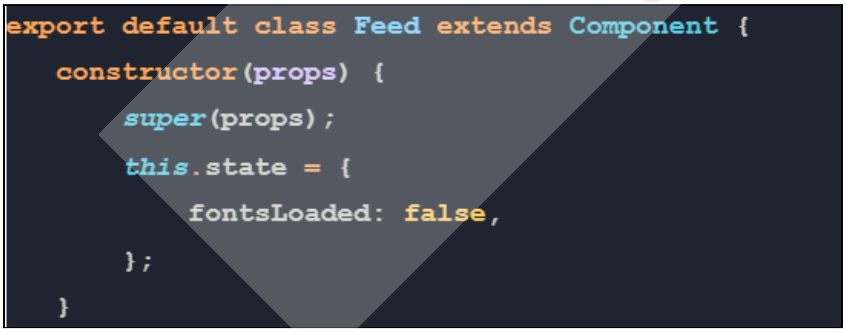
Topic	TEXTINPUT AND STYLING	
Class Description	**In this class, the students will be creating the CreateStory.js screen with a feature to preview images.**	
Class	**C83**	
Class time	**45 mins**	
Goal	• Create text inputs for our form. • Add preview image functionality.	

Resources Required	• Teacher Resources:
	  ○ Visual Studio Code Editor
	  ○ laptop with internet connectivity
	  ○ earphones with mic
	  ○ notebook and pen
	• Student Resources:
	  ○ Visual Studio Code Editor
	  ○ laptop with internet connectivity
	  ○ 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		
Teacher starts slideshow **from slides 1 to 12** Refer to speaker notes and follow the instructions on each slide.		
Activity details		**Solution/Guidelines**

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

Run the presentation from slide 1 to slide 4. **The following are the warm-up session deliverables:** ● Connect the student to the Trial class. ● Variables and objects. ● Warm-Up quiz session.	Click on the slide show tab and present the slides.
QnA Session	
Question	**Answer**
Which of the following is true about StoryCard.js? A. It is the boilerplate code. B. It already has the fonts loaded. C. It has essential modules imported for us. D. All of the above	**D**
In the following piece of code why do set fontsLoaded as false?  A. fonts are not loaded initially B. it is the default setting C. because we want our fonts to be false all the time D. fonts can be changed by it	**A**
Continue the warm-up session	

```
export default class Feed extends Component {
   constructor(props) {
       super(props);
       this.state = {
           fontsLoaded: false,
       };
   }
```

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

Activity details	Solution/Guidelines
Run the presentation from slide 5 to slide 12 to set the problem statement. **The following are the warm-up session deliverables:** • Introduce students to the coding environment - Workspace, blocks, and output. • Steps to write and run the code. • Introduce the concepts of Teacher-Led Activity.	Narrate the story by using hand gestures and voice modulation methods to bring in more student interest.
Teacher ends slideshow	
TEACHER-LED ACTIVITY - 15 mins	
Teacher Initiates Screen Share	
CHALLENGE • **Create the form for users to enter new stories.** • **Preview images that the user selects.**	
Step 2: Teacher-led Activity (15 min)	*The teacher opens the code from the previous class or downloads the code from* *Teacher Activity 1* ***Note -** If the student and/or teacher is using the snack editor for these classes, please refer to the support document in Teacher Activity 4.* We will start by working on the CreateStory screen. This screen will have a form through which the user will be able to submit a new story. This screen would look like -

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

In the code that we have just cloned, we can observe that we already have a template for our ***CreateStory.js*** in which we have the fonts loaded, constructor added, etc.

```
83t > screens > JS CreateStory.js > CreateStory > render
1   import React, { Component } from 'react';
2   import { Text, View } from 'react-native';
3
4   import AppLoading from 'expo-app-loading';
5   import * as Font from 'expo-font';
6
7   let customFonts = {
8       'Bubblegum-Sans': require('../assets/fonts/BubblegumSans-Regular.ttf'),
9   };
```

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

```
constructor(props) {
    super(props);
    this.state = {
        fontsLoaded: false
    };
}

async _loadFontsAsync() {
    await Font.loadAsync(customFonts);
    this.setState({ fontsLoaded: true });
}

componentDidMount() {
    this._loadFontsAsync();
}
```

