Topic	MULTIPLE FILTERS TRYOUT	
Class Description	**Students will learn to add multiple filters options to try out different frames in the app based on data collected after face detection.**	
Class	**C183**	
Class time	**45 mins**	
Goal	• Learn to create and add multiple face filters on the face.	
Resources Required	• Teacher Resources: ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ smartphone ○ earphones with mic ○ notebook and pen • Student Resources: ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ smartphone ○ earphones with mic ○ notebook and pen	
Class structure	**Warm-Up** **Teacher-led Activity** **Student-led Activity** **Wrap-Up**	**5 mins** **15 mins** **20 mins** **5 mins**

CONTEXT

- **Design App UI**
- **Adding multiple face filters.**

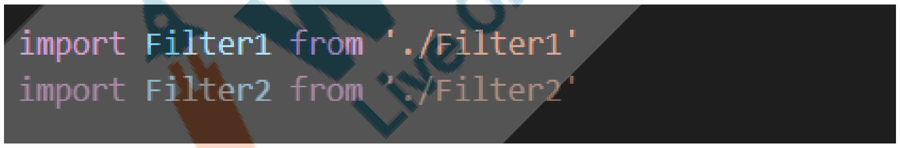
Class Steps	Teacher Action	Student Action

Step 1: Warm-Up (5 mins)	Hi, how are you? Great!	**ESR:** I am good!
	Can you tell me what we have learned in the previous class? Yes. Correct! In the previous class we learned to use the **faces** array data(that we received using FaceDetector) to place the filter images over some facial features. In today's class we are going to learn to add multiple filters options at the bottom of the app screen, from which users can select to try out different frame filters. Are you excited? Let's get started then.	**ESR:** ● We learned how to add face filter images using the data received from the Expo FaceDetector module. **ESR**: Yes.

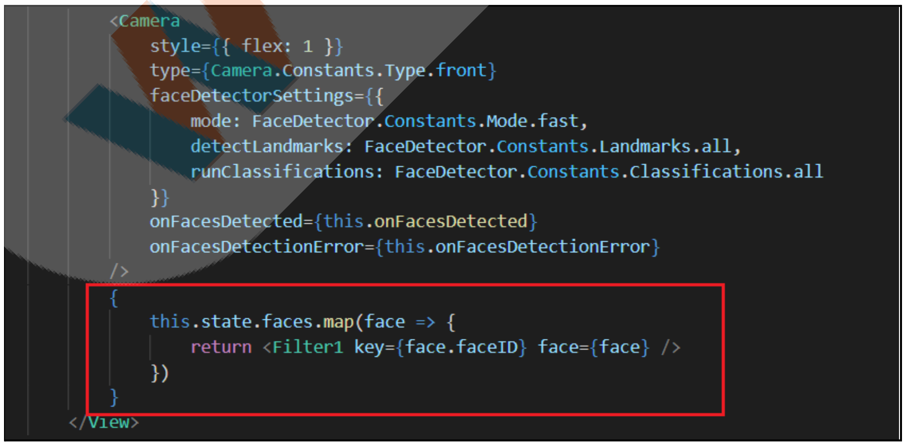
	Click on the Quiz Time *button on the bottom right corner of your screen to start the In-Class Quiz.* *A quiz will be visible to both you and the student.* *Encourage the student to answer the quiz question.* *The student may choose the wrong option, help the student to think correctly about the question and then answer again.* *After the student selects the correct option, the* End Quiz *button will start appearing on your screen.* *Click the End quiz to close the quiz pop-up and continue the class.*	
Teacher Initiates Screen Share		
CHALLENGE • **Design App UI.** • **Adding multiple face filters.**		

Step 2: Teacher-led Activity (15 mins)	*<The teacher clones the code Teacher Activity 1.* ***Note**: Do install node modules.>* ***[Teacher Activity 1]*** In the previous class we created two different filters, but we could only use one at a time to try out, right? *The teacher explains the previous class code from Teacher Activity 1.* In the **Main.js** file, We have the file **Filter1.js** and **Filter2.js** under the screens folder (from the previous class) and we have rendered only one filter image under **cameraStyle** <View> container.	**ESR**: Yes.

Previous class code section(./screens/Main.js)



