Topic	FILTERS CATEGORY	
Class Description	**Students will learn to add different categories of filters based on the different frame 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**

WARM-UP SESSION - 5 mins

CONTEXT

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

Teacher Starts Slideshow **Slide 1 to 3** Refer to speaker notes and follow the instructions on each slide.	
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** ● Greet the student. ● Revision of previous class activities.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Continue WARM-UP Session **Slide 4 to 10**	
Following are the session deliverables: ● Appreciate the student. ● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	

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?	**ESR:** ● We learned how to add scrollable options to choose from multiple face

	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.	filter images in the app. **ESR**: Varied. **ESR**: Yes.

Teacher Ends Slideshow

TEACHER-LED ACTIVITY - 15 mins

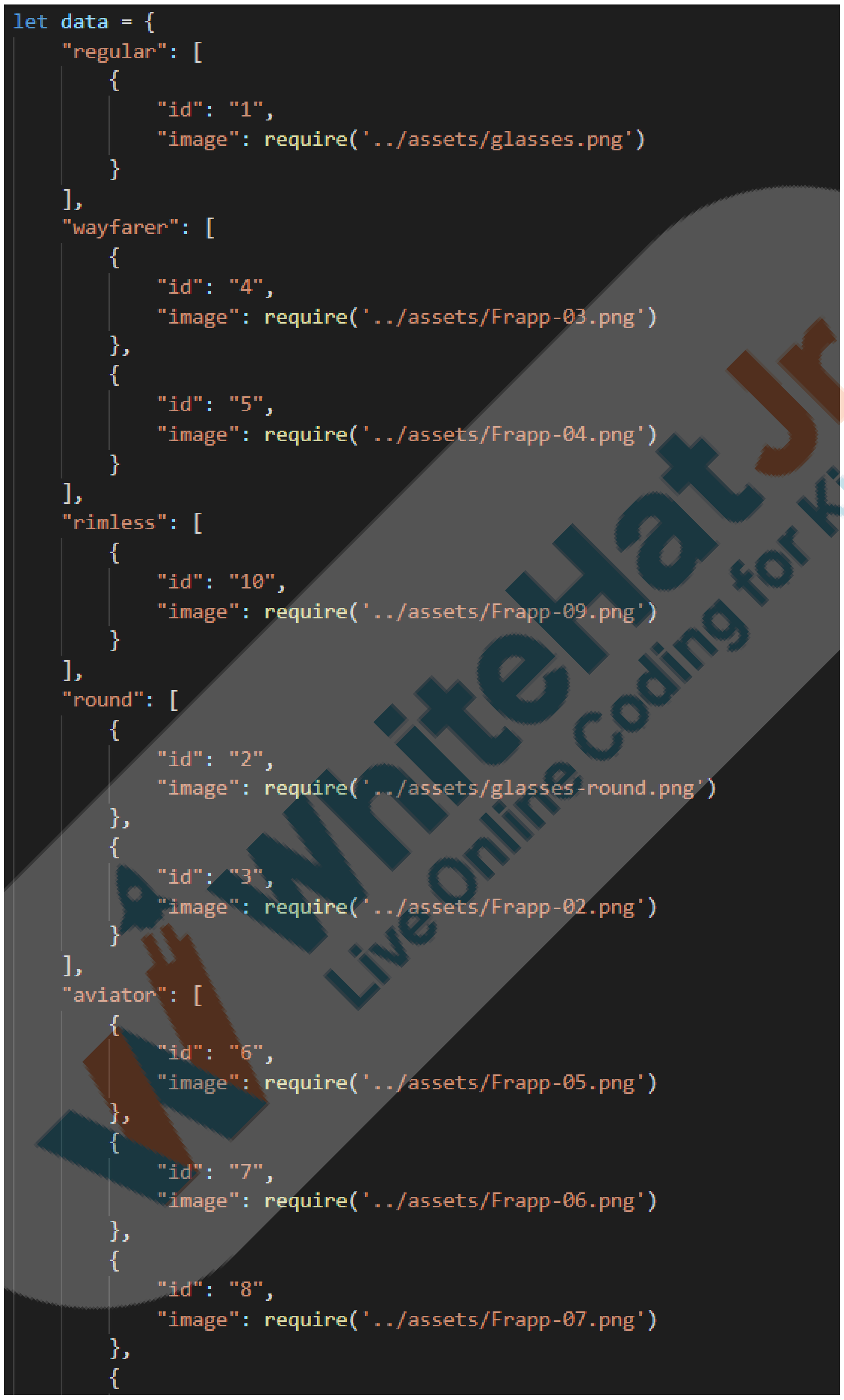
Teacher Initiates Screen Share

CHALLENGE

- **Design App UI.**

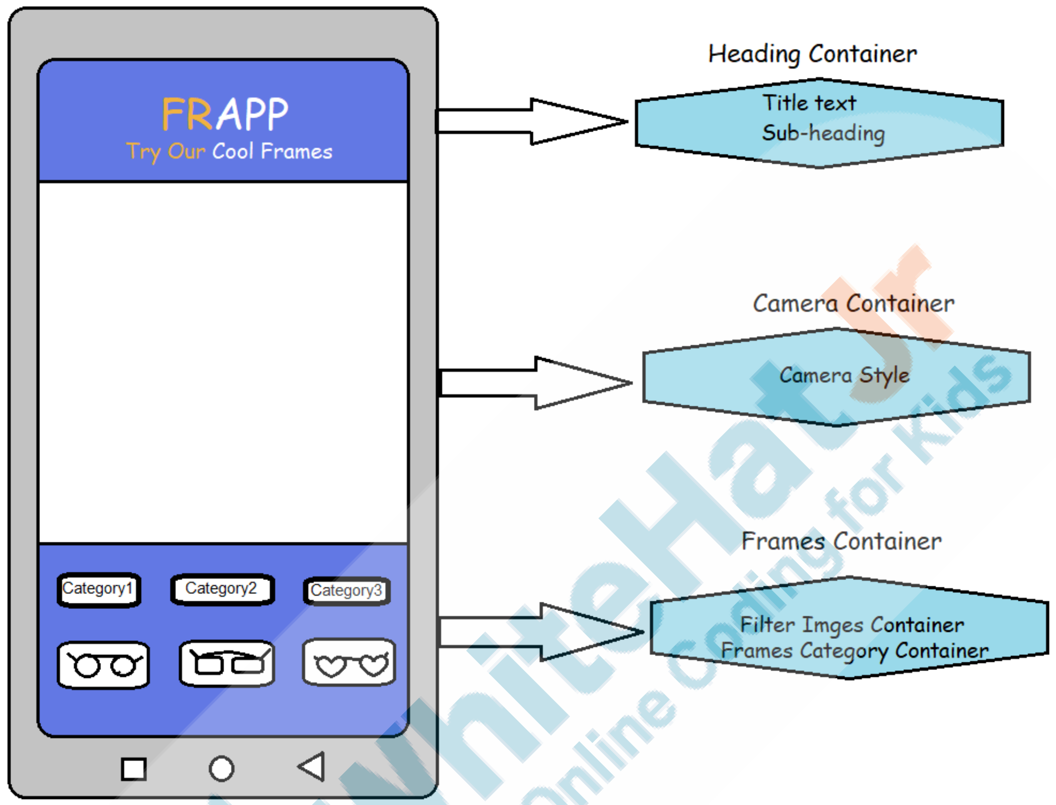
● **Divide multiple face filters into categories.**		
Step 2: Teacher-led Activity (15 mins)	*<The teacher clones the code Teacher Activity 1.* ***Note**: Do install node modules.>* ***[Teacher Activity 1]*** Before we can begin, can you tell me how we can categorise different frames in the app? ***Note**: 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 our images? ***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**.	**ESR**: Based on brand names, price, shape of the frames. **ESR**: Round, Square.

	Let’s modify the **data object variable** (defined in the previous class) with the images **id** and the image **source**, with the category names.	
./screens/Main.js		