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

```
render() {
    if (!this.state.fontsLoaded) {
        return <AppLoading />
    } else {
        <View style={styles.container}>
            <SafeAreaView style={styles.droidSafeArea} />
            <View style={styles.appTitle}>
                <View style={styles.appIcon}>
                    <Image
                        source={require("../assets/logo.png")}
                        style={styles.iconImage}
                    ></Image>
                </View>
                <View style={styles.appTitleTextContainer}>
                    <Text style={styles.appTitleText}>New Story</Text>
                </View>
            </View>
        </View>
    }
}
```

We also have the styles for it added in the boilerplate -

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

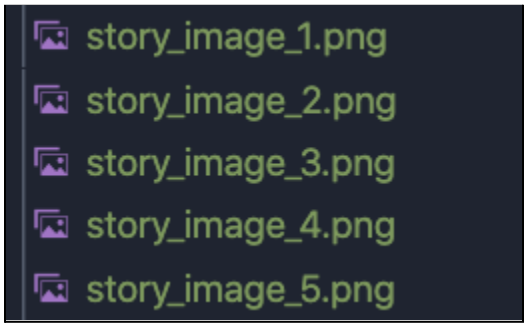
```
const styles = StyleSheet.create({
    container: {
        flex: 1,
        backgroundColor: "#15193c"
    },
    droidSafeArea: {
        marginTop: Platform.OS === "android" ? StatusBar.currentHeight : RFValue(35)
    },
    appTitle: {
        flex: 0.07,
        flexDirection: "row"
    },
    appIcon: {
        flex: 0.3,
        justifyContent: "center",
        alignItems: "center"
    },
    iconImage: {
        width: "100%",
        height: "100%",
        resizeMode: "contain"
    },
    appTitleTextContainer: {
        flex: 0.7,
        justifyContent: "center"
    },
    appTitleText: {
        color: "white",
        fontSize: RFValue(28),
        fontFamily: "Bubblegum-Sans"
    }
})
```

	We have a template mentioned above for our ***CreateStory.js*** screen in the boilerplate. Now can you guess what all fields would we need for our form?	**ESR:** 1. image 2. title 3. description 4. story

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

		5. moral
	That's right! For the image part, we want our app to look great. For that, instead of allowing the users to upload a picture, we will give them predefined images to choose from. This way, all the images in our app would be in alignment with the theme of the app. Now, it would be great if the user can see the image they are selecting, therefore we need to build it in a way where the users can preview an image. We can provide the users with a dropdown menu to choose the image. For that, let's install the following dependency - *expo install react-native-dropdown-picker*	
	We have about 5 images in our assets folder for our stories -	

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.



If we have to add a dropdown menu, it is important that we keep an image selected by default and display the preview of that image too. So let's do that -

```
import React, { Component } from "react";
import {
 View,
 Text,
 StyleSheet,
 SafeAreaView,
 Platform,
 StatusBar,
 Image,
 ScrollView,
 TextInput,
 Dimensions
} from "react-native";
import { RFValue } from "react-native-responsive-fontsize";
import DropDownPicker from "react-native-dropdown-picker";
```

We'll start by making the relevant imports in our **CreateStory.js**

```
constructor(props) {
    super(props);
    this.state = {
      fontsLoaded: false,
      previewImage: "image_1",
```

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

```
  };
}
```

Next, we will add a new state **previewImage** with **image_1** as the value. This is because we want to keep the first image as the previewImage by default.

Next, in the render() function inside the else condition, we will add a dictionary **preview_images** with key as **image_1, image_2, image_3, image_4** and **image_5**. The values would be the respective paths of the images.

```
render() {
    if (!this.state.fontsLoaded) {
        return <AppLoading />;
    } else {
        let preview_images = {
            "image_1": require("../assets/story_image_1.png"),
            "image_2": require("../assets/story_image_2.png"),
            "image_3": require("../assets/story_image_3.png"),
            "image_4": require("../assets/story_image_4.png"),
            "image_5": require("../assets/story_image_5.png")
        }
```

Lastly, we will add a scroll view into our screen containing code for preview image -

```
<View style={styles.fieldsContainer}>
   <ScrollView>
       <Image
           source={preview_images[this.state.previewImage]}
           style={styles.previewImage}
       ></Image>
       <View style={{ height: RFValue(this.state.dropdownHeight)
}}>
           <DropDownPicker
               items={[
```

