PRO-C185

WhiteHat Jr
Live Online Coding for Kids

COMPLETING FRAPP

What is our GOAL for this MODULE?

We revised the concepts of stack navigation in react-native to complete the face recognition app.

What did we ACHIEVE in the class TODAY?

- We learned to add stack navigation to complete FRApp

Which CONCEPTS/CODING BLOCKS did we cover today?

- expo-permissions
- expo-camera
- expo-face-detector
- <Camera/>, <SafeAreaView/>,<ScrollView/>,<Platform/> components
- react-native-responsive-fontsize library
- RFPercentage, RFValue
- createStackNavigator

How did we DO the activities?

1. Install dependencies for navigation
    - expo install react-native-gesture-handler
    - expo install react-native-reanimated
    - expo install react-native-screens
    - expo install react-native-safe-area-context
    - expo install @react-native-community/masked-view
2. Create Stack Navigator:

```
import 'react-native-gesture-handler';
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';

import Home from "./screens/Home";
import Main from "./screens/Main";

const Stack = createStackNavigator();

function App() {
 return (
   <NavigationContainer>
     <Stack.Navigator initialRouteName="Home" screenOptions={{
       headerShown: false
     }}>
       <Stack.Screen name="Home" component={Home} />
       <Stack.Screen name="Main" component={Main} />
     </Stack.Navigator>
   </NavigationContainer>
 );
}

export default App;
```


3. Create a component Home inside the screens folder.

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

export default class Home extends Component {
   render() {
       return (
           <View
               style={{
                   flex: 1,
                   justifyContent: "center",
                   alignItems: "center"
               }}>
               <Text>Home Screen!</Text>
           </View>
       )
   }
}
```

4. Create the header of the screen with App Name and App's Catch Phrase and add a style to the components.

```
            <View style={styles.container}>
               <SafeAreaView style={styles.droidSafeArea} />
               <View style={styles.headingContainer}>
                   <View style={{ flexDirection: 'row', flexWrap: 'wrap' }}>
                       <Text style={styles.titleText1}>FR</Text><Text
style={styles.titleText2}>APP</Text>
                   </View>
                   <View style={{ flexDirection: 'row', flexWrap: 'wrap' }}>
                       <Text style={styles.subheading1}>Try Out</Text><Text
style={styles.subheading2}> Cool Frames</Text>
                   </View>
               </View>
            </View>
```

```
container: {
      flex: 1,
      backgroundColor: "#6278e4"
  },
  droidSafeArea: {
      marginTop: Platform.OS === "android" ? StatusBar.currentHeight : 0
  },
  headingContainer: {
      flex: 0.2,
      alignItems: 'center',
      justifyContent: 'center'
  },
  titleText1: {
      fontSize: RFValue(30),
      fontWeight: "bold",
      color: "#efb141",
      fontStyle: 'italic',
      textShadowColor: 'rgba(0, 0, 0, 0.75)',
      textShadowOffset: { width: -3, height: 3 },
      textShadowRadius: 1
  },
  titleText2: {
      fontSize: RFValue(30),
      fontWeight: "bold",
      color: "white",
```

```
      fontStyle: 'italic',
      textShadowColor: 'rgba(0, 0, 0, 0.75)',
      textShadowOffset: { width: -3, height: 3 },
      textShadowRadius: 1
  },
  subheading1: {
      fontSize: RFValue(20),
      color: "#efb141",
      fontStyle: 'italic',
      textShadowColor: 'rgba(0, 0, 0, 0.75)',
      textShadowOffset: { width: -3, height: 3 },
      textShadowRadius: 1
  },
  subheading2: {
      fontSize: RFValue(20),
      color: "white",
      fontStyle: 'italic',
      textShadowColor: 'rgba(0, 0, 0, 0.75)',
      textShadowOffset: { width: -3, height: 3 },
      textShadowRadius: 1
  },
```

5. Create the second section, for text and images.

```
<View style={styles.contentContainer}>
                    <View style={{ flex: 0.5 }}>
                        <Text style={styles.contentText}>Experience the virtual
experience of trying out different frames from our wide collection on your mobile phone
before making any purchase, just how you would in an offline store!</Text>
                    </View>
                    <View style={{ flexDirection: "row", flex: 0.25 }}>
                        <View style={{ flex: 0.5 }}>
                            <Image source={require('../assets/Frapp-03.png')} style={{
height: 64, width: 160 }} />
                        </View>
                        <View style={{ flex: 0.5 }}>
```

```
                            <Image source={require('../assets/Frapp-09.png')} style={{
height: 64, width: 160 }} />
                        </View>
                    </View>
                    <View style={{ flexDirection: "row", flex: 0.25 }}>
                        <View style={{ flex: 0.5 }}>
                            <Image source={require('../assets/Frapp-02.png')} style={{
height: 64, width: 160 }} />
                        </View>
                        <View style={{ flex: 0.5 }}>
                            <Image source={require('../assets/Frapp-08.png')} style={{
height: 64, width: 160 }} />
                        </View>
                    </View>
                </View>
```

```
contentContainer: {
        flex: 0.6,
        margin: RFValue(5),
        borderRadius: RFValue(15),
        backgroundColor: "white",
        height: "100%",
        padding: RFValue(20)
    },
    contentText: {
        fontSize: RFValue(17),
        fontStyle: 'italic',
        fontWeight: "bold"
    },
```

6. Create the final section with the button.

```
<View style={styles.buttonContainer}>
                    <TouchableOpacity onPress={() =>
this.props.navigation.navigate("Main")}>
                        <View style={styles.button}>
                            <Text style={styles.buttonText}>Try Now!</Text>
                        </View>
                    </TouchableOpacity>
                </View>
```

7. Add style to the button

```
buttonContainer: {
      flex: 0.2,
      justifyContent: "center",
      alignItems: "center"
  },
  button: {
      backgroundColor: "#efb141",
      paddingLeft: RFValue(50),
      paddingRight: RFValue(50),
      paddingTop: RFValue(20),
      paddingBottom: RFValue(20),
      borderRadius: RFValue(20)
  },
  buttonText: {
      fontSize: RFValue(25),
      fontStyle: 'italic',
      color: "white",
      textShadowColor: 'rgba(0, 0, 0, 0.75)',
      textShadowOffset: { width: -1, height: 1 },
      textShadowRadius: 1
  }
```

8. Add the navigation on the onPress() event of our <TouchableOpacity/>

```
<View style={styles.buttonContainer}>
                    <TouchableOpacity onPress={() =>
this.props.navigation.navigate("Main")}>
                        <View style={styles.button}>
                            <Text style={styles.buttonText}>Try Now!</Text>
                        </View>
                    </TouchableOpacity>
                </View>
```

9. Test the output.

We have successfully learned to apply face filters using the data received from the FaceDetector API expo module.

What's NEXT?

In the next class, we will be revising some of the concepts we had learned earlier for Virtual Reality, so we can create something completely out of our own imagination!

EXTEND YOUR KNOWLEDGE:

- You can refer to the link below to explore more about expo FaceDetector
  expo FaceDetector
- You can refer to the link below to explore more about Stack Navigation
  Stack Navigation