Topic	FILTERS CATEGORY	
Class Description	**Students will learn to add different categories of the filters based on the different frames options.**	
Class	**C184**	
Class time	**45 mins**	
Goal	● Learn to add categories for 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**

<u>CONTEXT</u>

- **Design App UI**
- **Dividing multiple face filters into categories.**

Class Steps	Teacher Action	Student Action

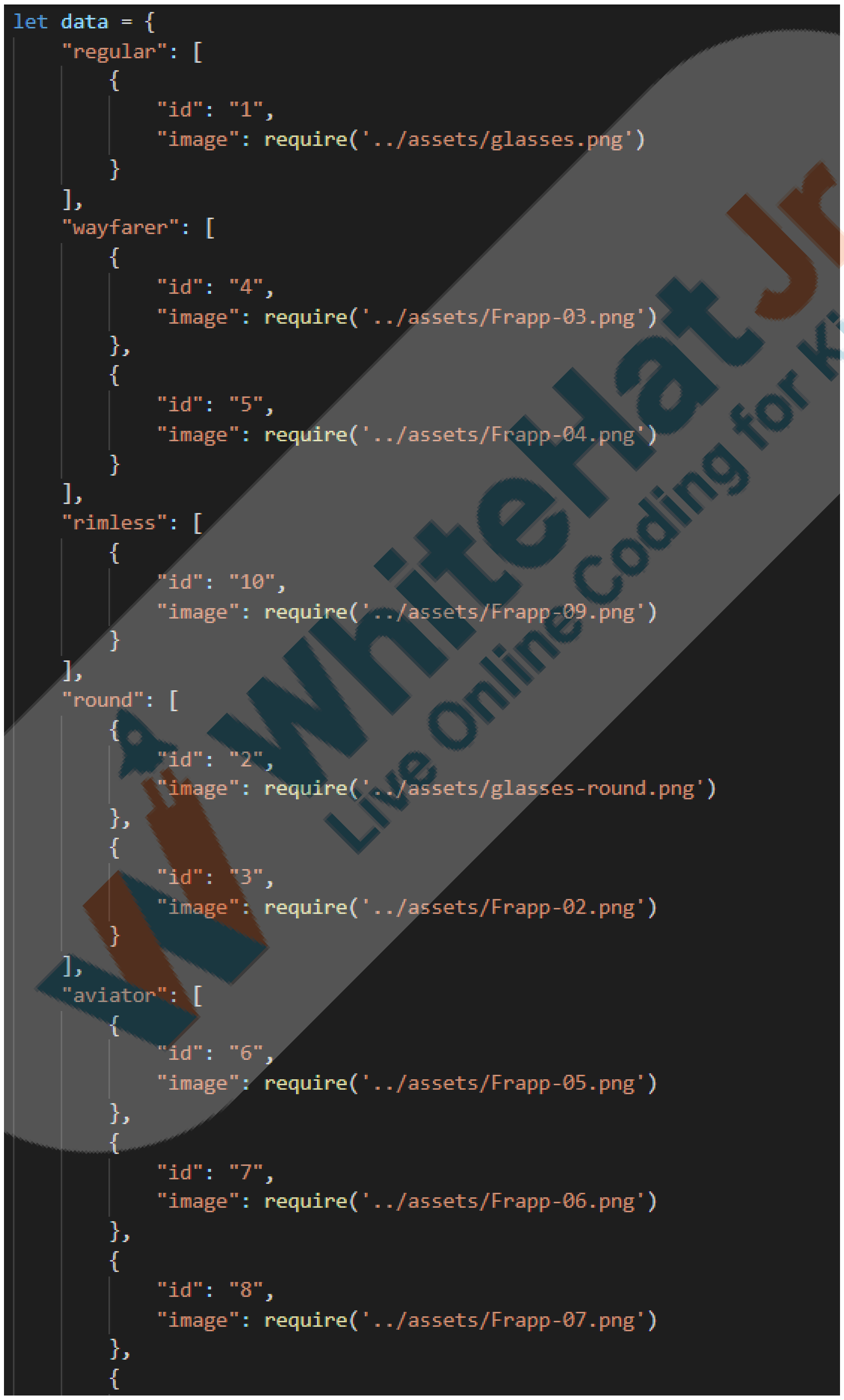
Step 1: Warm-Up (5 mins)	Hi, how are you? Great!	**ESR:** I am good!
	Can you tell me what we have learned in the previous class? Great! Today we will be adding categories of frames filters. Users will be able to select frames to try on based on the category. Can you explain why it should be made into categories? Well when we make categories of the product in an app, it helps users to easily find and browse the product they are looking for and this is really to increase user experience using the app. Are you excited? Let’s get started then.	**ESR:** ● We learned how to add scrollable options to choose from multiple face filter images in the app. **ESR**: Varied. **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		
<u>CHALLENGE</u> ● **Design App UI** ● **Dividing multiple face filters into categories.**		