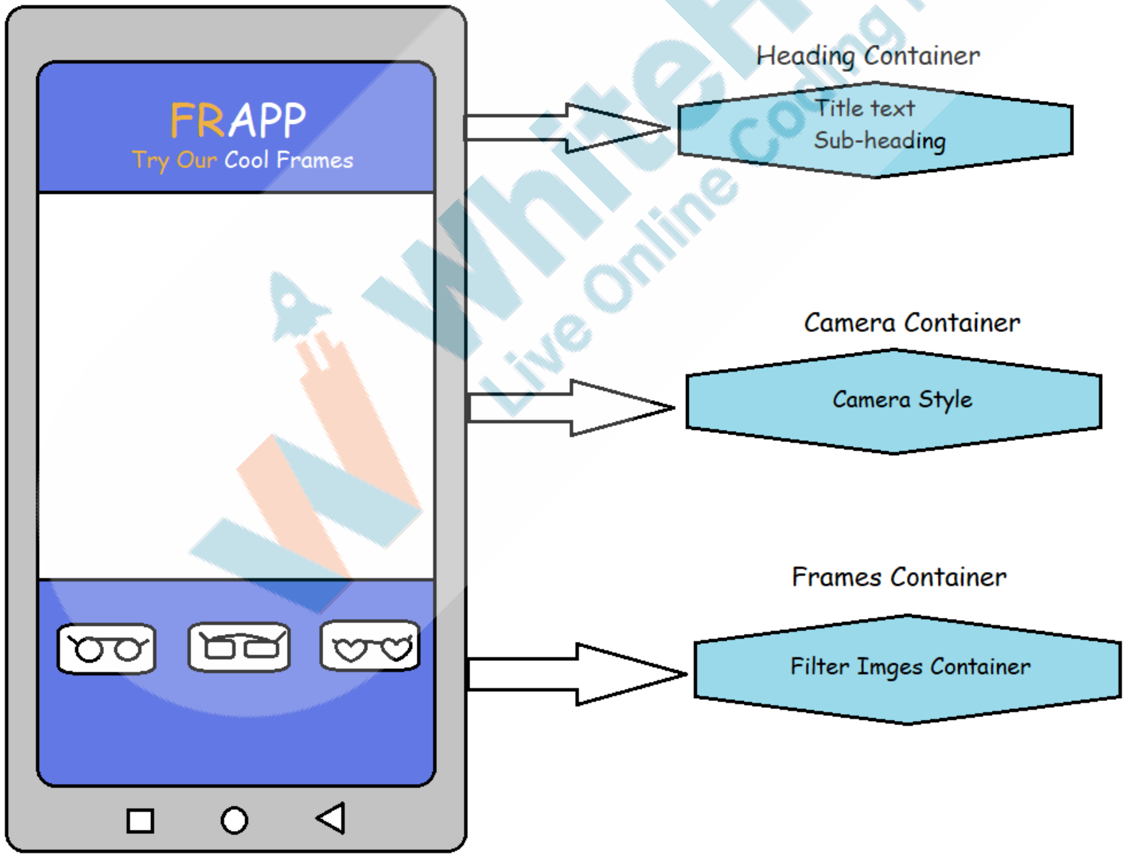
```
import Filter1 from './Filter1'
import Filter2 from './Filter2'
```



```
        <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 => {
                return <Filter1 key={face.faceID} face={face} />
            })
        }
    </View>
```

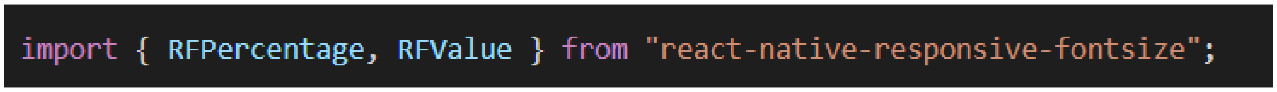
	Now we would want our app users to choose from multiple glasses (spectacles) filters and try them out over the eyes. For this let's begin designing the app from the top. *<Open the image from Teacher Activity 2 and discuss the design of the app>* First we would want to set the name of the app and some other text that we want to show as a subheading in the app. The name and subheading will be in two different colors. Then we would need a camera section. After this we are going to add an image container for each frame at the bottom and the user can tap on the image to try out the frame. Users can also scroll horizontally from right to left for more frames filters.	

	To summarize will we need: **Heading Container**: To render name of the app and some other information heading. **Camera Container**: To style the camera section **Frames Container**: To add frames images using Image Container for each frame image.	



	Let’s begin with heading now. To add the headings we are going to use the <Text> component. The most important thing for any app is how responsive the app is to different devices! This means no matter which device we will use to run the app, the UI components(likes images, texts, etc.) of the app must not get distorted. To make these texts responsive and adjust according to the screen size, we will install one of the React Native libraries. **npm install react-native-responsive-fontsize --save** And now we can import **RFPercentage** and **RFValue** from this in our app, which we will use to style the <Text> component. **RFPercentage**: Sets the font size with respect to the height of the device(in percentage) **RFValue**: Sets the font size based on standardScreenHeight.	

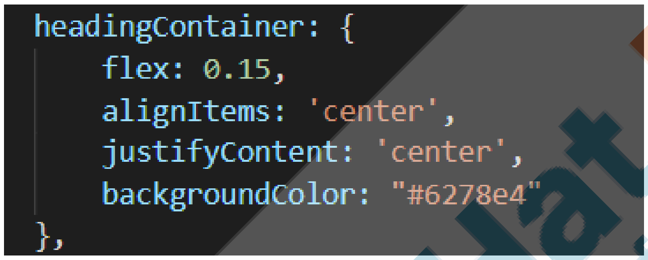
./screens/Main.js



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

Now we can add the style containers for the heading.

