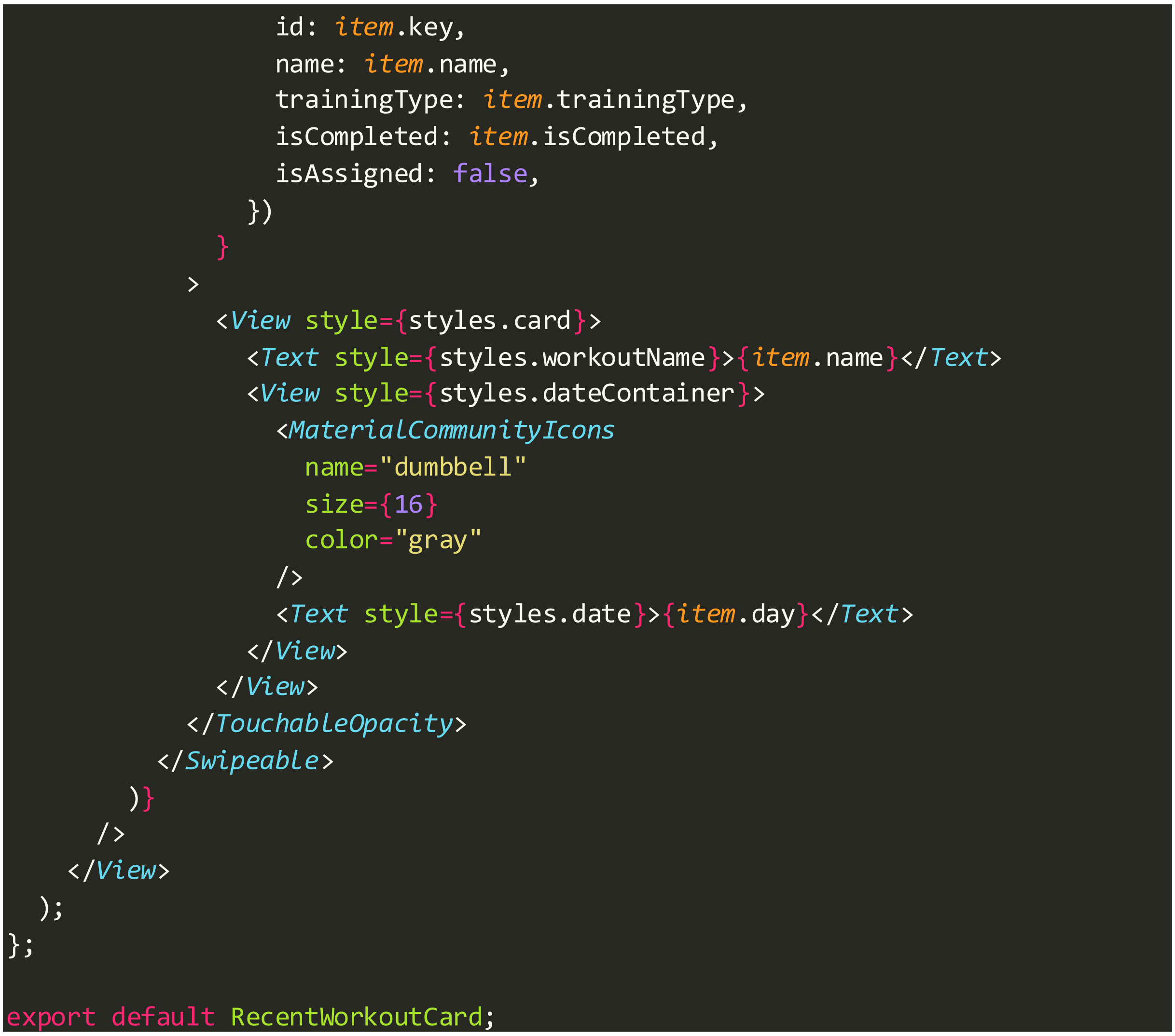
```
};

export default WorkoutScreen;
```

RecentWorkoutcards.js

```
//for swiping
const renderRightActions = (progress, dragX, item) => {

  const user = getAuth().currentUser;

  const trans = dragX.interpolate({
    inputRange: [0, 50, 100, 101],
    outputRange: [0, 0, 0, 1],
  });
  return (
    <TouchableOpacity
      style={styles.deleteBtn}
      onPress={() => {
        const docRef = doc(db, "users", user.uid, "workouts", item.key);
        deleteDoc(docRef);
      }}
    >
      <Ionicons name="trash-bin" size={40} color="red" />
      <Animated.Text
        style={[
          styles.deleteText,
          {
            transform: [{ translateX: trans }],
          },
        ]}
      >
        Delete
      </Animated.Text>
    </TouchableOpacity>
  );
};



const RecentWorkoutCard = () => {
  const user = getAuth().currentUser;
  const [workouts, setWorkouts] = useState([]);
  const navigation = useNavigation();

  // getting from firestore
  const GetWorkout = async () => {
    // get user
    if (user) {
```

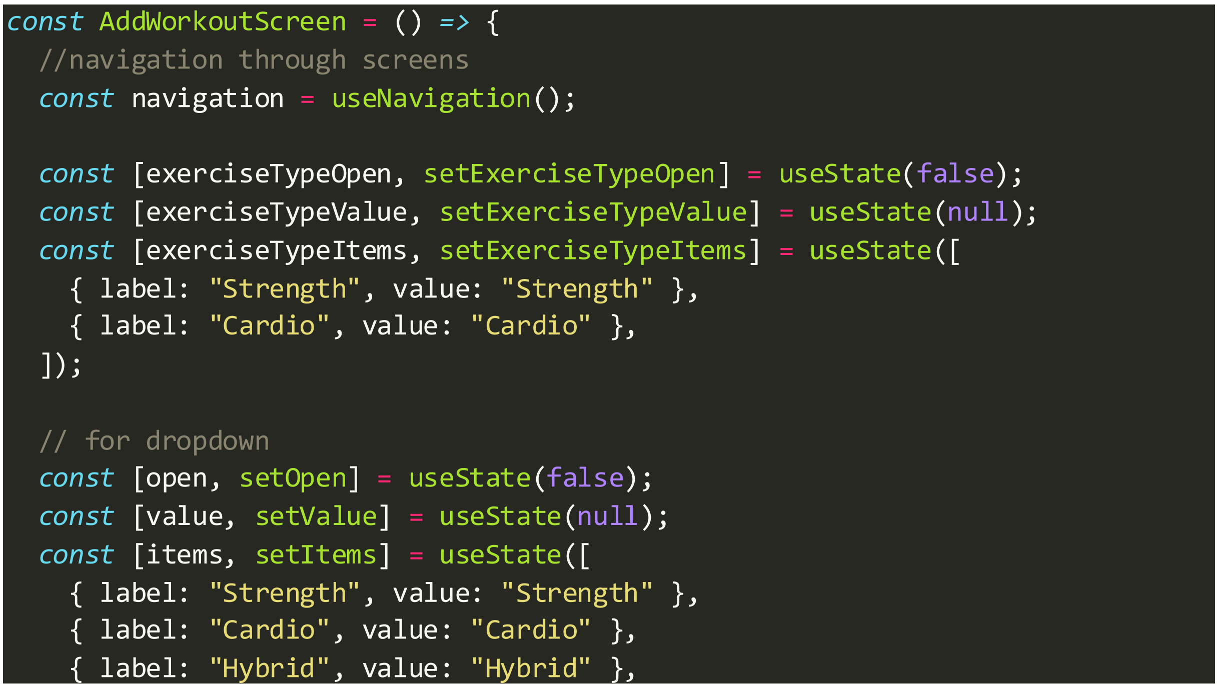


```
      const docRef = doc(db, "users", user.uid);
      const colRef = collection(docRef, "workouts");
      const q = await query(
        colRef,
        orderBy("createdAt", "desc"),
        limit(3),
        // Add a where clause to filter by documents created within the last 3
items
        where("createdAt", ">", new Date(Date.now() - 3 * 24 * 60 * 60 *
1000))
      );
      const subscriber = onSnapshot(q, (snapshot) => {
        let newWorkouts = [];
        snapshot.docs.forEach((doc) => {
          newWorkouts.push({ ...doc.data(), key: doc.id });
        });
        setWorkouts(newWorkouts);
        console.log(newWorkouts);
      });
      return () => subscriber();
    }
  };

  useEffect(() => {
    GetWorkout();
  }, []);

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Recent Workouts</Text>
      {/* List for rendering items */}
      <FlatList
        data={workouts}
        keyExtractor={(item) => item.key}
        style={{ flex: 1, overflow: "scroll", }}
        renderItem={({ item }) => (
          <Swipeable
            renderRightActions={(progress, dragX) =>
              renderRightActions(progress, dragX, item)
            }
          >
            <TouchableOpacity
              style={styles.cardContainer}
              onPress={() =>
                navigation.navigate("CreatedWorkout", {
                  day: item.day,
                  notes: item.notes,
                  exercises: item.exercises,
```

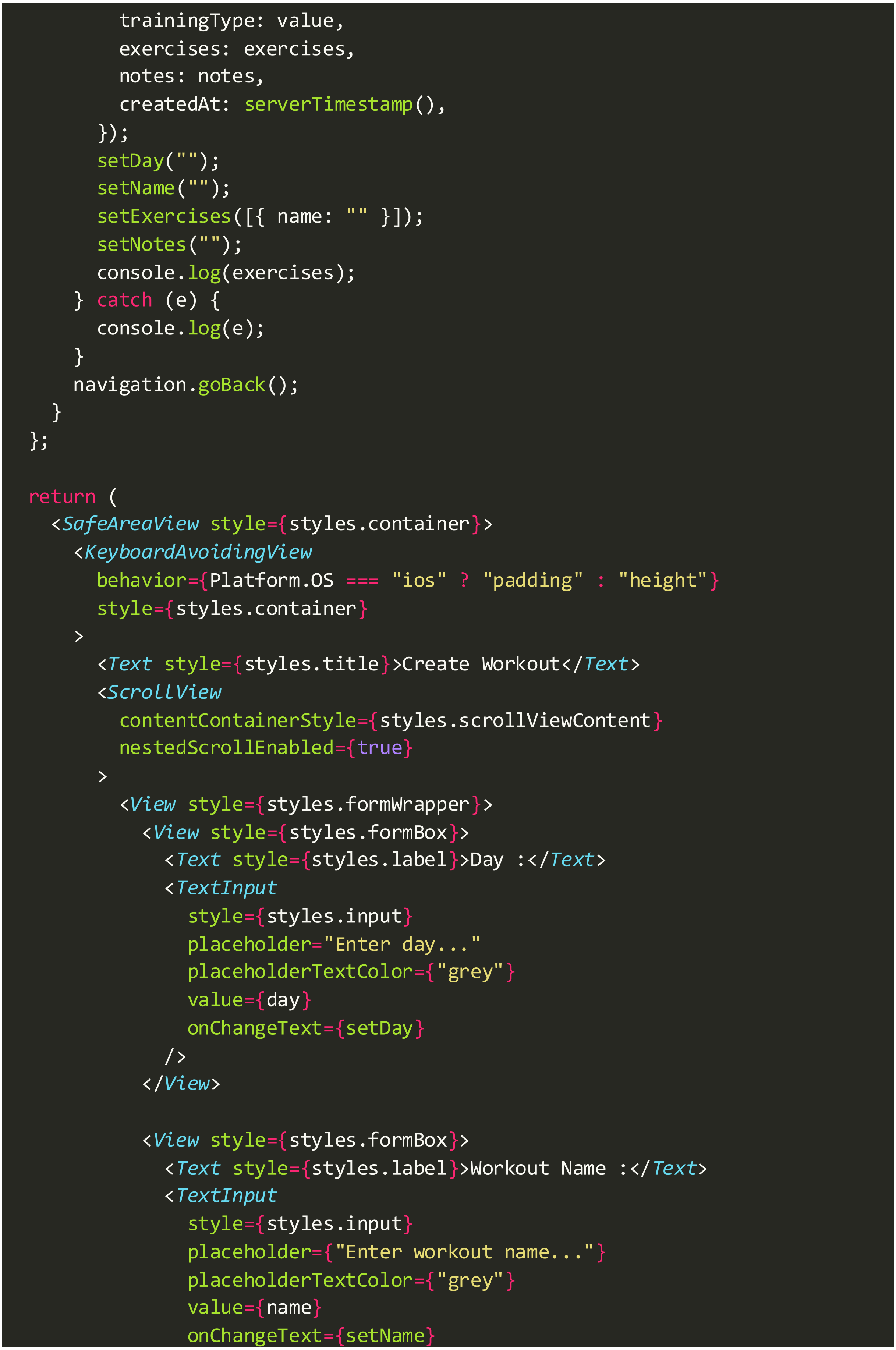


```
                  id: item.key,
                  name: item.name,
                  trainingType: item.trainingType,
                  isCompleted: item.isCompleted,
                  isAssigned: false,
                })
              }
            >
              <View style={styles.card}>
                <Text style={styles.workoutName}>{item.name}</Text>
                <View style={styles.dateContainer}>
                  <MaterialCommunityIcons
                    name="dumbbell"
                    size={16}
                    color="gray"
                  />
                  <Text style={styles.date}>{item.day}</Text>
                </View>
              </View>
            </TouchableOpacity>
          </Swipeable>
        )}
      />
    </View>
  );
};

export default RecentWorkoutCard;
```

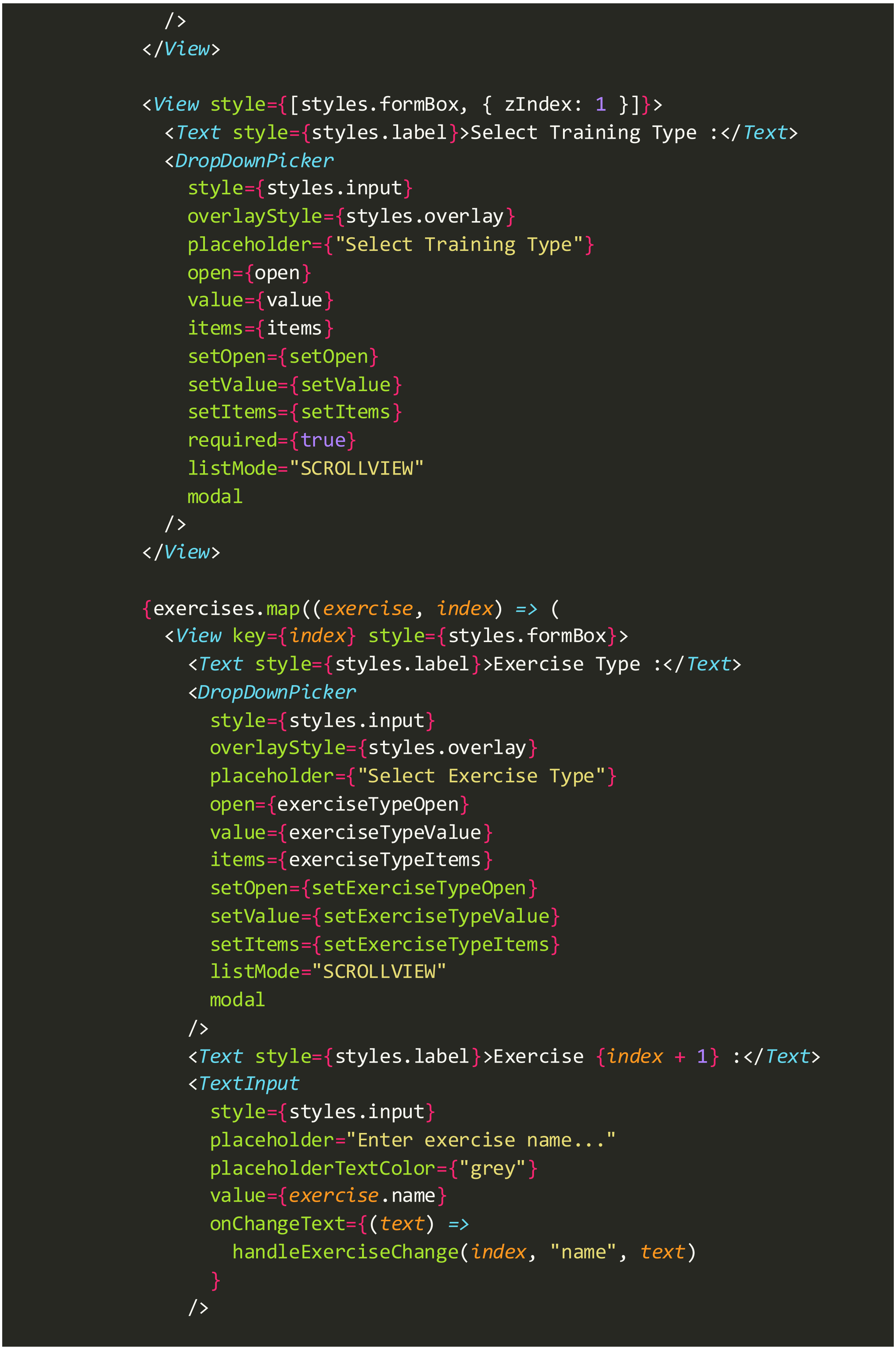
AddWorkoutScreen.js



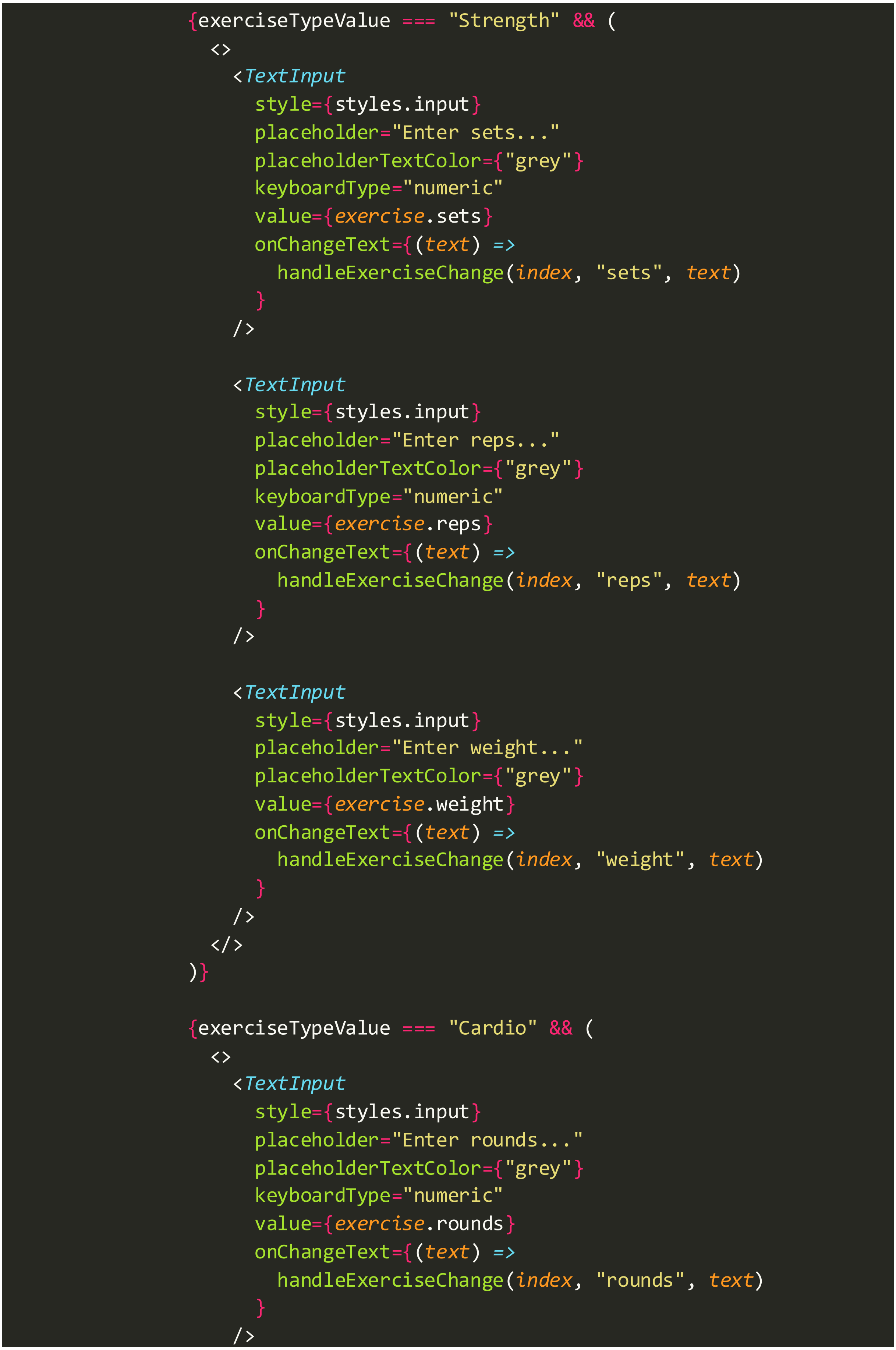
```
const AddWorkoutScreen = () => {
  //navigation through screens
  const navigation = useNavigation();

  const [exerciseTypeOpen, setExerciseTypeOpen] = useState(false);
  const [exerciseTypeValue, setExerciseTypeValue] = useState(null);
  const [exerciseTypeItems, setExerciseTypeItems] = useState([
    { label: "Strength", value: "Strength" },
    { label: "Cardio", value: "Cardio" },
  ]);

  // for dropdown
  const [open, setOpen] = useState(false);
  const [value, setValue] = useState(null);
  const [items, setItems] = useState([
    { label: "Strength", value: "Strength" },
    { label: "Cardio", value: "Cardio" },
    { label: "Hybrid", value: "Hybrid" },
  ]);

  const [day, setDay] = useState("");
  const [name, setName] = useState("");
  const [exercises, setExercises] = useState([
    { name: "", sets: "", reps: "", weight: "", videoLink: "", rounds: "",
    time: "", },
  ]);
  const [notes, setNotes] = useState("");

  const handleAddExercise = () => {
    setExercises([
      ...exercises,
      {
        name: "",
        sets: "",
        reps: "",
        weight: "",
        videoLink: "",
        rounds: "",
        time: "",
        exerciseType: exerciseTypeValue,
      },
    ]);
  };

  const handleRemoveExercise = (index) => {
    const newExercises = [...exercises];
    newExercises.splice(index, 1);
    setExercises(newExercises);
  };

  const handleExerciseChange = (index, field, value) => {
    const newExercises = [...exercises];
    newExercises[index][field] = value;
    setExercises(newExercises);
  };

  //Create in Firesotre
  const AddWorkout = async () => {
    const user = getAuth().currentUser;
    if (user) {
      try {
        const docRef = doc(db, "users", user.uid);
        const colRef = collection(docRef, "workouts");
        await addDoc(colRef, {
          day: day,
          name: name,
```



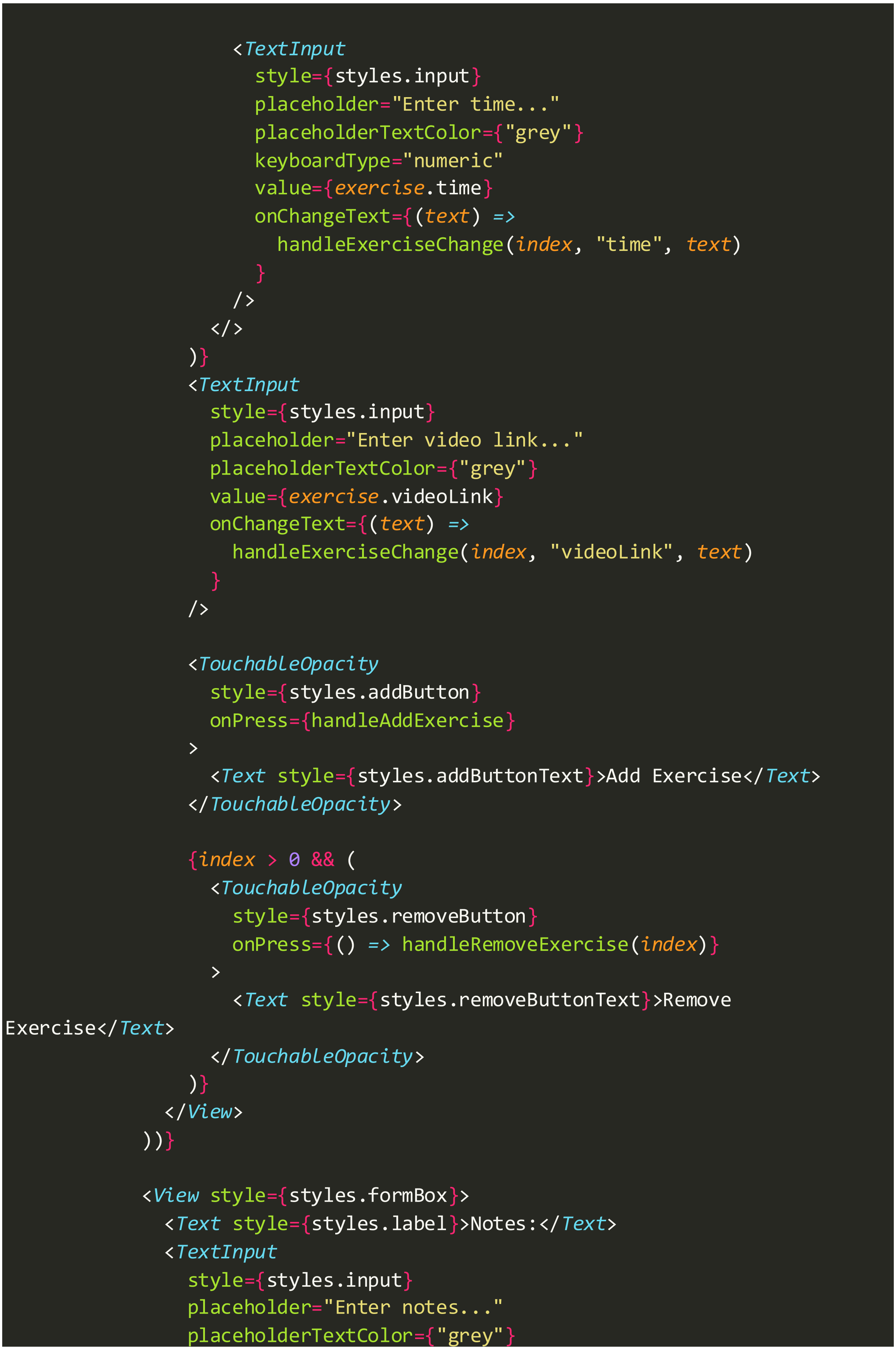
```
        trainingType: value,
        exercises: exercises,
        notes: notes,
        createdAt: serverTimestamp(),
      });
      setDay("");
      setName("");
      setExercises([{ name: "" }]);
      setNotes("");
      console.log(exercises);
    } catch (e) {
      console.log(e);
    }
    navigation.goBack();
  }
};

return (
  <SafeAreaView style={styles.container}>
    <KeyboardAvoidingView
      behavior={Platform.OS === "ios" ? "padding" : "height"}
      style={styles.container}
    >
      <Text style={styles.title}>Create Workout</Text>
      <ScrollView
        contentContainerStyle={styles.scrollViewContent}
        nestedScrollEnabled={true}
      >
        <View style={styles.formWrapper}>
          <View style={styles.formBox}>
            <Text style={styles.label}>Day :</Text>
            <TextInput
              style={styles.input}
              placeholder="Enter day..."
              placeholderTextColor={"grey"}
              value={day}
              onChangeText={setDay}
            />
          </View>

          <View style={styles.formBox}>
            <Text style={styles.label}>Workout Name :</Text>
            <TextInput
              style={styles.input}
              placeholder={"Enter workout name..."}
              placeholderTextColor={"grey"}
              value={name}
              onChangeText={setName}
```



```
  />
</View>

<View style={[styles.formBox, { zIndex: 1 }]}>
  <Text style={styles.label}>Select Training Type :</Text>
  <DropDownPicker
    style={styles.input}
    overlayStyle={styles.overlay}
    placeholder={"Select Training Type"}
    open={open}
    value={value}
    items={items}
    setOpen={setOpen}
    setValue={setValue}
    setItems={setItems}
    required={true}
    listMode="SCROLLVIEW"
    modal
  />
</View>

{exercises.map((exercise, index) => (
  <View key={index} style={styles.formBox}>
    <Text style={styles.label}>Exercise Type :</Text>
    <DropDownPicker
      style={styles.input}
      overlayStyle={styles.overlay}
      placeholder={"Select Exercise Type"}
      open={exerciseTypeOpen}
      value={exerciseTypeValue}
      items={exerciseTypeItems}
      setOpen={setExerciseTypeOpen}
      setValue={setExerciseTypeValue}
      setItems={setExerciseTypeItems}
      listMode="SCROLLVIEW"
      modal
    />
    <Text style={styles.label}>Exercise {index + 1} :</Text>
    <TextInput
      style={styles.input}
      placeholder="Enter exercise name..."
      placeholderTextColor={"grey"}
      value={exercise.name}
      onChangeText={(text) =>
        handleExerciseChange(index, "name", text)
      }
    />
```



```
{exerciseTypeValue === "Strength" && (
  <>
    <TextInput
      style={styles.input}
      placeholder="Enter sets..."
      placeholderTextColor={"grey"}
      keyboardType="numeric"
      value={exercise.sets}
      onChangeText={(text) =>
        handleExerciseChange(index, "sets", text)
      }
    />

    <TextInput
      style={styles.input}
      placeholder="Enter reps..."
      placeholderTextColor={"grey"}
      keyboardType="numeric"
      value={exercise.reps}
      onChangeText={(text) =>
        handleExerciseChange(index, "reps", text)
      }
    />

    <TextInput
      style={styles.input}
      placeholder="Enter weight..."
      placeholderTextColor={"grey"}
      value={exercise.weight}
      onChangeText={(text) =>
        handleExerciseChange(index, "weight", text)
      }
    />
  </>
)}

{exerciseTypeValue === "Cardio" && (
  <>
    <TextInput
      style={styles.input}
      placeholder="Enter rounds..."
      placeholderTextColor={"grey"}
      keyboardType="numeric"
      value={exercise.rounds}
      onChangeText={(text) =>
        handleExerciseChange(index, "rounds", text)
      }
    />
```