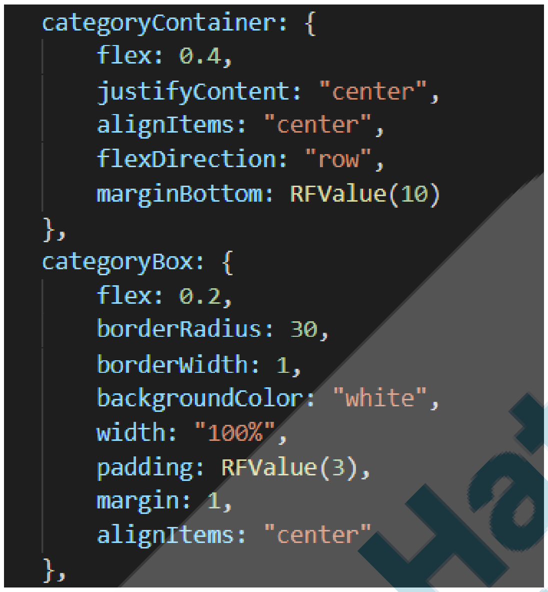
```
let data = {
    "regular": [
        {
            "id": "1",
            "image": require('../assets/glasses.png')
        }
    ],
    "wayfarer": [
        {
            "id": "4",
            "image": require('../assets/Frapp-03.png')
        },
        {
            "id": "5",
            "image": 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 the 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 it 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 the 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

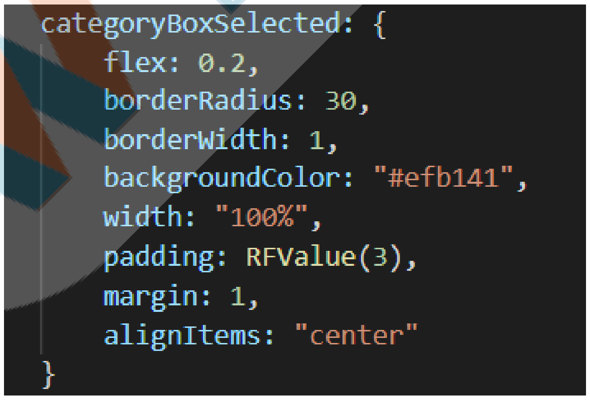


```
categoryContainer: {
    flex: 0.4,
    justifyContent: "center",
    alignItems: "center",
    flexDirection: "row",
    marginBottom: RFValue(10)
},
categoryBox: {
    flex: 0.2,
    borderRadius: 30,
    borderWidth: 1,
    backgroundColor: "white",
    width: "100%",
    padding: RFValue(3),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

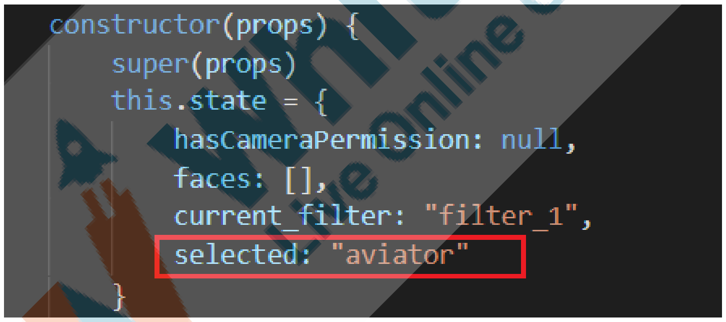
Also, to show which category of the frames is currently selected, we can highlight it with a different colour.