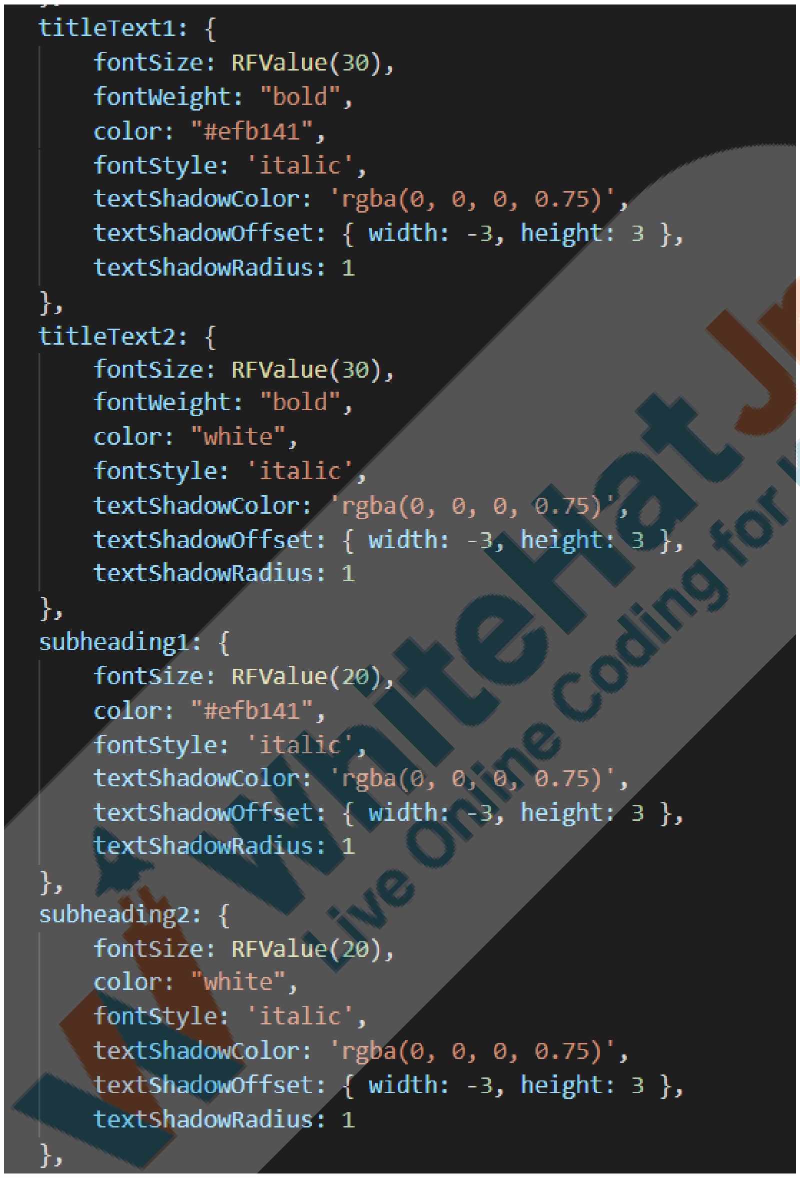
./screens/Main.js



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

The name of the app and subheading will have two different colors, for this we can divide the styling into two different parts, each for one color.

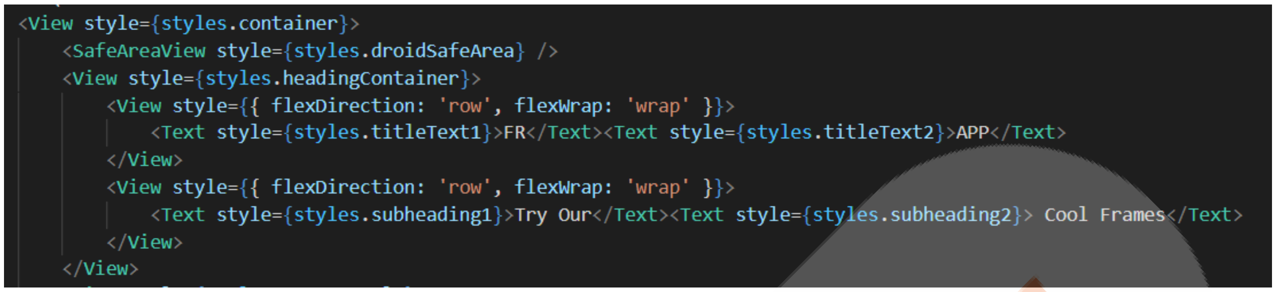
./screens/Main.js



```
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
},
```

	Now we can add these in the <View> and <Text> component inside the return() method of the Main component.	