Step 2: Teacher-led Activity (15 mins)	*<The teacher clones the code Teacher Activity 1.* ***Note**: Do install node modules.>* ***[Teacher Activity 1]*** Before we can begin, can you tell me how we can categorise different frames in the app? ***Note**: Let the student come up with a few of his/her own ideas. Encourage the student to be more involved in the discussion.* Well we will be keeping it simple! We distribute the frame based on the shape of the frames. What do you think could be the possible categories for the images of the that we have? ***Note**: Let the student come up with a few of his/her own ideas. Encourage the student to be more involved in the discussion.* We will be dividing them in five categories, **regular**, **wayfarer**, **rimless**, **round**, **aviator**. Let's modify the **data object variable**(defined in the previous class)	**ESR**: Based on brand names, price, shape of the frames. **ESR**: Round, Square.

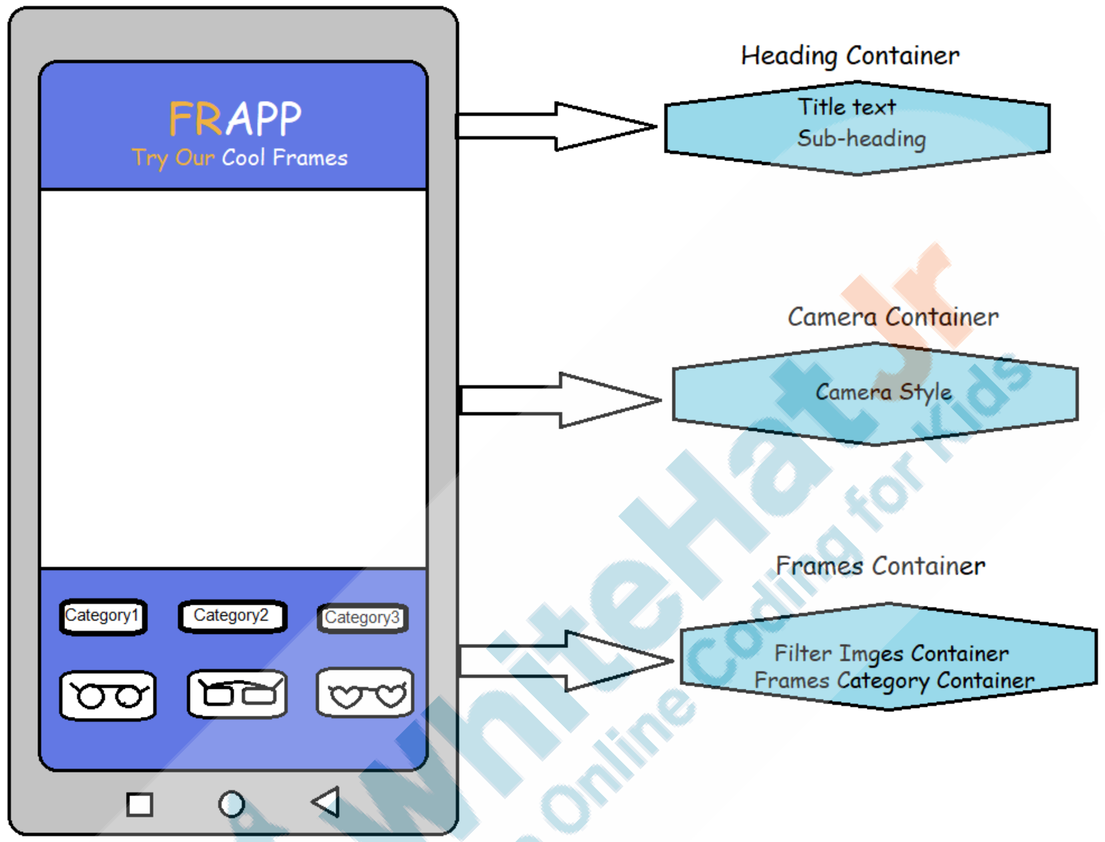
	with the images **id** and the image **source**, with the category names.	