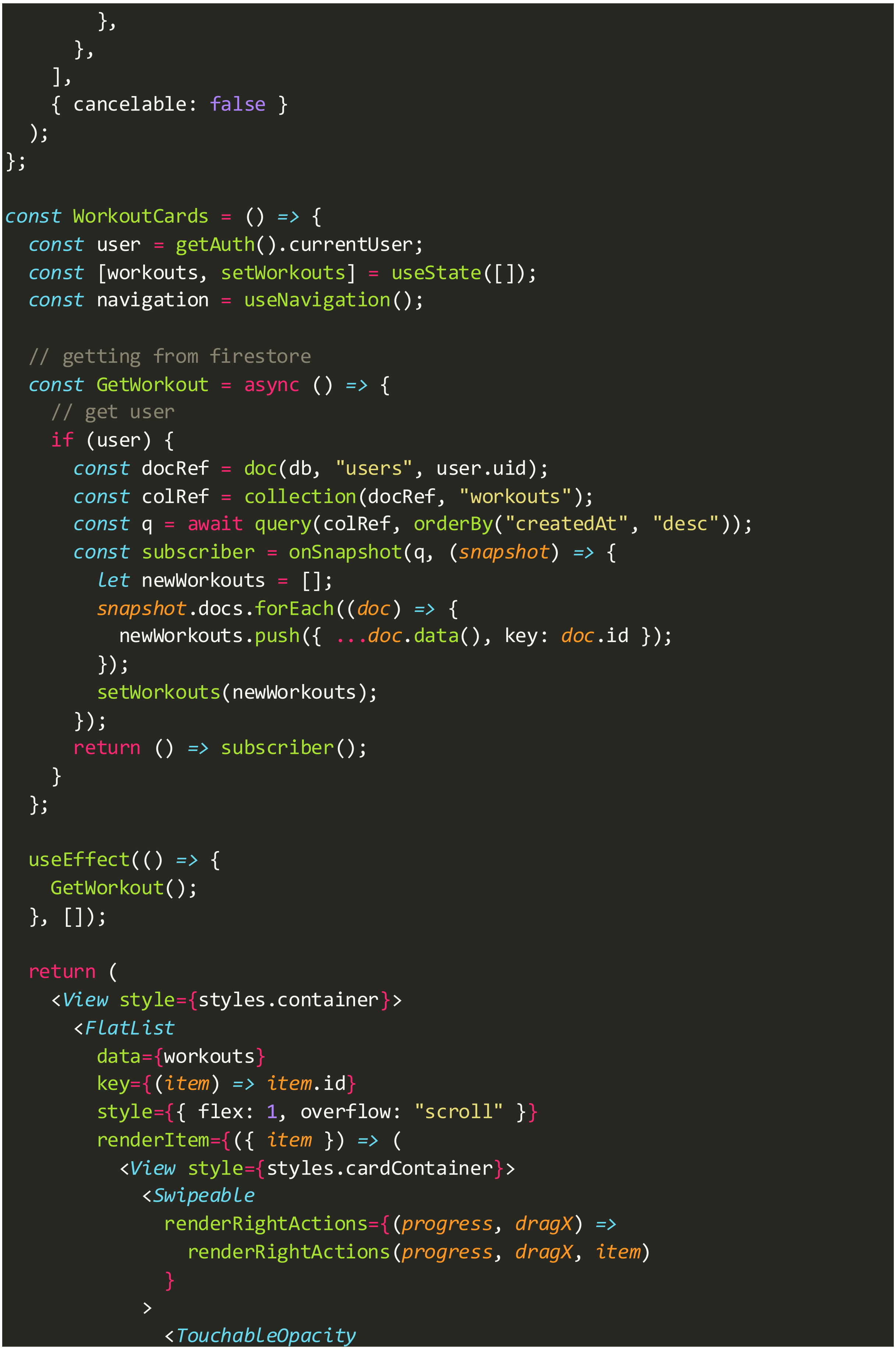
```
                    <TextInput
                      style={styles.input}
                      placeholder="Enter time..."
                      placeholderTextColor={"grey"}
                      keyboardType="numeric"
                      value={exercise.time}
                      onChangeText={(text) =>
                        handleExerciseChange(index, "time", text)
                      }
                    />
                  </>
                )}
                <TextInput
                  style={styles.input}
                  placeholder="Enter video link..."
                  placeholderTextColor={"grey"}
                  value={exercise.videoLink}
                  onChangeText={(text) =>
                    handleExerciseChange(index, "videoLink", text)
                  }
                />

                <TouchableOpacity
                  style={styles.addButton}
                  onPress={handleAddExercise}
                >
                  <Text style={styles.addButtonText}>Add Exercise</Text>
                </TouchableOpacity>

                {index > 0 && (
                  <TouchableOpacity
                    style={styles.removeButton}
                    onPress={() => handleRemoveExercise(index)}
                  >
                    <Text style={styles.removeButtonText}>Remove
Exercise</Text>
                  </TouchableOpacity>
                )}
              </View>
            ))}

            <View style={styles.formBox}>
              <Text style={styles.label}>Notes:</Text>
              <TextInput
                style={styles.input}
                placeholder="Enter notes..."
                placeholderTextColor={"grey"}
                multiline={true}
                value={notes}
                onChangeText={setNotes}
              />
            </View>

            <TouchableOpacity style={styles.addButton} onPress={AddWorkout}>
              <Text style={styles.addButtonText}>Submit</Text>
            </TouchableOpacity>
            <TouchableOpacity style={styles.cancelButton} onPress={() =>
navigation.goBack()}>
              <Text style={styles.addButtonText}>Cancel</Text>
            </TouchableOpacity>
          </View>
        </ScrollView>
      </KeyboardAvoidingView>
    </SafeAreaView>
  );
};

export default AddWorkoutScreen;
```

AllWorkoutScreen.js

```
const AllWorkoutScreen = () => {
  //getting the user data
  const user = getAuth().currentUser;

  //setting the state
  const [exercises, setExercises] = useState([]);
  const navigation = useNavigation();

  return (
    <SafeAreaView style={styles.heading}>
      <View style={styles.header}>
        {/* heading */}
        <TouchableOpacity onPress={() => navigation.goBack()}
style={{right:15}}>
          <Ionicons name="arrow-back" size={30} color="#0792F9" />
        </TouchableOpacity>
        <Text style={styles.heading}>Workout List</Text>
        <Pressable>
          <Entypo name="menu" size={30} color="black" />
        </Pressable>
      </View>

      <WorkoutCards/>

    </SafeAreaView>
  );
};

export default AllWorkoutScreen;
```

WorkoutCards.js

```
//for swiping
const renderRightActions = (progress, dragX, item) => {
  const user = getAuth().currentUser;

  const trans = dragX.interpolate({
    inputRange: [0, 50, 100, 101],
    outputRange: [0, 0, 0, 1],
  });
  return (
    <TouchableOpacity style={styles.deleteBtn} onPress={() =>
DeleteUser(item)}>
      <Ionicons name="trash-bin" size={40} color="red" />
      <Animated.Text
        style={[
          styles.deleteText,
          {
            transform: [{ translateX: trans }],
          },
        ]}
      >
        Delete
      </Animated.Text>
    </TouchableOpacity>
  );
};

const DeleteUser = (item) => {
  const user = getAuth().currentUser;

  Alert.alert(
    "Delete Account",
    "Are you sure you want to delete this workout?",
    [
      {
        text: "Cancel",
        style: "cancel",
      },
      {
        text: "Delete",
        onPress: () => {
          const docRef = doc(db, "users", user.uid, "workouts", item.key);
          deleteDoc(docRef);
```

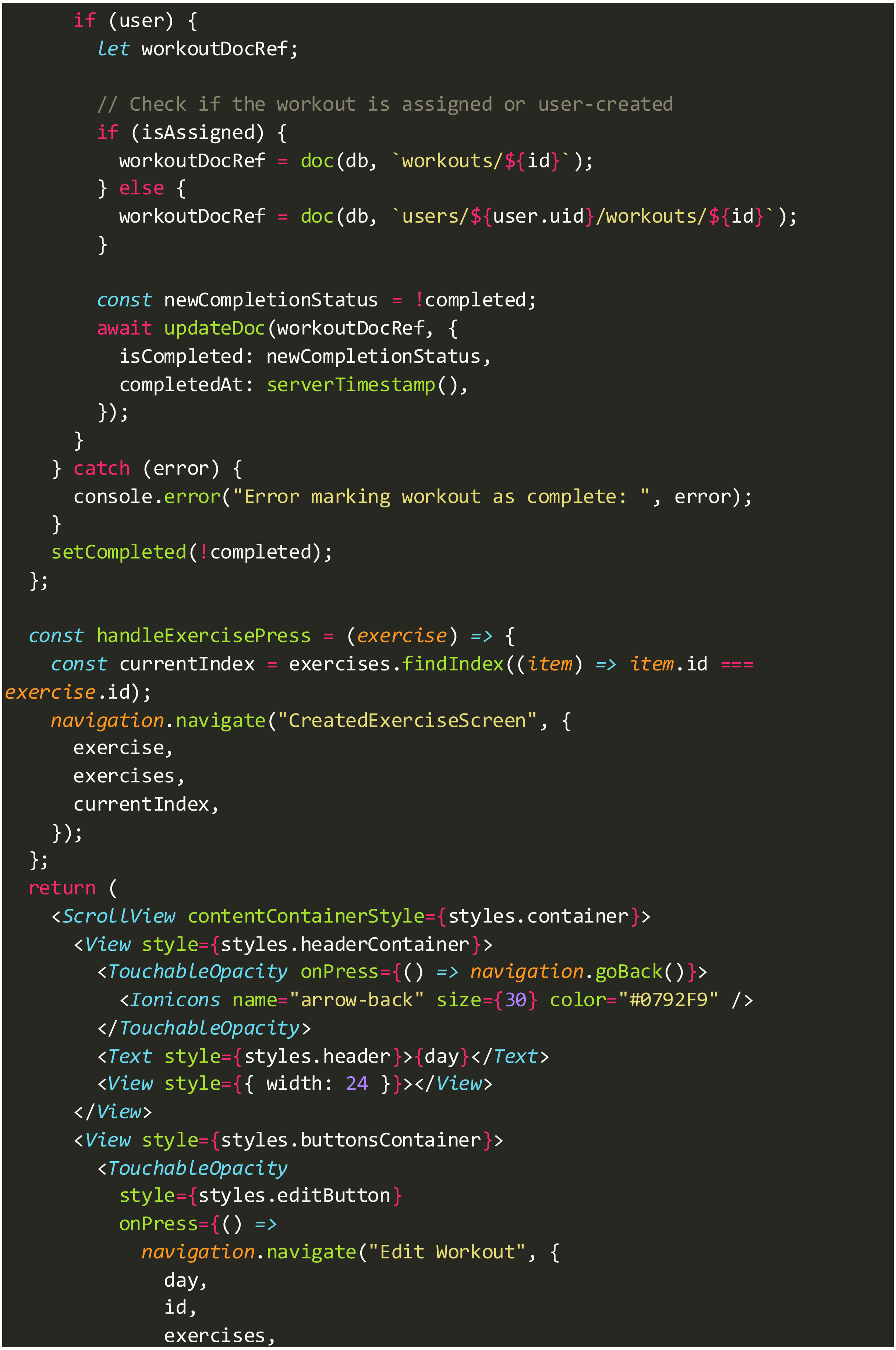


```
        },
      },
    ],
    { cancelable: false }
  );
};

const WorkoutCards = () => {
  const user = getAuth().currentUser;
  const [workouts, setWorkouts] = useState([]);
  const navigation = useNavigation();

  // getting from firestore
  const GetWorkout = async () => {
    // get user
    if (user) {
      const docRef = doc(db, "users", user.uid);
      const colRef = collection(docRef, "workouts");
      const q = await query(colRef, orderBy("createdAt", "desc"));
      const subscriber = onSnapshot(q, (snapshot) => {
        let newWorkouts = [];
        snapshot.docs.forEach((doc) => {
          newWorkouts.push({ ...doc.data(), key: doc.id });
        });
        setWorkouts(newWorkouts);
      });
      return () => subscriber();
    }
  };

  useEffect(() => {
    GetWorkout();
  }, []);

  return (
    <View style={styles.container}>
      <FlatList
        data={workouts}
        key={(item) => item.id}
        style={{ flex: 1, overflow: "scroll" }}
        renderItem={({ item }) => (
          <View style={styles.cardContainer}>
            <Swipeable
              renderRightActions={(progress, dragX) =>
                renderRightActions(progress, dragX, item)
              }
            >
              <TouchableOpacity
```

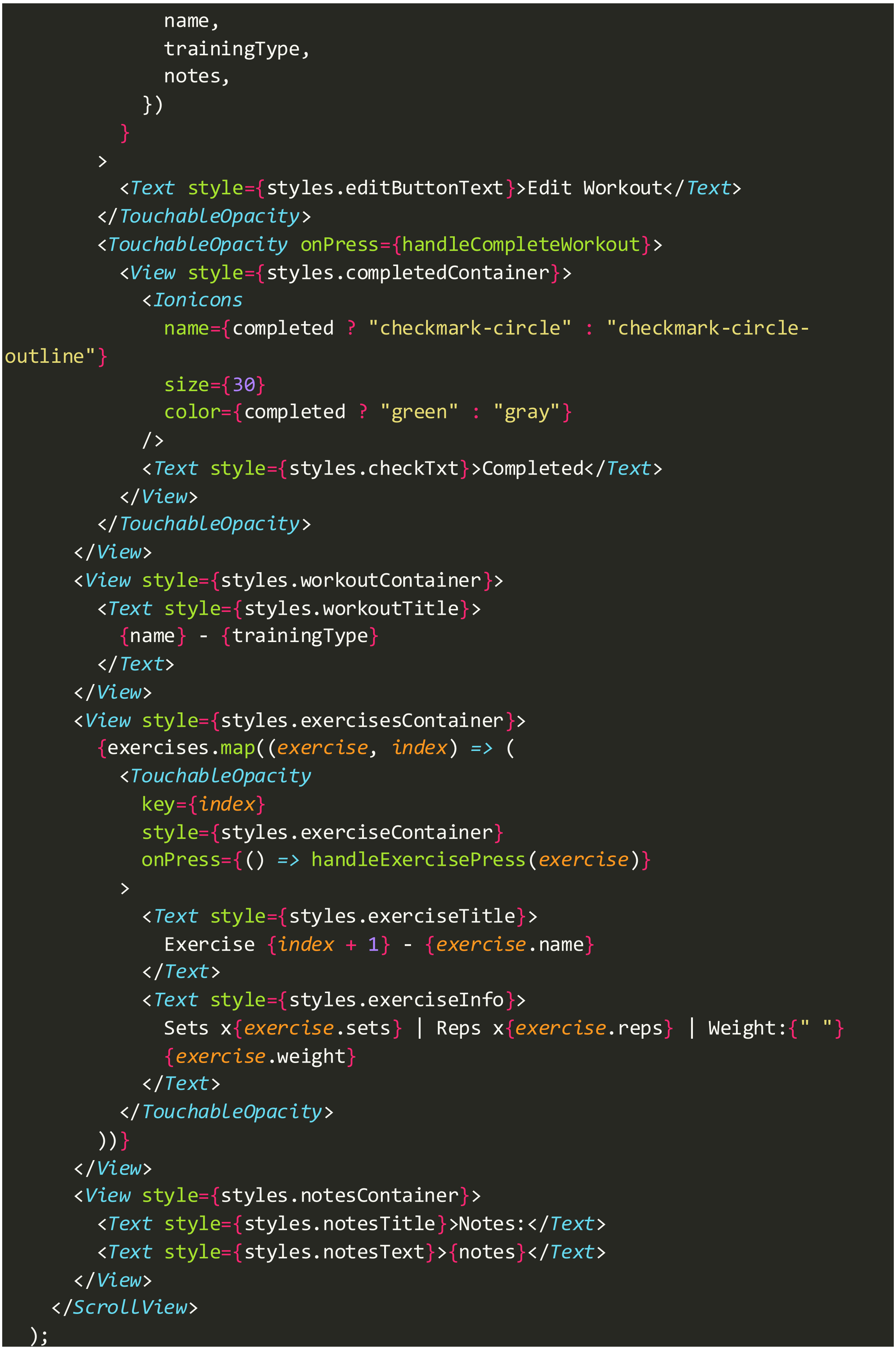
```
                style={styles.card}
                onPress={() =>
                  navigation.navigate("CreatedWorkout", {
                    day: item.day,
                    exercises: item.exercises,
                    id: item.key,
                    name: item.name,
                    trainingType: item.trainingType,
                    notes: item.notes,
                    isCompleted: item.isCompleted,
                    isAssigned: false,
                  })
                }
              >
                <View style={styles.header}>
                  <Text style={styles.workoutName}>{item.name}</Text>
                  <MaterialCommunityIcons
                    name="dumbbell"
                    size={24}
                    color="black"
                  />
                </View>
                <View style={styles.typeContainer}>
                  <Text style={styles.trainingType}>{item.trainingType}</Text>
                  <Text style={styles.day}>{item.day}</Text>
                </View>
              </TouchableOpacity>
            </Swipeable>
          </View>
        )}
      />
    </View>
  );
};

export default WorkoutCards;
```

CreatedWorkoutScreen.js

```
const CreatedWorkout = ({ route, navigation }) => {
  const { day, exercises, name, trainingType, notes, id, isCompleted,
isAssigned } = route.params;
  const [completed, setCompleted] = useState(isCompleted);
  const user = getAuth().currentUser;


  const handleCompleteWorkout = async () => {
    try {
```



```
      if (user) {
        let workoutDocRef;

        // Check if the workout is assigned or user-created
        if (isAssigned) {
          workoutDocRef = doc(db, `workouts/${id}`);
        } else {
          workoutDocRef = doc(db, `users/${user.uid}/workouts/${id}`);
        }

        const newCompletionStatus = !completed;
        await updateDoc(workoutDocRef, {
          isCompleted: newCompletionStatus,
          completedAt: serverTimestamp(),
        });
      }
    } catch (error) {
      console.error("Error marking workout as complete: ", error);
    }
    setCompleted(!completed);
  };

  const handleExercisePress = (exercise) => {
    const currentIndex = exercises.findIndex((item) => item.id ===
exercise.id);
    navigation.navigate("CreatedExerciseScreen", {
      exercise,
      exercises,
      currentIndex,
    });
  };
  return (
    <ScrollView contentContainerStyle={styles.container}>
      <View style={styles.headerContainer}>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Ionicons name="arrow-back" size={30} color="#0792F9" />
        </TouchableOpacity>
        <Text style={styles.header}>{day}</Text>
        <View style={{ width: 24 }}></View>
      </View>
      <View style={styles.buttonsContainer}>
        <TouchableOpacity
          style={styles.editButton}
          onPress={() =>
            navigation.navigate("Edit Workout", {
              day,
              id,
              exercises,
```



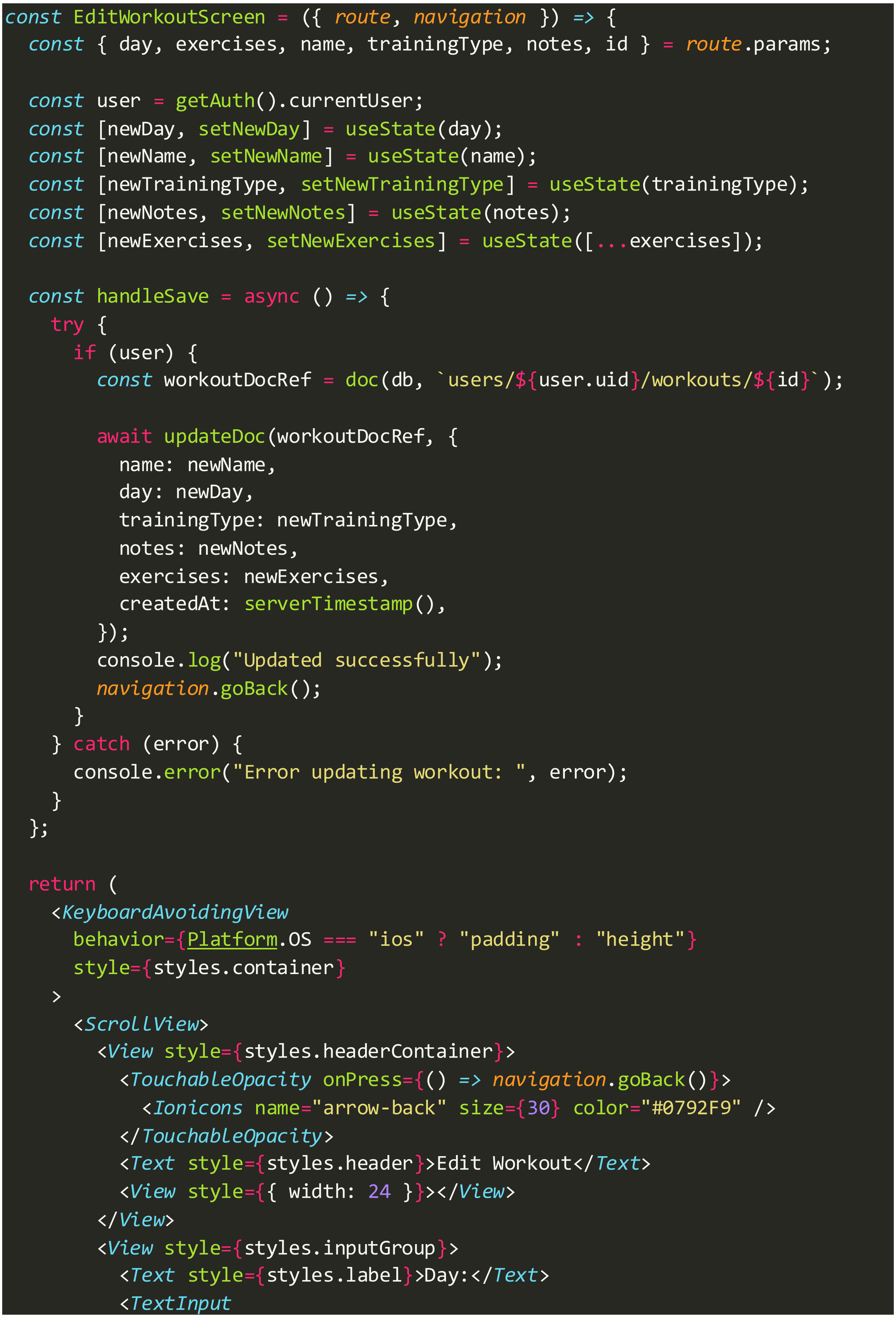
```
              name,
              trainingType,
              notes,
            })
          }
        >
          <Text style={styles.editButtonText}>Edit Workout</Text>
        </TouchableOpacity>
        <TouchableOpacity onPress={handleCompleteWorkout}>
          <View style={styles.completedContainer}>
            <Ionicons
              name={completed ? "checkmark-circle" : "checkmark-circle-
outline"}
              size={30}
              color={completed ? "green" : "gray"}
            />
            <Text style={styles.checkTxt}>Completed</Text>
          </View>
        </TouchableOpacity>
      </View>
      <View style={styles.workoutContainer}>
        <Text style={styles.workoutTitle}>
          {name} - {trainingType}
        </Text>
      </View>
      <View style={styles.exercisesContainer}>
        {exercises.map((exercise, index) => (
          <TouchableOpacity
            key={index}
            style={styles.exerciseContainer}
            onPress={() => handleExercisePress(exercise)}
          >
            <Text style={styles.exerciseTitle}>
              Exercise {index + 1} - {exercise.name}
            </Text>
            <Text style={styles.exerciseInfo}>
              Sets x{exercise.sets} | Reps x{exercise.reps} | Weight:{" "}
              {exercise.weight}
            </Text>
          </TouchableOpacity>
        ))}
      </View>
      <View style={styles.notesContainer}>
        <Text style={styles.notesTitle}>Notes:</Text>
        <Text style={styles.notesText}>{notes}</Text>
      </View>
    </ScrollView>
  );
```

```
};

export default CreatedWorkout;
```

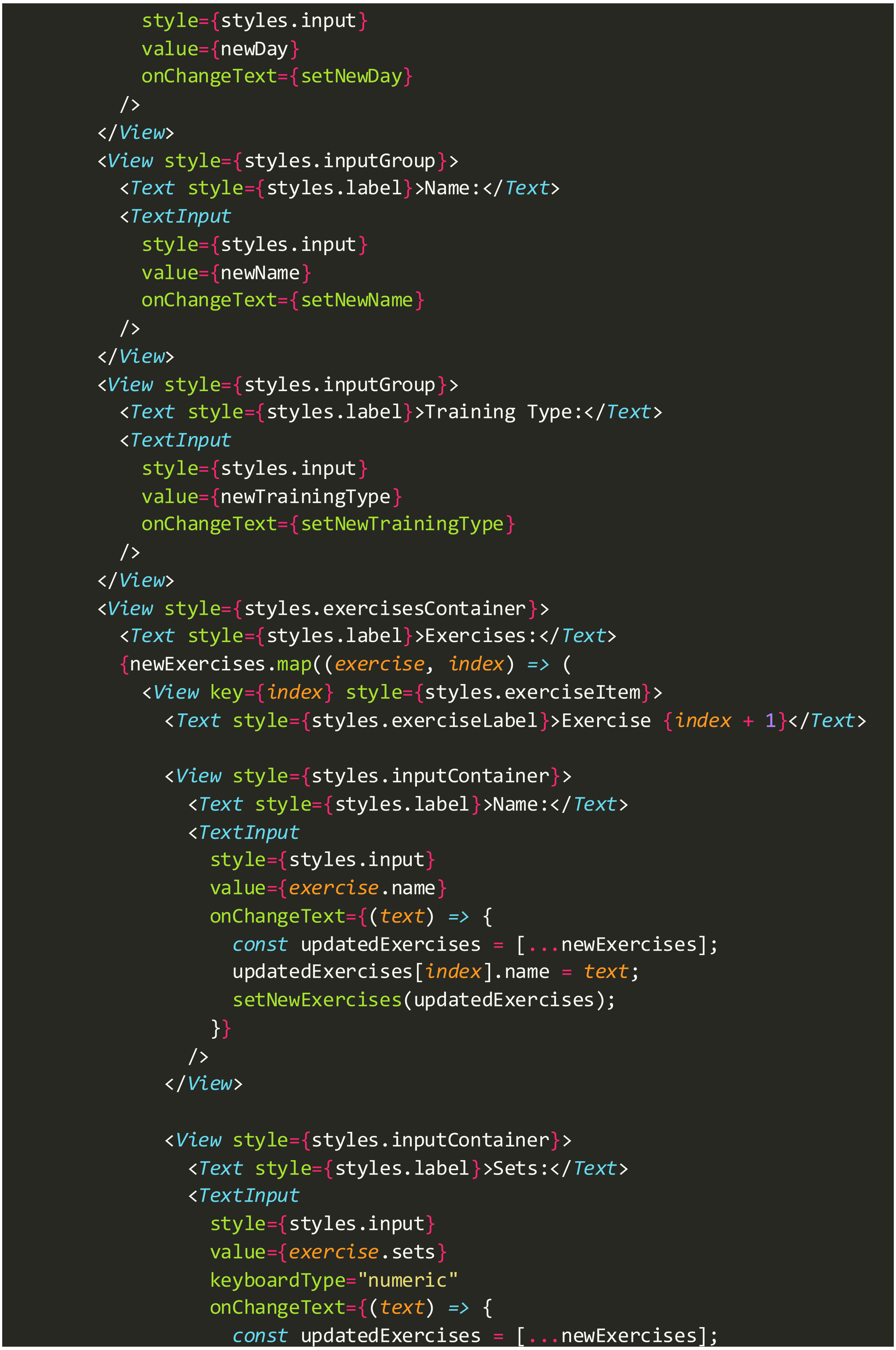
CreatedExerciseScreen.js

```
const CreatedExerciseScreen = ({ route }) => {
  //gets params from last page
  const { exercise, exercises, currentIndex } = route.params;
  const navigation = useNavigation();
  const [videoId, setVideoId] = useState("");

  const extractId = () => {
    if (exercise.videoLink) {
      const extractedId = exercise.videoLink.slice(-11);
      setVideoId(extractedId);
    }
  };

  useEffect(() => {
    extractId();
  }, []);

  return (
    <SafeAreaView style={styles.container}>
      {videoId ? (
        <YoutubePlayer height={300} play={false} videoId={videoId} />
      ) : (
        <Text style={styles.title}></Text>
      )}

      <Text style={styles.title}>{exercise.name}</Text>
      <Text style={styles.sets}>x{exercise.sets} Sets</Text>
      <Text style={styles.sets}>x{exercise.reps} Reps</Text>
      <View style={styles.buttonContainer}>
        <TouchableOpacity
          style={styles.exitBtn}
          onPress={() => navigation.goBack()}
        >
          <Text
            style={styles.btnText}>BACK</Text>
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  );
};

export default CreatedExerciseScreen;
```

