

## App Development (Code Review)

### Table of Contents

<b>Introduction .....</b>	<b>1</b>
<b>Feedback and Suggestions: .....</b>	<b>1</b>
<b>Conclusion: .....</b>	<b>4</b>

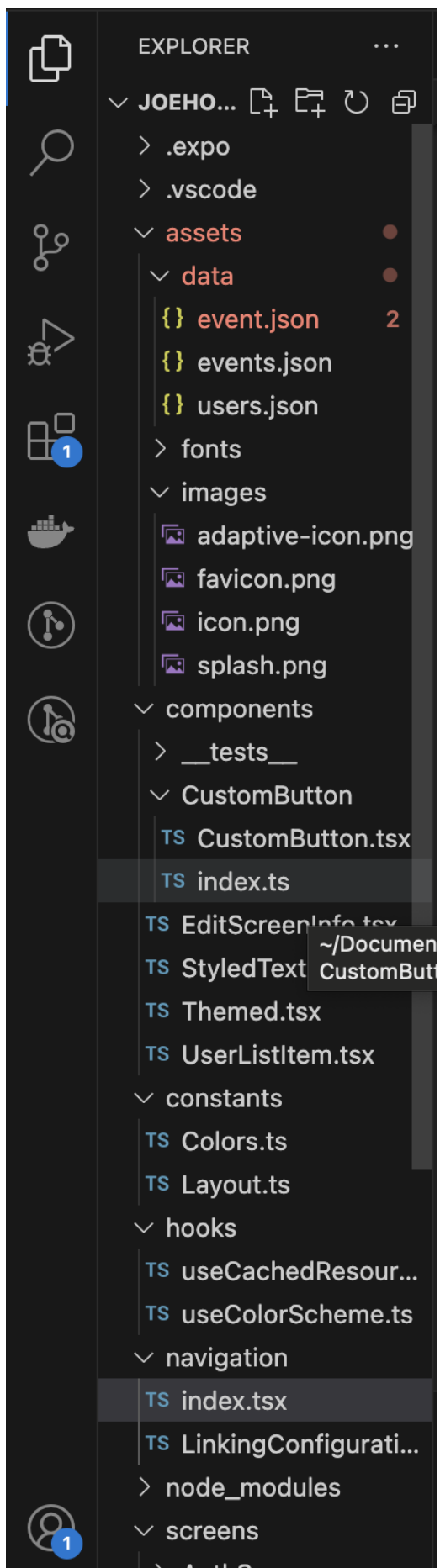
### Introduction

I was the main developer in the group, so my task was to create an app for making events and socialize with people during our lunch break. The provided code is a React Native navigation setup using React Navigation. It defines the navigation structure of the app, including authentication, modals, and a bottom tab navigator with two main screens. The code also incorporates some additional features, such as a custom tab bar icon and a pressable user icon in the header.