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"
}
```

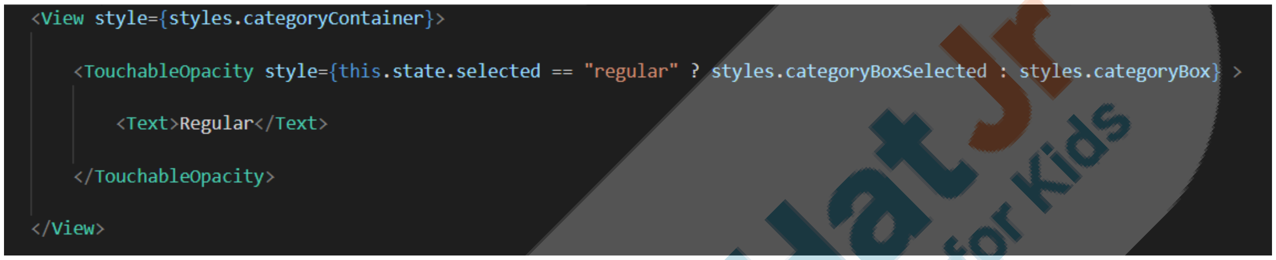
	Once we have the styling ready, what should we do next? Perfect! Since we need to show the frames of the selected category only, we should first know which frame category is selected, right? To do this we would need a state variable. Let's take a variable called, **selected**, with the initial category as "**aviator**".	**ESR**: We need to show the frames of the selected category only. **ESR**: Yes.
./screens/Main.js 		
	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 it's state value to highlight the selected category.

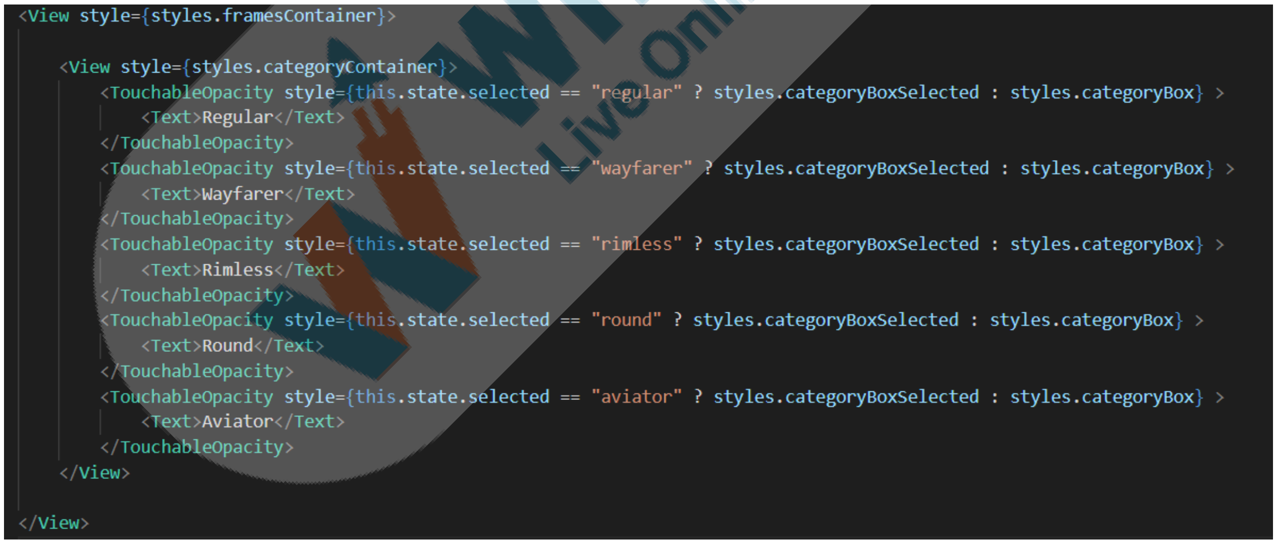
./screens/Main.js



```
<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

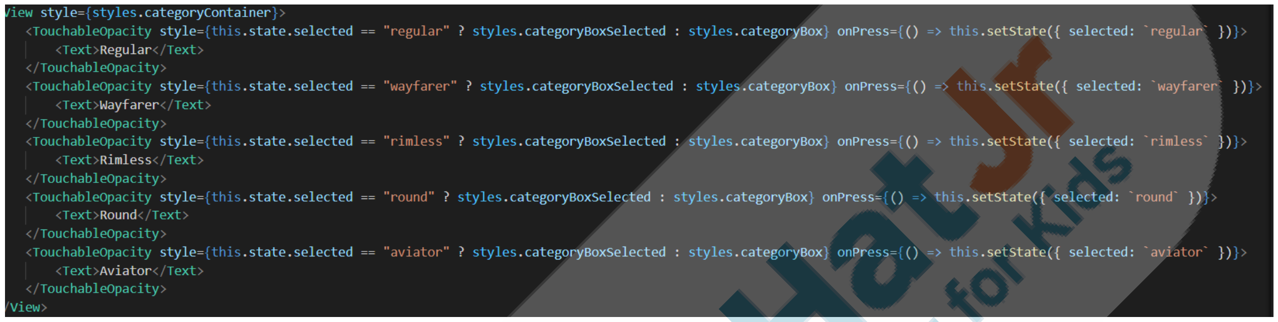


```
<View style={styles.framesContainer}>

    <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.	

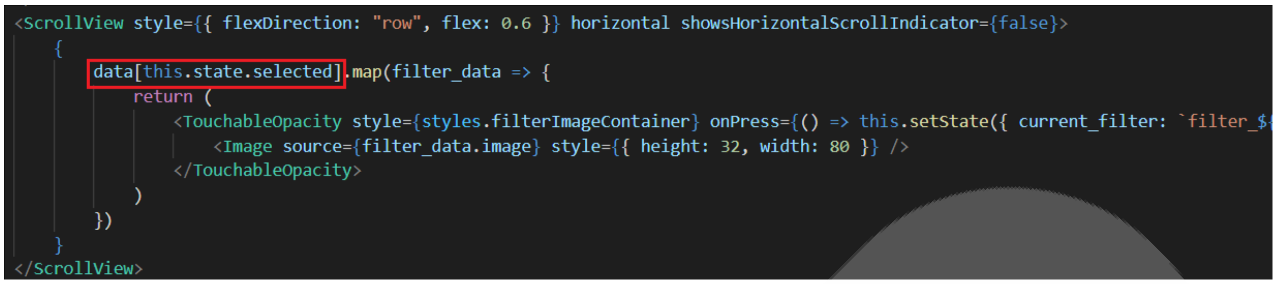
./screens/Main.js