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

```
                { label: "Image 1", value: "image_1" },
                { label: "Image 2", value: "image_2" },
                { label: "Image 3", value: "image_3" },
                { label: "Image 4", value: "image_4" },
                { label: "Image 5", value: "image_5" },
              ]}
              defaultValue={this.state.previewImage}
              open={this.state.dropdownHeight == 170 ? true :
false}
              onOpen={() => {
                this.setState({ dropdownHeight: 170 });
              }}
              onClose={() => {
                this.setState({ dropdownHeight: 40 });
              }}
              style={{
                backgroundColor: "transparent",
                borderWidth: 1,
                borderColor: "white",
              }}
              textStyle={{
                color: this.state.dropdownHeight == 170 ?
"black" : "white",
                fontFamily: "Bubblegum-Sans",
              }}
              onSelectItem={(item) =>
                this.setState({
                  previewImage: item.value,
                })
              }
            />
```

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

```
        </View>
    </ScrollView>
</View>
<View style={{ flex: 0.08 }} />
```

Here, we have added a **<ScrollView>** in which our code for the preview image is there. It is a scrollview because later on, more fields for input will be added.

We then have an **<Image>** tag that displays the image in preview from the state.

We then finally have the **<DropDownPicker>** component which is our dropdown menu.

We have specified the dropdown options with the help of the **items** attribute and we have given it the default value with the **defaultValue** attribute.

We give some styles to the container in which the options will be displayed on clicking the dropdown with the **style** attribute.

We are keeping its background color as **transparent** because we want the dropdown field to have the color same as the background.

Similarly, we are styling the texts using the **textStyle** attribute.

We have added an **onSelectItem** attribute which changes the state **previewImage** to the newly selected option.

We will also add the styling of these -

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

```
fieldsContainer: {
    flex: 0.85
},
previewImage: {
    width: "93%",
    height: RFValue(250),
    alignSelf: "center",
    borderRadius: RFValue(10),
    marginVertical: RFValue(10),
    resizeMode: "contain"
}
```

Here, our **fieldsContainer** will take 85% of the height, but since it's inside a **<ScrollView>**, it will be scrollable.

	Now once the dropdown's value is changed, it will call the **setState()** function which will re-render the entire screen. Since we are displaying the preview image based on the state, this time, the previewed image will be the one user has selected. By now, the app looks like -	

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

	Great! Now we are left with the fields for **title, description, story and moral**. It's your turn.	

Teacher Stops Screen Share

STUDENT-LED ACTIVITY - 20 mins

	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.**

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

<table>
<tr><td colspan="3"><u>ACTIVITY</u>
• Complete the form by adding all the text input fields and styling.</td></tr>
<tr><td colspan="3">Teacher starts slideshow for slide 13 to 14
Refer to speaker notes and follow the instructions on each slide.</td></tr>
<tr><td>Step 3: Student-Led Activity (20 mins)</td><td>Please refer to <Student Activity 1> to clone the repository with the code we just created.</td><td>Student refers to <Student Activity 1> and clones the repository</td></tr>
<tr><td></td><td>Now, please add the remaining 4 fields.

Remember that our input fields would look the same, but the title field is small and the fields for description, story and moral would be big.

For that, we can use attributes like multiLine and numberOfLines on the <TextInput> component.

Please create the fields.

Teacher helps and guides the student in creating the fields.</td><td>Student writes the code to create the fields and styles them.</td></tr>
<tr><td colspan="3">

```
<TextInput
  style={styles.inputFont}
  onChangeText={title => this.setState({ title })}
  placeholder={"Title"}
  placeholderTextColor="white"
```

</td></tr>
</table>

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

```
            />

            <TextInput
              style={[
                styles.inputFont,
                styles.inputFontExtra,
                styles.inputTextBig
              ]}
              onChangeText={description => this.setState({ description
})}
              placeholder={"Description"}
              multiline={true}
              numberOfLines={4}
              placeholderTextColor="white"
            />
            <TextInput
              style={[
                styles.inputFont,
                styles.inputFontExtra,
                styles.inputTextBig
              ]}
              onChangeText={story => this.setState({ story })}
              placeholder={"Story"}
              multiline={true}
              numberOfLines={20}
              placeholderTextColor="white"
            />

            <TextInput
              style={[
                styles.inputFont,
                styles.inputFontExtra,
```

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

```
            styles.inputTextBig
          ]}
          onChangeText={moral => this.setState({ moral })}
          placeholder={"Moral of the story"}
          multiline={true}
          numberOfLines={4}
          placeholderTextColor="white"
        />
```

Here, inside the **<ScrollView>**, we have added 4 more views with **<TextInput>** in it. These text inputs contain the styles for their styling.