./screens/Main.js



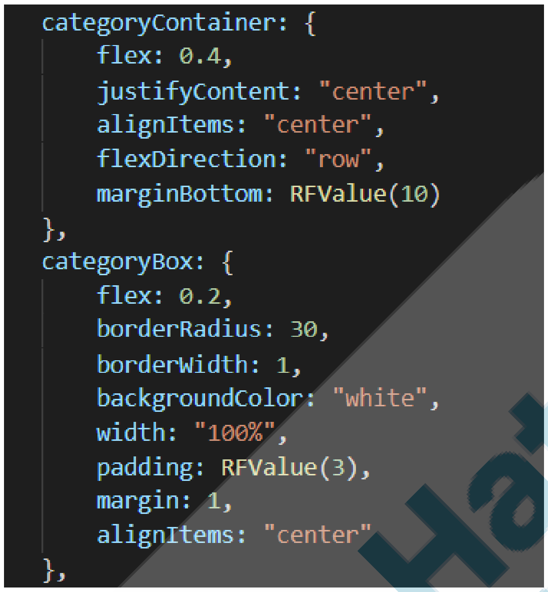
```
let data = {
    "regular": [
        {
            "id": "1",
            "image": require('../assets/glasses.png')
        }
    ],
    "wayfarer": [
        {
            "id": "4",
            "image": require('../assets/Frapp-03.png')
        },
        {
            "id": "5",
            "image": require('../assets/Frapp-04.png')
        }
    ],
    "rimless": [
        {
            "id": "10",
            "image": require('../assets/Frapp-09.png')
        }
    ],
    "round": [
        {
            "id": "2",
            "image": require('../assets/glasses-round.png')
        },
        {
            "id": "3",
            "image": require('../assets/Frapp-02.png')
        }
    ],
    "aviator": [
        {
            "id": "6",
            "image": require('../assets/Frapp-05.png')
        },
        {
            "id": "7",
            "image": require('../assets/Frapp-06.png')
        },
        {
            "id": "8",
            "image": require('../assets/Frapp-07.png')
        },
        {
```

	Remember we had: **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. Since we want to add the categories of the frame filters, let's discuss the design again. What do you do where can we add the categories section for the frames filters in our app? One of the options could be to keep at the bottom of the screen. *<Open the image from Teacher Activity 2 and discuss the updated design of the app>* We will be adding the **categories for the frames** of glasses at bottom of the screen in the **Frames Container**, just above the filter images.	 **ESR**: Varied.



	Now let's add the style of the categories container in the Stylesheet. We can have the white colored round boxes for the categories to display.	

./screens/Main.js

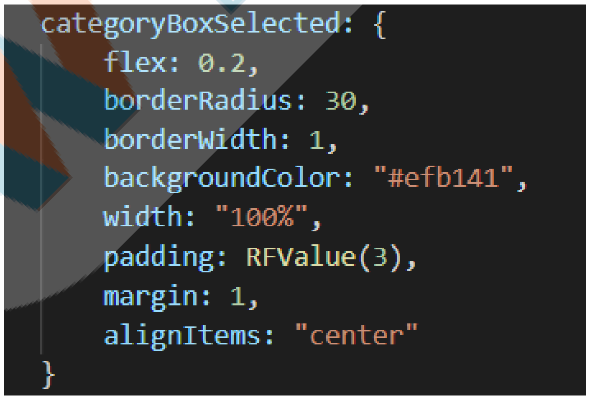


```
categoryContainer: {
    flex: 0.4,
    justifyContent: "center",
    alignItems: "center",
    flexDirection: "row",
    marginBottom: RFValue(10)
},
categoryBox: {
    flex: 0.2,
    borderRadius: 30,
    borderWidth: 1,
    backgroundColor: "white",
    width: "100%",
    padding: RFValue(3),
    margin: 1,
    alignItems: "center"
},
```

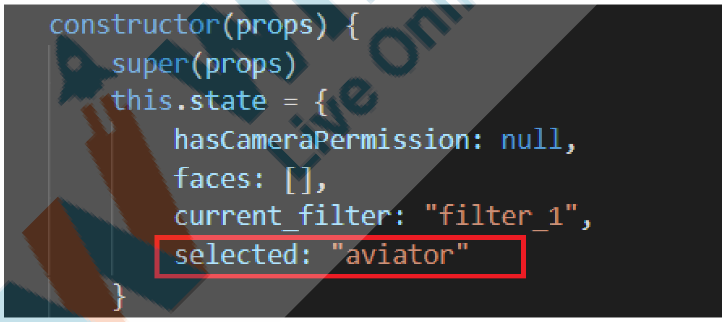
Also to show which category of the frames is currently selected, we can highlight it with a different color.

We can add styling for the box of the selected category.