```
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>
```

	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 the 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

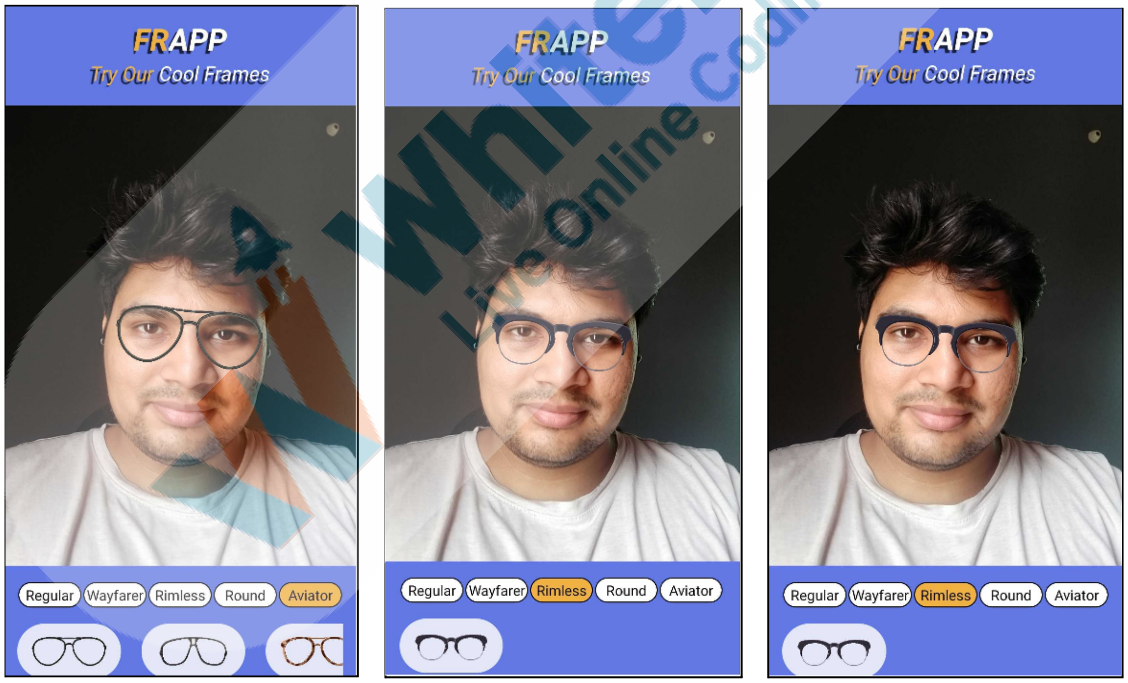


```
<ScrollView style={{ flexDirection: "row", flex: 0.6 }} horizontal showsHorizontalScrollIndicator={false}>
    {
        data[this.state.selected].map(filter_data => {
            return (
                <TouchableOpacity style={styles.filterImageContainer} onPress={() => this.setState({ current_filter: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

Now let's test the final output using expo.

