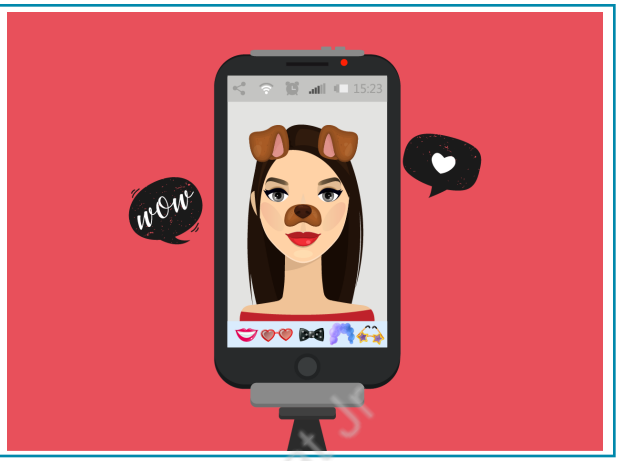
PRO-C183

MULTIPLE FILTERS TRYOUT



What is our GOAL for this MODULE?

We learned to add multiple filter options to try out different frames in the app based on data collected after face detection.

What did we ACHIEVE in the class TODAY?

- We learned to create and add multiple face filters on the face

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

How did we DO the activities?

1. Install '**react-native-responsive-fontsize**' library and import RFPercentage, RFValue and other dependencies

```
import { RFPercentage, RFValue } from "react-native-responsive-fontsize";
```

```
import {
    StyleSheet,
    Text,
    View,
    SafeAreaView,
    StatusBar,
    Platform,
    ScrollView,
    TouchableOpacity,
    Image
} from 'react-native';
```

2. Add the style for each container:
    - Heading
    - App name text 1
    - App name text 2
    - Subheading text 1
    - Subheading text 2
    - Frames container
    - Images container

```
headingContainer: {
    flex: 0.15,
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: "#6278e4"
},
```

```
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
},
```

```
framesContainer: {
    flex: 0.2,
    paddingLeft: RFValue(20),
    paddingRight: RFValue(20),
    paddingTop: RFValue(30),
    backgroundColor: "#6278e4"
},
filterImageContainer: {
    height: RFPercentage(8),
    width: RFPercentage(15),
    justifyContent: "center",
    alignItems: "center",
    backgroundColor: "#e4e7f8",
    borderRadius: 30,
    marginRight: 20
}
```

3. Write a return method to render text

```
<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 Our</Text><Text style={styles.subheading2}> Cool Frames</Text>
        </View>
    </View>
```

4. Add an image data object.

```
let data = [
    {
        "id": "1",
        "image": require('../assets/glasses.png')
    },
    {
        "id": "2",
        "image": require('../assets/glasses-round.png')
    },

]
```

5. Add a state variable current_filter to change image filter value and update the state variable value in render() method.

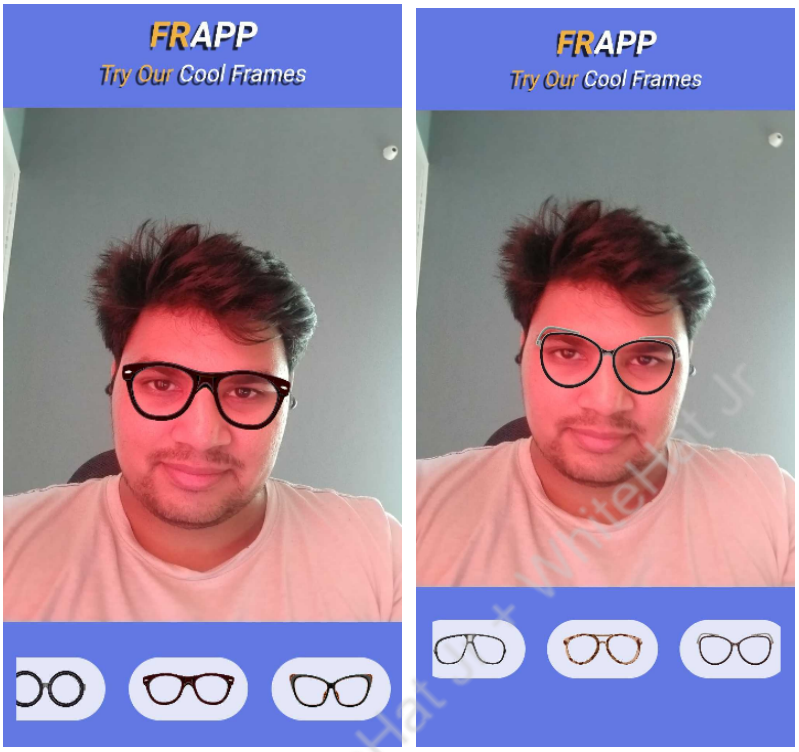
```
constructor(props) {
    super(props)
    this.state = {
        hasCameraPermission: null,
        faces: [],
        current_filter: "filter_1"
    }
```

```
<Camera
    style={{ flex: 1 }}
    type={Camera.Constants.Type.front}
    faceDetectorSettings={{
        mode: FaceDetector.Constants.Mode.fast,
        detectLandmarks: FaceDetector.Constants.Landmarks.all,
        runClassifications: FaceDetector.Constants.Classifications.all
    }}
    onFacesDetected={this.onFacesDetected}
    onFacesDetectionError={this.onFacesDetectionError}
/>
{
    this.state.faces.map(face => {
        if (this.state.current_filter === "filter_1") {
            return <Filter1 key={face.faceID} face={face} />
        } else if (this.state.current_filter === "filter_2") {
            return <Filter2 key={face.faceID} face={face} />
        }
    })
}
```

6. Write a return method to render images

```
<View style={styles.framesContainer}>
    <ScrollView style={{ flexDirection: "row" }} horizontal showsHorizontalScrollIndicator={false}>
        {
            data.map(filter_data => {
                return (
                    <TouchableOpacity style={styles.filterImageContainer} onPress={() => this.setState({ current_filter: `filter_${filter_data.id}` })}>
                        <Image source={filter_data.image} style={{ height: 32, width: 80 }} />
                    </TouchableOpacity>
                )
            })
        }
    </ScrollView>
</View>
```

7. Test the output by selecting the frames:



We have successfully learned to add multiple filters using the FaceDetector API in the expo.

What's NEXT?

In the next class, we will learn to add different categories of the filters based on the different frames options.

EXTEND YOUR KNOWLEDGE:

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