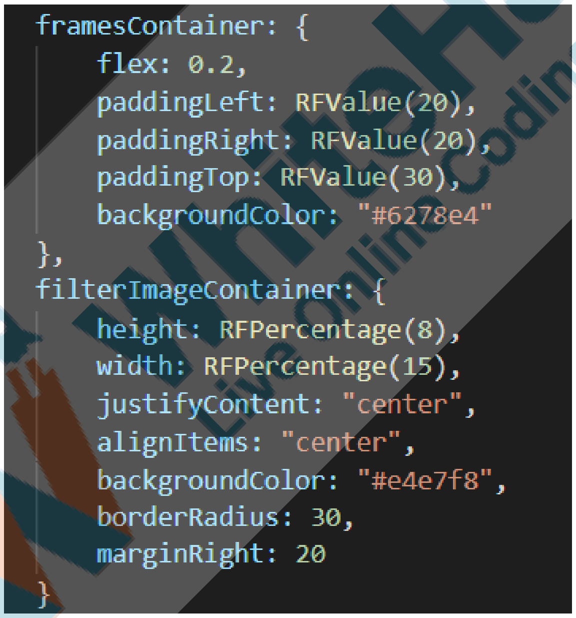
./screens/Main.js



```
<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>
```

Now let's add the styling image container for the two filters that we created in the previous class.

./screens/Main.js



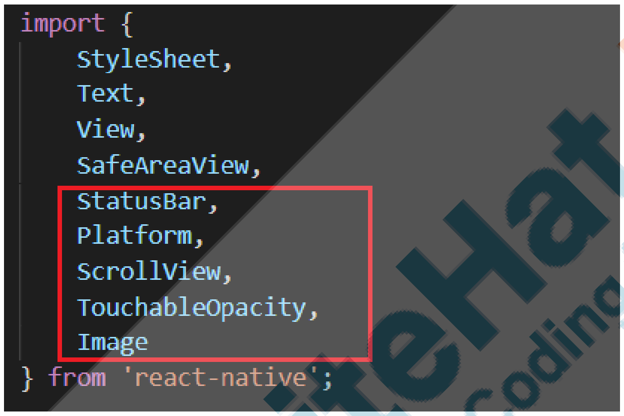
```
framesContainer: {
    flex: 0.2,
    paddingLeft: RFValue(20),
    paddingRight: RFValue(20),
    paddingTop: RFValue(30),
    backgroundColor: "#6278e4"
},
filterImageContainer: {
    height: RFPercentage(8),
    width: RFPercentage(15),
    justifyContent: "center",
    alignItems: "center",
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

Since we want to add many filters, we should be able scroll through all the filters to choose.

Then we can tap on the filter image to try that filter out.

	Can you tell me which react components can be used for this? Superb! Let's import these components.	**ESR**: We can <ScrollView> and <TouchableOpacity>

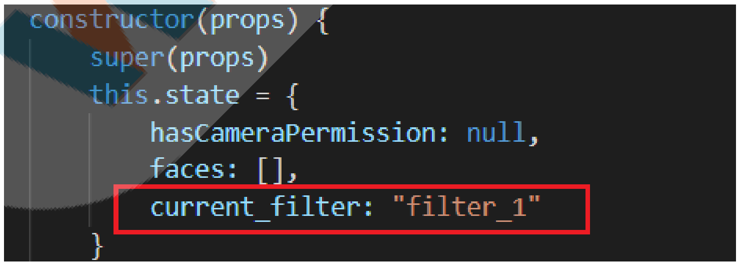
./screens/Main.js



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

	Now let's take a **state variable**, **current_filter**, to keep a track of which filter is being rendered on the face. By default we will use the first filter. So let's keep the value of the variable as "**filter_1**"	

./screens/Main.js



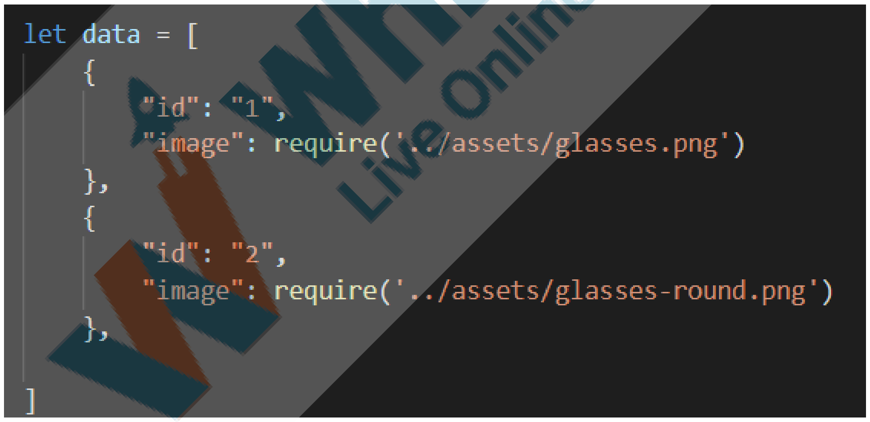
```
constructor(props) {
    super(props)
    this.state = {
        hasCameraPermission: null,
        faces: [],
        current_filter: "filter_1"
    }
```

	Now we will have to **update** the **value of the state variable** based on which image filter is tapped so that its respective filter can be rendered.	

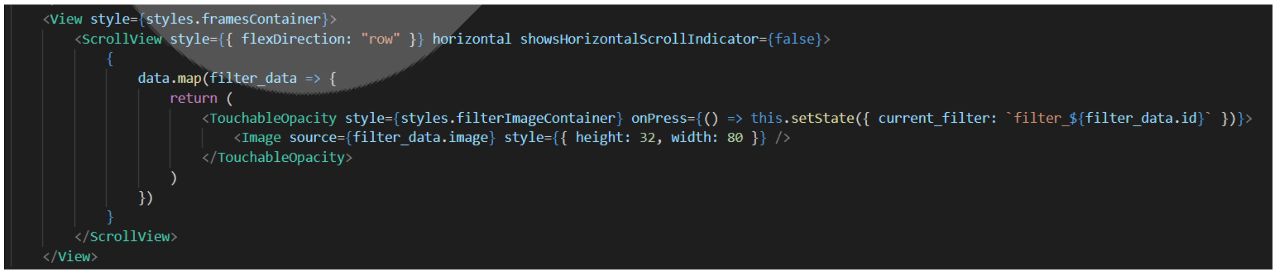
For this we will:

- Take a **data** object to assign a unique id to each image source that can be used to update the state variable value.
- Add the **<ScrollView>** with flexDirection as “row” which will help us scroll from right to left horizontally.
- Loop through the **data** object to add **<TouchableOpacity>** component:
    - Add **onPress()** method to set current_filter state value using **setState()** method.
    - Add the image source.

./screens/Main.js



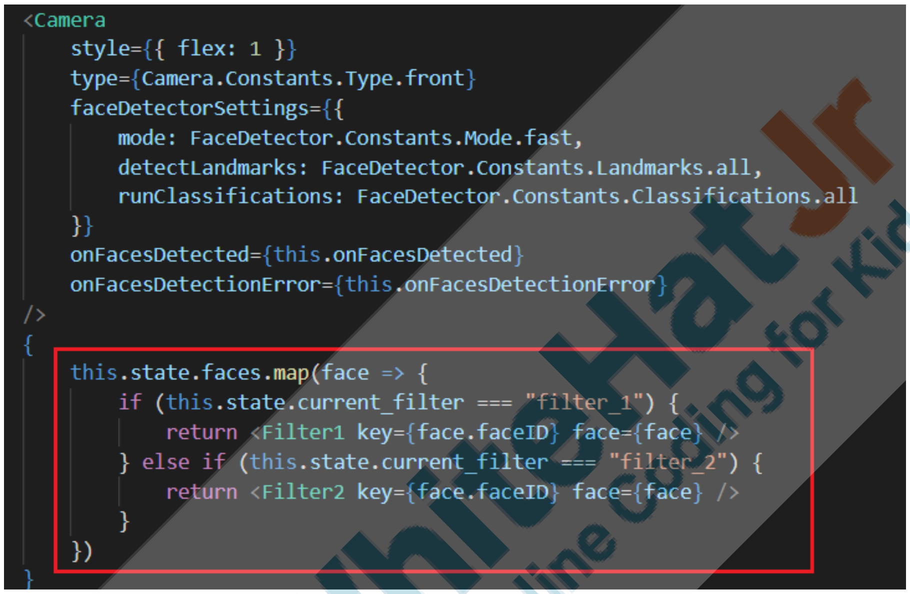
```
let data = [
    {
        "id": "1",
        "image": require('../assets/glasses.png')
    },
    {
        "id": "2",
        "image": require('../assets/glasses-round.png')
    },