They also contain an **onChangeText** attribute, which sets their value as a state.

We have used the **placeholder** attribute for setting the placeholders and we have specified if the particular textbox is **multiline** or not. If it is, we have set the number of lines with the **numberOfLines** attribute and finally, we set the color of the placeholders with **placeholderTextColor** attribute.

We also add the relevant styles for these -

```
inputFont: {
  height: RFValue(40),
  borderColor: "white",
  borderWidth: RFValue(1),
  borderRadius: RFValue(10),
  paddingLeft: RFValue(10),
  color: "white",
  fontFamily: "Bubblegum-Sans"
},
inputFontExtra: {
  marginTop: RFValue(15)
},
inputTextBig: {
  textAlignVertical: "top",
  padding: RFValue(5)
```

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

```
}
```

height of **undefined** means that it can have the height it needs to take. For us, if we have a big textbox with 4 lines, then it can take as much height as it wants.

	With this, our app looks like -	

	Okay, now you might be wondering why we haven't added a submit button. We'll do that when we integrate our app with Firebase later.	

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 5 mins	
FEEDBACK • **Appreciate the student for their attentiveness in the class.** • **Get them to play around with different ideas.**	
Teacher can show slideshow **from slides 15 to 24** Refer to speaker notes and follow the instructions on each slide.	
Run the presentation from slide 15 to slide 24. **Following are the warm up session deliverables:** • **Explain the facts and trivias** • **Next class challenge** • **Project for the day** • **Additional Activity**	Guide the student to develop the project and share with us.
QnA Session	
Question	**Answer**
To implement dropdown functionality what do we use? A. DropDown B. DropPicker C. DropDownPicker D. DownDropPicker	**C**
To allow the user to add multiple lines in TextInput which prop should we use? A. multiLine B. multipleLines.	**A**

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

C. style D. placeholder		
Which prop of the TextInput is used to set the state of the constructor? A. onChangeText B. placeholder C. multiLine D. placeholderTextColor		**A**
End the quiz panel		
Step 4: **Wrap-Up** **(5 min)**	Let's quickly wrap up today's class.	
	Amazing work today! You get a “hats-off”. In the next class, we’ll be working on stack navigation and adding the text-to-speech functionality. Alright. See you in next class.	*Make sure you have given at least 2 Hats Off during the class for:*
Project Overview **Spectagram Stage - 3** **Goal of the Project:**		*The students engage with the teacher over the project.*

© 2022 - BYJU’S Future School.
Note: This document is the original copyright of BYJU’S Future School.
Please don't share, download or copy this file without permission.

In Class 83, we've created text input for our form and also added the preview image functionality. In this project, you will practice the concepts learned in the class by building the **CreatePost** component which contains the form. *This is a continuation project of 81 & 82, please make sure to finish that before attempting this one. **Story:** Jenny is a photographer. She wants to share pictures taken by her with others. At the same time, she wants to create a space for others to share their talent too. She has decided to create a social media app. She has asked for your help to create an app. Guide Jenny to create a text input component to get the caption and also add the preview image functionality.		
Teacher ends slideshow		
Teacher Clicks		
ADDITIONAL ACTIVITY		
Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.* Use these as guiding questions: • What happened today?	*The student uses the markdown editor to write their reflections in a reflection journal.*

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

	○ 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?	

Activity	Activity Name	Links
Teacher Activity 1	Teacher Boilerplate Code	https://github.com/pro-whitehatjr/Story-Telling-App-83-TB
Teacher Activity 2	Reference Code	https://github.com/pro-whitehatjr/ST-83-Solution
Teacher Activity 3	Teacher Aid	https://drive.google.com/file/d/1WA1BQff4dmgv5BInU3f_imk4vlpvAyMa/view?usp=sharing
Teacher Activity 4	Snack Support Document	https://docs.google.com/document/d/11vq49uJQCfdxaUUzOoY7A65aau0kZqNMFhObZH-e71Y/edit?usp=sharing
Student Activity 1	Boilerplate Code	https://github.com/pro-whitehatjr/ST-83-Boilerplate
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_V3_C83_LITE_withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/6d8a0e11-dddd-4865-bacd-078f6a029be4.pdf

© 2022 - BYJU'S Future School.
Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

© 2022 - BYJU’S Future School.
Note: This document is the original copyright of BYJU’S Future School.
Please don't share, download or copy this file without permission.