### Feedback and Suggestions:

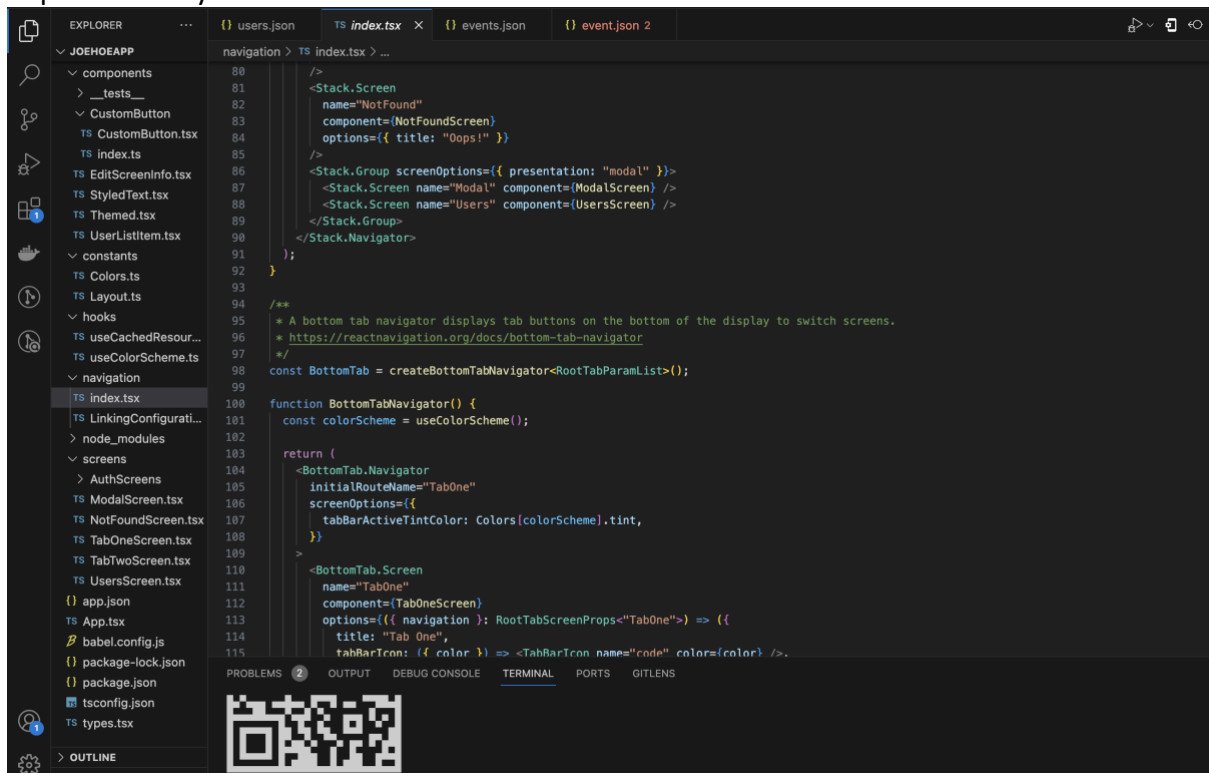
#### 1. Code Organization:

The code is well-organized, separating concerns into different files and components, enhancing readability.



## 2. Authentication Handling:

The authentication check (`isAuthenticated`) is hard-coded. It would be beneficial to implement a dynamic check based on the actual authentication status.