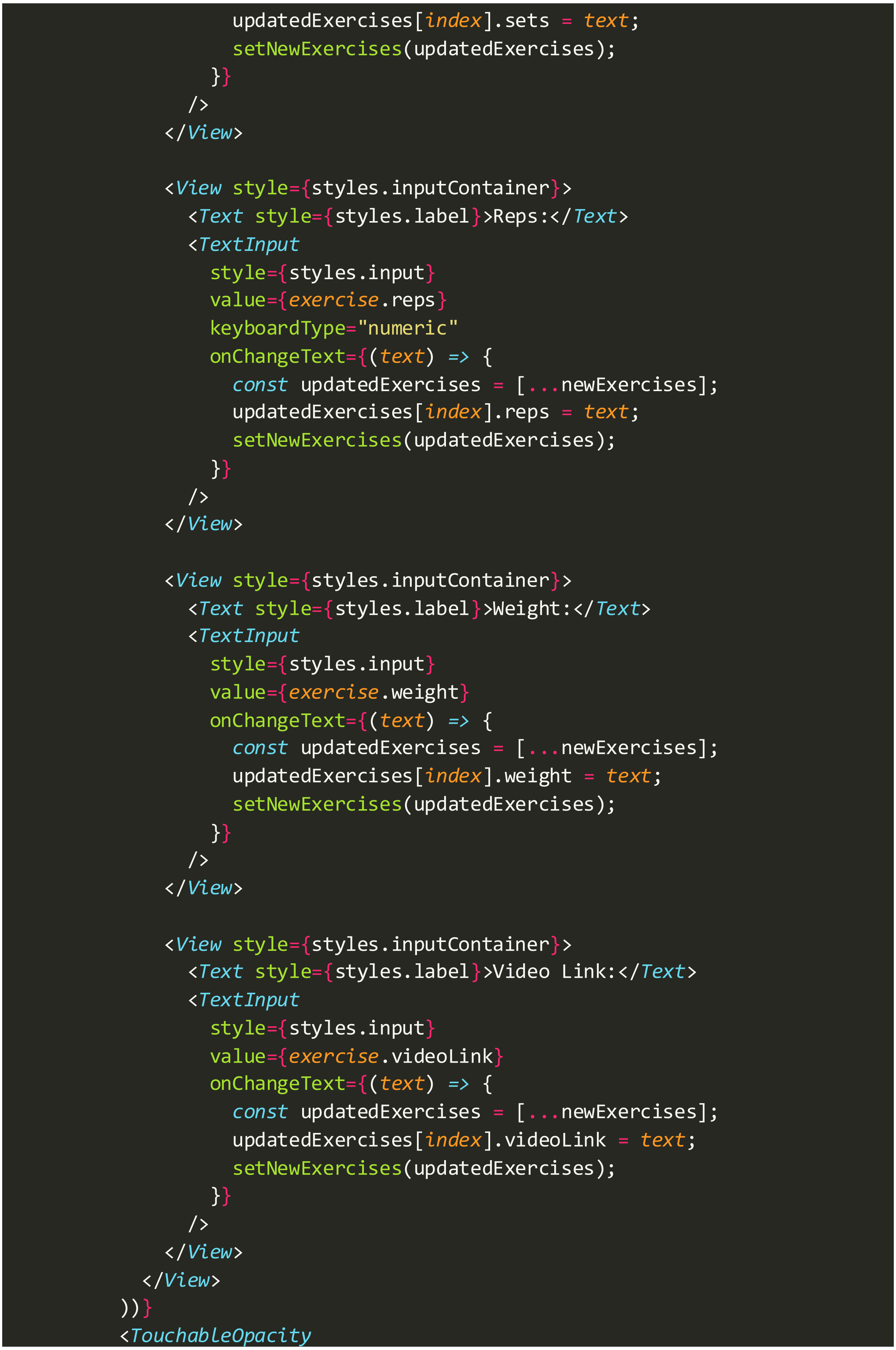
EditWorkout.js



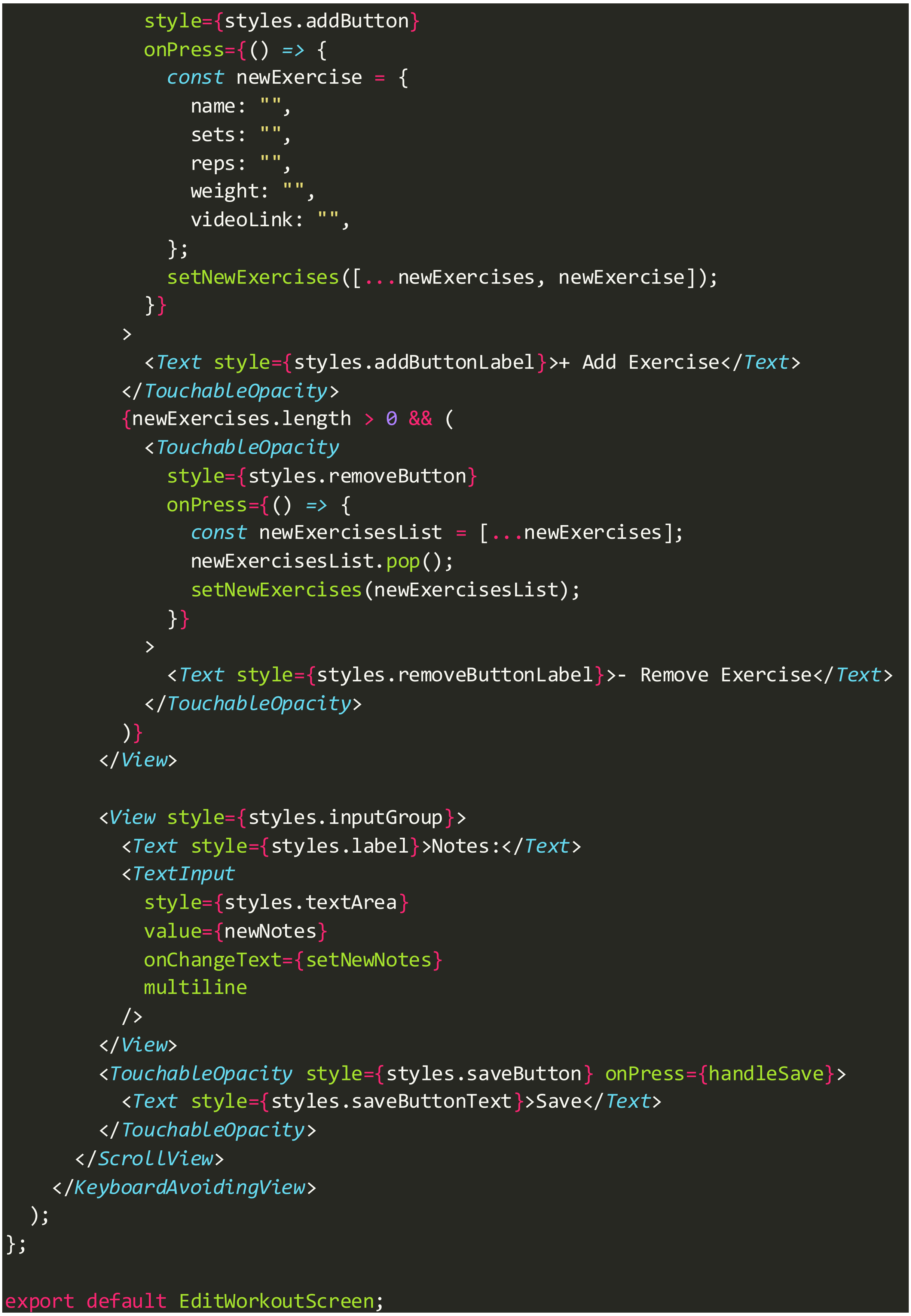
```
const EditWorkoutScreen = ({ route, navigation }) => {
  const { day, exercises, name, trainingType, notes, id } = route.params;

  const user = getAuth().currentUser;
  const [newDay, setNewDay] = useState(day);
  const [newName, setNewName] = useState(name);
  const [newTrainingType, setNewTrainingType] = useState(trainingType);
  const [newNotes, setNewNotes] = useState(notes);
  const [newExercises, setNewExercises] = useState([...exercises]);

  const handleSave = async () => {
    try {
      if (user) {
        const workoutDocRef = doc(db, `users/${user.uid}/workouts/${id}`);

        await updateDoc(workoutDocRef, {
          name: newName,
          day: newDay,
          trainingType: newTrainingType,
          notes: newNotes,
          exercises: newExercises,
          createdAt: serverTimestamp(),
        });
        console.log("Updated successfully");
        navigation.goBack();
      }
    } catch (error) {
      console.error("Error updating workout: ", error);
    }
  };

  return (
    <KeyboardAvoidingView
      behavior={Platform.OS === "ios" ? "padding" : "height"}
      style={styles.container}
    >
      <ScrollView>
        <View style={styles.headerContainer}>
          <TouchableOpacity onPress={() => navigation.goBack()}>
            <Ionicons name="arrow-back" size={30} color="#0792F9" />
          </TouchableOpacity>
          <Text style={styles.header}>Edit Workout</Text>
          <View style={{ width: 24 }}></View>
        </View>
        <View style={styles.inputGroup}>
          <Text style={styles.label}>Day:</Text>
          <TextInput
```



```
            style={styles.input}
            value={newDay}
            onChangeText={setNewDay}
          />
        </View>
        <View style={styles.inputGroup}>
          <Text style={styles.label}>Name:</Text>
          <TextInput
            style={styles.input}
            value={newName}
            onChangeText={setNewName}
          />
        </View>
        <View style={styles.inputGroup}>
          <Text style={styles.label}>Training Type:</Text>
          <TextInput
            style={styles.input}
            value={newTrainingType}
            onChangeText={setNewTrainingType}
          />
        </View>
        <View style={styles.exercisesContainer}>
          <Text style={styles.label}>Exercises:</Text>
          {newExercises.map((exercise, index) => (
            <View key={index} style={styles.exerciseItem}>
              <Text style={styles.exerciseLabel}>Exercise {index + 1}</Text>

              <View style={styles.inputContainer}>
                <Text style={styles.label}>Name:</Text>
                <TextInput
                  style={styles.input}
                  value={exercise.name}
                  onChangeText={(text) => {
                    const updatedExercises = [...newExercises];
                    updatedExercises[index].name = text;
                    setNewExercises(updatedExercises);
                  }}
                />
              </View>

              <View style={styles.inputContainer}>
                <Text style={styles.label}>Sets:</Text>
                <TextInput
                  style={styles.input}
                  value={exercise.sets}
                  keyboardType="numeric"
                  onChangeText={(text) => {
                    const updatedExercises = [...newExercises];
```

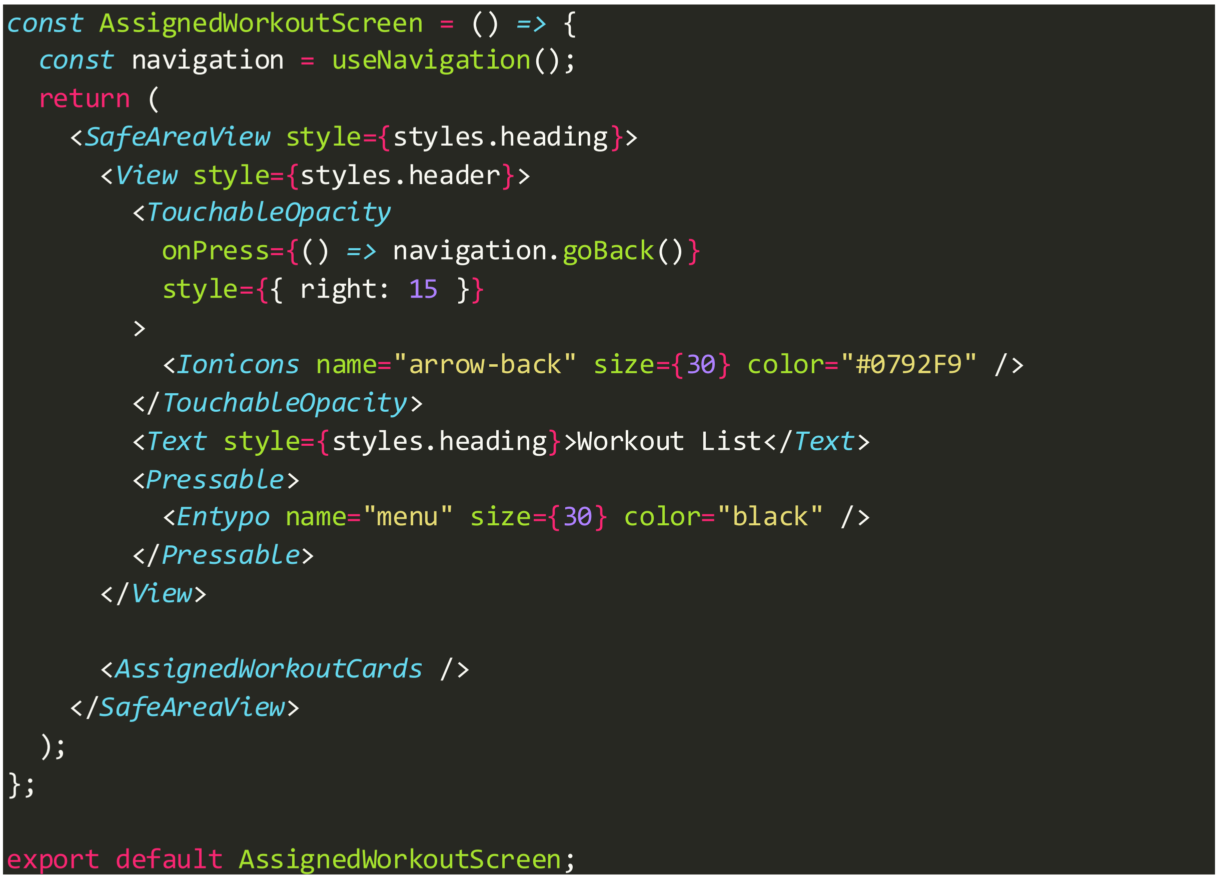


```
                    updatedExercises[index].sets = text;
                    setNewExercises(updatedExercises);
                  }}
                />
              </View>

              <View style={styles.inputContainer}>
                <Text style={styles.label}>Reps:</Text>
                <TextInput
                  style={styles.input}
                  value={exercise.reps}
                  keyboardType="numeric"
                  onChangeText={(text) => {
                    const updatedExercises = [...newExercises];
                    updatedExercises[index].reps = text;
                    setNewExercises(updatedExercises);
                  }}
                />
              </View>

              <View style={styles.inputContainer}>
                <Text style={styles.label}>Weight:</Text>
                <TextInput
                  style={styles.input}
                  value={exercise.weight}
                  onChangeText={(text) => {
                    const updatedExercises = [...newExercises];
                    updatedExercises[index].weight = text;
                    setNewExercises(updatedExercises);
                  }}
                />
              </View>

              <View style={styles.inputContainer}>
                <Text style={styles.label}>Video Link:</Text>
                <TextInput
                  style={styles.input}
                  value={exercise.videoLink}
                  onChangeText={(text) => {
                    const updatedExercises = [...newExercises];
                    updatedExercises[index].videoLink = text;
                    setNewExercises(updatedExercises);
                  }}
                />
              </View>
            </View>
          ))}
          <TouchableOpacity
```



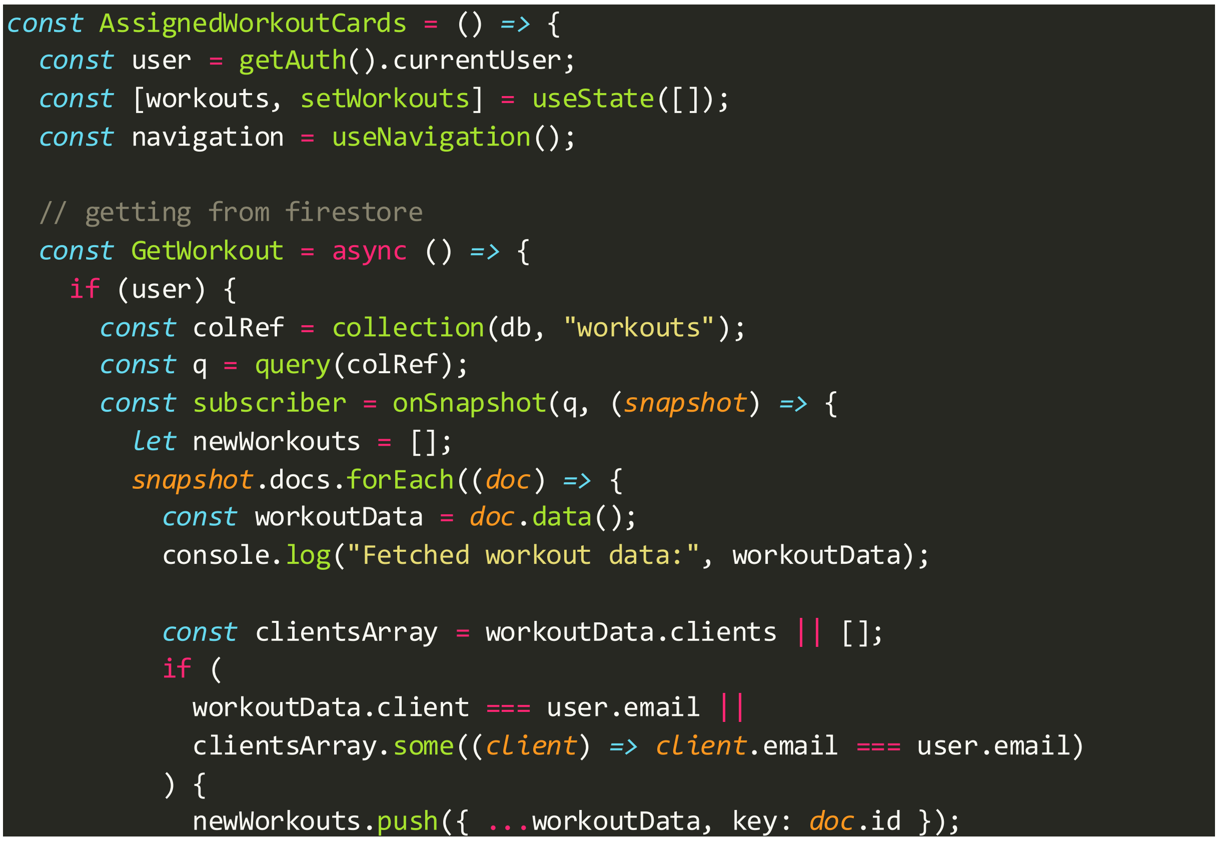
```
            style={styles.addButton}
            onPress={() => {
              const newExercise = {
                name: "",
                sets: "",
                reps: "",
                weight: "",
                videoLink: "",
              };
              setNewExercises([...newExercises, newExercise]);
            }}
          >
            <Text style={styles.addButtonLabel}>+ Add Exercise</Text>
          </TouchableOpacity>
          {newExercises.length > 0 && (
            <TouchableOpacity
              style={styles.removeButton}
              onPress={() => {
                const newExercisesList = [...newExercises];
                newExercisesList.pop();
                setNewExercises(newExercisesList);
              }}
            >
              <Text style={styles.removeButtonLabel}>- Remove Exercise</Text>
            </TouchableOpacity>
          )}
        </View>

        <View style={styles.inputGroup}>
          <Text style={styles.label}>Notes:</Text>
          <TextInput
            style={styles.textArea}
            value={newNotes}
            onChangeText={setNewNotes}
            multiline
          />
        </View>
        <TouchableOpacity style={styles.saveButton} onPress={handleSave}>
          <Text style={styles.saveButtonText}>Save</Text>
        </TouchableOpacity>
      </ScrollView>
    </KeyboardAvoidingView>
  );
};

export default EditWorkoutScreen;
```

AssignedWorkouts.js

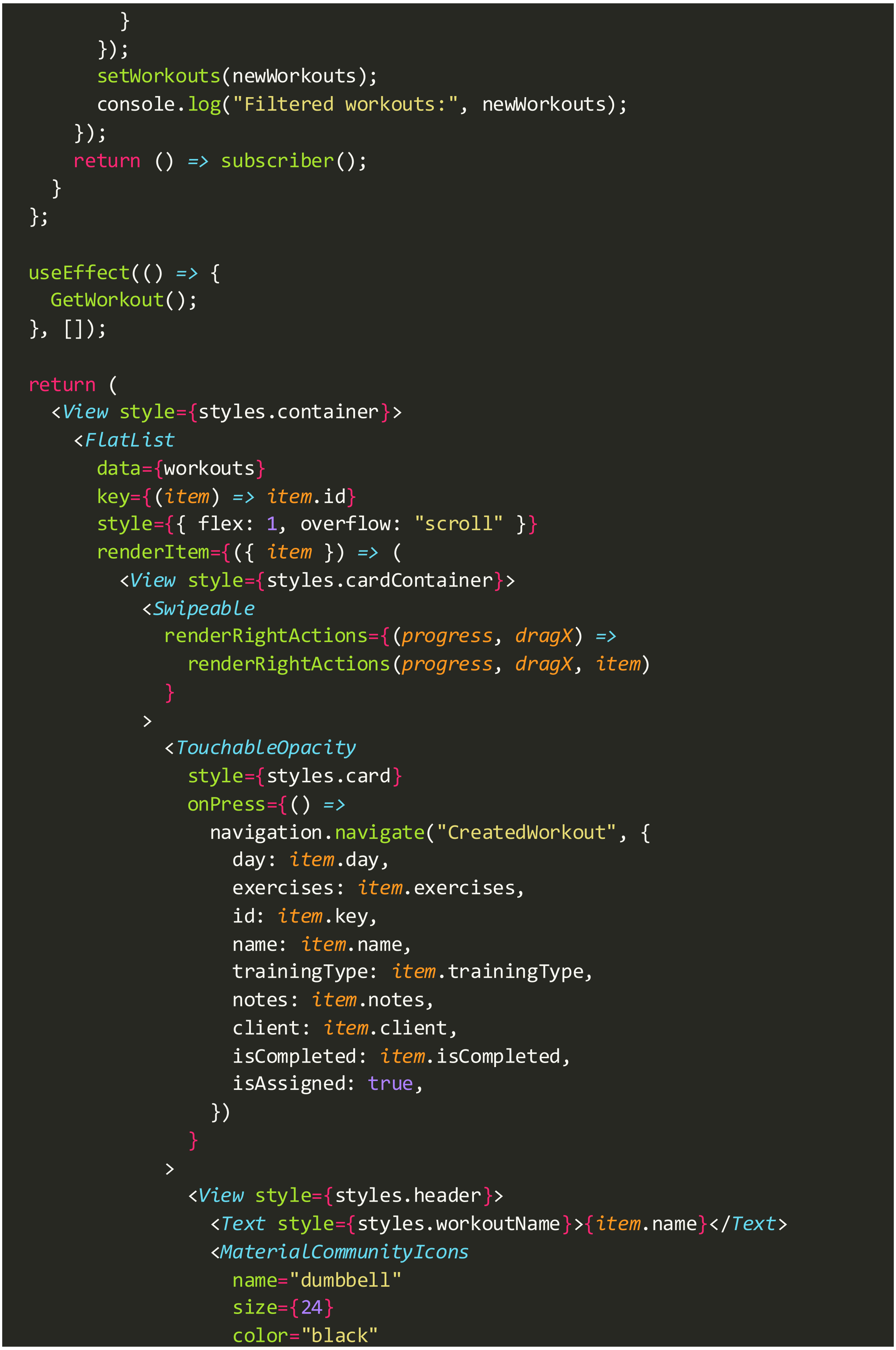


```
const AssignedWorkoutScreen = () => {
  const navigation = useNavigation();
  return (
    <SafeAreaView style={styles.heading}>
      <View style={styles.header}>
        <TouchableOpacity
          onPress={() => navigation.goBack()}
          style={{ right: 15 }}
        >
          <Ionicons name="arrow-back" size={30} color="#0792F9" />
        </TouchableOpacity>
        <Text style={styles.heading}>Workout List</Text>
        <Pressable>
          <Entypo name="menu" size={30} color="black" />
        </Pressable>
      </View>

      <AssignedWorkoutCards />
    </SafeAreaView>
  );
};

export default AssignedWorkoutScreen;
```

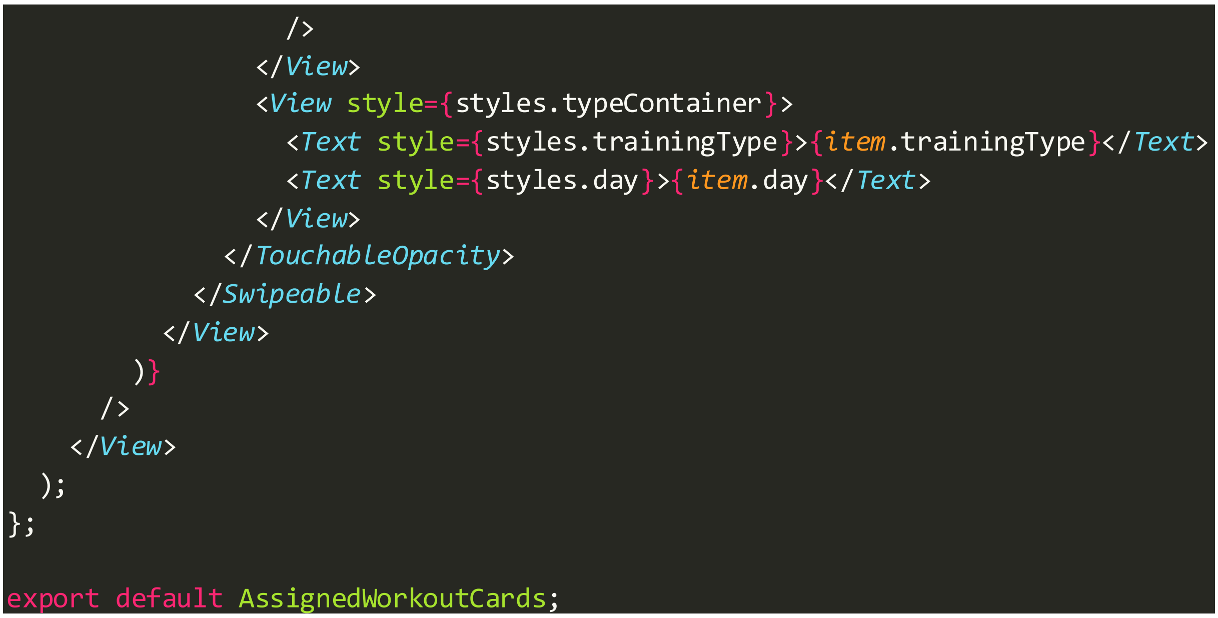
AssignedWorkoutCards.js



```
const AssignedWorkoutCards = () => {
  const user = getAuth().currentUser;
  const [workouts, setWorkouts] = useState([]);
  const navigation = useNavigation();

  // getting from firestore
  const GetWorkout = async () => {
    if (user) {
      const colRef = collection(db, "workouts");
      const q = query(colRef);
      const subscriber = onSnapshot(q, (snapshot) => {
        let newWorkouts = [];
        snapshot.docs.forEach((doc) => {
          const workoutData = doc.data();
          console.log("Fetched workout data:", workoutData);

          const clientsArray = workoutData.clients || [];
          if (
            workoutData.client === user.email ||
            clientsArray.some((client) => client.email === user.email)
          ) {
            newWorkouts.push({ ...workoutData, key: doc.id });
```