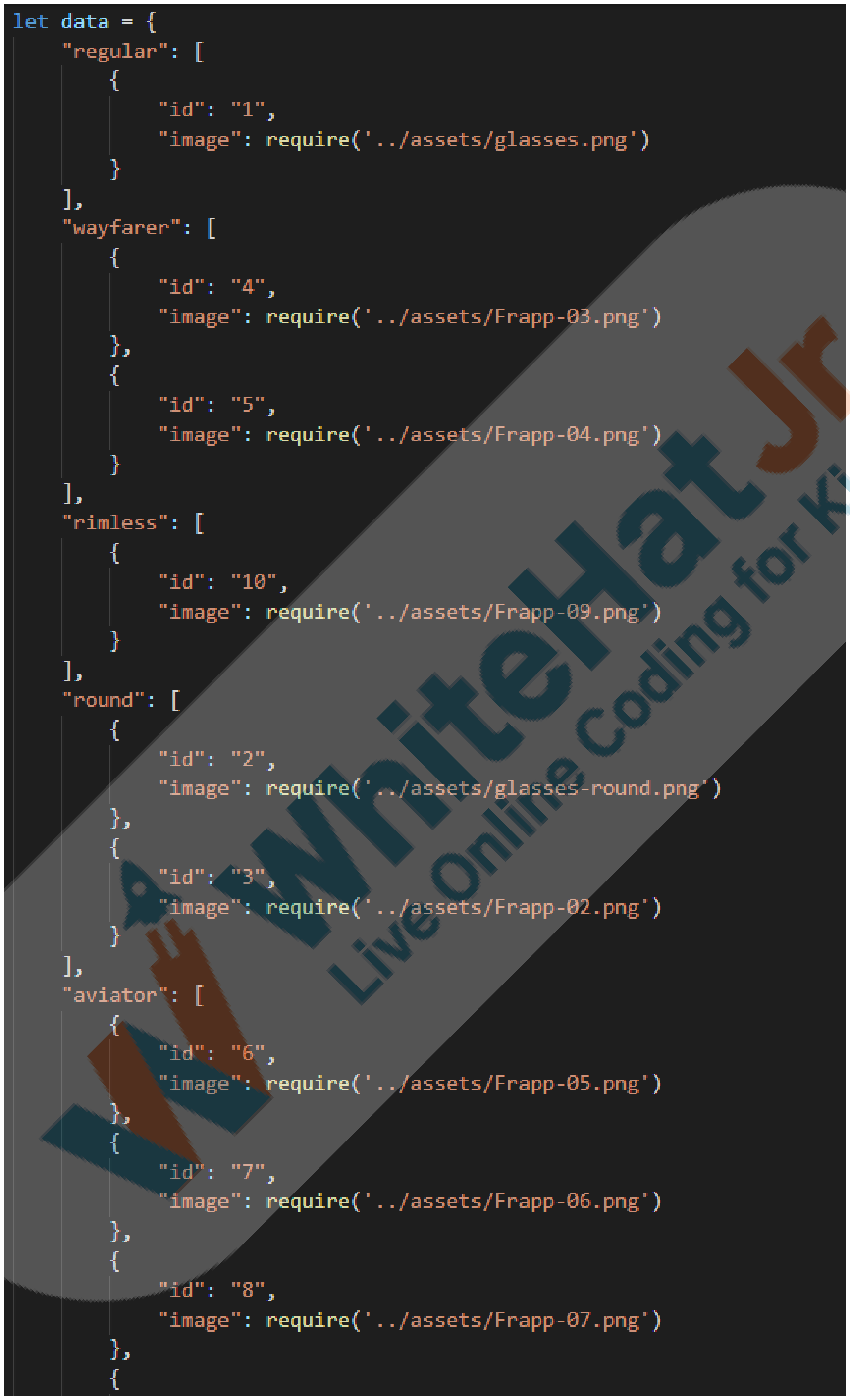
Navigate to the working directory and open cmd.

Command: ***expo start -c***



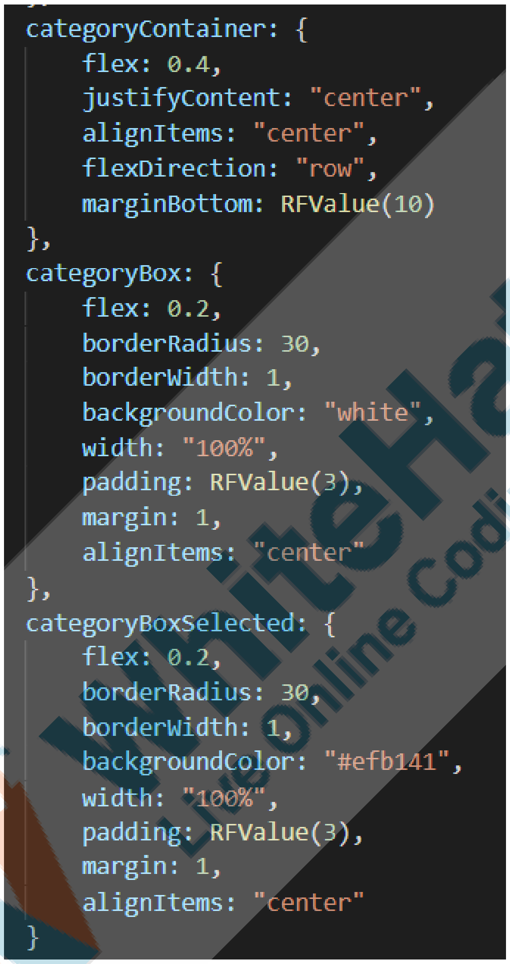
	Now you will add categories for filters in the app. Are you excited?	**ESR**: Yes!
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
Teacher Starts Slideshow **Slide 11 to 12** Refer to speaker notes and follow the instructions on each slide.		
We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.		
Teacher Ends Slideshow		
STUDENT-LED ACTIVITY - 20 mins		
• **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 update data variables with the categories.	