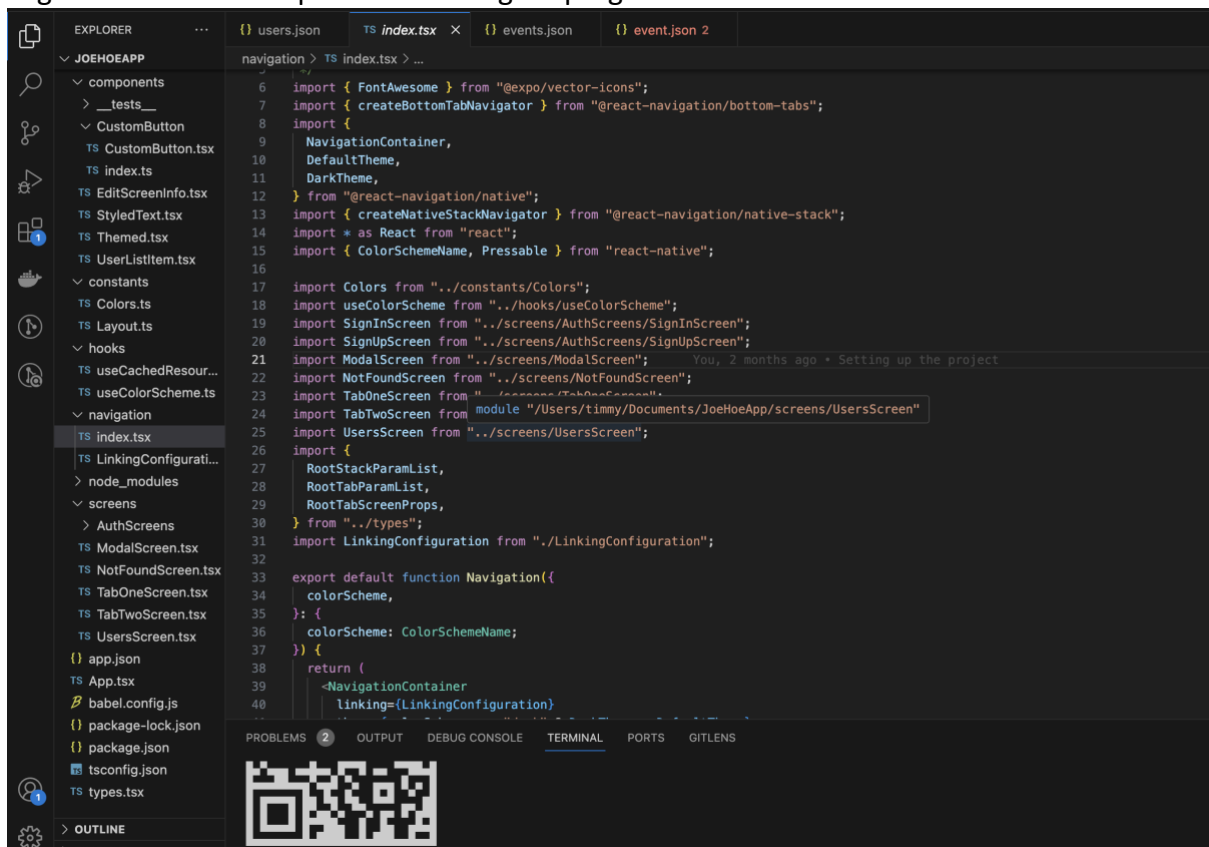
```
navigation > TS index.tsx > ...
80 //>
81 <Stack.Screen
82   name="NotFound"
83   component={NotFoundScreen}
84   options={{ title: "Oops!" }}
85 />
86 <Stack.Group screenOptions={{ presentation: "modal" }}>
87   <Stack.Screen name="Modal" component={ModalScreen} />
88   <Stack.Screen name="Users" component={UsersScreen} />
89 </Stack.Group>
90 </Stack.Navigator>
91 );
92 }
93
94 /**
95  * A bottom tab navigator displays tab buttons on the bottom of the display to switch screens.
96  * https://reactnavigation.org/docs/bottom-tab-navigator
97  */
98 const BottomTab = createBottomTabNavigator<RootTabParamList>();
99
100 function BottomTabNavigator() {
101   const colorScheme = useColorScheme();
102
103   return (
104     <BottomTab.Navigator
105       initialRouteName="TabOne"
106       screenOptions={{
107         tabBarActiveTintColor: Colors[colorScheme].tint,
108       }}
109     >
110       <BottomTab.Screen
111         name="TabOne"
112         component={TabOneScreen}
113         options={({ navigation }) => ({
114           title: "Tab One",
115           tabBarIcon: ({ color }) => <TabBarIcon name="code" color={color} />,
116         })}
117       />
118     </BottomTab.Navigator>
119   );
120 }
```

## 3. Conditional Rendering:

The conditional rendering for authentication could be further improved. Consider redirecting unauthenticated users to a login screen instead of having separate navigation stacks.

## 4. Navigation Structure:

The navigation structure is clear and concise. However, ensure that the app's navigation flow aligns with the user experience and logical progression.



## 5. Styling and Theming:

Styling is consistent and follows a theme based on the color scheme. Make sure to maintain this consistency throughout the entire application.

## 6. Pressable Component:

The usage of Pressable for the user icon is a good choice for incorporating interactivity. Ensure that this interaction is intuitive for users.

## 7. Custom Tab Bar Icon:

The custom tab bar icon is a nice touch, providing a visual representation of the tab. Ensure that it aligns with the overall design language of the app.

## Conclusion:

The code demonstrates a solid foundation for navigation in a React Native application. The organization, styling, and additional features contribute to a positive user experience. Addressing the mentioned points will enhance the overall code quality and maintainability. Great job!