]
```



```
<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>
```

	Now we will return the **Filter1** and **Filter2** component(created in the previous class) based on the value of the **current_filter** state variable.	

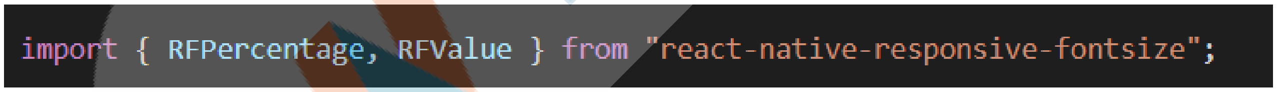
./screens/Main.js



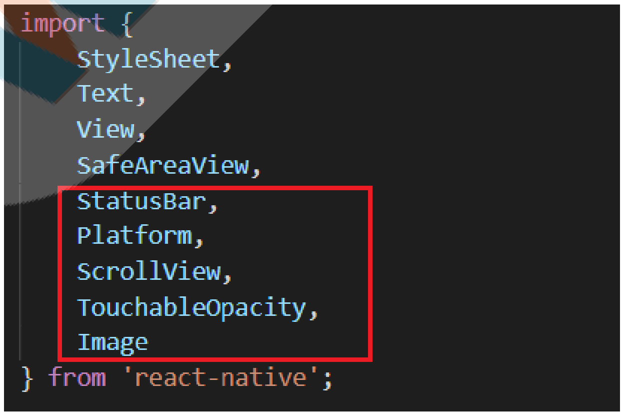
```
<Camera
    style={{ flex: 1 }}
    type={Camera.Constants.Type.front}
    faceDetectorSettings={{
        mode: FaceDetector.Constants.Mode.fast,
        detectLandmarks: FaceDetector.Constants.Landmarks.all,
        runClassifications: FaceDetector.Constants.Classifications.all
    }}
    onFacesDetected={this.onFacesDetected}
    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} />
        }
    })
}
```

	Now let’s test the output using expo.	
	That’s really amazing! We added multiple glasses filters. We can try out any filter just by tapping on it. Now you will have to add multiple filters in the app. Are you excited?	**ESR**: Yes!
Teacher Stops Screen Share		

	Now it's your turn. Please share your screen with me.	
● Ask the student to press the ESC key to come back to the panel. ● Guide the student to start screen share. ● Teacher gets into fullscreen.		
ACTIVITY **● Design the app UI. ● Add multiple filters in the app.**		
Step 3: Student-led Activity (20 mins)	*The teacher guides the student to clone the code from Student Activity 1.* ***[Student Activity 1]*** ***Note****: The student will repeat teacher activity for different filter images.* *Guide the student to create and set up the react project.*	
	Guide the student to import library component.	

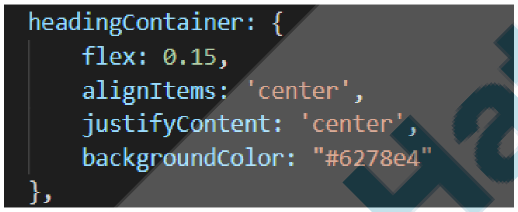