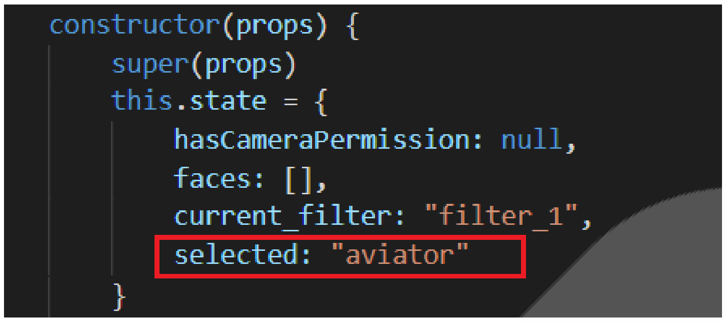


```
let data = {
    "regular": [
        {
            "id": "1",
            "image": require('../assets/glasses.png')
        }
    ],
    "wayfarer": [
        {
            "id": "4",
            "image": require('../assets/Frapp-03.png')
        },
        {
            "id": "5",
            "image": 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:	

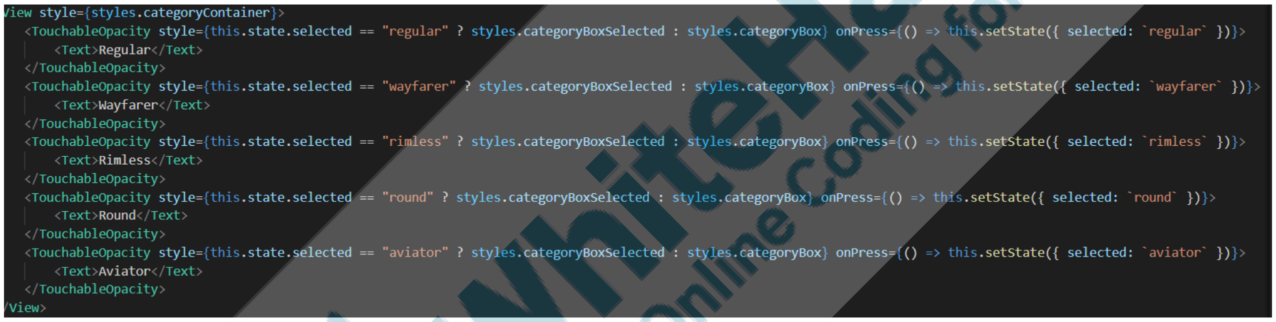
	• *Category Container* • *Category Box* • *Category Box Selected*	
		
	Guide the student to define the "selected" state variable.	