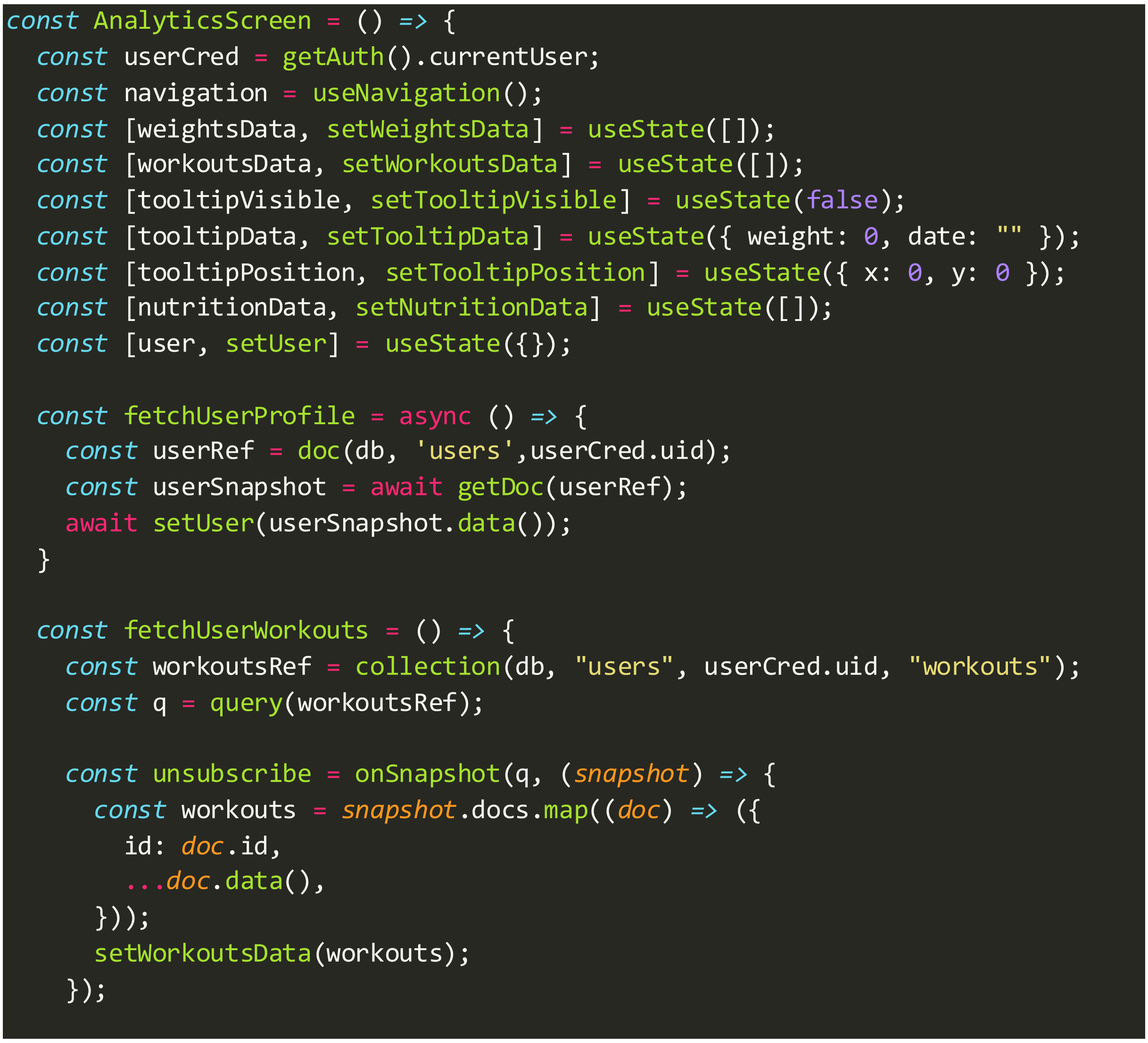


```
        }
      });
      setWorkouts(newWorkouts);
      console.log("Filtered workouts:", newWorkouts);
    });
    return () => subscriber();
  }
};

useEffect(() => {
  GetWorkout();
}, []);

return (
  <View style={styles.container}>
    <FlatList
      data={workouts}
      key={(item) => item.id}
      style={{ flex: 1, overflow: "scroll" }}
      renderItem={({ item }) => (
        <View style={styles.cardContainer}>
          <Swipeable
            renderRightActions={(progress, dragX) =>
              renderRightActions(progress, dragX, item)
            }
          >
            <TouchableOpacity
              style={styles.card}
              onPress={() =>
                navigation.navigate("CreatedWorkout", {
                  day: item.day,
                  exercises: item.exercises,
                  id: item.key,
                  name: item.name,
                  trainingType: item.trainingType,
                  notes: item.notes,
                  client: item.client,
                  isCompleted: item.isCompleted,
                  isAssigned: true,
                })
              }
            >
              <View style={styles.header}>
                <Text style={styles.workoutName}>{item.name}</Text>
                <MaterialCommunityIcons
                  name="dumbbell"
                  size={24}
                  color="black"
```

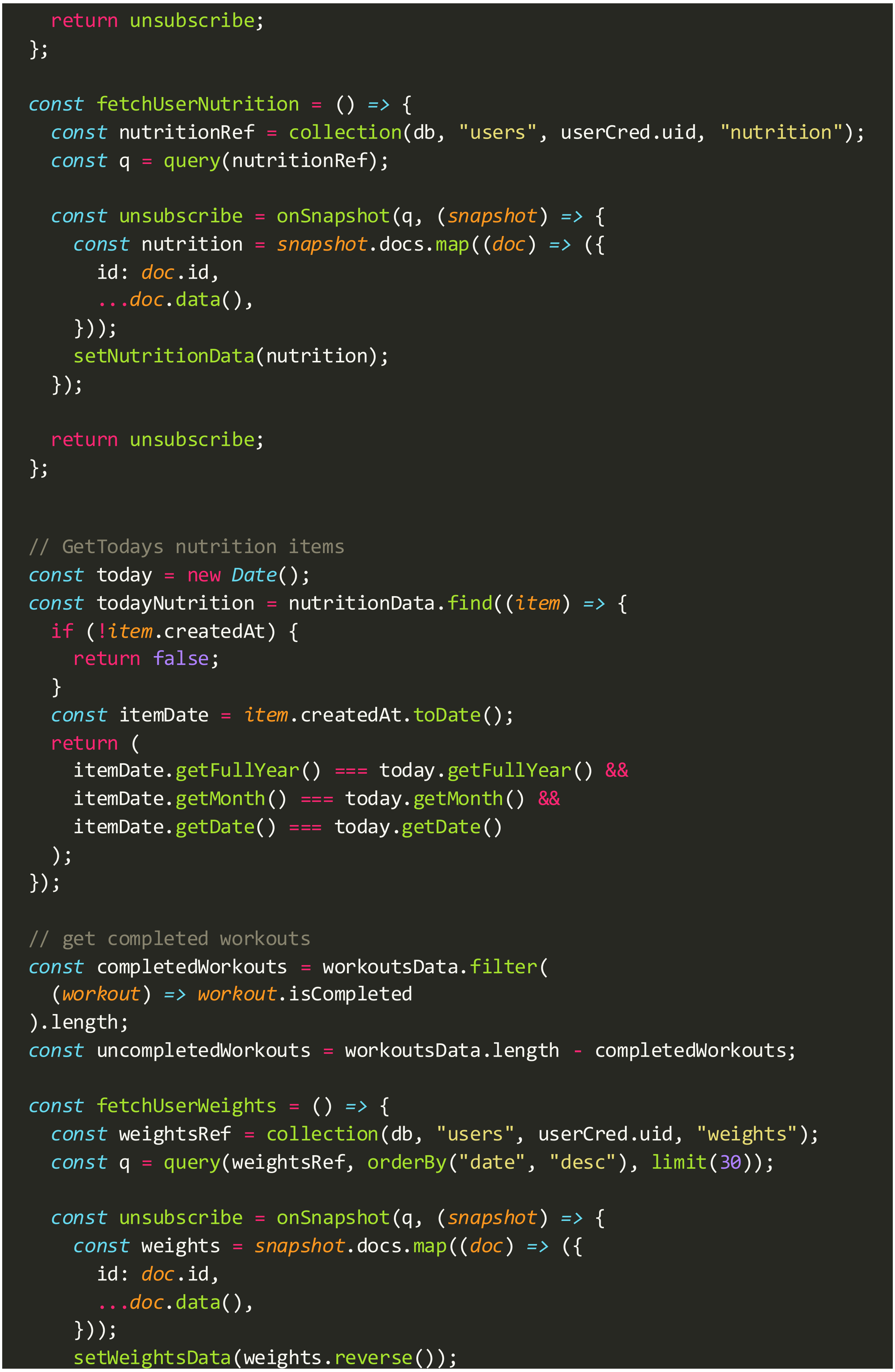


```
                  />
                </View>
                <View style={styles.typeContainer}>
                  <Text style={styles.trainingType}>{item.trainingType}</Text>
                  <Text style={styles.day}>{item.day}</Text>
                </View>
              </TouchableOpacity>
            </Swipeable>
          </View>
        )}
      />
    </View>
  );
};

export default AssignedWorkoutCards;
```

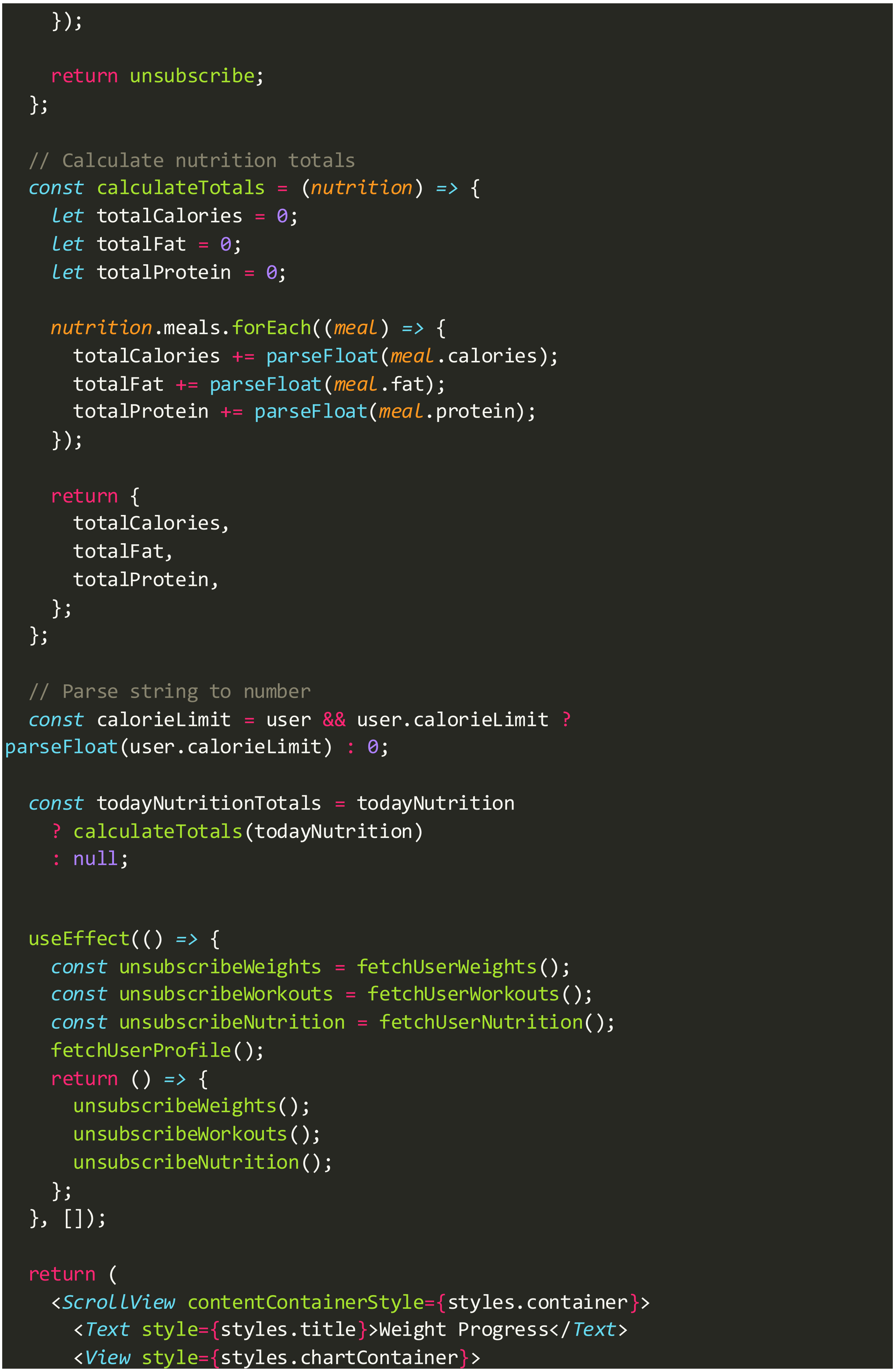
AnalyticsScreen.js



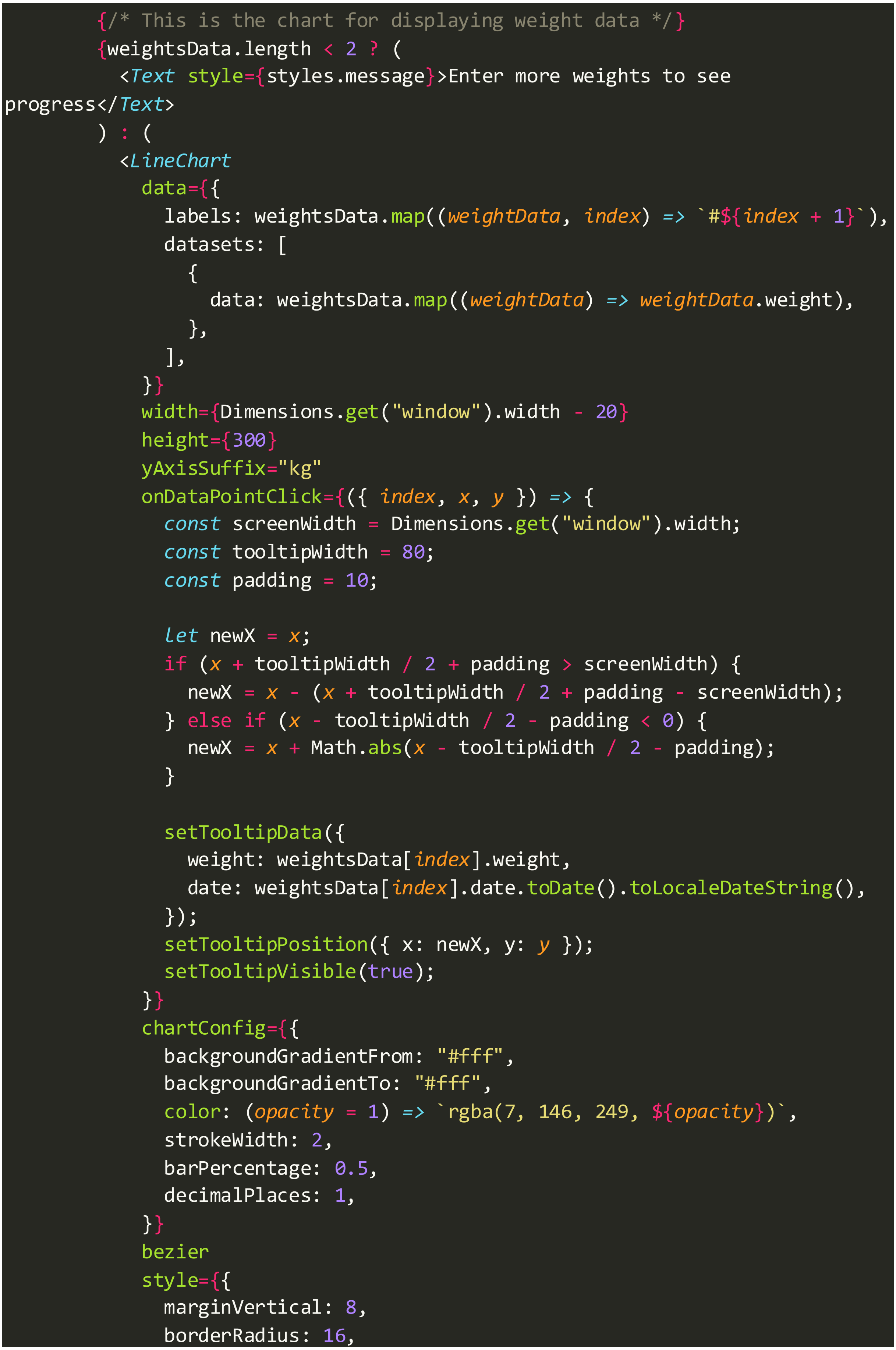
```
const AnalyticsScreen = () => {
  const userCred = getAuth().currentUser;
  const navigation = useNavigation();
  const [weightsData, setWeightsData] = useState([]);
  const [workoutsData, setWorkoutsData] = useState([]);
  const [tooltipVisible, setTooltipVisible] = useState(false);
  const [tooltipData, setTooltipData] = useState({ weight: 0, date: "" });
  const [tooltipPosition, setTooltipPosition] = useState({ x: 0, y: 0 });
  const [nutritionData, setNutritionData] = useState([]);
  const [user, setUser] = useState({});

  const fetchUserProfile = async () => {
    const userRef = doc(db, 'users',userCred.uid);
    const userSnapshot = await getDoc(userRef);
    await setUser(userSnapshot.data());
  }

  const fetchUserWorkouts = () => {
    const workoutsRef = collection(db, "users", userCred.uid, "workouts");
    const q = query(workoutsRef);

    const unsubscribe = onSnapshot(q, (snapshot) => {
      const workouts = snapshot.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
      setWorkoutsData(workouts);
    });
```



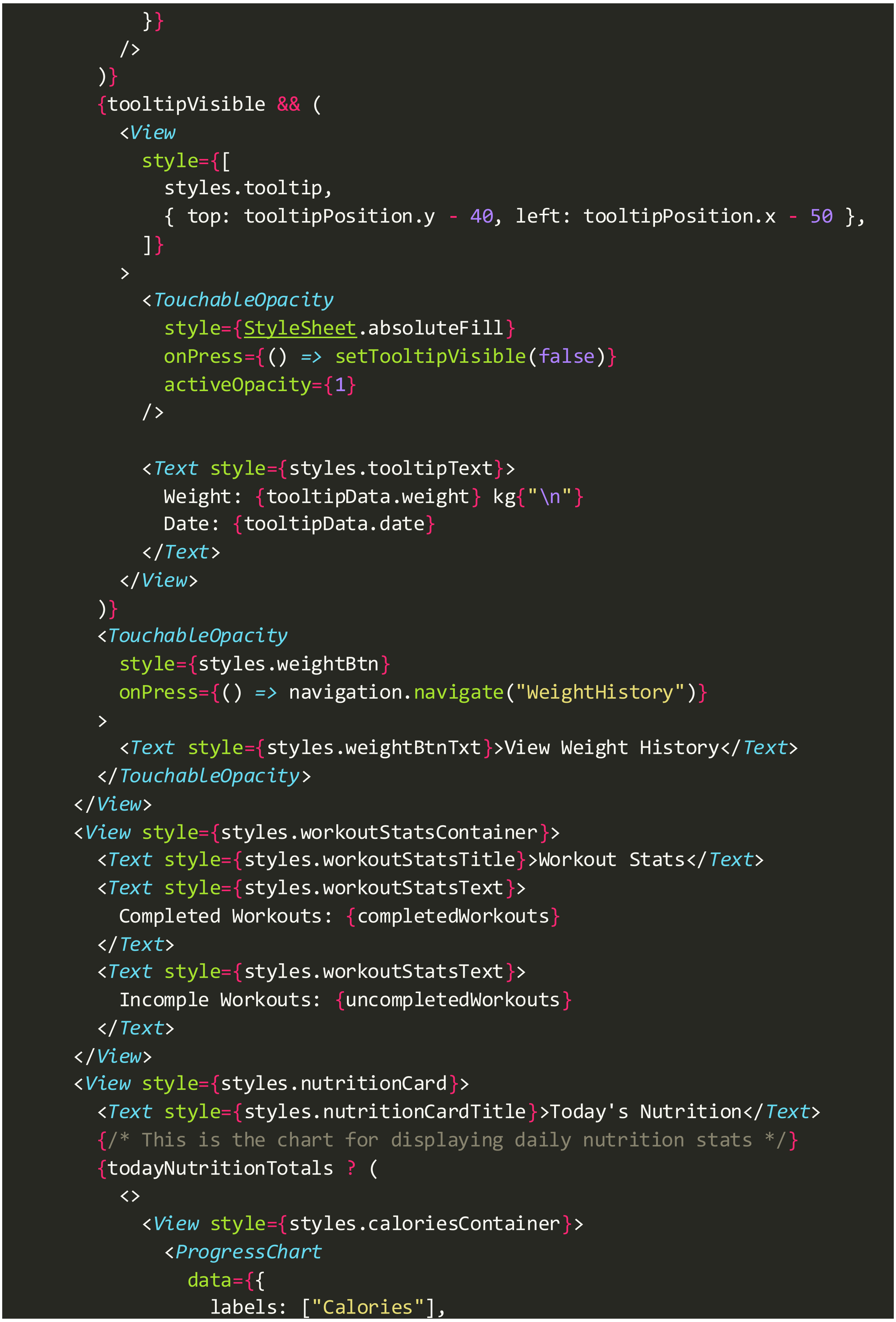
```
  return unsubscribe;
};

const fetchUserNutrition = () => {
  const nutritionRef = collection(db, "users", userCred.uid, "nutrition");
  const q = query(nutritionRef);

  const unsubscribe = onSnapshot(q, (snapshot) => {
    const nutrition = snapshot.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    setNutritionData(nutrition);
  });

  return unsubscribe;
};


// GetTodays nutrition items
const today = new Date();
const todayNutrition = nutritionData.find((item) => {
  if (!item.createdAt) {
    return false;
  }
  const itemDate = item.createdAt.toDate();
  return (
    itemDate.getFullYear() === today.getFullYear() &&
    itemDate.getMonth() === today.getMonth() &&
    itemDate.getDate() === today.getDate()
  );
});

// get completed workouts
const completedWorkouts = workoutsData.filter(
  (workout) => workout.isCompleted
).length;
const uncompletedWorkouts = workoutsData.length - completedWorkouts;

const fetchUserWeights = () => {
  const weightsRef = collection(db, "users", userCred.uid, "weights");
  const q = query(weightsRef, orderBy("date", "desc"), limit(30));

  const unsubscribe = onSnapshot(q, (snapshot) => {
    const weights = snapshot.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    setWeightsData(weights.reverse());
```



```
    });

    return unsubscribe;
  };

  // Calculate nutrition totals
  const calculateTotals = (nutrition) => {
    let totalCalories = 0;
    let totalFat = 0;
    let totalProtein = 0;

    nutrition.meals.forEach((meal) => {
      totalCalories += parseFloat(meal.calories);
      totalFat += parseFloat(meal.fat);
      totalProtein += parseFloat(meal.protein);
    });

    return {
      totalCalories,
      totalFat,
      totalProtein,
    };
  };

  // Parse string to number
  const calorieLimit = user && user.calorieLimit ?
parseFloat(user.calorieLimit) : 0;

  const todayNutritionTotals = todayNutrition
    ? calculateTotals(todayNutrition)
    : null;


  useEffect(() => {
    const unsubscribeWeights = fetchUserWeights();
    const unsubscribeWorkouts = fetchUserWorkouts();
    const unsubscribeNutrition = fetchUserNutrition();
    fetchUserProfile();
    return () => {
      unsubscribeWeights();
      unsubscribeWorkouts();
      unsubscribeNutrition();
    };
  }, []);

  return (
    <ScrollView contentContainerStyle={styles.container}>
      <Text style={styles.title}>Weight Progress</Text>
      <View style={styles.chartContainer}>
```



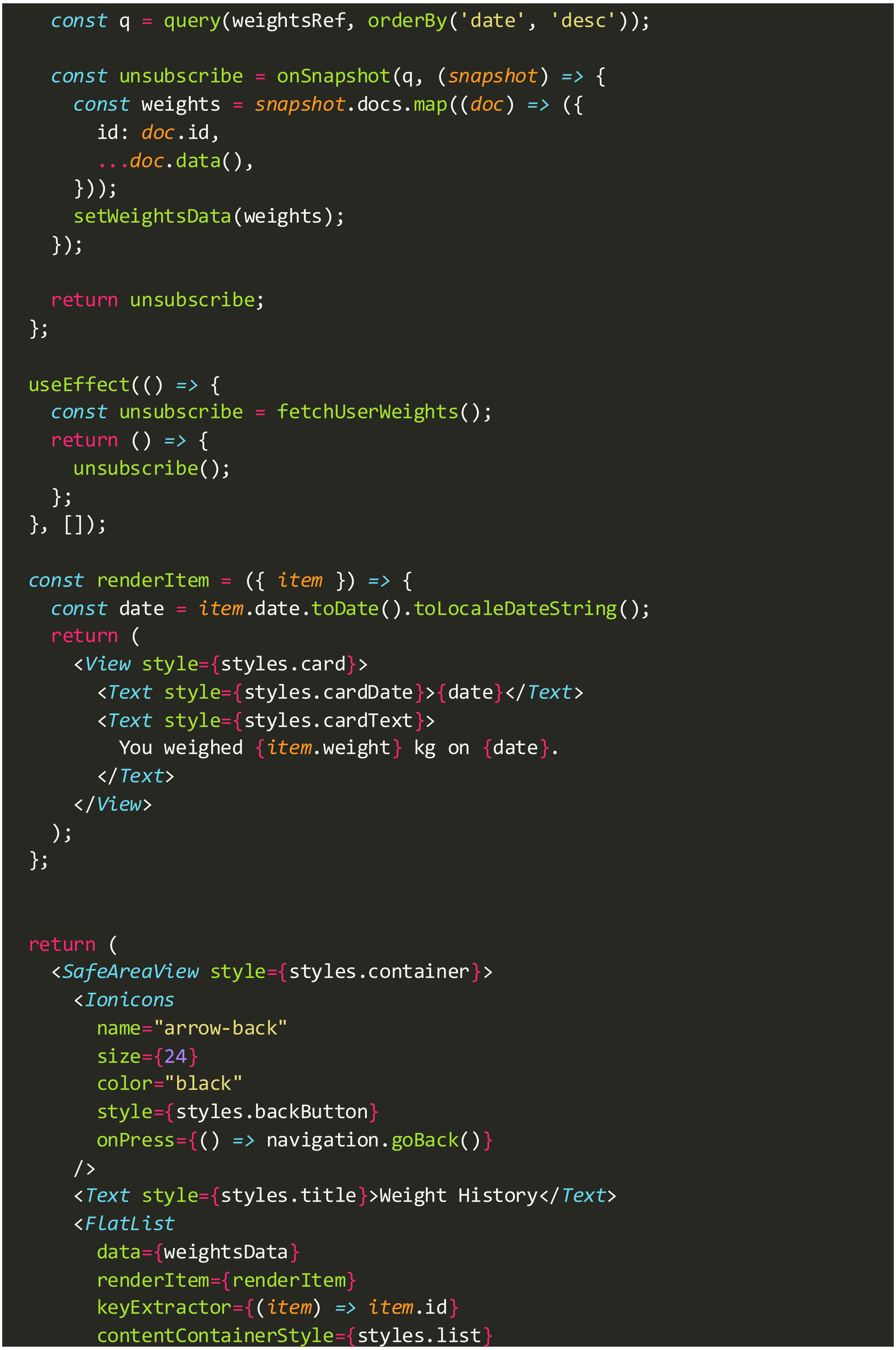
```
        {/* This is the chart for displaying weight data */}
        {weightsData.length < 2 ? (
          <Text style={styles.message}>Enter more weights to see
progress</Text>
        ) : (
          <LineChart
            data={{
              labels: weightsData.map((weightData, index) => `#${index + 1}`),
              datasets: [
                {
                  data: weightsData.map((weightData) => weightData.weight),
                },
              ],
            }}
            width={Dimensions.get("window").width - 20}
            height={300}
            yAxisSuffix="kg"
            onDataPointClick={({ index, x, y }) => {
              const screenWidth = Dimensions.get("window").width;
              const tooltipWidth = 80;
              const padding = 10;

              let newX = x;
              if (x + tooltipWidth / 2 + padding > screenWidth) {
                newX = x - (x + tooltipWidth / 2 + padding - screenWidth);
              } else if (x - tooltipWidth / 2 - padding < 0) {
                newX = x + Math.abs(x - tooltipWidth / 2 - padding);
              }

              setTooltipData({
                weight: weightsData[index].weight,
                date: weightsData[index].date.toDate().toLocaleDateString(),
              });
              setTooltipPosition({ x: newX, y: y });
              setTooltipVisible(true);
            }}
            chartConfig={{
              backgroundGradientFrom: "#fff",
              backgroundGradientTo: "#fff",
              color: (opacity = 1) => `rgba(7, 146, 249, ${opacity})`,
              strokeWidth: 2,
              barPercentage: 0.5,
              decimalPlaces: 1,
            }}
            bezier
            style={{
              marginVertical: 8,
              borderRadius: 16,
```



```
            }}
          />
        )}
        {tooltipVisible && (
          <View
            style={[
              styles.tooltip,
              { top: tooltipPosition.y - 40, left: tooltipPosition.x - 50 },
            ]}
          >
            <TouchableOpacity
              style={StyleSheet.absoluteFill}
              onPress={() => setTooltipVisible(false)}
              activeOpacity={1}
            />

            <Text style={styles.tooltipText}>
              Weight: {tooltipData.weight} kg{"\n"}
              Date: {tooltipData.date}
            </Text>
          </View>
        )}
        <TouchableOpacity
          style={styles.weightBtn}
          onPress={() => navigation.navigate("WeightHistory")}
        >
          <Text style={styles.weightBtnTxt}>View Weight History</Text>
        </TouchableOpacity>
      </View>
      <View style={styles.workoutStatsContainer}>
        <Text style={styles.workoutStatsTitle}>Workout Stats</Text>
        <Text style={styles.workoutStatsText}>
          Completed Workouts: {completedWorkouts}
        </Text>
        <Text style={styles.workoutStatsText}>
          Incomple Workouts: {uncompletedWorkouts}
        </Text>
      </View>
      <View style={styles.nutritionCard}>
        <Text style={styles.nutritionCardTitle}>Today's Nutrition</Text>
        {/* This is the chart for displaying daily nutrition stats */}
        {todayNutritionTotals ? (
          <>
            <View style={styles.caloriesContainer}>
              <ProgressChart
                data={{
                  labels: ["Calories"],
                  data: [todayNutritionTotals.totalCalories / (calorieLimit ||
1)],
                }}
                width={Dimensions.get("window").width}
                height={200}
                strokeWidth={8}
                radius={75}
                chartConfig={{
                  backgroundGradientFrom: "#fff",
                  backgroundGradientTo: "#fff",
                  color: (opacity = 1) => `rgba(7, 146, 249, ${opacity})`,
                  strokeWidth: 2,
                  barPercentage: 0.5,
                }}
                hideLegend={true}
              />
              <Text style={styles.caloriesText}>
                {todayNutritionTotals.totalCalories} / {user.calorieLimit}
kcal
              </Text>
            </View>
            <View style={styles.nutritionDetailsContainer}>
              <Text style={styles.nutritionCardText}>
                Fat: {todayNutritionTotals.totalFat}g
              </Text>
              <Text style={styles.nutritionCardText}>
                Protein: {todayNutritionTotals.totalProtein}g
              </Text>
            </View>
          </>
        ) : (
          <Text style={styles.nutritionCardText}>No data available</Text>
        )}
      </View>
    </ScrollView>
  );
};

export default AnalyticsScreen;
```