```
import { RFPercentage, RFValue } from "react-native-responsive-fontsize";
```

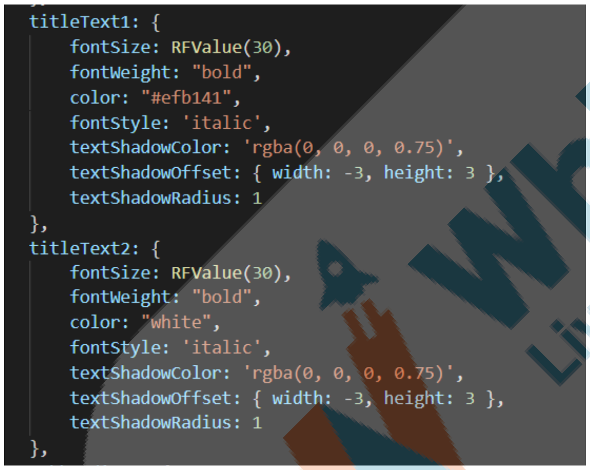


```
import {
    StyleSheet,
    Text,
    View,
    SafeAreaView,
    StatusBar,
    Platform,
    ScrollView,
    TouchableOpacity,
    Image
} from 'react-native';
```

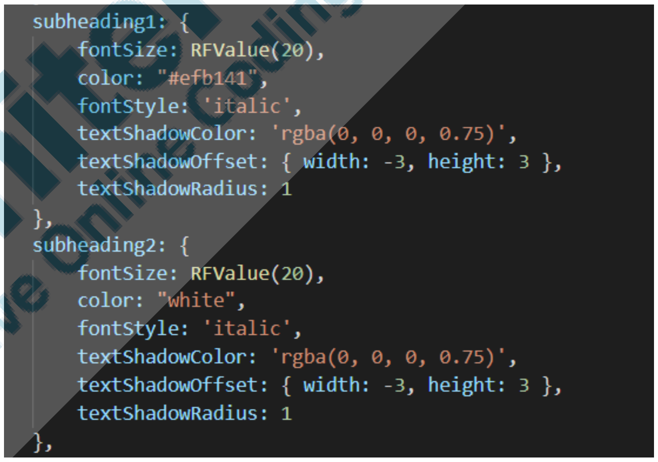
	Guide the student to add the style for each container: ● *Heading* ● *App name text 1* ● *App name text 2* ● *Subheading text 1* ● *Subheading text 2* ● *Frames container* ● *Images container*	



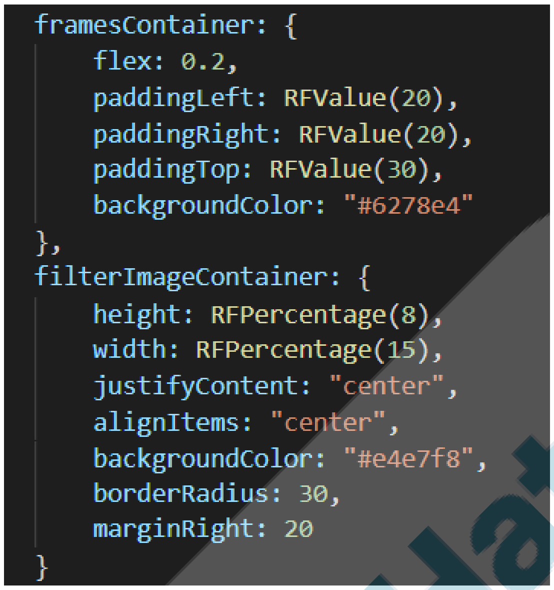
```
headingContainer: {
    flex: 0.15,
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: "#6278e4"
},
```



