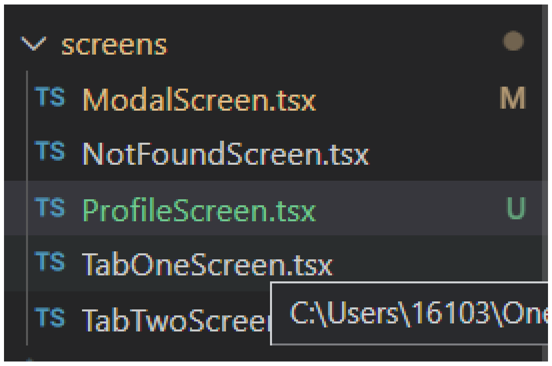
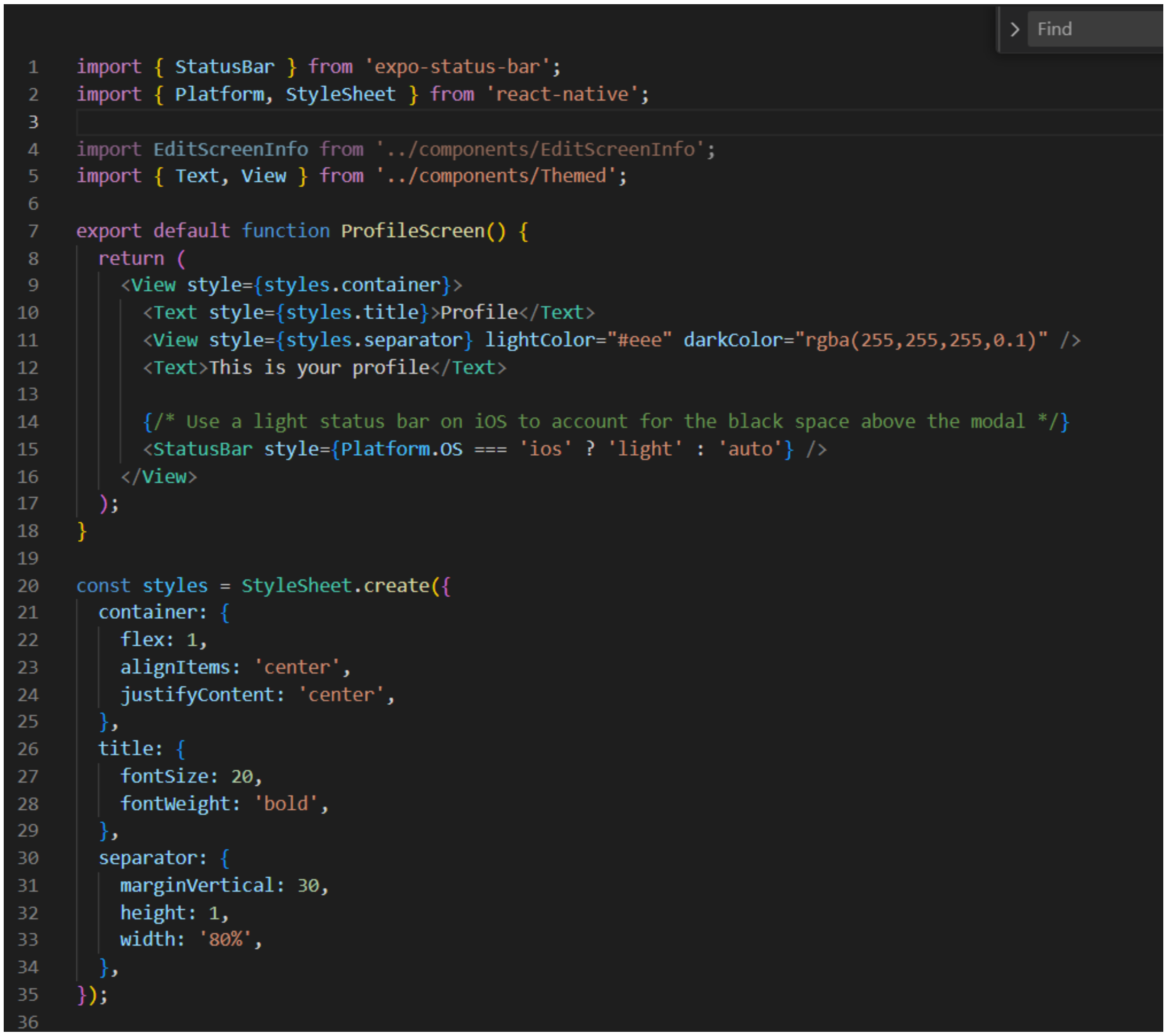
Getting the Screen to Work:

In order to create a new screen, you will have to first create a new file called {name}screen.tsx. For now I would recommend copy and pasting the basic layout of another screen. Make this file in the screens folder. For this guide, I will be making ProfileScreen.tsx.





```
import { StatusBar } from 'expo-status-bar';
import { Platform, StyleSheet } from 'react-native';

import EditScreenInfo from '../components/EditScreenInfo';
import { Text, View } from '../components/Themed';

export default function ProfileScreen() {
  return (
    <View style={styles.container}>
      <Text style={styles.title}>Profile</Text>
      <View style={styles.separator} lightColor="#eee" darkColor="rgba(255,255,255,0.1)" />
      <Text>This is your profile</Text>

      {/* Use a light status bar on iOS to account for the black space above the modal */}
      <StatusBar style={Platform.OS === 'ios' ? 'light' : 'auto'} />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',
  },
  separator: {
    marginVertical: 30,
    height: 1,
    width: '80%',
  },
});

```

Next, we will have to define the screen in types.tsx. It should be at the very bottom of your directory in VSCode. Here you will see multiple exports. Two of which being:

```
export type RootStackParamList = {
```

and

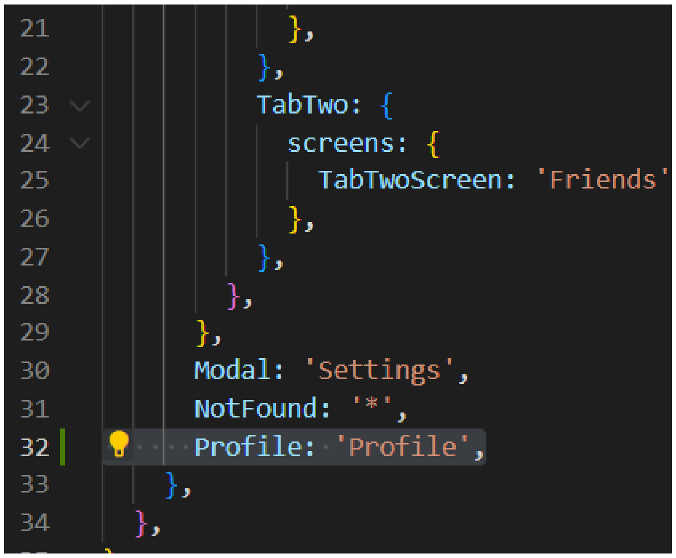
```
export type RootTabParamList = {
```

Under RootStackParamList is where we can store most screens. Under RootTabParamList is where the most important tabs are. I believe if you want a tab at the bottom for your screen you will need to have it here. You also need to have it here if you want the screen to pop op first. For ProfileScreen I will put define it in RootStackParamList the following way:

```
export type RootStackParamList = {
  Root: NavigatorScreenParams<RootTabParamList> | undefined;
  Modal: undefined;
  Profile: undefined;
  NotFound: undefined;
};
```

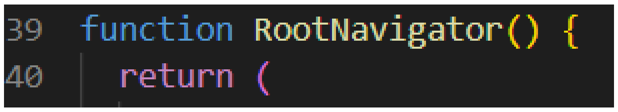
I believe you can just put “undefined” I do not know another definition use or reason why to put it as undefined but let me know if there is a different use.

Next, in LinkingConfigurations.ts, under the structure, where you find Modal and NotFound, link the screem as follows:



```
              },
            },
            TabTwo: {
              screens: {
                TabTwoScreen: 'Friends'
              },
            },
          },
        },
        Modal: 'Settings',
        NotFound: '*',
        Profile: 'Profile',
      },
    },
```

Going into index.tssx, we will need to define what type of screen this will be. Go to:



```
function RootNavigator() {
  return (
```

Go into the Stack.Navigator you will want to define the kind of screen. For a pop-up-like screen we will be using a “modal” style. This is shown in the template as:

```
      <Stack.Group screenOptions={{ presentation: 'modal' }}>
        <Stack.Screen name="Modal" component={ModalScreen} />
      </Stack.Group>
```

In order to show the Profile screen as a modal, I would do the following:

```
      <Stack.Group screenOptions={{ presentation: 'modal' }}>
        <Stack.Screen name="Profile" component={ProfileScreen} />
      </Stack.Group>
```

If I did not want to have it be a modal, I wanted it to be just a normal screen, I would design it just like the NotFound screen:

```
      <Stack.Screen name="NotFound" component={NotFoundScreen} options={{ title: 
'Oops!' }} />
```