WeightHistoryScreen.js

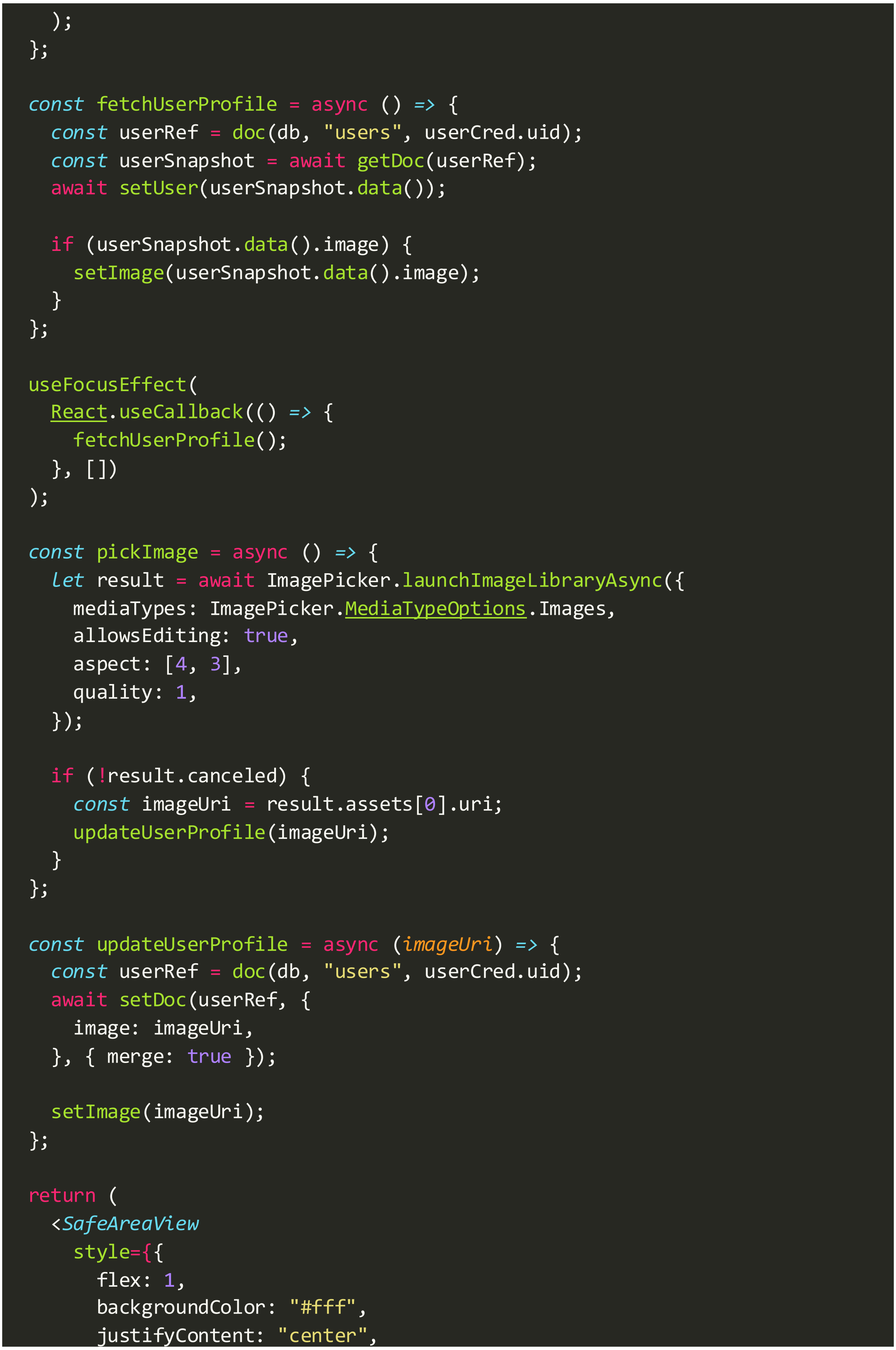
```
const WeightHistoryScreen = () => {
  const userCred = getAuth().currentUser;
  const [weightsData, setWeightsData] = useState([]);
  const navigation = useNavigation();

  const fetchUserWeights = () => {
    const weightsRef = collection(db, 'users', userCred.uid, 'weights');
```



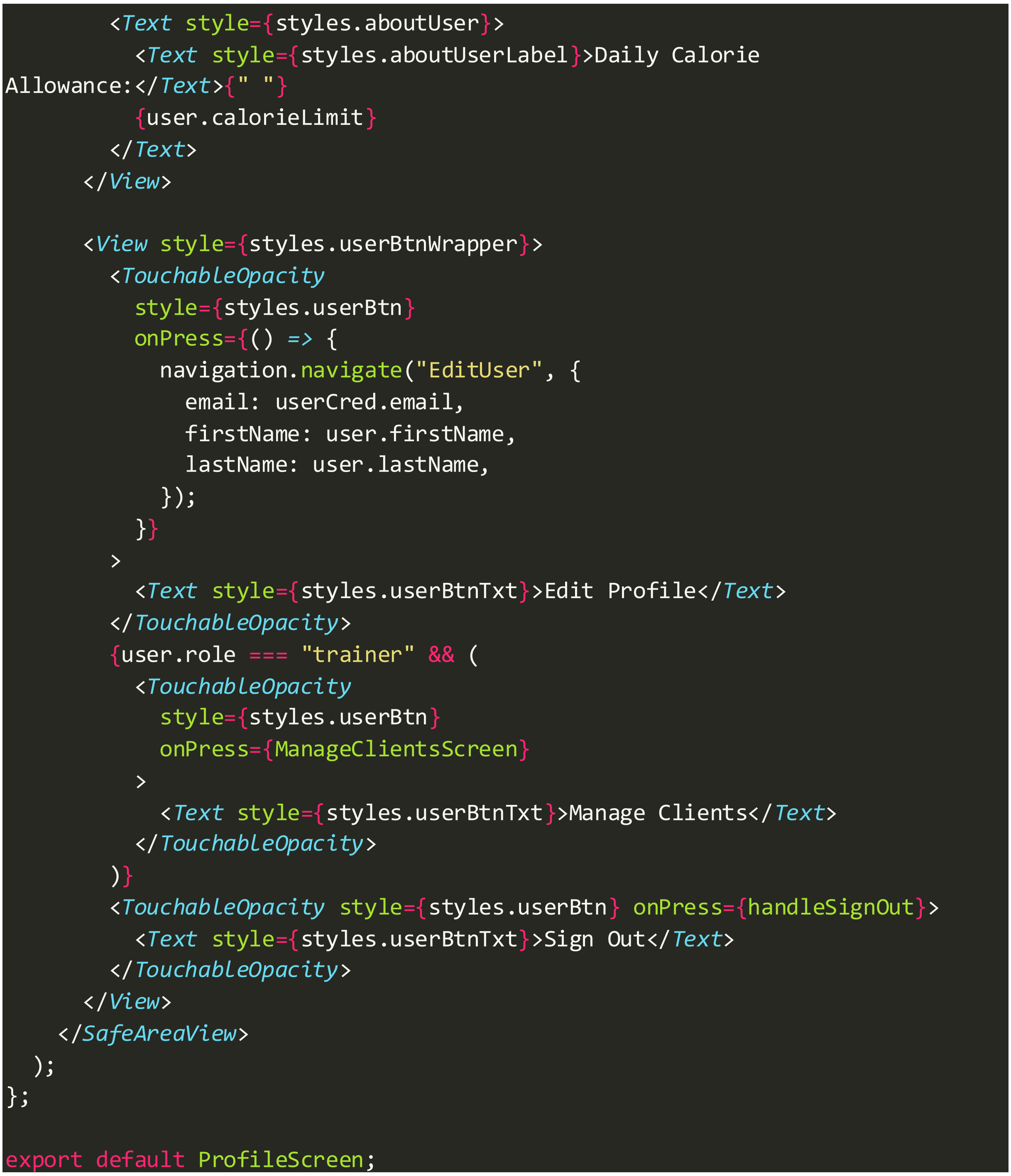
```
  const q = query(weightsRef, orderBy('date', 'desc'));

  const unsubscribe = onSnapshot(q, (snapshot) => {
    const weights = snapshot.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    setWeightsData(weights);
  });

  return unsubscribe;
};

useEffect(() => {
  const unsubscribe = fetchUserWeights();
  return () => {
    unsubscribe();
  };
}, []);

const renderItem = ({ item }) => {
  const date = item.date.toDate().toLocaleDateString();
  return (
    <View style={styles.card}>
      <Text style={styles.cardDate}>{date}</Text>
      <Text style={styles.cardText}>
        You weighed {item.weight} kg on {date}.
      </Text>
    </View>
  );
};


return (
  <SafeAreaView style={styles.container}>
    <Ionicons
      name="arrow-back"
      size={24}
      color="black"
      style={styles.backButton}
      onPress={() => navigation.goBack()}
    />
    <Text style={styles.title}>Weight History</Text>
    <FlatList
      data={weightsData}
      renderItem={renderItem}
      keyExtractor={(item) => item.id}
      contentContainerStyle={styles.list}
      />
    </SafeAreaView>
  );
};

export default WeightHistoryScreen;
```

ProfileScreen.js

```
const ProfileScreen = () => {
  const auth = getAuth();
  const userCred = auth.currentUser;
  const [user, setUser] = useState({});
  const [image, setImage] = useState(null);
  const navigation = useNavigation();

  const ManageClientsScreen = () => {
    navigation.navigate("ManageClients");
  };

  const handleSignOut = async () => {
    auth.signOut().catch((error) => alert(error.message));
  };

  const DeleteUser = () => {
    Alert.alert(
      "Delete Account",
      "Are you sure you want to delete this account? All of your data will be
lost.",
      [
        {
          text: "Cancel",
          style: "cancel",
        },
        {
          text: "Delete",
          onPress: () => {
            deleteUser(userCred)
              .then(() => {
                console.log("Deleted", userCred);
              })
              .catch((error) => {
                console.log("error:", error);
              });
          },
        },
      ],
      { cancelable: false }
```

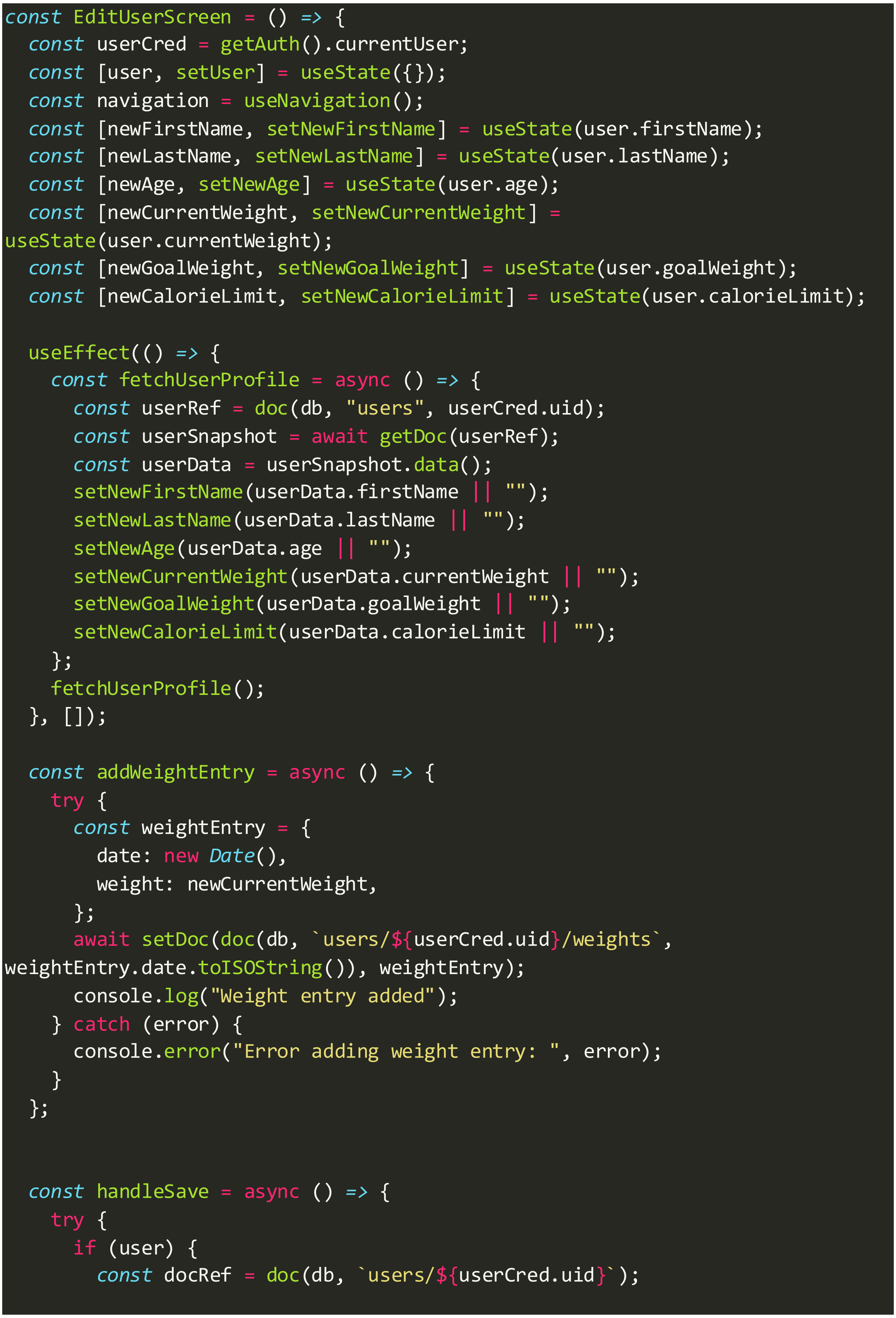


```
  );
};

const fetchUserProfile = async () => {
  const userRef = doc(db, "users", userCred.uid);
  const userSnapshot = await getDoc(userRef);
  await setUser(userSnapshot.data());

  if (userSnapshot.data().image) {
    setImage(userSnapshot.data().image);
  }
};

useFocusEffect(
  React.useCallback(() => {
    fetchUserProfile();
  }, [])
);

const pickImage = async () => {
  let result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ImagePicker.MediaTypeOptions.Images,
    allowsEditing: true,
    aspect: [4, 3],
    quality: 1,
  });

  if (!result.canceled) {
    const imageUri = result.assets[0].uri;
    updateUserProfile(imageUri);
  }
};

const updateUserProfile = async (imageUri) => {
  const userRef = doc(db, "users", userCred.uid);
  await setDoc(userRef, {
    image: imageUri,
  }, { merge: true });

  setImage(imageUri);
};

return (
  <SafeAreaView
    style={{
      flex: 1,
      backgroundColor: "#fff",
      justifyContent: "center",
      alignItems: "center",
    }}
  >
    <TouchableOpacity style={styles.delete} onPress={DeleteUser}>
      <Text style={{ marginTop: 50, fontWeight: "bold" }}>
        Delete Account
      </Text>
      <Entypo
        name="remove-user"
        size={40}
        style={{
          color: "darkred",
          marginTop: 20,
          position: "absolute",
          right: 10,
          top: 50,
        }}
      />
    </TouchableOpacity>
    <Text style={styles.userName}>Profile</Text>
    <TouchableOpacity onPress={pickImage}>
      <Image
        style={styles.userImg}
        source={image ? { uri: image } : require("../../../assets/blank-
user.jpg")}
      />
      <View style={styles.editBtn}>
        <Entypo name="plus" size={24} color="white" />
      </View>
    </TouchableOpacity>
    <View style={styles.userCard}>
      <Text style={styles.userName}>
        {user.firstName} {user.lastName}
      </Text>
      <Text style={styles.aboutUser}>
        <Text style={styles.aboutUserLabel}>Email:</Text> {userCred.email}
      </Text>
      <Text style={styles.aboutUser}>
        <Text style={styles.aboutUserLabel}>Age:</Text> {user.age}
      </Text>
      <Text style={styles.aboutUser}>
        <Text style={styles.aboutUserLabel}>Current Weight:</Text>{" "}
        {user.currentWeight}
      </Text>
      <Text style={styles.aboutUser}>
        <Text style={styles.aboutUserLabel}>Goal Weight:</Text>{" "}
        {user.goalWeight}
      </Text>
```

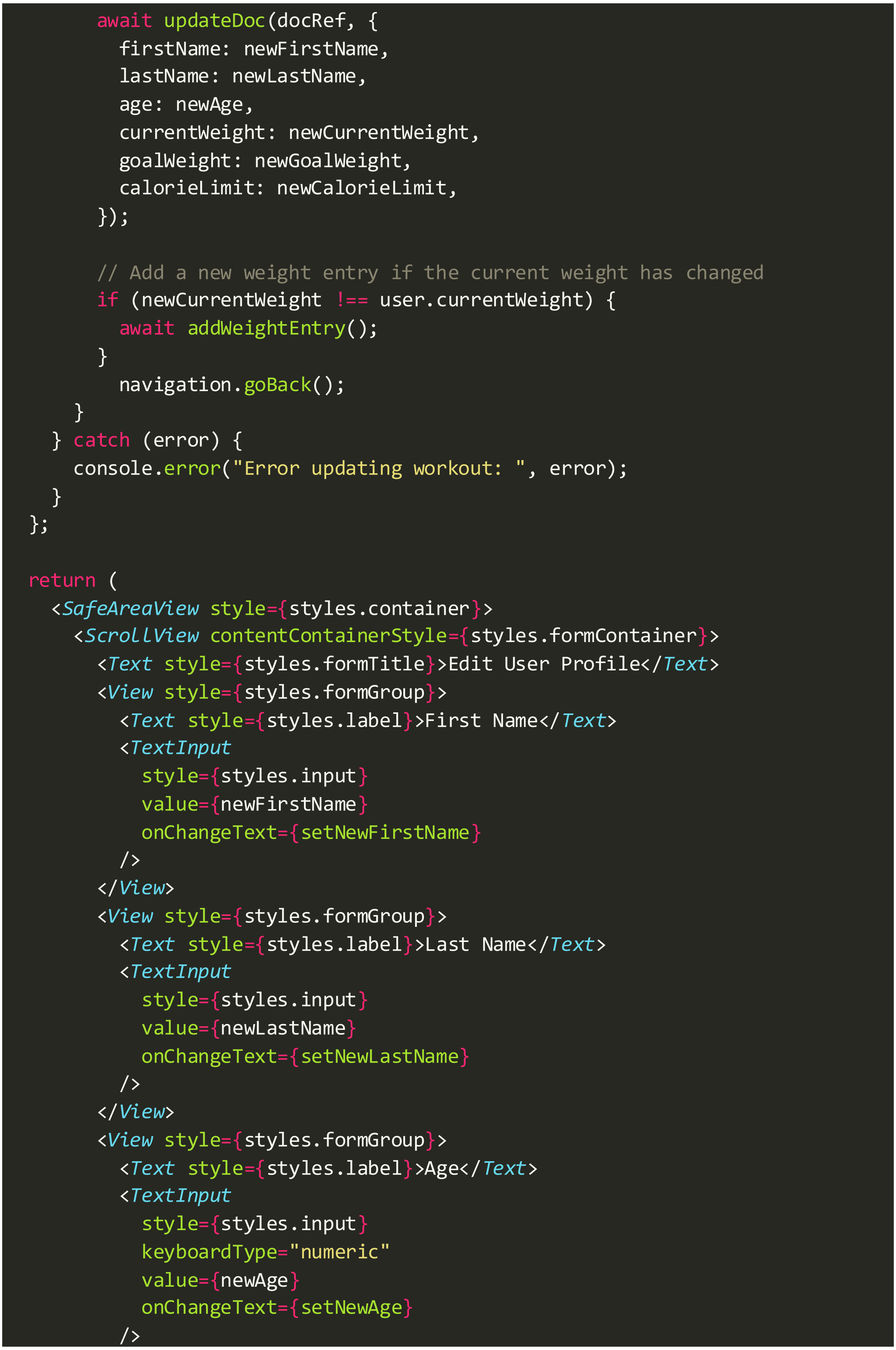


```
        <Text style={styles.aboutUser}>
          <Text style={styles.aboutUserLabel}>Daily Calorie
Allowance:</Text>{" "}
          {user.calorieLimit}
        </Text>
      </View>

      <View style={styles.userBtnWrapper}>
        <TouchableOpacity
          style={styles.userBtn}
          onPress={() => {
            navigation.navigate("EditUser", {
              email: userCred.email,
              firstName: user.firstName,
              lastName: user.lastName,
            });
          }}
        >
          <Text style={styles.userBtnTxt}>Edit Profile</Text>
        </TouchableOpacity>
        {user.role === "trainer" && (
          <TouchableOpacity
            style={styles.userBtn}
            onPress={ManageClientsScreen}
          >
            <Text style={styles.userBtnTxt}>Manage Clients</Text>
          </TouchableOpacity>
        )}
        <TouchableOpacity style={styles.userBtn} onPress={handleSignOut}>
          <Text style={styles.userBtnTxt}>Sign Out</Text>
        </TouchableOpacity>
      </View>
    </SafeAreaView>
  );
};

export default ProfileScreen;
```

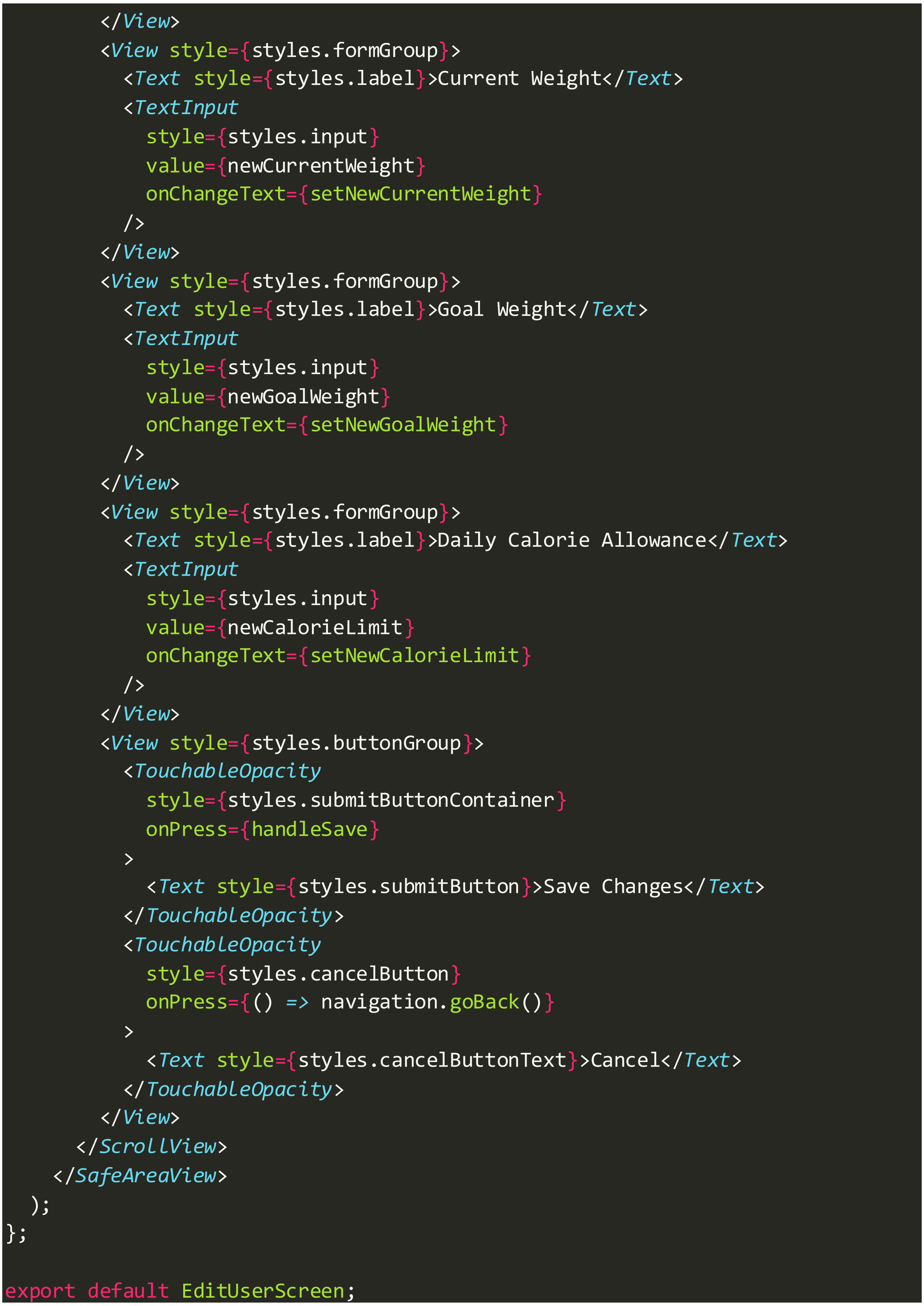
EditUserScreen.js



```
const EditUserScreen = () => {
  const userCred = getAuth().currentUser;
  const [user, setUser] = useState({});
  const navigation = useNavigation();
  const [newFirstName, setNewFirstName] = useState(user.firstName);
  const [newLastName, setNewLastName] = useState(user.lastName);
  const [newAge, setNewAge] = useState(user.age);
  const [newCurrentWeight, setNewCurrentWeight] =
useState(user.currentWeight);
  const [newGoalWeight, setNewGoalWeight] = useState(user.goalWeight);
  const [newCalorieLimit, setNewCalorieLimit] = useState(user.calorieLimit);

  useEffect(() => {
    const fetchUserProfile = async () => {
      const userRef = doc(db, "users", userCred.uid);
      const userSnapshot = await getDoc(userRef);
      const userData = userSnapshot.data();
      setNewFirstName(userData.firstName || "");
      setNewLastName(userData.lastName || "");
      setNewAge(userData.age || "");
      setNewCurrentWeight(userData.currentWeight || "");
      setNewGoalWeight(userData.goalWeight || "");
      setNewCalorieLimit(userData.calorieLimit || "");
    };
    fetchUserProfile();
  }, []);

  const addWeightEntry = async () => {
    try {
      const weightEntry = {
        date: new Date(),
        weight: newCurrentWeight,
      };
      await setDoc(doc(db, `users/${userCred.uid}/weights`,
weightEntry.date.toISOString()), weightEntry);
      console.log("Weight entry added");
    } catch (error) {
      console.error("Error adding weight entry: ", error);
    }
  };


  const handleSave = async () => {
    try {
      if (user) {
        const docRef = doc(db, `users/${userCred.uid}`);
```