./screens/Main.js



```
categoryBoxSelected: {
    flex: 0.2,
    borderRadius: 30,
    borderWidth: 1,
    backgroundColor: "#efb141",
    width: "100%",
    padding: RFValue(3),
    margin: 1,
    alignItems: "center"
}
```

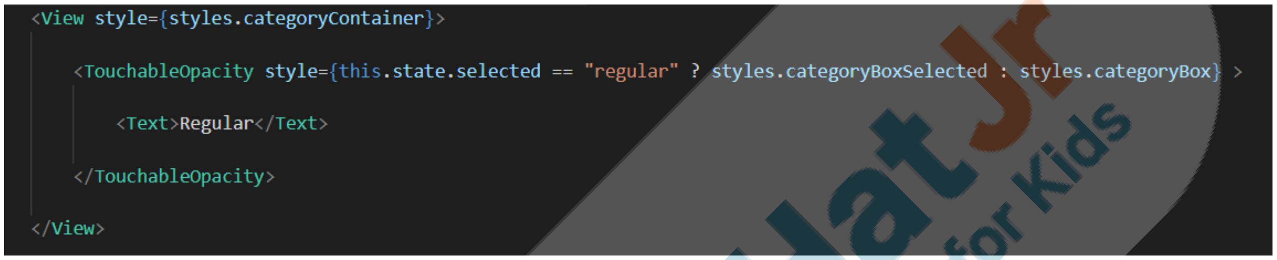
	Once we have styling ready, what should we do next? Prefect! Since we need to show the frames of the selected category only, we should first know which frame category is selected, right? To do this we would need a state variable. Let's take a variable called, **selected**, with the initial category as "**aviator**".	**ESR**: We need to show the frames of the selected category only. **ESR**:Yes.
./screens/Main.js  `constructor(props) {` `super(props)` `this.state = {` `hasCameraPermission: null,` `faces: [],` `current_filter: "filter_1",` `selected: "aviator"` `}`		
	Next step would be to render the categories in the app as the child of **framesContainer <View>** we created in the previous class. Which component(s) can be used to render the categories that can be tapped?	**ESR**: <TouchableOpacity> and <Text>

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

Perfect!

Also we will set the style of the <TouchableOpacity> based on the it’s state value to highlight the selected category.

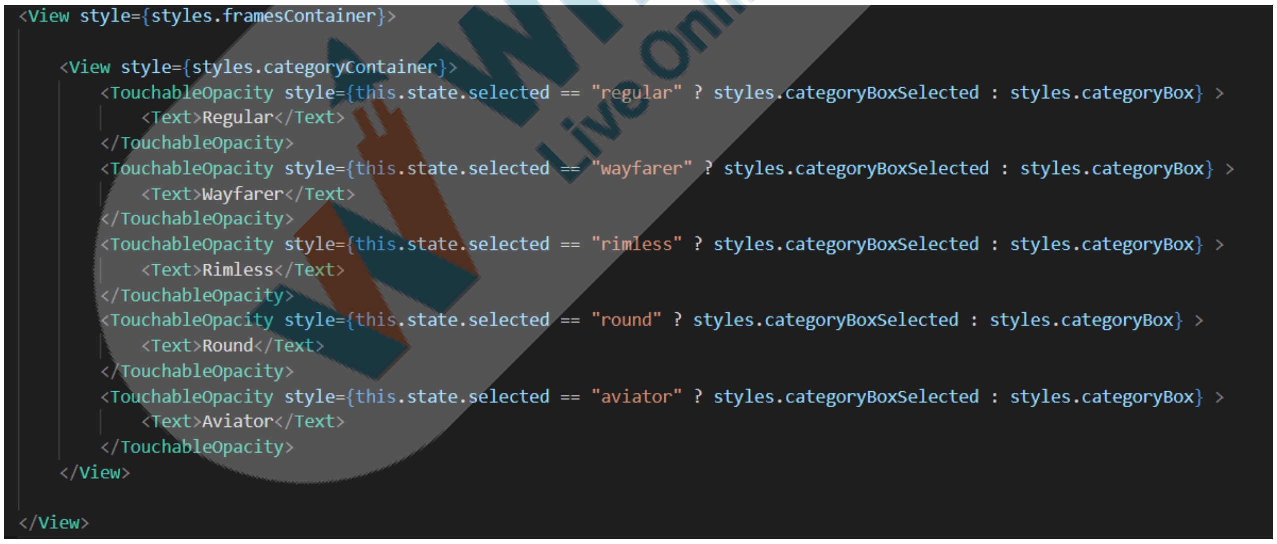
./screens/Main.js



```
<View style={styles.categoryContainer}>

    <TouchableOpacity style={this.state.selected == "regular" ? styles.categoryBoxSelected : styles.categoryBox} >

        <Text>Regular</Text>

    </TouchableOpacity>

</View>
```

We can render all the five categories using <TouchableOpacity> and <Text> components.