```
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
},
```

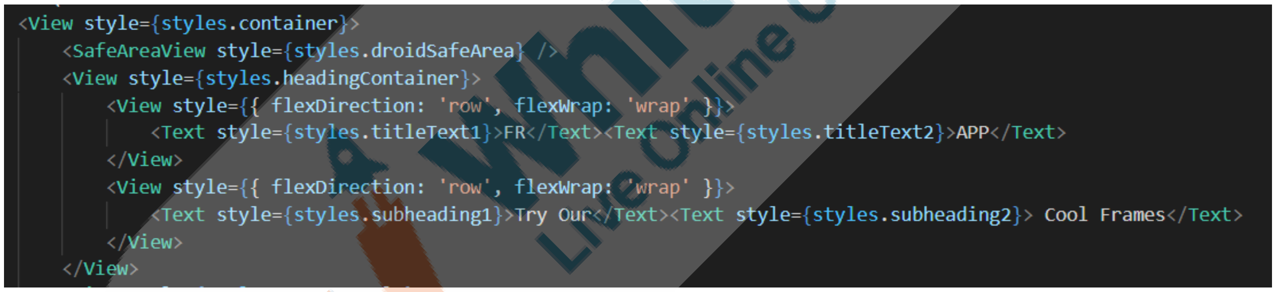


```
subheading1: {
    fontSize: RFValue(20),
    color: "#efb141",
    fontStyle: 'italic',
    textShadowColor: 'rgba(0, 0, 0, 0.75)',
    textShadowOffset: { width: -3, height: 3 },
    textShadowRadius: 1
},
subheading2: {
    fontSize: RFValue(20),
    color: "white",
    fontStyle: 'italic',
 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



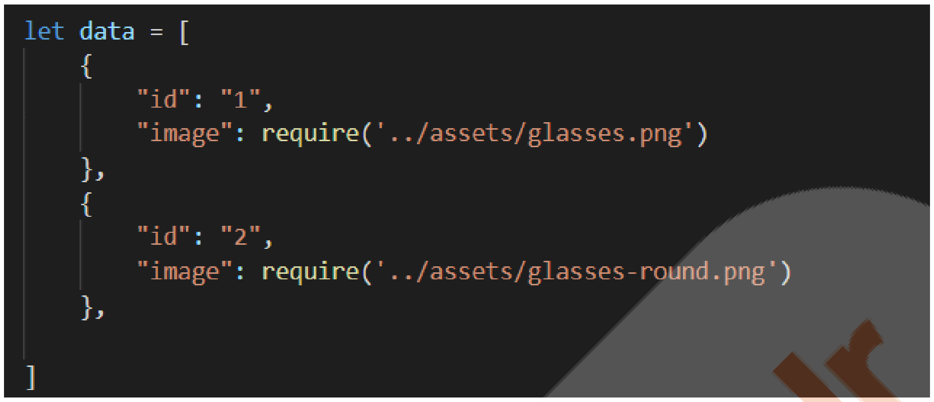
```
framesContainer: {
    flex: 0.2,
    paddingLeft: RFValue(20),
    paddingRight: RFValue(20),
    paddingTop: RFValue(30),
    backgroundColor: "#6278e4"
},
filterImageContainer: {
    height: RFPercentage(8),
    width: RFPercentage(15),
    justifyContent: "center",
    alignItems: "center",
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

Guide the student to write a return method to render text



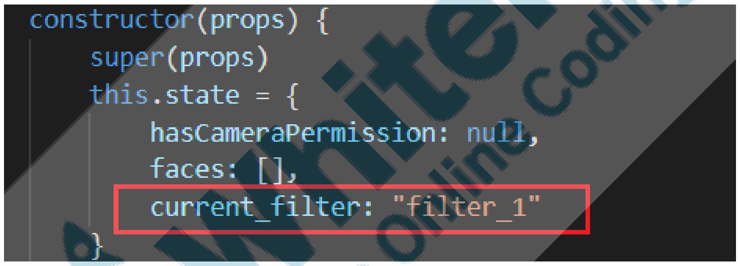
```
<View style={styles.container}>
    <SafeAreaView style={styles.droidSafeArea} />
    <View style={styles.headingContainer}>
        <View style={{ flexDirection: 'row', flexWrap: 'wrap' }}>
            <Text style={styles.titleText1}>FR</Text><Text style={styles.titleText2}>APP</Text>
        </View>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

Guide the student to add an image data object.