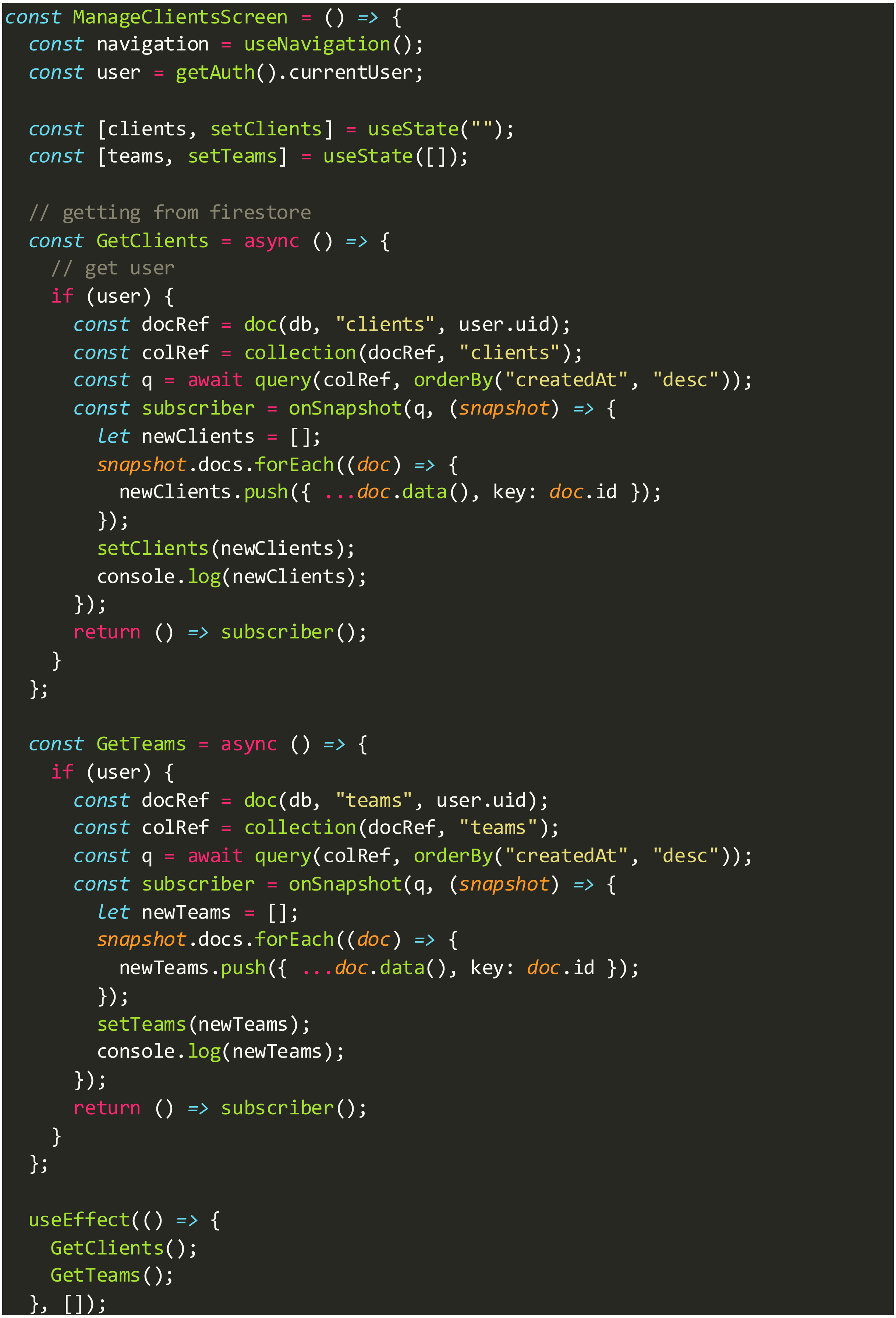


```
      await updateDoc(docRef, {
        firstName: newFirstName,
        lastName: newLastName,
        age: newAge,
        currentWeight: newCurrentWeight,
        goalWeight: newGoalWeight,
        calorieLimit: newCalorieLimit,
      });

      // Add a new weight entry if the current weight has changed
      if (newCurrentWeight !== user.currentWeight) {
        await addWeightEntry();
      }
        navigation.goBack();
    }
  } catch (error) {
    console.error("Error updating workout: ", error);
  }
};

return (
  <SafeAreaView style={styles.container}>
    <ScrollView contentContainerStyle={styles.formContainer}>
      <Text style={styles.formTitle}>Edit User Profile</Text>
      <View style={styles.formGroup}>
        <Text style={styles.label}>First Name</Text>
        <TextInput
          style={styles.input}
          value={newFirstName}
          onChangeText={setNewFirstName}
        />
      </View>
      <View style={styles.formGroup}>
        <Text style={styles.label}>Last Name</Text>
        <TextInput
          style={styles.input}
          value={newLastName}
          onChangeText={setNewLastName}
        />
      </View>
      <View style={styles.formGroup}>
        <Text style={styles.label}>Age</Text>
        <TextInput
          style={styles.input}
          keyboardType="numeric"
          value={newAge}
          onChangeText={setNewAge}
        />
```

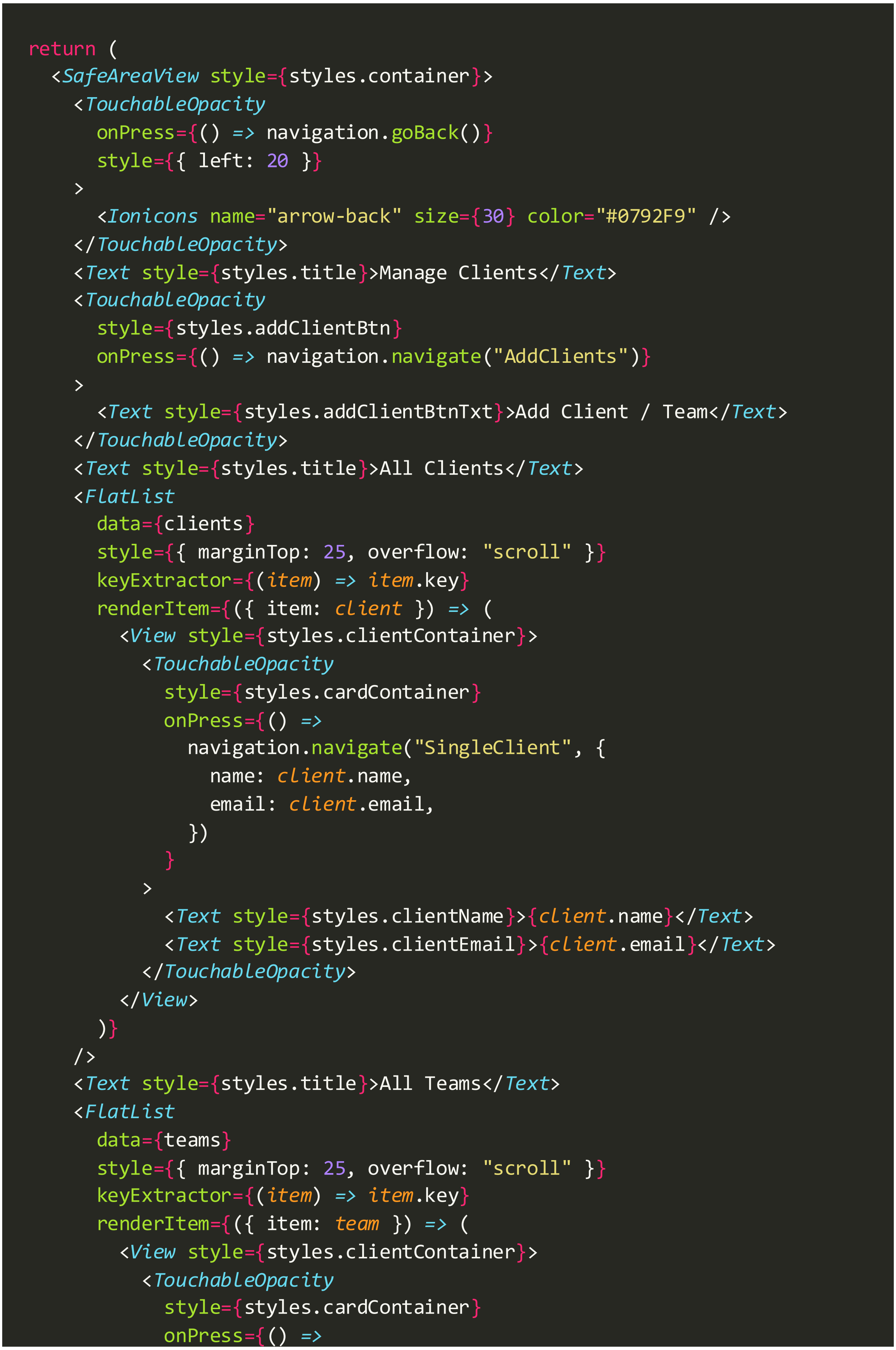


```
        </View>
        <View style={styles.formGroup}>
          <Text style={styles.label}>Current Weight</Text>
          <TextInput
            style={styles.input}
            value={newCurrentWeight}
            onChangeText={setNewCurrentWeight}
          />
        </View>
        <View style={styles.formGroup}>
          <Text style={styles.label}>Goal Weight</Text>
          <TextInput
            style={styles.input}
            value={newGoalWeight}
            onChangeText={setNewGoalWeight}
          />
        </View>
        <View style={styles.formGroup}>
          <Text style={styles.label}>Daily Calorie Allowance</Text>
          <TextInput
            style={styles.input}
            value={newCalorieLimit}
            onChangeText={setNewCalorieLimit}
          />
        </View>
        <View style={styles.buttonGroup}>
          <TouchableOpacity
            style={styles.submitButtonContainer}
            onPress={handleSave}
          >
            <Text style={styles.submitButton}>Save Changes</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={styles.cancelButton}
            onPress={() => navigation.goBack()}
          >
            <Text style={styles.cancelButtonText}>Cancel</Text>
          </TouchableOpacity>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};

export default EditUserScreen;
```

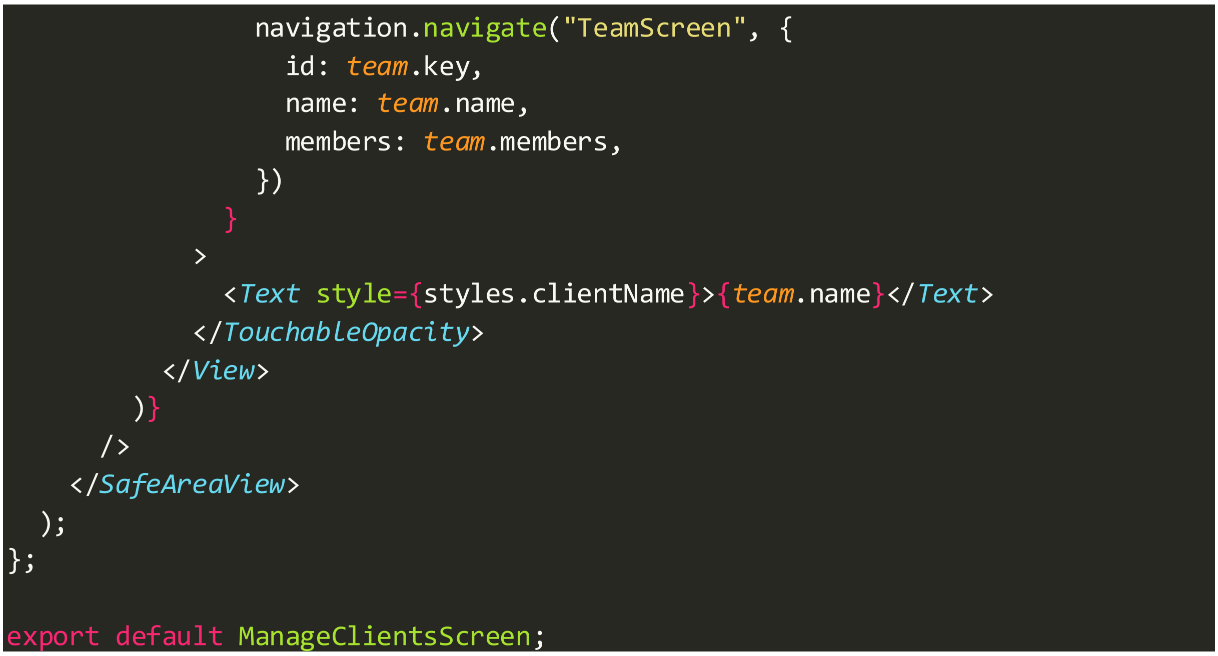
ManageClientsScreen.js



```javascript
const ManageClientsScreen = () => {
  const navigation = useNavigation();
  const user = getAuth().currentUser;

  const [clients, setClients] = useState("");
  const [teams, setTeams] = useState([]);

  // getting from firestore
  const GetClients = async () => {
    // get user
    if (user) {
      const docRef = doc(db, "clients", user.uid);
      const colRef = collection(docRef, "clients");
      const q = await query(colRef, orderBy("createdAt", "desc"));
      const subscriber = onSnapshot(q, (snapshot) => {
        let newClients = [];
        snapshot.docs.forEach((doc) => {
          newClients.push({ ...doc.data(), key: doc.id });
        });
        setClients(newClients);
        console.log(newClients);
      });
      return () => subscriber();
    }
  };

  const GetTeams = async () => {
    if (user) {
      const docRef = doc(db, "teams", user.uid);
      const colRef = collection(docRef, "teams");
      const q = await query(colRef, orderBy("createdAt", "desc"));
      const subscriber = onSnapshot(q, (snapshot) => {
        let newTeams = [];
        snapshot.docs.forEach((doc) => {
          newTeams.push({ ...doc.data(), key: doc.id });
        });
        setTeams(newTeams);
        console.log(newTeams);
      });
      return () => subscriber();
    }
  };

  useEffect(() => {
    GetClients();
    GetTeams();
  }, []);
```

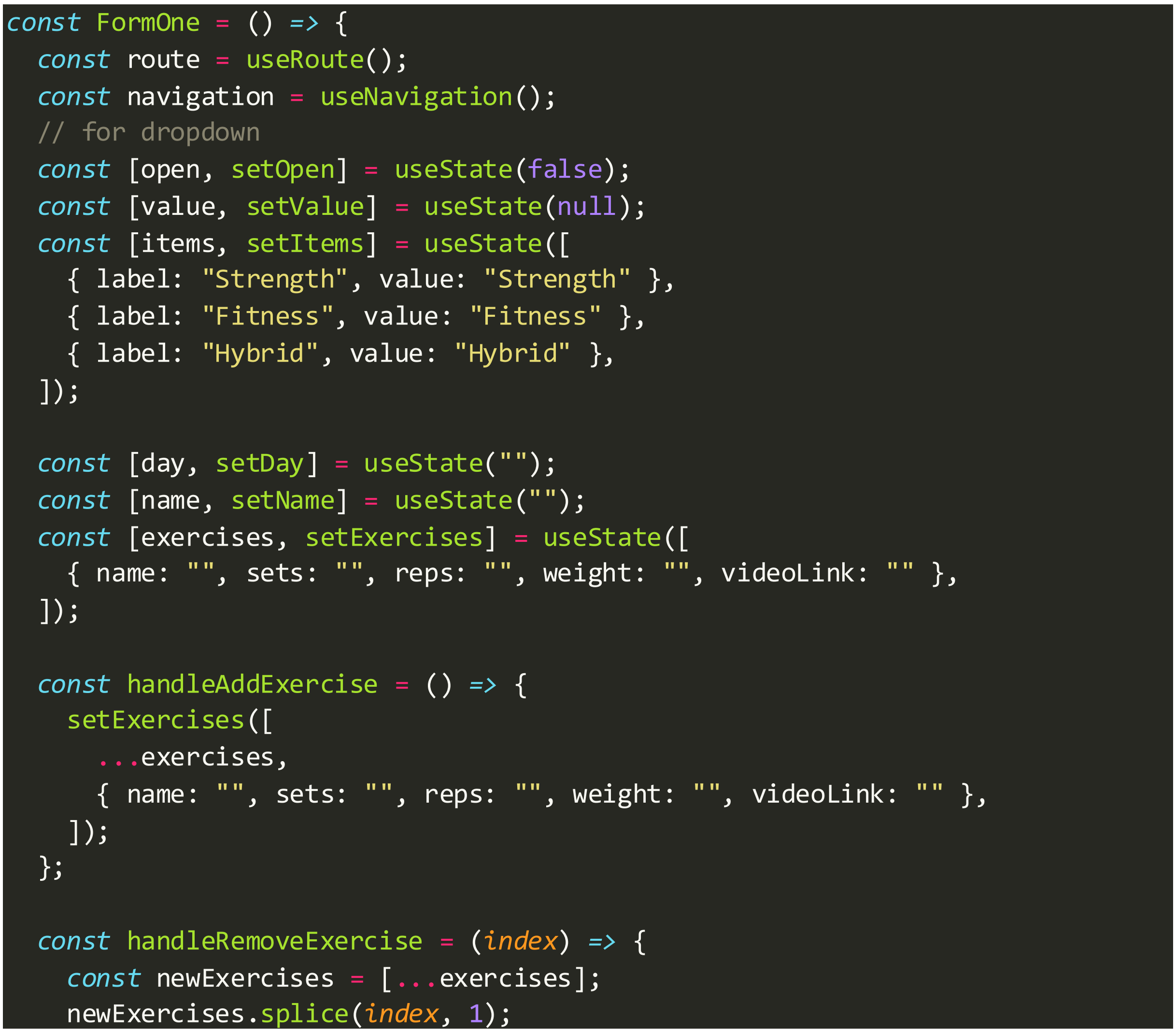


```
return (
  <SafeAreaView style={styles.container}>
    <TouchableOpacity
      onPress={() => navigation.goBack()}
      style={{ left: 20 }}
    >
      <Ionicons name="arrow-back" size={30} color="#0792F9" />
    </TouchableOpacity>
    <Text style={styles.title}>Manage Clients</Text>
    <TouchableOpacity
      style={styles.addClientBtn}
      onPress={() => navigation.navigate("AddClients")}
    >
      <Text style={styles.addClientBtnTxt}>Add Client / Team</Text>
    </TouchableOpacity>
    <Text style={styles.title}>All Clients</Text>
    <FlatList
      data={clients}
      style={{ marginTop: 25, overflow: "scroll" }}
      keyExtractor={(item) => item.key}
      renderItem={({ item: client }) => (
        <View style={styles.clientContainer}>
          <TouchableOpacity
            style={styles.cardContainer}
            onPress={() =>
              navigation.navigate("SingleClient", {
                name: client.name,
                email: client.email,
              })
            }
          >
            <Text style={styles.clientName}>{client.name}</Text>
            <Text style={styles.clientEmail}>{client.email}</Text>
          </TouchableOpacity>
        </View>
      )}
    />
    <Text style={styles.title}>All Teams</Text>
    <FlatList
      data={teams}
      style={{ marginTop: 25, overflow: "scroll" }}
      keyExtractor={(item) => item.key}
      renderItem={({ item: team }) => (
        <View style={styles.clientContainer}>
          <TouchableOpacity
            style={styles.cardContainer}
            onPress={() =>
```

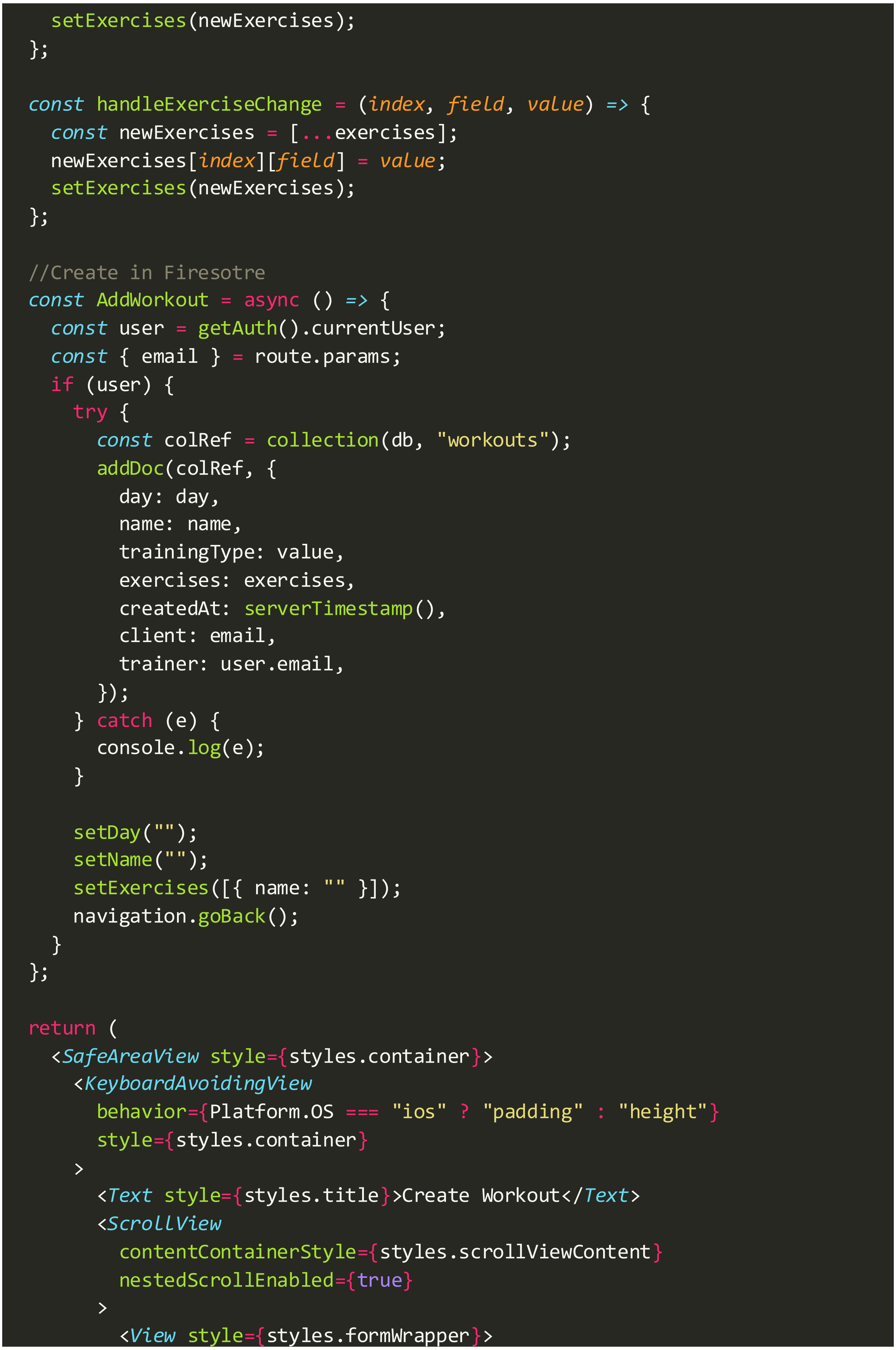


```
                navigation.navigate("TeamScreen", {
                  id: team.key,
                  name: team.name,
                  members: team.members,
                })
              }
            >
              <Text style={styles.clientName}>{team.name}</Text>
            </TouchableOpacity>
          </View>
        )}
      />
    </SafeAreaView>
  );
};

export default ManageClientsScreen;
```