```
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"
}
```



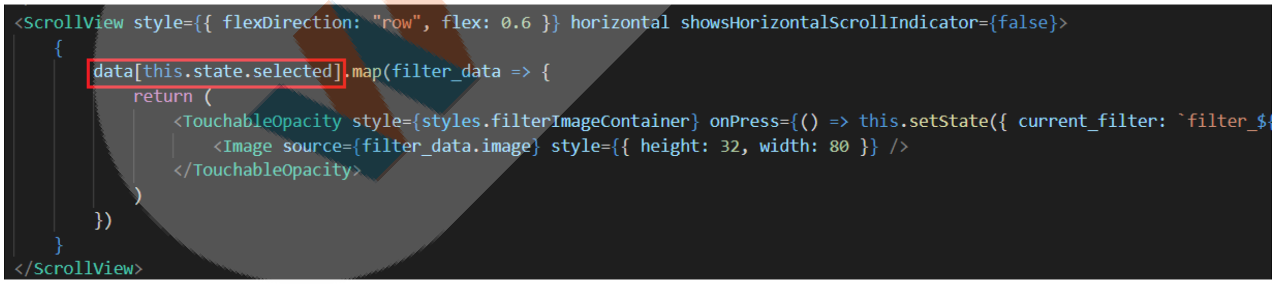
```
constructor(props) {
    super(props)
    this.state = {
        hasCameraPermission: null,
        faces: [],
        current_filter: "filter_1",
        selected: "aviator"
    }
```

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



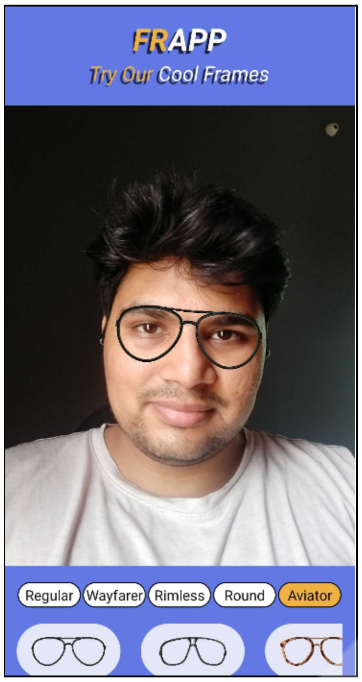
```
iew style={styles.categoryContainer}>
  <TouchableOpacity style={this.state.selected == "regular" ? styles.categoryBoxSelected : styles.categoryBox} onPress={() => this.setState({ selected: `regular` })}>
      <Text>Regular</Text>
  </TouchableOpacity>
  <TouchableOpacity style={this.state.selected == "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>
```

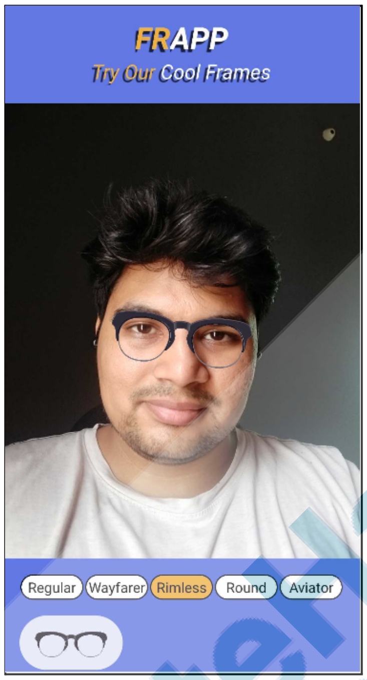
Guide the student to render based on the state variable value.

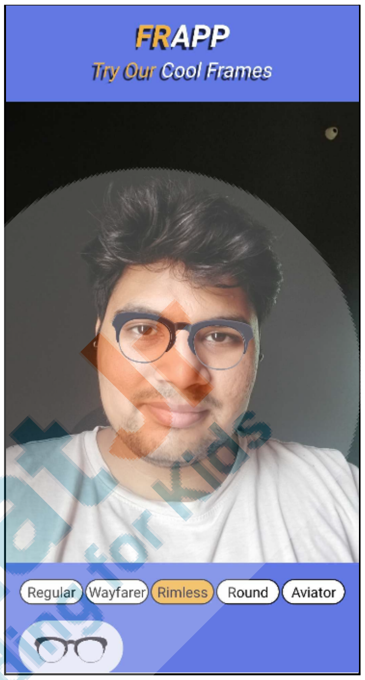


```
<ScrollView style={{ flexDirection: "row", flex: 0.6 }} horizontal showsHorizontalScrollIndicator={false}>
    {
        data[this.state.selected].map(filter_data => {
            return (
                <TouchableOpacity style={styles.filterImageContainer} onPress={() => this.setState({ current_filter: `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

WRAP UP SESSION - 5 mins

Teacher Starts Slideshow
Slide 13 to 16

Activity details
Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

Continue WRAP-UP Session
Slide 17 to 22

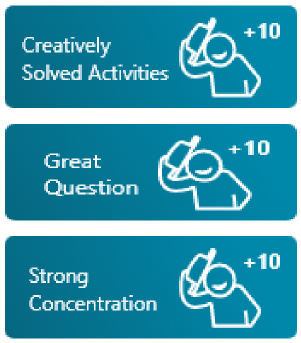
Activity Details

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

<u>FEEDBACK</u>

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	**Student Action**
You get Hats off for your excellent work!	*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 DISCUSSION
Refer the document below in Activity Links Sections

Teacher Clicks ✖ End Class		
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
Teacher Reference 1	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/ed8f5e78-ab08-44ec-97ab-e514c0e7d75c.pdf

Teacher Reference 2	Project Solution	https://github.com/whitehatjr/AR-PRO-C184
Teacher Reference 3	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/e2587acf-98b2-4973-9438-316c2375b258.html
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/ae459b04-f1ce-4c7e-b80a-33ad42899787.pdf