| Topic	Home Screen and Navigation
Class Description	The student learns to create buttons using touchable opacity. The student also learns to navigate from the Home screen to other screens.
Class	C77
Class time	50 mins
Goal	- Planning different elements in the app's home screen and setting their placements to achieve a nice-looking UI. - Add elements to the screen such as buttons and images.
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	10 mins
	Teacher-Led Activity	15 mins
	Student-Led Activity	20 mins
	Wrap-Up	5 mins

WARM-UP SESSION - 10 mins

Teacher starts slideshow 🖥 from slides 1 to 11

Refer to speaker notes and follow the instructions on each slide.

Activity details	
Hey <student name>. How are you? It's great to see you! Are you excited to learn something new today? ***Run the presentation from slide 1 to slide 4.*** **The following are the warm-up session deliverables:** ● Connecting students to the previous class.	**ESR**: I am doing great!
Question	

Choose the correct block of code that can be used to add background images to Homescreen.
 	
A.
```\n<Background\n   source={require('../assets/stars.gif')}\n   style={styles.backgroundImage}>\n```
B.
```\n<Image\n   source={require('../assets/stars.gif')}\n   style={styles.backgroundImage}>\n```
C.
```\n<ImageBackground\n   source={require('../assets/stars.gif')}\n   style={styles.backgroundImage}>\n```
D.
```\n<BackgroundImage\n   source={require('../assets/stars.gif')}\n   style={styles.backgroundImage}>\n```
	C

Choose the correct block of code to use the onpress property of <Touchable opacity> for adding navigation to spacecraft screen.	D
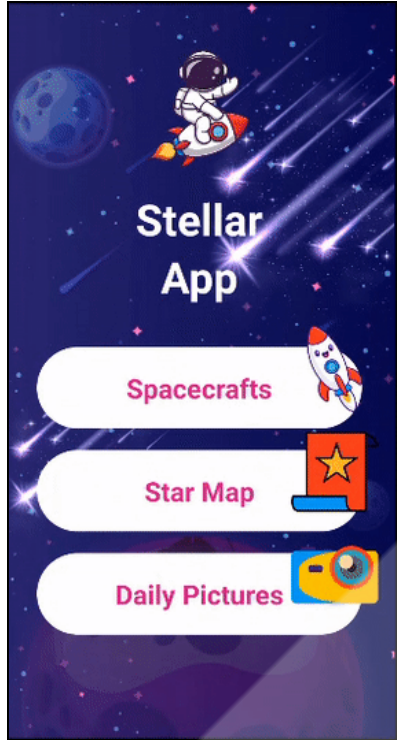
A.
```
{/* <TouchableOpacity
    style={styles.routeCard} onPress={() =>
  this.props.navigation("SpaceCraft")
}> */}
```

B.
```
{/* <TouchableOpacity
    style={styles.routeCard} onPress={() =>
  navigation.navigate("SpaceCraft")
}> */}
```

C.
```
{/* <TouchableOpacity
    style={styles.routeCard} onPress={() =>
  this.props("SpaceCraft")
}> */}
```

D.
```
{/* <TouchableOpacity
    style={styles.routeCard} onPress={() =>
  this.props.navigation.navigate("SpaceCraft")
}> */}
```

Continue the warm-up session	
Teacher Action	**Student Action**

Run the presentation from slide 5 to slide 11 to set the problem statement.	Narrate the slides by using hand gestures and voice modulation methods to bring in more interest in students
Following are the session deliverables: ● **Add a Title to the Home Screen.** ● **Create buttons for different screens**	

Teacher ends slideshow

TEACHER-LED ACTIVITY- 1 (15 mins)

● **Teacher Initiates Screen Share**

ACTIVITY
● **Add the Title to the Home Screen on our app.**
● **Create buttons for different screens on our app.**

Teacher Action	Student Action
<The teacher opens code from the previous class, Teacher Activity 1> When you open any app what are the things that you see? Yes, and this makes it very helpful and easy to have all the functionality in one place. It makes the app more user-friendly.	**ESR:** We see that there is the main screen. This main screen contains buttons to go to different screens or some other functionality such as settings or changing profile picture.
Similarly, we'll be doing the same thing in our app to make it more user-friendly. Let's get started then.	