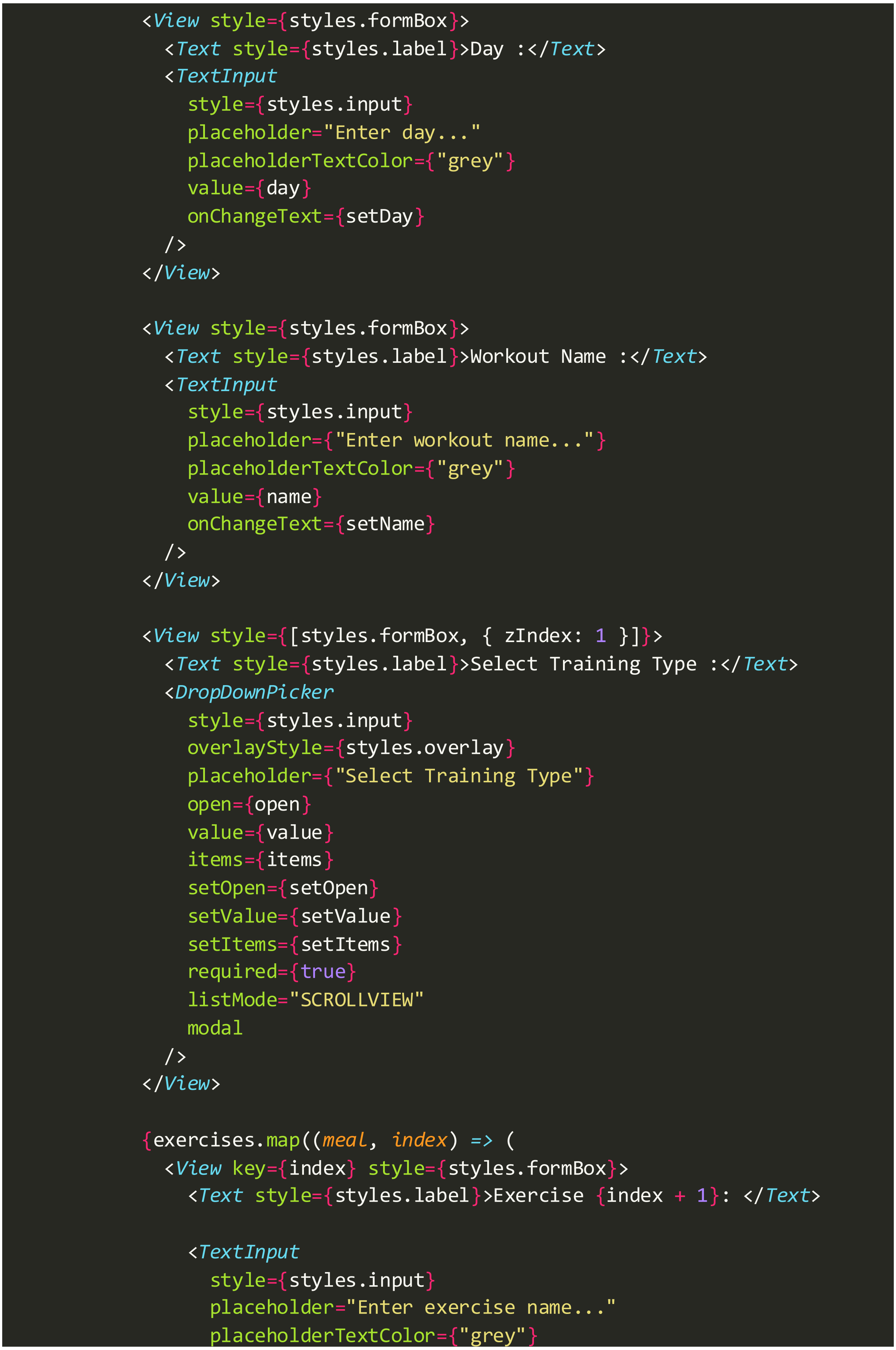
SingleClientScreen.js



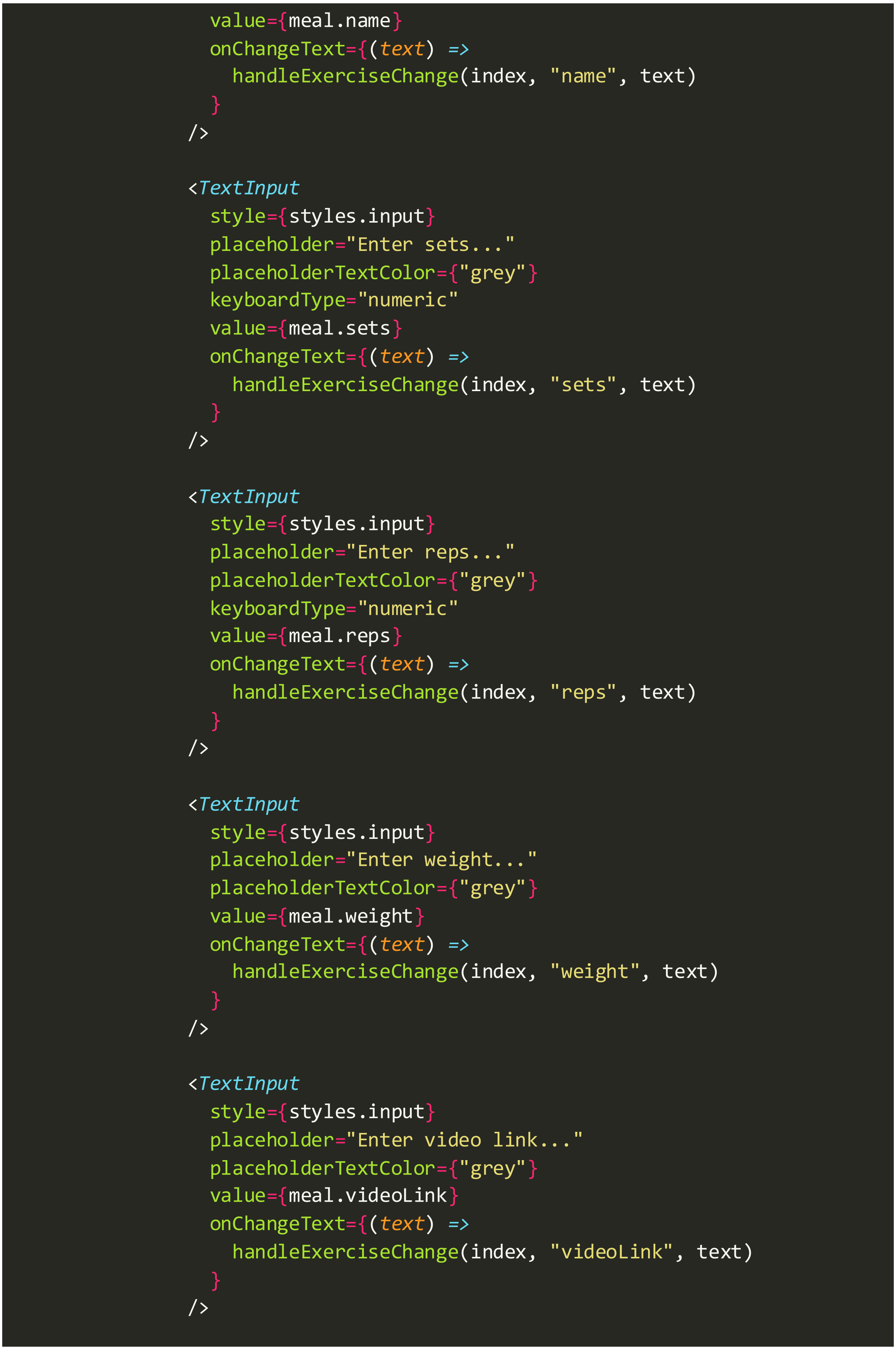
```
const FormOne = () => {
  const route = useRoute();
  const navigation = useNavigation();
  // for dropdown
  const [open, setOpen] = useState(false);
  const [value, setValue] = useState(null);
  const [items, setItems] = useState([
    { label: "Strength", value: "Strength" },
    { label: "Fitness", value: "Fitness" },
    { label: "Hybrid", value: "Hybrid" },
  ]);

  const [day, setDay] = useState("");
  const [name, setName] = useState("");
  const [exercises, setExercises] = useState([
    { name: "", sets: "", reps: "", weight: "", videoLink: "" },
  ]);

  const handleAddExercise = () => {
    setExercises([
      ...exercises,
      { name: "", sets: "", reps: "", weight: "", videoLink: "" },
    ]);
  };

  const handleRemoveExercise = (index) => {
    const newExercises = [...exercises];
    newExercises.splice(index, 1);
```



```
  setExercises(newExercises);
};

const handleExerciseChange = (index, field, value) => {
  const newExercises = [...exercises];
  newExercises[index][field] = value;
  setExercises(newExercises);
};

//Create in Firesotre
const AddWorkout = async () => {
  const user = getAuth().currentUser;
  const { email } = route.params;
  if (user) {
    try {
      const colRef = collection(db, "workouts");
      addDoc(colRef, {
        day: day,
        name: name,
        trainingType: value,
        exercises: exercises,
        createdAt: serverTimestamp(),
        client: email,
        trainer: user.email,
      });
    } catch (e) {
      console.log(e);
    }

    setDay("");
    setName("");
    setExercises([{ name: "" }]);
    navigation.goBack();
  }
};

return (
  <SafeAreaView style={styles.container}>
    <KeyboardAvoidingView
      behavior={Platform.OS === "ios" ? "padding" : "height"}
      style={styles.container}
    >
      <Text style={styles.title}>Create Workout</Text>
      <ScrollView
        contentContainerStyle={styles.scrollViewContent}
        nestedScrollEnabled={true}
      >
        <View style={styles.formWrapper}>
```

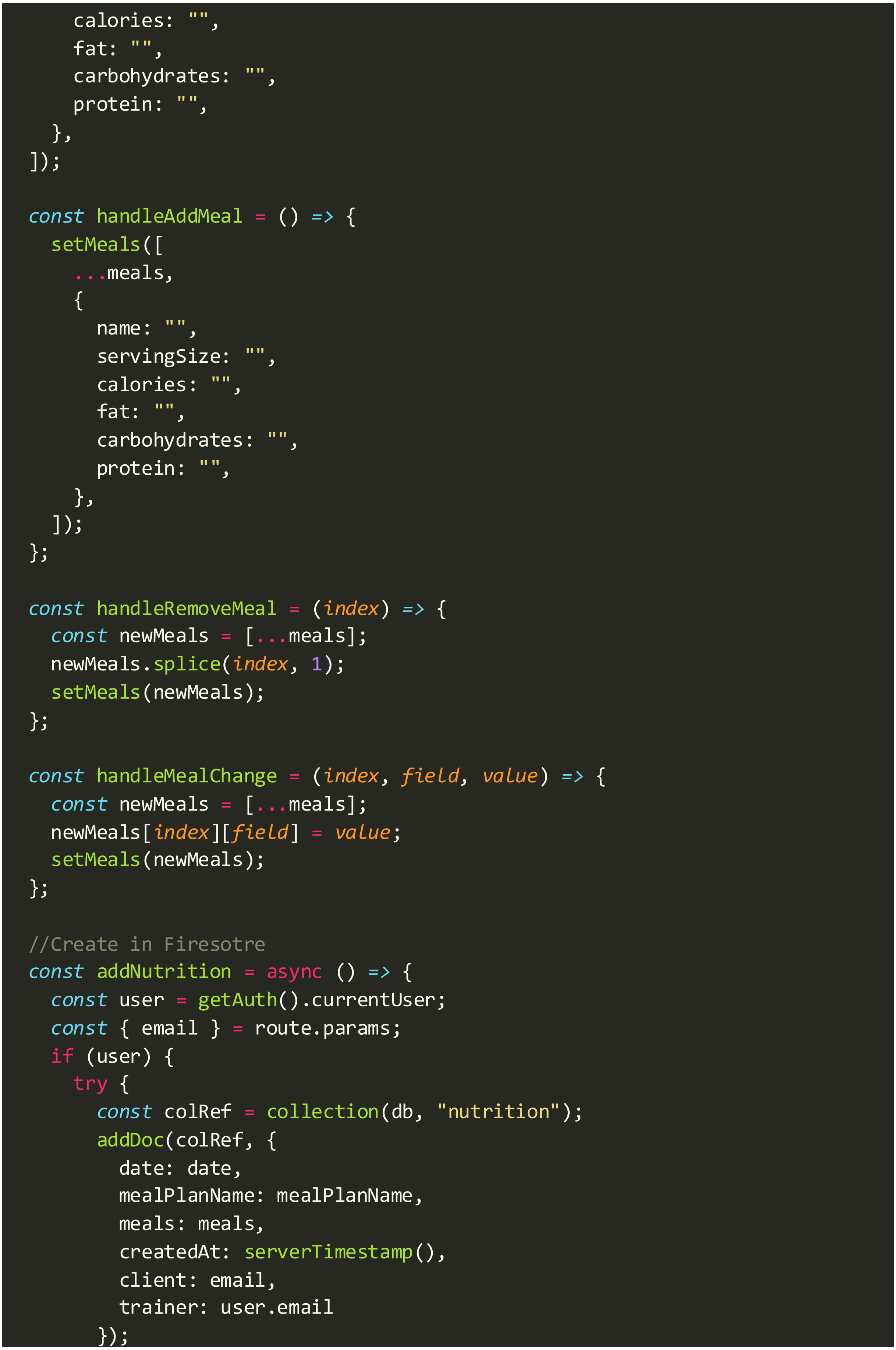


```
          <View style={styles.formBox}>
            <Text style={styles.label}>Day :</Text>
            <TextInput
              style={styles.input}
              placeholder="Enter day..."
              placeholderTextColor={"grey"}
              value={day}
              onChangeText={setDay}
            />
          </View>

          <View style={styles.formBox}>
            <Text style={styles.label}>Workout Name :</Text>
            <TextInput
              style={styles.input}
              placeholder={"Enter workout name..."}
              placeholderTextColor={"grey"}
              value={name}
              onChangeText={setName}
            />
          </View>

          <View style={[styles.formBox, { zIndex: 1 }]}>
            <Text style={styles.label}>Select Training Type :</Text>
            <DropDownPicker
              style={styles.input}
              overlayStyle={styles.overlay}
              placeholder={"Select Training Type"}
              open={open}
              value={value}
              items={items}
              setOpen={setOpen}
              setValue={setValue}
              setItems={setItems}
              required={true}
              listMode="SCROLLVIEW"
              modal
            />
          </View>

          {exercises.map((meal, index) => (
            <View key={index} style={styles.formBox}>
              <Text style={styles.label}>Exercise {index + 1}: </Text>

              <TextInput
                style={styles.input}
                placeholder="Enter exercise name..."
                placeholderTextColor={"grey"}
```

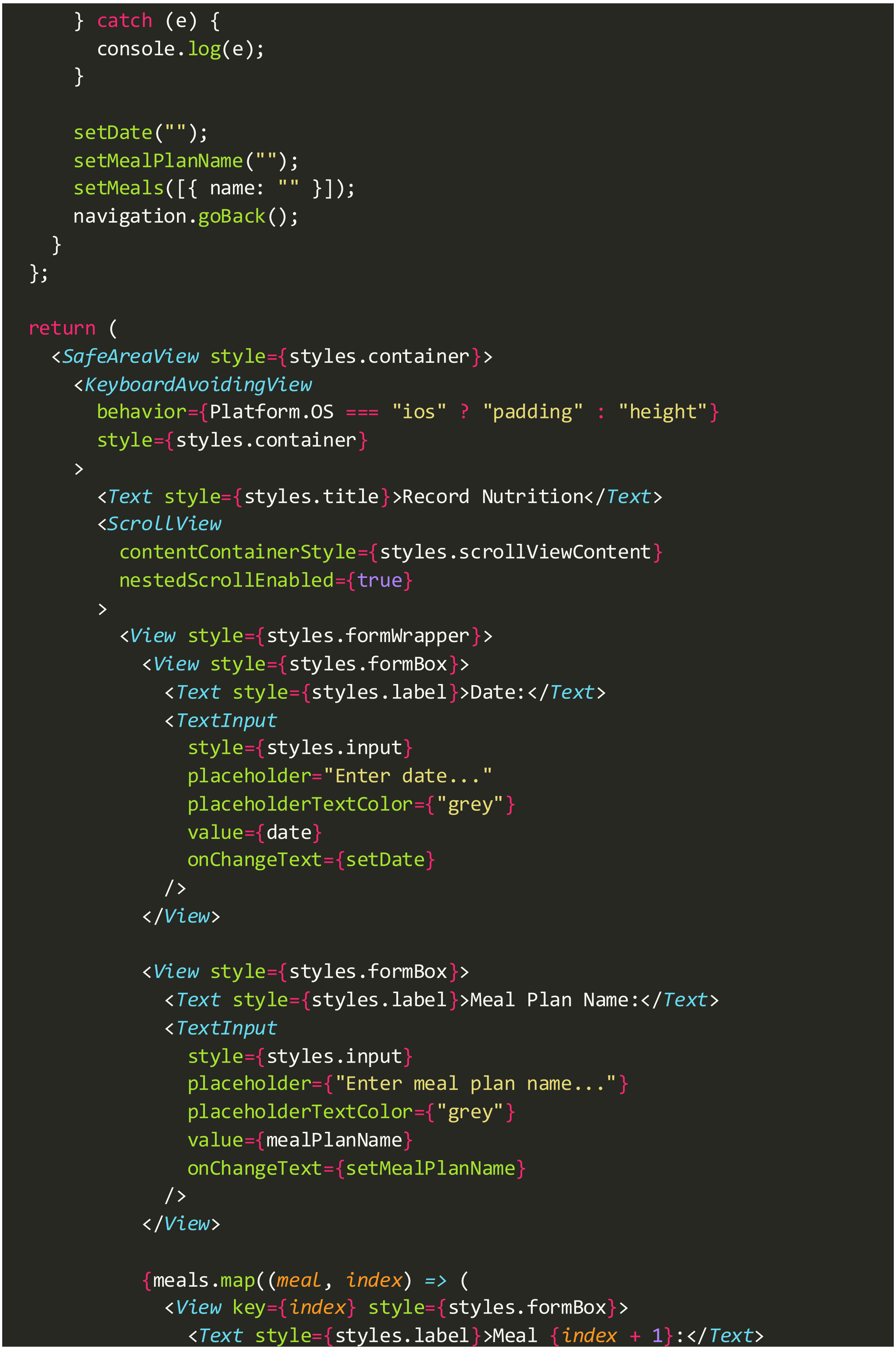


```
              value={meal.name}
              onChangeText={(text) =>
                handleExerciseChange(index, "name", text)
              }
            />

            <TextInput
              style={styles.input}
              placeholder="Enter sets..."
              placeholderTextColor={"grey"}
              keyboardType="numeric"
              value={meal.sets}
              onChangeText={(text) =>
                handleExerciseChange(index, "sets", text)
              }
            />

            <TextInput
              style={styles.input}
              placeholder="Enter reps..."
              placeholderTextColor={"grey"}
              keyboardType="numeric"
              value={meal.reps}
              onChangeText={(text) =>
                handleExerciseChange(index, "reps", text)
              }
            />

            <TextInput
              style={styles.input}
              placeholder="Enter weight..."
              placeholderTextColor={"grey"}
              value={meal.weight}
              onChangeText={(text) =>
                handleExerciseChange(index, "weight", text)
              }
            />

            <TextInput
              style={styles.input}
              placeholder="Enter video link..."
              placeholderTextColor={"grey"}
              value={meal.videoLink}
              onChangeText={(text) =>
                handleExerciseChange(index, "videoLink", text)
              }
            />
```

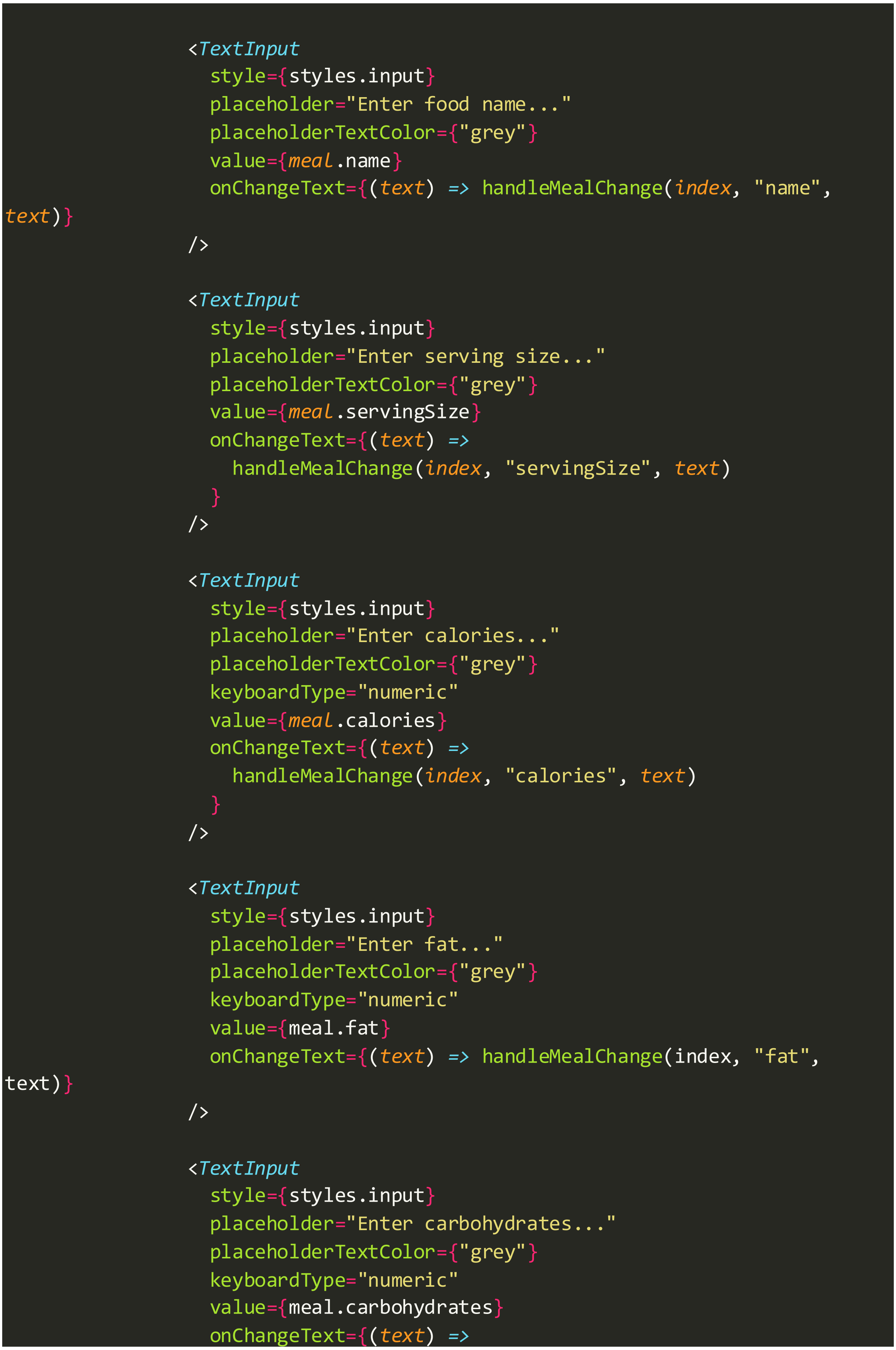
```
                <TouchableOpacity
                  style={styles.addButton}
                  onPress={handleAddExercise}
                >
                  <Text style={styles.addButtonText}>Add Exersie</Text>
                </TouchableOpacity>

                {index > 0 && (
                  <TouchableOpacity
                    style={styles.removeButton}
                    onPress={() => handleRemoveExercise(index)}
                  >
                    <Text style={styles.removeButtonText}>Remove
Exercise</Text>
                  </TouchableOpacity>
                )}
              </View>
            ))}

            <View style={styles.formBox}>
              <Text style={styles.label}>Notes:</Text>
              <TextInput
                style={styles.input}
                placeholder="Enter notes..."
                placeholderTextColor={"grey"}
                multiline={true}
              />
            </View>

            <TouchableOpacity style={styles.addButton} onPress={AddWorkout}>
              <Text style={styles.addButtonText}>Submit</Text>
            </TouchableOpacity>
          </View>
        </ScrollView>
      </KeyboardAvoidingView>
    </SafeAreaView>
  );
};

const FormTwo = () => {
  const route = useRoute();
  const navigation = useNavigation();
  const [date, setDate] = useState("");
  const [mealPlanName, setMealPlanName] = useState("");
  const [meals, setMeals] = useState([
    {
      name: "",
      servingSize: "",
```



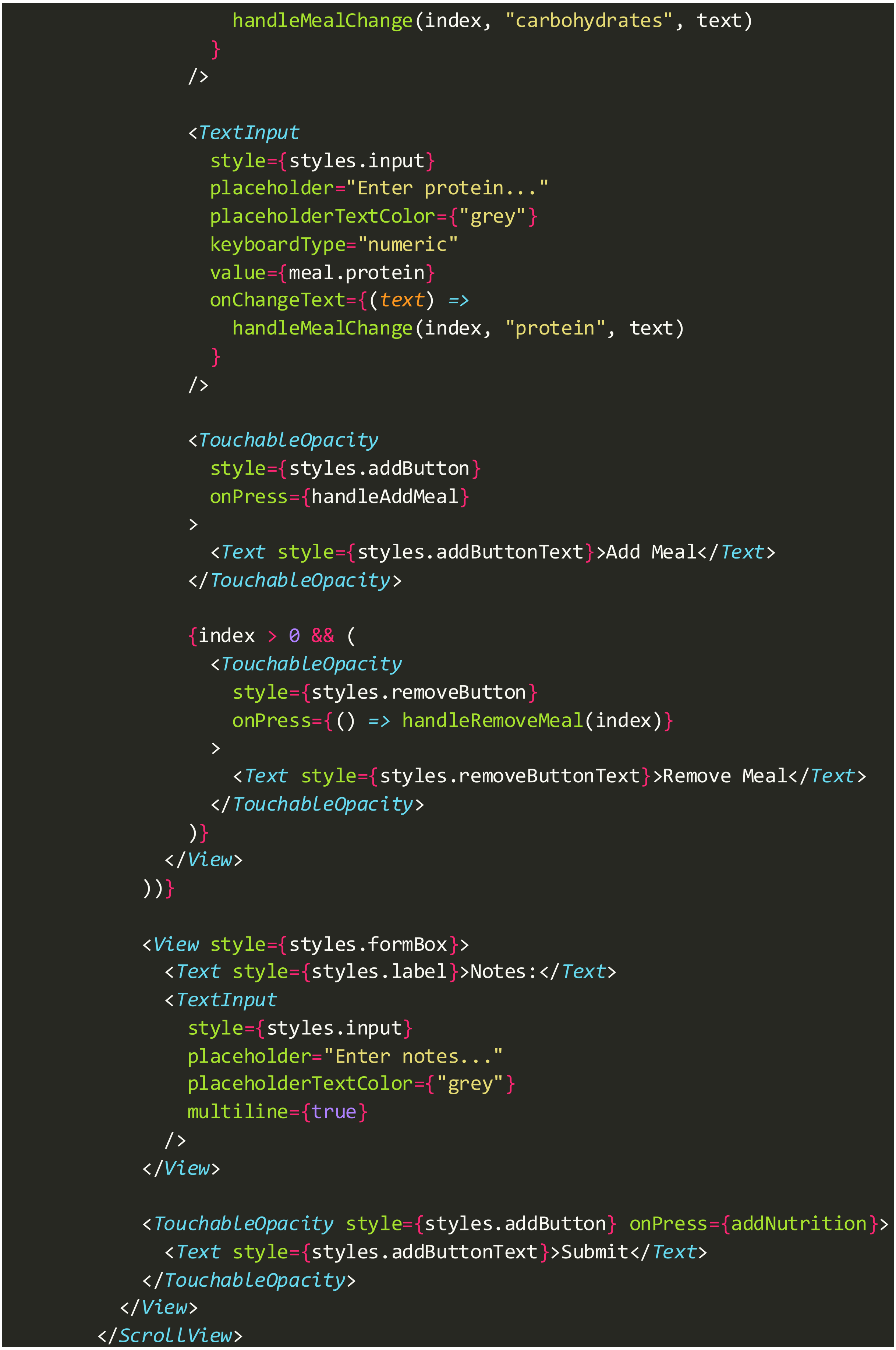
```
    calories: "",
    fat: "",
    carbohydrates: "",
    protein: "",
  },
]);

const handleAddMeal = () => {
  setMeals([
    ...meals,
    {
      name: "",
      servingSize: "",
      calories: "",
      fat: "",
      carbohydrates: "",
      protein: "",
    },
  ]);
};

const handleRemoveMeal = (index) => {
  const newMeals = [...meals];
  newMeals.splice(index, 1);
  setMeals(newMeals);
};

const handleMealChange = (index, field, value) => {
  const newMeals = [...meals];
  newMeals[index][field] = value;
  setMeals(newMeals);
};

//Create in Firesotre
const addNutrition = async () => {
  const user = getAuth().currentUser;
  const { email } = route.params;
  if (user) {
    try {
      const colRef = collection(db, "nutrition");
      addDoc(colRef, {
        date: date,
        mealPlanName: mealPlanName,
        meals: meals,
        createdAt: serverTimestamp(),
        client: email,
        trainer: user.email
      });
```



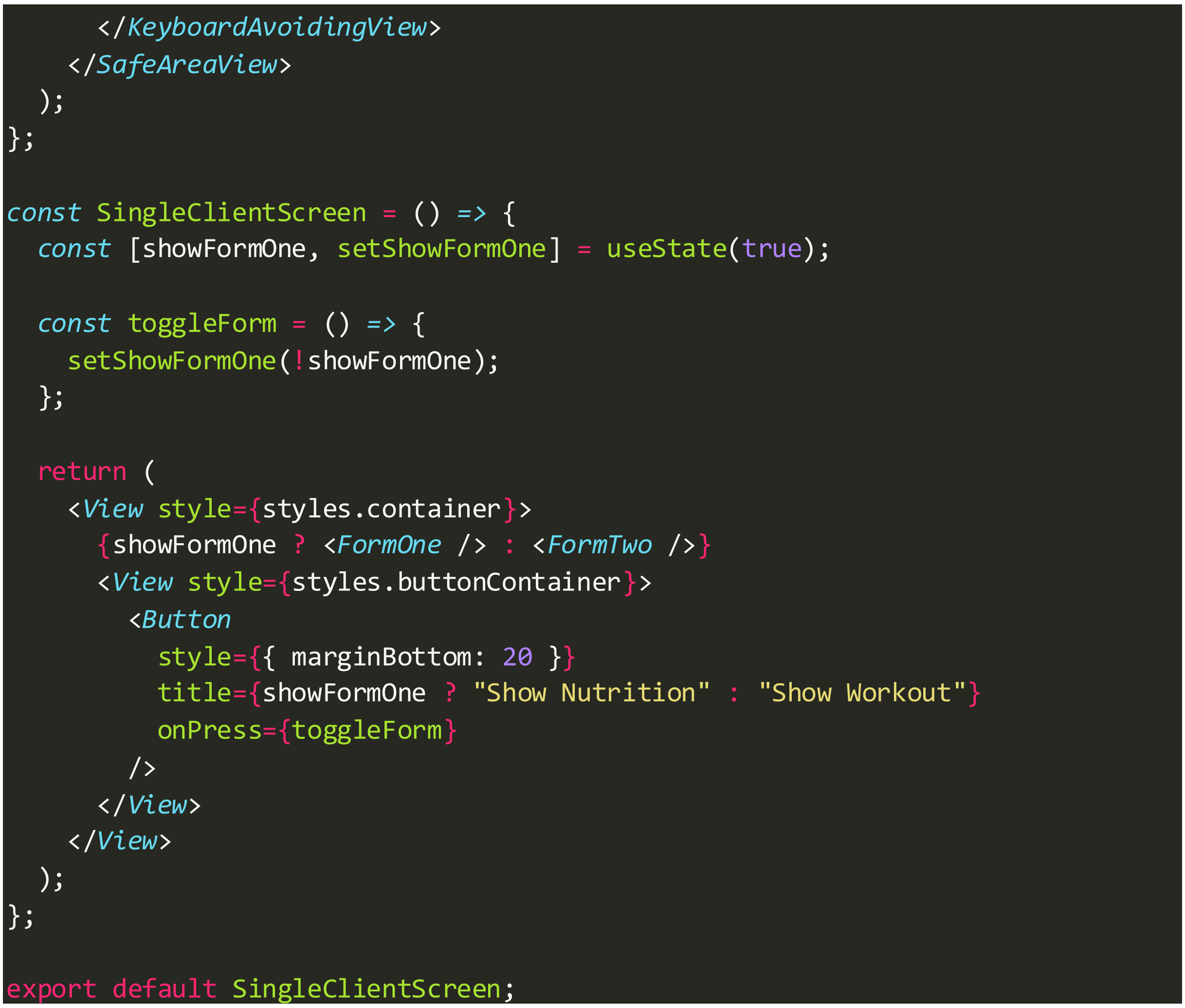
```
    } catch (e) {
      console.log(e);
    }

    setDate("");
    setMealPlanName("");
    setMeals([{ name: "" }]);
    navigation.goBack();
  }
};

return (
  <SafeAreaView style={styles.container}>
    <KeyboardAvoidingView
      behavior={Platform.OS === "ios" ? "padding" : "height"}
      style={styles.container}
    >
      <Text style={styles.title}>Record Nutrition</Text>
      <ScrollView
        contentContainerStyle={styles.scrollViewContent}
        nestedScrollEnabled={true}
      >
        <View style={styles.formWrapper}>
          <View style={styles.formBox}>
            <Text style={styles.label}>Date:</Text>
            <TextInput
              style={styles.input}
              placeholder="Enter date..."
              placeholderTextColor={"grey"}
              value={date}
              onChangeText={setDate}
            />
          </View>

          <View style={styles.formBox}>
            <Text style={styles.label}>Meal Plan Name:</Text>
            <TextInput
              style={styles.input}
              placeholder={"Enter meal plan name..."}
              placeholderTextColor={"grey"}
              value={mealPlanName}
              onChangeText={setMealPlanName}
            />
          </View>

          {meals.map((meal, index) => (
            <View key={index} style={styles.formBox}>
              <Text style={styles.label}>Meal {index + 1}:</Text>
```



```
                <TextInput
                  style={styles.input}
                  placeholder="Enter food name..."
                  placeholderTextColor={"grey"}
                  value={meal.name}
                  onChangeText={(text) => handleMealChange(index, "name",
text)}
                />

                <TextInput
                  style={styles.input}
                  placeholder="Enter serving size..."
                  placeholderTextColor={"grey"}
                  value={meal.servingSize}
                  onChangeText={(text) =>
                    handleMealChange(index, "servingSize", text)
                  }
                />

                <TextInput
                  style={styles.input}
                  placeholder="Enter calories..."
                  placeholderTextColor={"grey"}
                  keyboardType="numeric"
                  value={meal.calories}
                  onChangeText={(text) =>
                    handleMealChange(index, "calories", text)
                  }
                />

                <TextInput
                  style={styles.input}
                  placeholder="Enter fat..."
                  placeholderTextColor={"grey"}
                  keyboardType="numeric"
                  value={meal.fat}
                  onChangeText={(text) => handleMealChange(index, "fat",
text)}
                />

                <TextInput
                  style={styles.input}
                  placeholder="Enter carbohydrates..."
                  placeholderTextColor={"grey"}
                  keyboardType="numeric"
                  value={meal.carbohydrates}
                  onChangeText={(text) =>
```