./screens/Main.js

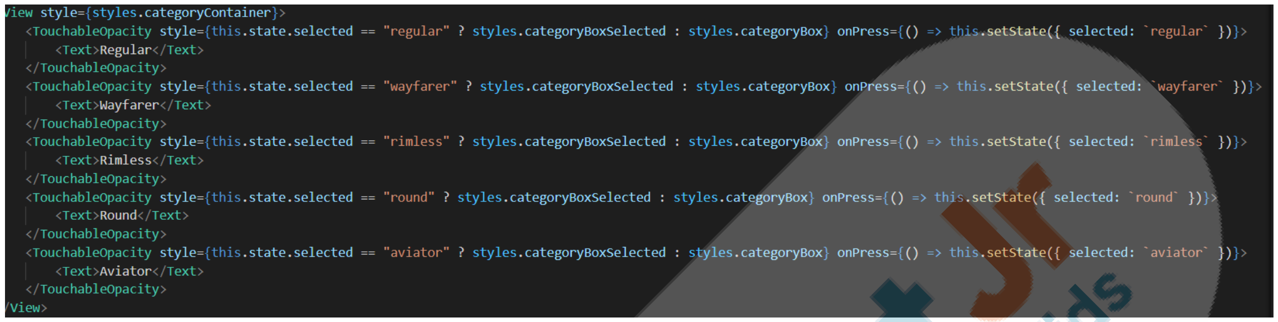


```
<View style={styles.framesContainer}>

    <View style={styles.categoryContainer}>
        <TouchableOpacity style={this.state.selected == "regular" ? styles.categoryBoxSelected : styles.categoryBox} >
            <Text>Regular</Text>
        </TouchableOpacity>
        <TouchableOpacity style={this.state.selected == "wayfarer" ? styles.categoryBoxSelected : styles.categoryBox} >
            <Text>Wayfarer</Text>
        </TouchableOpacity>
        <TouchableOpacity style={this.state.selected == "rimless" ? styles.categoryBoxSelected : styles.categoryBox} >
            <Text>Rimless</Text>
        </TouchableOpacity>
        <TouchableOpacity style={this.state.selected == "round" ? styles.categoryBoxSelected : styles.categoryBox} >
            <Text>Round</Text>
        </TouchableOpacity>
        <TouchableOpacity style={this.state.selected == "aviator" ? styles.categoryBoxSelected : styles.categoryBox} >
            <Text>Aviator</Text>
        </TouchableOpacity>
    </View>

</View>
```

Now we can **update the state** variable **selected** using the setState**()** method inside onPress() method of the <TouchableOpacity> component.

	We can set the value based on the category selected.	

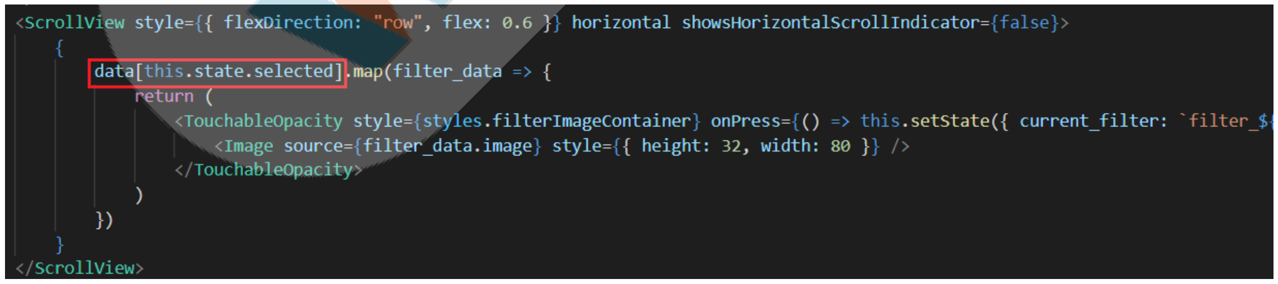
./screens/Main.js



```
View style={styles.categoryContainer}>
  <TouchableOpacity style={this.state.selected == "regular" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `regular` })}>
      <Text>Regular</Text>
  </TouchableOpacity>
  <TouchableOpacity style={this.state.selected == "wayfarer" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `wayfarer` })}>
      <Text>Wayfarer</Text>
  </TouchableOpacity>
  <TouchableOpacity style={this.state.selected == "rimless" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `rimless` })}>
      <Text>Rimless</Text>
  </TouchableOpacity>
  <TouchableOpacity style={this.state.selected == "round" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `round` })}>
      <Text>Round</Text>
  </TouchableOpacity>
  <TouchableOpacity style={this.state.selected == "aviator" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `aviator` })}>
      <Text>Aviator</Text>
  </TouchableOpacity>
/View>
```

	Now categories are there in the app which can be selected and deselected. But still when we will run the app all frames render in UI at once. We need to show only the frames selected for a particular category. Any idea how we can do that? We can do this by looping through data variables based on the value of the "selected" state.	**ESR**: Varied.