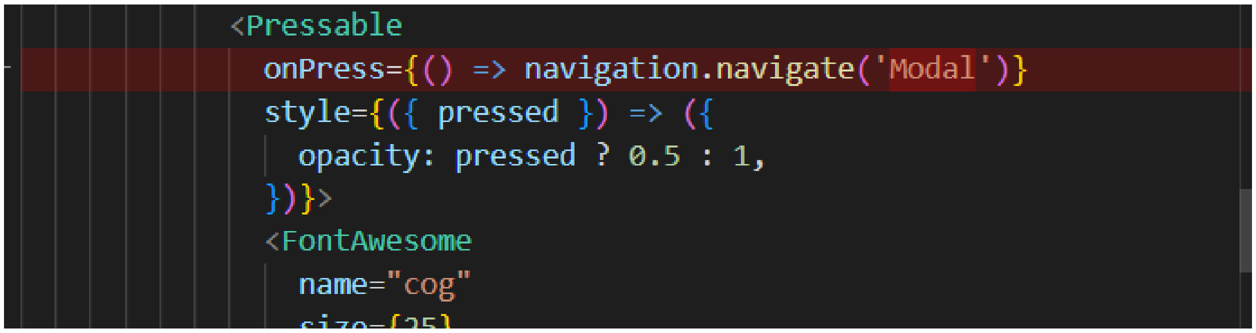
I will format the line as the following if I want the profile to be a full screen:

```
      <Stack.Screen name="Profile" component={ProfileScreen} options={{ title: 
'Profile' }} />
```

Testing the Screen Using the Template

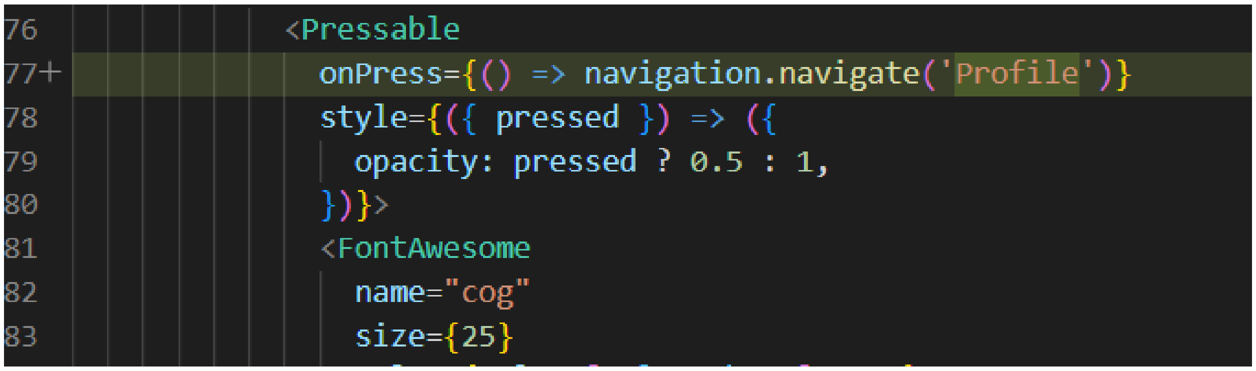
The fastest and easiest way to see your screen is to go into index.tsx and replace the modal with your screens identifier.

The original will look like this:



```
<Pressable
  onPress={() => navigation.navigate('Modal')}
  style={({ pressed }) => ({
    opacity: pressed ? 0.5 : 1,
  })}>
  <FontAwesome
    name="cog"
    size={25}
```

Replacing will look like this:



```
<Pressable
  onPress={() => navigation.navigate('Profile')}
  style={({ pressed }) => ({
    opacity: pressed ? 0.5 : 1,
  })}>
  <FontAwesome
    name="cog"
    size={25}
```

Incorporating Buttons with Screen Changes

In the screen in which you want the button, make a button, pressable, or touchable opacity.

With in the opening bracket (this is a TouchableOpacitty for example), paste the following:

```
<TouchableOpacity onPress={() => navigation.replace('Profile')}
style={styles.link}>
```

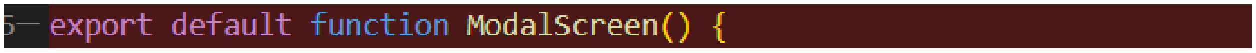
The only change you will make is replacing ‘NotFound’ or ‘Profile’ with the identifier of your screen

You will also need to import one thing into your screen:

```
import { RootStackScreenProps } from '../types';
```

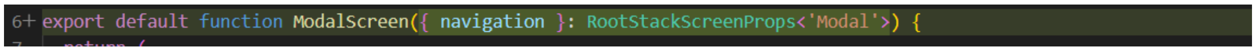
The final thing you will need to do is:

In the screen you have the button on, add to the function name to show the following change:



```
export default function ModalScreen() {
```

To



```
export default function ModalScreen({ navigation }: RootStackScreenProps<'Modal'>) {
```

The name in the ' ' should match the identifier we have been using.