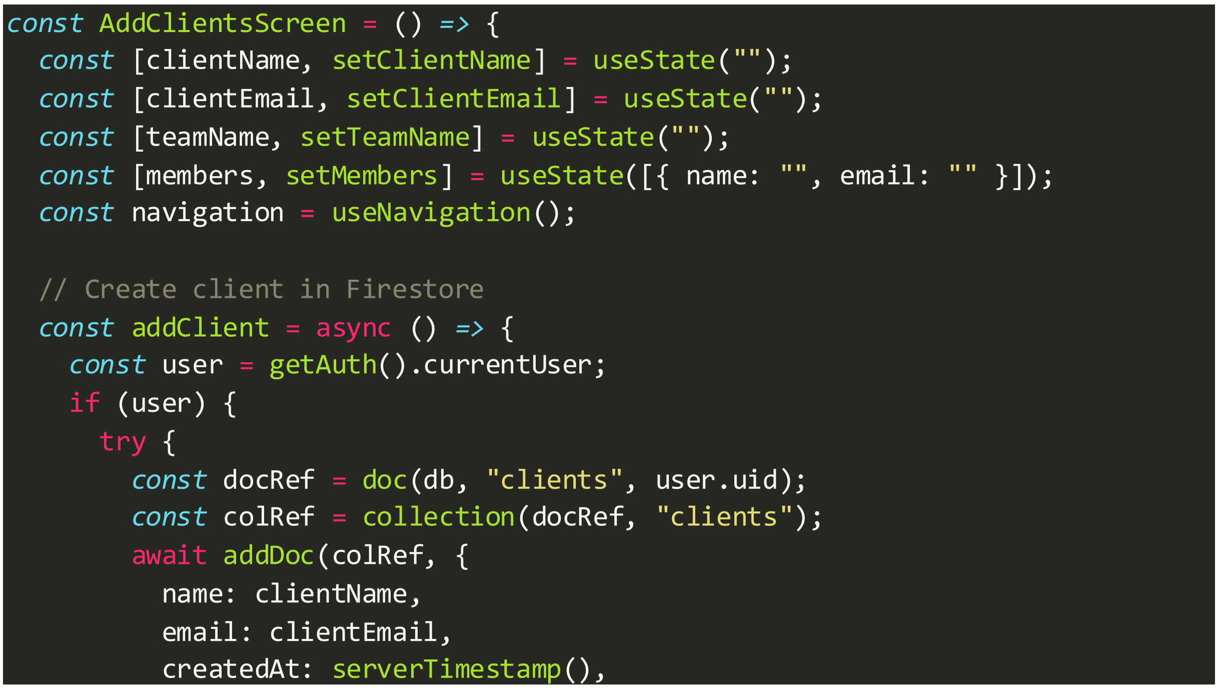


```
                    handleMealChange(index, "carbohydrates", text)
                  }
                />

                <TextInput
                  style={styles.input}
                  placeholder="Enter protein..."
                  placeholderTextColor={"grey"}
                  keyboardType="numeric"
                  value={meal.protein}
                  onChangeText={(text) =>
                    handleMealChange(index, "protein", text)
                  }
                />

                <TouchableOpacity
                  style={styles.addButton}
                  onPress={handleAddMeal}
                >
                  <Text style={styles.addButtonText}>Add Meal</Text>
                </TouchableOpacity>

                {index > 0 && (
                  <TouchableOpacity
                    style={styles.removeButton}
                    onPress={() => handleRemoveMeal(index)}
                  >
                    <Text style={styles.removeButtonText}>Remove Meal</Text>
                  </TouchableOpacity>
                )}
              </View>
            ))}

            <View style={styles.formBox}>
              <Text style={styles.label}>Notes:</Text>
              <TextInput
                style={styles.input}
                placeholder="Enter notes..."
                placeholderTextColor={"grey"}
                multiline={true}
              />
            </View>

            <TouchableOpacity style={styles.addButton} onPress={addNutrition}>
              <Text style={styles.addButtonText}>Submit</Text>
            </TouchableOpacity>
          </View>
        </ScrollView>
```

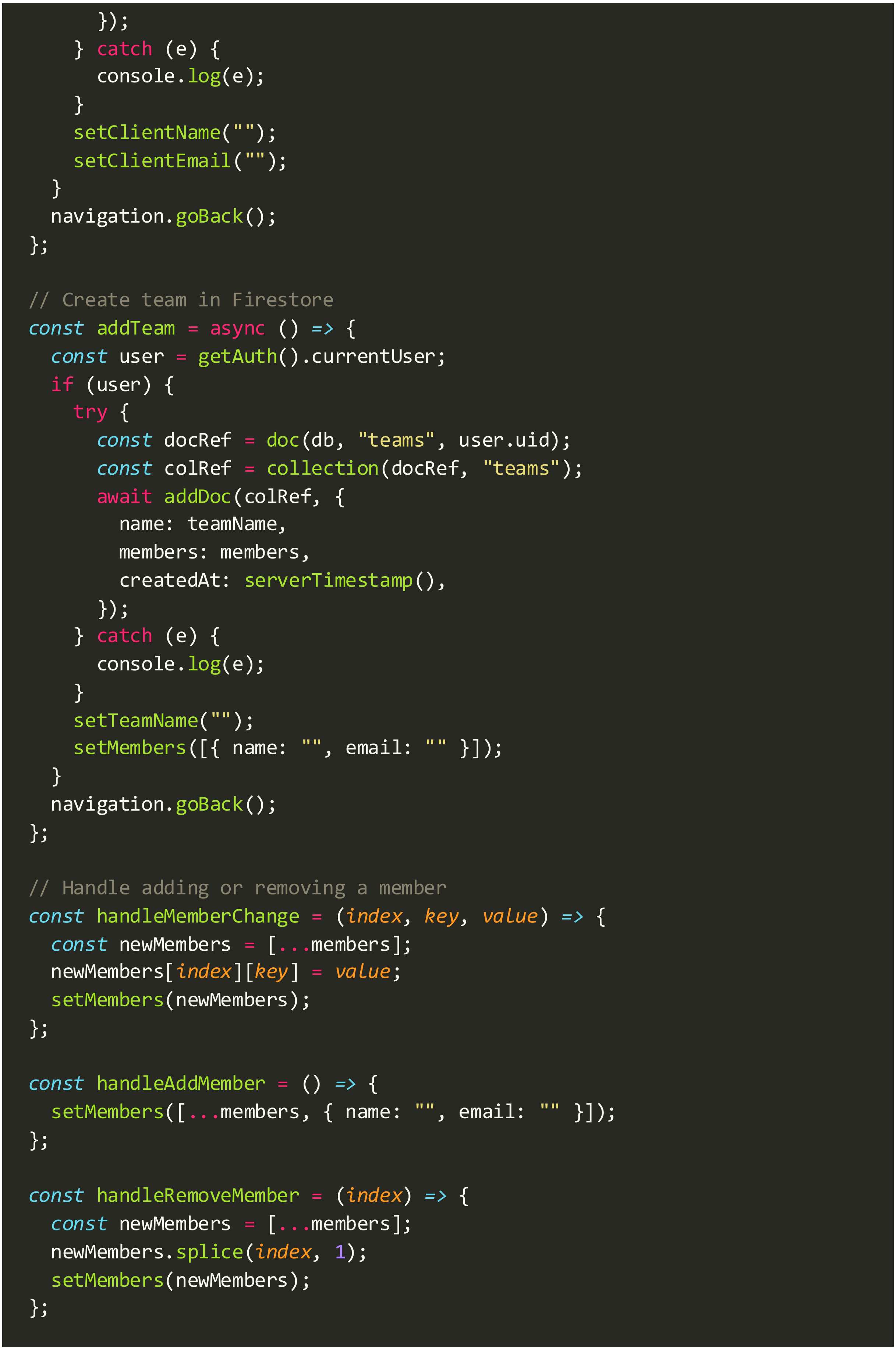


```
      </KeyboardAvoidingView>
    </SafeAreaView>
  );
};

const SingleClientScreen = () => {
  const [showFormOne, setShowFormOne] = useState(true);

  const toggleForm = () => {
    setShowFormOne(!showFormOne);
  };

  return (
    <View style={styles.container}>
      {showFormOne ? <FormOne /> : <FormTwo />}
      <View style={styles.buttonContainer}>
        <Button
          style={{ marginBottom: 20 }}
          title={showFormOne ? "Show Nutrition" : "Show Workout"}
          onPress={toggleForm}
        />
      </View>
    </View>
  );
};

export default SingleClientScreen;
```

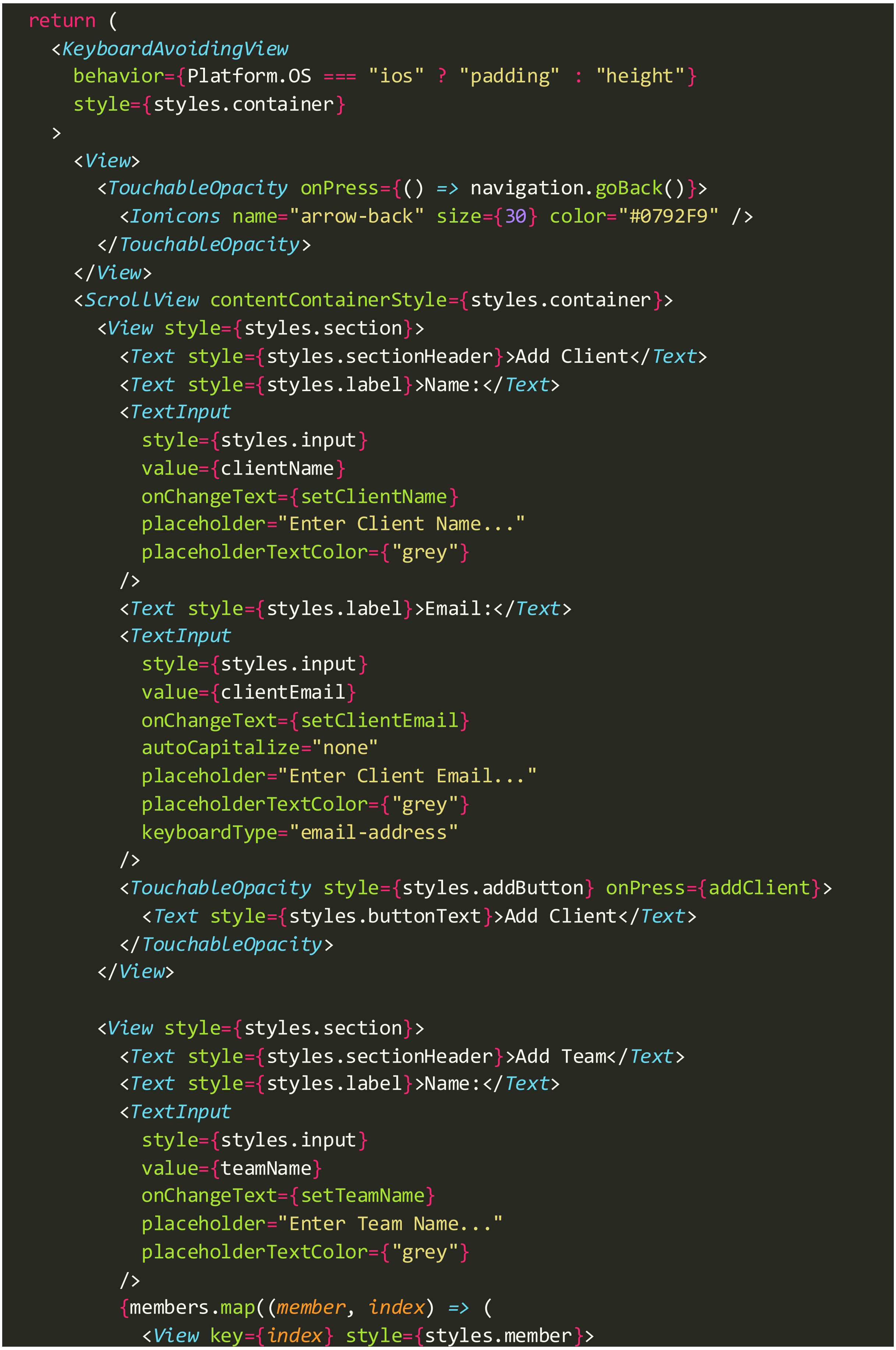
AddClients / Team Screen.js



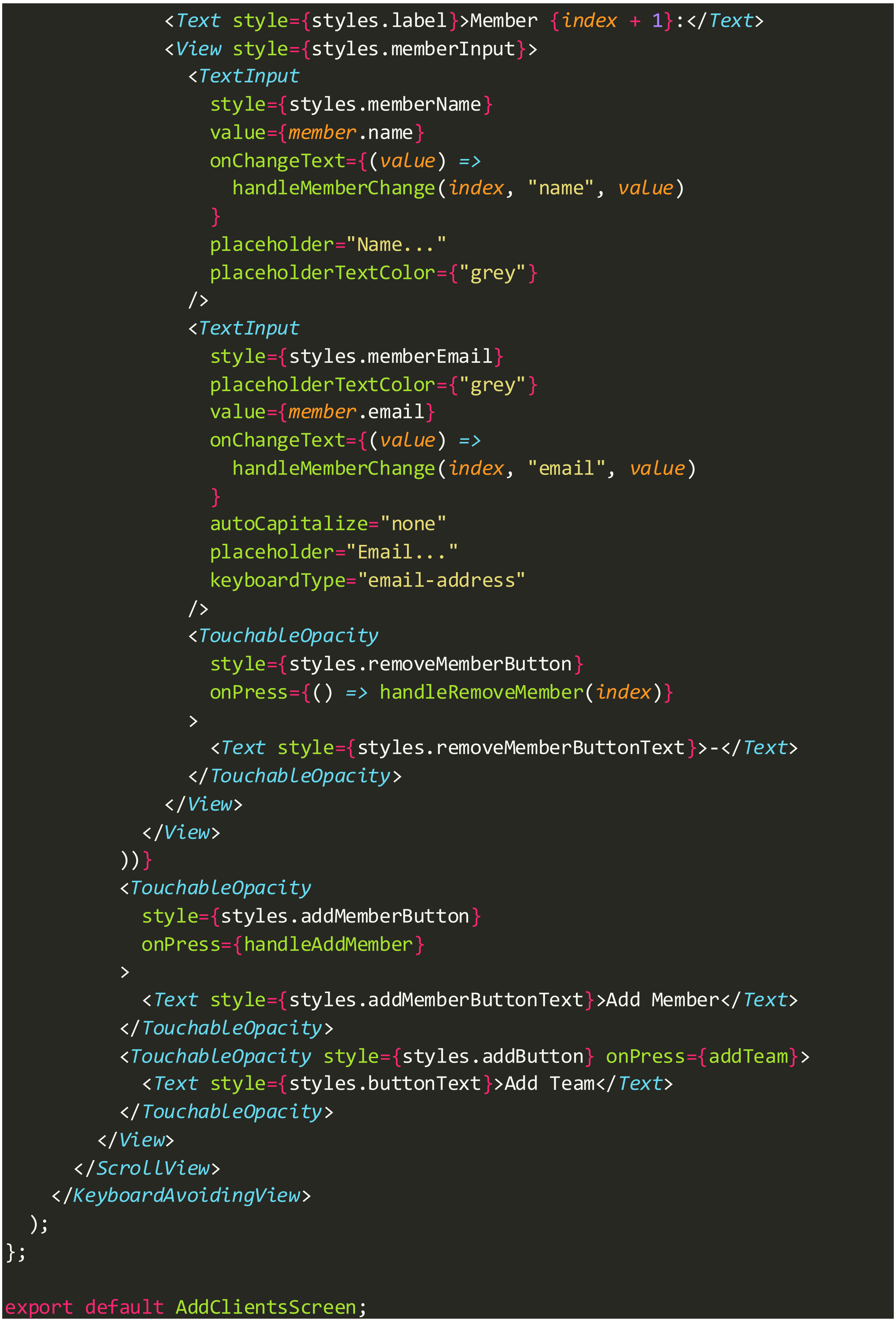
```
const AddClientsScreen = () => {
  const [clientName, setClientName] = useState("");
  const [clientEmail, setClientEmail] = useState("");
  const [teamName, setTeamName] = useState("");
  const [members, setMembers] = useState([{ name: "", email: "" }]);
  const navigation = useNavigation();

  // Create client in Firestore
  const addClient = async () => {
    const user = getAuth().currentUser;
    if (user) {
      try {
        const docRef = doc(db, "clients", user.uid);
        const colRef = collection(docRef, "clients");
        await addDoc(colRef, {
          name: clientName,
          email: clientEmail,
          createdAt: serverTimestamp(),
```



```
      });
    } catch (e) {
      console.log(e);
    }
    setClientName("");
    setClientEmail("");
  }
  navigation.goBack();
};

// Create team in Firestore
const addTeam = async () => {
  const user = getAuth().currentUser;
  if (user) {
    try {
      const docRef = doc(db, "teams", user.uid);
      const colRef = collection(docRef, "teams");
      await addDoc(colRef, {
        name: teamName,
        members: members,
        createdAt: serverTimestamp(),
      });
    } catch (e) {
      console.log(e);
    }
    setTeamName("");
    setMembers([{ name: "", email: "" }]);
  }
  navigation.goBack();
};

// Handle adding or removing a member
const handleMemberChange = (index, key, value) => {
  const newMembers = [...members];
  newMembers[index][key] = value;
  setMembers(newMembers);
};

const handleAddMember = () => {
  setMembers([...members, { name: "", email: "" }]);
};

const handleRemoveMember = (index) => {
  const newMembers = [...members];
  newMembers.splice(index, 1);
  setMembers(newMembers);
};
```



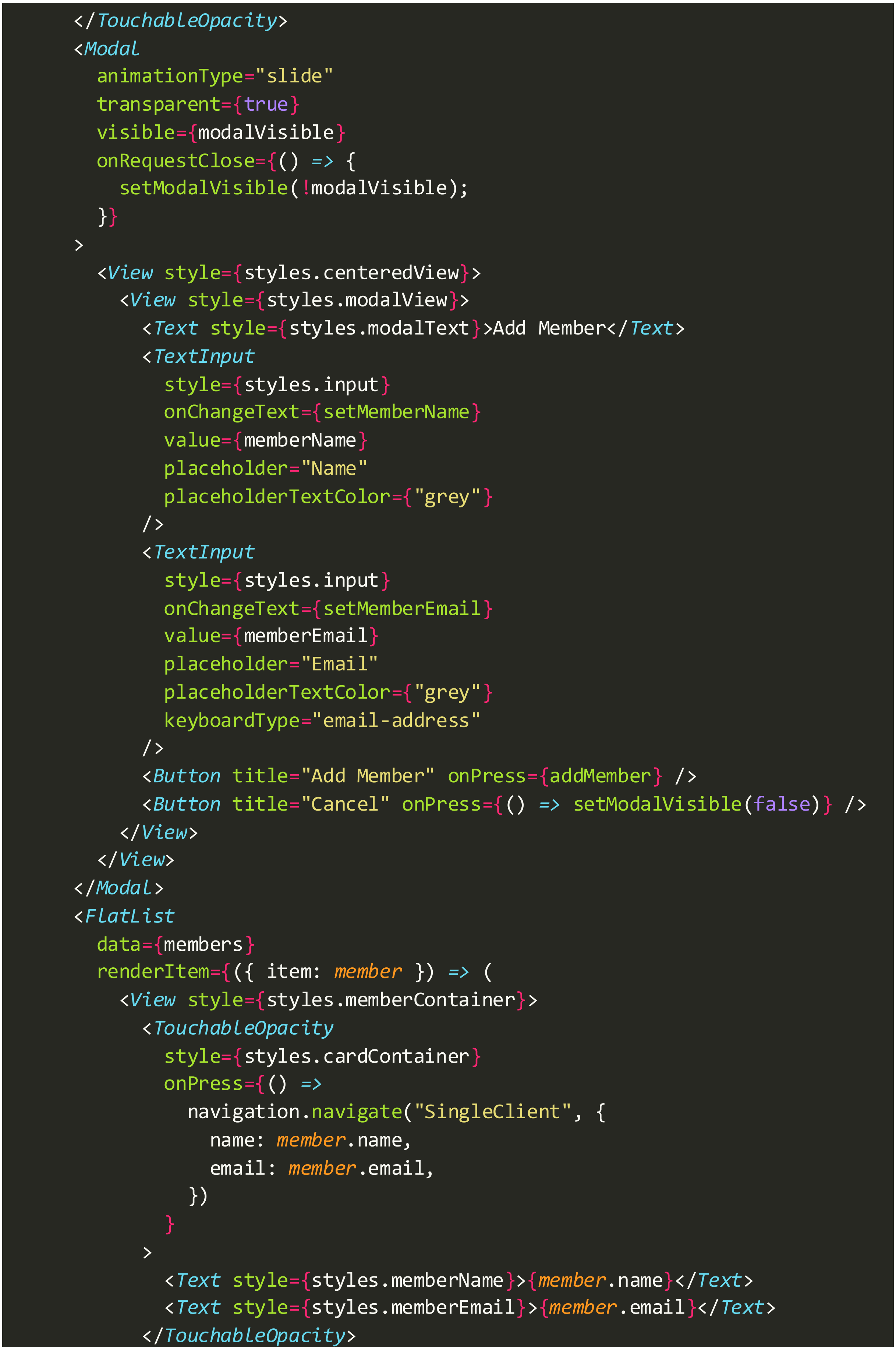
```
  return (
    <KeyboardAvoidingView
      behavior={Platform.OS === "ios" ? "padding" : "height"}
      style={styles.container}
    >
      <View>
        <TouchableOpacity onPress={() => navigation.goBack()}>
          <Ionicons name="arrow-back" size={30} color="#0792F9" />
        </TouchableOpacity>
      </View>
      <ScrollView contentContainerStyle={styles.container}>
        <View style={styles.section}>
          <Text style={styles.sectionHeader}>Add Client</Text>
          <Text style={styles.label}>Name:</Text>
          <TextInput
            style={styles.input}
            value={clientName}
            onChangeText={setClientName}
            placeholder="Enter Client Name..."
            placeholderTextColor={"grey"}
          />
          <Text style={styles.label}>Email:</Text>
          <TextInput
            style={styles.input}
            value={clientEmail}
            onChangeText={setClientEmail}
            autoCapitalize="none"
            placeholder="Enter Client Email..."
            placeholderTextColor={"grey"}
            keyboardType="email-address"
          />
          <TouchableOpacity style={styles.addButton} onPress={addClient}>
            <Text style={styles.buttonText}>Add Client</Text>
          </TouchableOpacity>
        </View>

        <View style={styles.section}>
          <Text style={styles.sectionHeader}>Add Team</Text>
          <Text style={styles.label}>Name:</Text>
          <TextInput
            style={styles.input}
            value={teamName}
            onChangeText={setTeamName}
            placeholder="Enter Team Name..."
            placeholderTextColor={"grey"}
          />
          {members.map((member, index) => (
            <View key={index} style={styles.member}>
```



```
              <Text style={styles.label}>Member {index + 1}:</Text>
              <View style={styles.memberInput}>
                <TextInput
                  style={styles.memberName}
                  value={member.name}
                  onChangeText={(value) =>
                    handleMemberChange(index, "name", value)
                  }
                  placeholder="Name..."
                  placeholderTextColor={"grey"}
                />
                <TextInput
                  style={styles.memberEmail}
                  placeholderTextColor={"grey"}
                  value={member.email}
                  onChangeText={(value) =>
                    handleMemberChange(index, "email", value)
                  }
                  autoCapitalize="none"
                  placeholder="Email..."
                  keyboardType="email-address"
                />
                <TouchableOpacity
                  style={styles.removeMemberButton}
                  onPress={() => handleRemoveMember(index)}
                >
                  <Text style={styles.removeMemberButtonText}>-</Text>
                </TouchableOpacity>
              </View>
            </View>
          ))}
          <TouchableOpacity
            style={styles.addMemberButton}
            onPress={handleAddMember}
          >
            <Text style={styles.addMemberButtonText}>Add Member</Text>
          </TouchableOpacity>
          <TouchableOpacity style={styles.addButton} onPress={addTeam}>
            <Text style={styles.buttonText}>Add Team</Text>
          </TouchableOpacity>
        </View>
      </ScrollView>
    </KeyboardAvoidingView>
  );
};

export default AddClientsScreen;
```

TeamScreen.js

```
const TeamScreen = ({ route }) => {
  const { id, name, members } = route.params;
  const navigation = useNavigation();
  const [modalVisible, setModalVisible] = useState(false);
  const [memberName, setMemberName] = useState("");
  const [memberEmail, setMemberEmail] = useState("");

  const addMember = async () => {
    if (memberName.trim() !== "" && memberEmail.trim() !== "") {
      const newMember = { name: memberName, email: memberEmail };

      // Get current user
      const userCred = getAuth().currentUser;

      // Update the team members in Firestore database
      const teamRef = doc(db, "teams", userCred.uid, "teams", id);

      try {
        await updateDoc(teamRef, {
          members: arrayUnion(newMember),
        });
        setModalVisible(false);
        setMemberName("");
        setMemberEmail("");
        navigation.goBack();
      } catch (error) {
        console.error("Error updating team members: ", error);
      }
    }
  };

  return (
    <SafeAreaView style={styles.container}>
      <TouchableOpacity onPress={() => navigation.goBack()} style={{ left: 20
}}>
        <Ionicons name="arrow-back" size={30} color="#0792F9" />
      </TouchableOpacity>
      <Text style={styles.title}>{name}</Text>
      <TouchableOpacity style={styles.button} onPress={() =>
navigation.navigate("AssignAll", {
        members: members,
      })}>
        <Text style={styles.buttonText}>Assign All</Text>
      </TouchableOpacity>
      <TouchableOpacity style={styles.button} onPress={() =>
setModalVisible(true)}>
        <Text style={styles.buttonText}>Add Members</Text>
```



```
      </TouchableOpacity>
      <Modal
        animationType="slide"
        transparent={true}
        visible={modalVisible}
        onRequestClose={() => {
          setModalVisible(!modalVisible);
        }}
      >
        <View style={styles.centeredView}>
          <View style={styles.modalView}>
            <Text style={styles.modalText}>Add Member</Text>
            <TextInput
              style={styles.input}
              onChangeText={setMemberName}
              value={memberName}
              placeholder="Name"
              placeholderTextColor={"grey"}
            />
            <TextInput
              style={styles.input}
              onChangeText={setMemberEmail}
              value={memberEmail}
              placeholder="Email"
              placeholderTextColor={"grey"}
              keyboardType="email-address"
            />
            <Button title="Add Member" onPress={addMember} />
            <Button title="Cancel" onPress={() => setModalVisible(false)} />
          </View>
        </View>
      </Modal>
      <FlatList
        data={members}
        renderItem={({ item: member }) => (
          <View style={styles.memberContainer}>
            <TouchableOpacity
              style={styles.cardContainer}
              onPress={() =>
                navigation.navigate("SingleClient", {
                  name: member.name,
                  email: member.email,
                })
              }
            >
              <Text style={styles.memberName}>{member.name}</Text>
              <Text style={styles.memberEmail}>{member.email}</Text>
            </TouchableOpacity>
          </View>
        )}
      />
    </SafeAreaView>
  );
};

export default TeamScreen;
```