./screens/Main.js

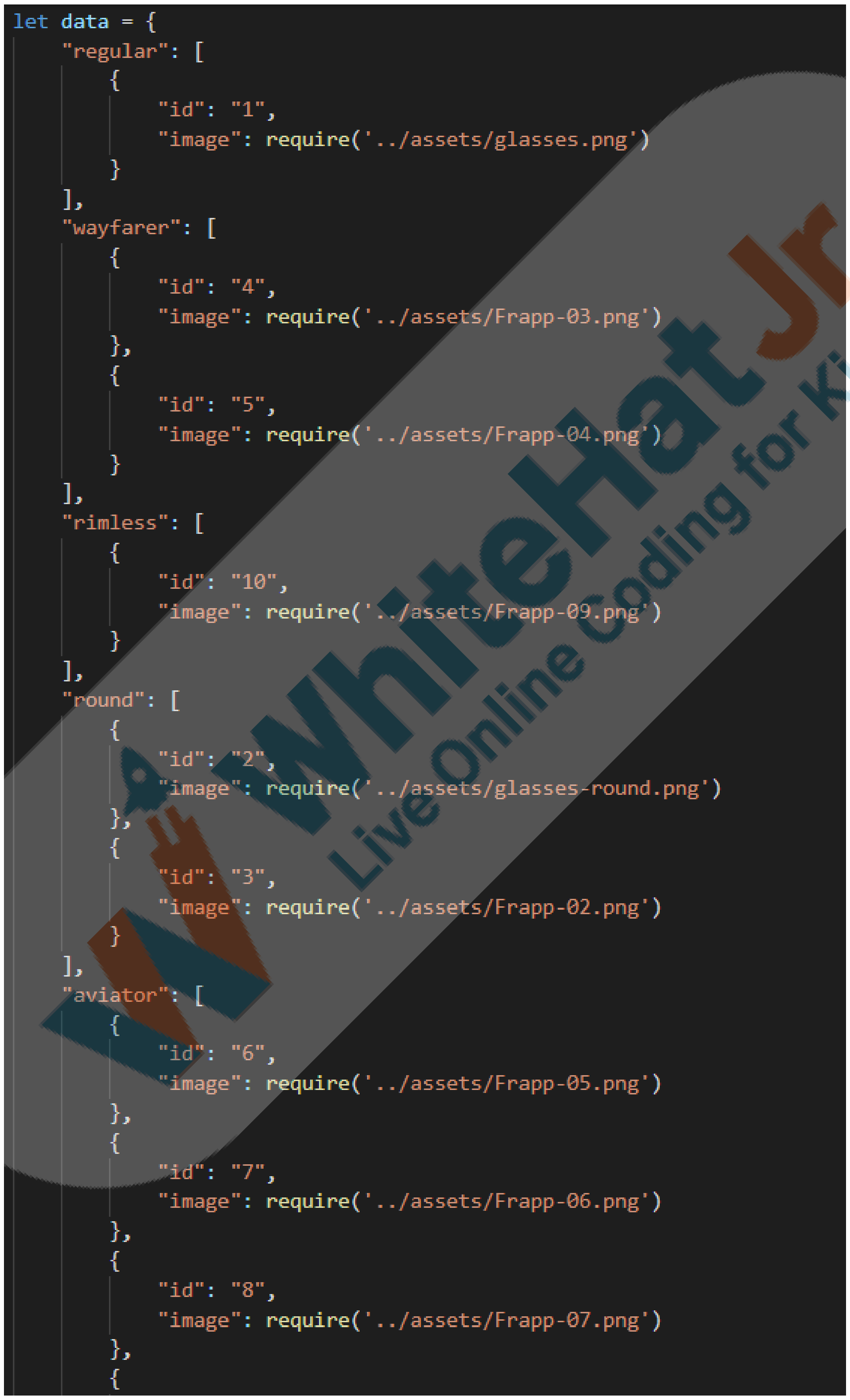


```
<ScrollView style={{ flexDirection: "row", flex: 0.6 }} horizontal showsHorizontalScrollIndicator={false}>
    {
        data[this.state.selected].map(filter_data => {
            return (
                <TouchableOpacity style={styles.filterImageContainer} onPress={() => this.setState({ current_filter: `filter_$
                    <Image source={filter_data.image} style={{ height: 32, width: 80 }} />
                </TouchableOpacity>
            )
        })
    }
</ScrollView>
```

	Now let's test the final output using expo.	