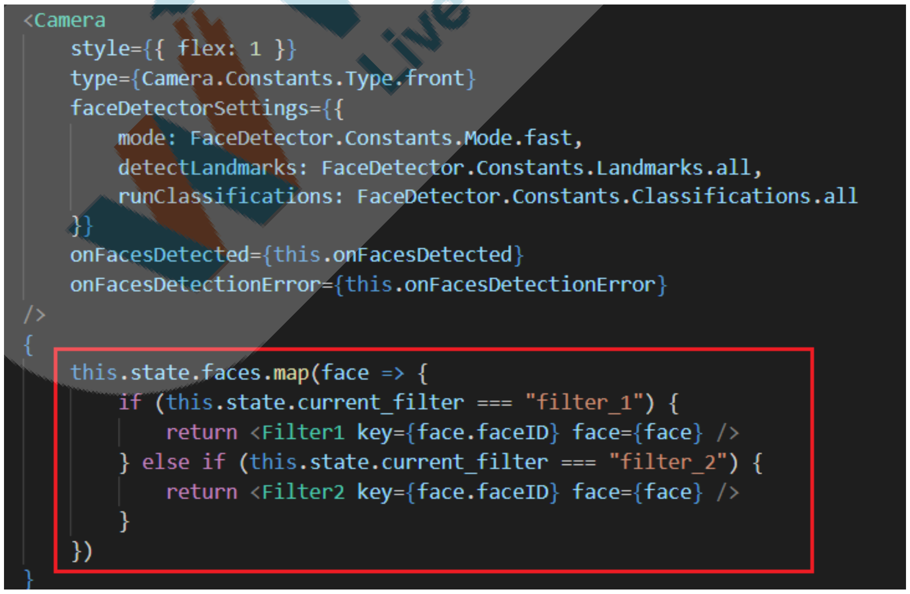


```
let data = [
    {
        "id": "1",
        "image": require('../assets/glasses.png')
    },
    {
        "id": "2",
        "image": require('../assets/glasses-round.png')
    },

]
```

*Guide the student to **add** a state variable **current_filter** to change image filter value and **update** the state variable value in render() method.*

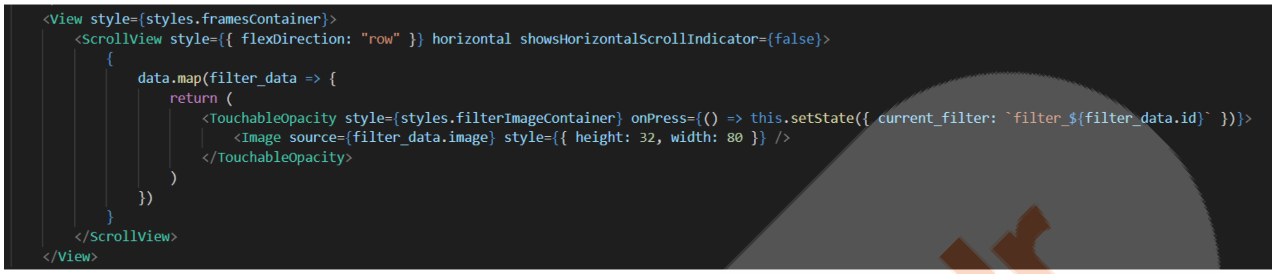


```
constructor(props) {
    super(props)
    this.state = {
        hasCameraPermission: null,
        faces: [],
        current_filter: "filter_1"
    }
```



```
<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} />
        }
    })
}
```

	Guide the student to 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>
```

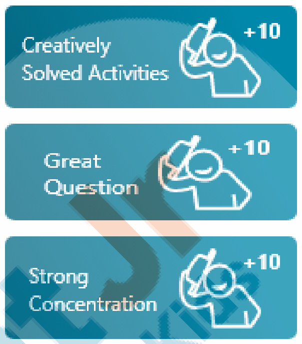
	Guide the student to test the output. ***Note**: Ask the student to add the other 8 filters in the app. Filter images can be found in the **assets** folder.*	

Teacher Guides Student to Stop Screen Share

FEEDBACK

- **Compliment the student for her/his effort in the class.**
- **Encourage the student to think and come up with their own solutions.**

Step 4: Wrap-Up (5 mins)	Let's quickly wrap-up today's class. What did we learn?	**ESR**: ● We learned to apply face filters using the data received from FaceDetector API expo module.
	Amazing work today! In the next class, we will learn how to switch between multiple filters to apply on the face.	*The student listens.*

	You get a “hats-off”. Alright. See you in next class.	*Make sure you have given at least 2 Hats Off during the class for:*  Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
Project Overview	**NAME** **Goal of the Project:** **Story:** <Yet To Updated>	
	Teacher Clicks	
Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class?	*The student uses the markdown editor to write their reflections in a reflection journal.*

	• What have I learned about programming and developing games? • What aspects of the class helped me? What did I find difficult?	

Activity	Activity Name	Links
Teacher Activity 1	Previous Class Code	https://github.com/whitehatjr/PRO-C182-Code-Ref
Teacher Activity 2	FRAPP Design Model	https://s3-whjr-v2-prod-bucket.whjr.online/9d54bf86-a6e0-4f66-b8fa-16e547591311.png
Teacher Activity 3	Final Reference Code	https://github.com/whitehatjr/PRO-C183-Code-Ref
Student Activity 1	Previous Class Code	https://github.com/whitehatjr/PRO-C182-Code-Ref