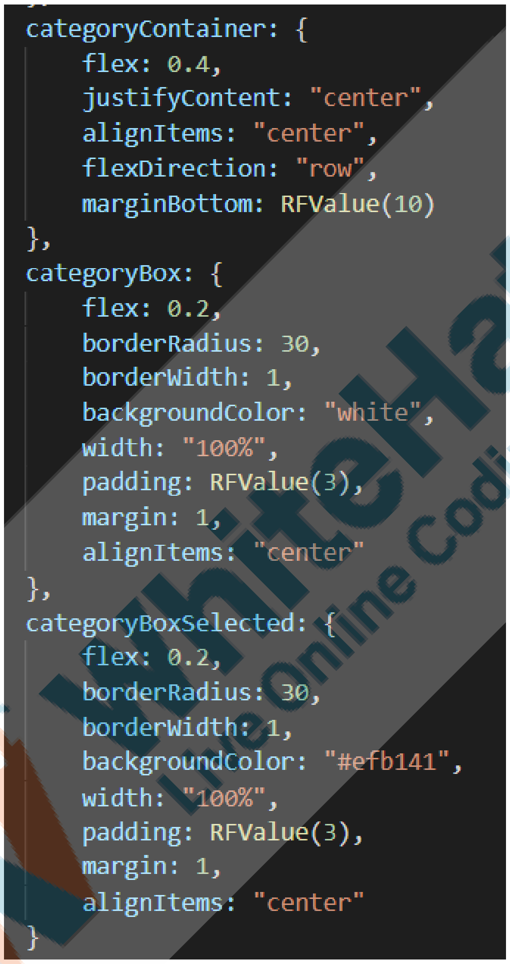
	Navigate to the working directory and open cmd. *Command:* ***expo start -c***	
	That's really interesting to add categories in the app! Now you will categories for filters in the app. Are you excited?	**ESR**: Yes!
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
● **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 update data variables with the categories.	



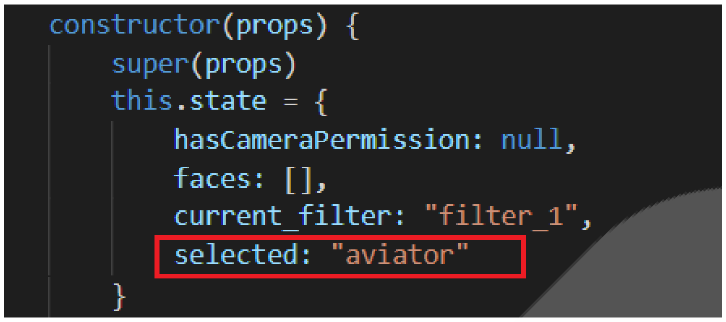
```
let data = {
    "regular": [
        {
            "id": "1",
            "image": require('../assets/glasses.png')
        }
    ],
    "wayfarer": [
        {
            "id": "4",
            "image": require('../assets/Frapp-03.png')
        },
        {
            "id": "5",
            "image": require('../assets/Frapp-04.png')
        }
    ],
    "rimless": [
        {
            "id": "10",
            "image": require('../assets/Frapp-09.png')
        }
    ],
    "round": [
        {
            "id": "2",
            "image": require('../assets/glasses-round.png')
        },
        {
            "id": "3",
            "image": require('../assets/Frapp-02.png')
        }
    ],
    "aviator": [
        {
            "id": "6",
            "image": require('../assets/Frapp-05.png')
        },
        {
            "id": "7",
            "image": require('../assets/Frapp-06.png')
        },
        {
            "id": "8",
            "image": require('../assets/Frapp-07.png')
        },
        {
```

	Guide the student to add the style for: ● *Category Container* ● *Category Box* ● *Category Box Selected*	



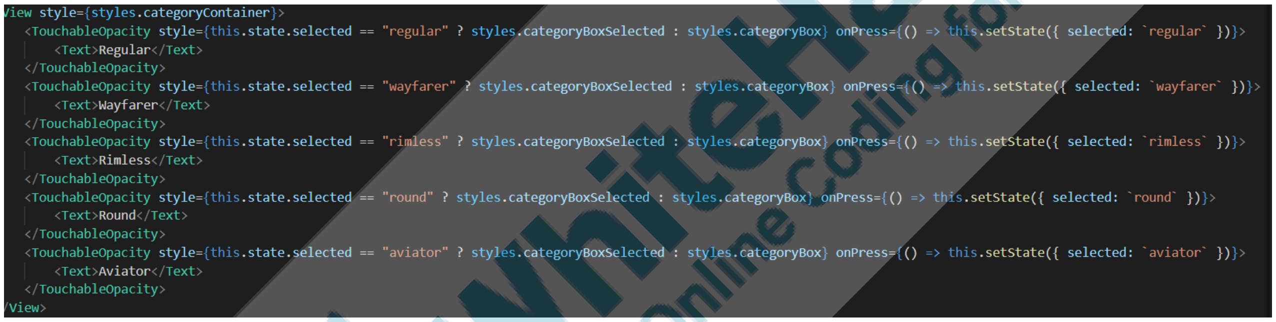
```
categoryContainer: {
    flex: 0.4,
    justifyContent: "center",
    alignItems: "center",
    flexDirection: "row",
    marginBottom: RFValue(10)
},
categoryBox: {
    flex: 0.2,
    borderRadius: 30,
    borderWidth: 1,
    backgroundColor: "white",
    width: "100%",
    padding: RFValue(3),
    margin: 1,
    alignItems: "center"
},
categoryBoxSelected: {
    flex: 0.2,
    borderRadius: 30,
    borderWidth: 1,
    backgroundColor: "#efb141",
    width: "100%",
    padding: RFValue(3),
    margin: 1,
    alignItems: "center"
}
```

	Guide the student to define the “selected” state variable.	



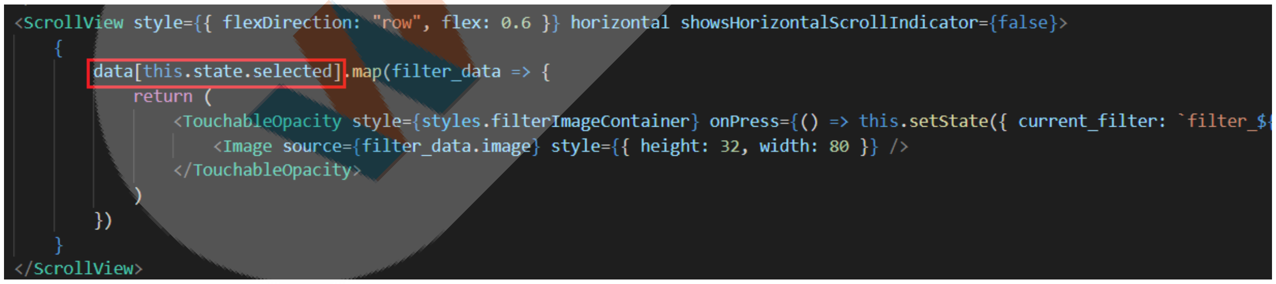
```
constructor(props) {
    super(props)
    this.state = {
        hasCameraPermission: null,
        faces: [],
        current_filter: "filter_1",
        selected: "aviator"
    }
```

Guide the student to write a return method to render text and category boxes and update the "selected" state value



```
iew style={styles.categoryContainer}>
  <TouchableOpacity style={this.state.selected == "regular" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `regular` })}>
      <Text>Regular</Text>
  </TouchableOpacity>
  <TouchableOpacity style={this.state.selected == "wayfarer" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `wayfarer` })}>
      <Text>Wayfarer</Text>
  </TouchableOpacity>
  <TouchableOpacity style={this.state.selected == "rimless" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `rimless` })}>
      <Text>Rimless</Text>
  </TouchableOpacity>
  <TouchableOpacity style={this.state.selected == "round" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `round` })}>
      <Text>Round</Text>
  </TouchableOpacity>
  <TouchableOpacity style={this.state.selected == "aviator" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `aviator` })}>
      <Text>Aviator</Text>
  </TouchableOpacity>
/View>
```

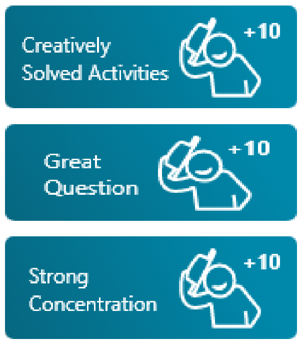
Guide the student to render based on the state variable value.



```
<ScrollView style={{ flexDirection: "row", flex: 0.6 }} horizontal showsHorizontalScrollIndicator={false}>
    {
        data[this.state.selected].map(filter_data => {
            return (
                <TouchableOpacity style={styles.filterImageContainer} onPress={() => this.setState({ current_filter: `filter_${
                    <Image source={filter_data.image} style={{ height: 32, width: 80 }} />
                </TouchableOpacity>
            )
        })
    }
</ScrollView>
```

Guide the student to test the output.

Teacher Guides Student to Stop Screen Share

<u>FEEDBACK</u> ● **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? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	*The student uses the markdown editor to write their reflections in a reflection journal.*

Activity	Activity Name	Links
Teacher Activity 1	Previous Class Code	https://github.com/whitehatjr/PRO-C183-Code-Ref
Teacher Activity 2	FRAPP Design Model	https://s3-whjr-v2-prod-bucket.whjr.online/baa4d669-2cc8-4284-bd31-8caf8426d739.png

Teacher Activity 3	Final Reference Code	https://github.com/whitehatjr/PRO-C184-Code-Ref
Student Activity 1	Previous Class Code	https://github.com/whitehatjr/PRO-C183-Code-Ref