So can you tell me what all elements do we want to have on our screen?	**ESR:** - We want to have a title on the screen. - And 3 buttons showing the titles of the 3 different screens. - Add background images for the screens.
Alright so let's first start with the Title for the app on the home screen. Where do we usually have the title for any app? What all React Native components do we require to give the title to the app? We have already imported the **Text** component and the **View** component from React Native. We'll also need to add some styles to the Title for that we'll add a stylesheet where we'll be adding the styles for all the components. Let's use these to create the title. Import the **Stylesheet** from the React Native. Create a title using the **Text** and the **View** component and adds styles in Stylesheet.	**ESR:** At the top of the screen. **ESR:** We'll require the Text and View components.

```
import React, { Component } from "react";
import {
    View,
    Text,
    StyleSheet,
} from "react-native";

export default class HomeScreen extends Component {

    render() {
        return (
            <View style={styles.container}>
                <Text style={styles.titleText}>ISS Tracker App</Text>
            </View>
        );
    }
}
```

Where do we see the Title?	**ESR:** We see the title hidden behind the status bar.
Yes!! The title appears there by default, do you know why this happens?	**ESR:** Varied!
This happens because the **View** starts to read the height from the top most corner which also includes the status bar of the device.	
To avoid this, React Native contains a component called the **SafeAreaView** using which we can exclude the status bar on each device.	
We'll also need to know the height of the status bar on each of the android devices, so we'll first import the status bar along with **SafeAreaView** from React Native. But the problem is that we have two OS (operating systems) Android and the Apple OS for phones, and the status bar only works for the android devices, so we'll need	

to check which operating system the user is using and then decide the height of the status bar.

Import the **SafeAreaView** from React Native and use it inside the View component.

We'll assign the height using the **marginTop** property in the stylesheet.

Now to check if the OS is **android** and then assign the height using the **StatusBar** component.

```jsx
import React, { Component } from "react";
import {
    View,
    Text,
    StyleSheet,
    SafeAreaView,
    TouchableOpacity,
    Platform,
    StatusBar,

} from "react-native";

export default class HomeScreen extends Component {


    render() {
        return (
            <View style={styles.container}>
                <SafeAreaView style={styles.droidSafeArea} />

                    <Text style={styles.titleText}>ISS Tracker App</Text>

            </View>
        );
    }
}

const styles = StyleSheet.create({
    container: {
        flex: 1
    },
    droidSafeArea: {
        marginTop: Platform.OS === "android" ? StatusBar.currentHeight : 0
    },
});
```

Now we can see that the Title has shifted below the status bar of the mobile device.

But is still that the place where we want the title?	**ESR:** No, we want it at the center.
Yes and to do so we'll just add the title inside another **View** and give styling such as **flex** to **0.25 justifyContent** to center, and **alignItems** to center.	
As this is our title we'll call it **titleBar**.	
Add the **Text** for the title inside the **View** component.	
Create a **titleBar** in the stylesheet and add the styles to it as shown in the below code snippets:	

```
export default class HomeScreen extends Component {

    render() {
        return (
            <View style={styles.container}>
                <SafeAreaView style={styles.droidSafeArea} />

                    <View style={styles.titleBar}>
                        <Text style={styles.titleText}>ISS Tracker App</Text>
                    </View>

            </View>
        );
    }
}
```

And its corresponding styles would be:

```
const styles = StyleSheet.create({
    container: {
        flex: 1
    },
    droidSafeArea: {
        marginTop: Platform.OS === "android" ? StatusBar.currentHeight : 0
    },
    titleBar: {
        flex: 0.15,
        justifyContent: "center",
        alignItems: "center"
    },
})
```

What is the next thing that we want on our screen?	**ESR:** We want to have 3 buttons for the 3 different screens.
Yes! Can you tell me how can we create them?	**ESR:** We can use the
We import the **TouchableOpacity** from the React Native	

and using the **Text** component we add the name on the button as "**ISS Location**".	**Touchableopacity** to create the buttons.
Similarly, we'll create another button for the **Meteors screen**.	
<The teacher codes to create another button for Meteors>	

```jsx
import React, { Component } from "react";
import {
    View,
    Text,
    StyleSheet,
    SafeAreaView,
    TouchableOpacity,
    Platform,
    StatusBar,
} from "react-native";
```

```jsx
export default class HomeScreen extends Component {

    render() {
        return (
            <View style={styles.container}>
                <SafeAreaView style={styles.droidSafeArea} />

                <View style={styles.titleBar}>
                    <Text style={styles.titleText}>ISS Tracker App</Text>
                </View>

                <TouchableOpacity
                >
                    <Text>ISS Location</Text>

                </TouchableOpacity>


            </View>
        );
    }
}
```

And to create similar button for **Meteors**, we do:

```
render() {
    return (
        <View style={styles.container}>
            <SafeAreaView style={styles.droidSafeArea} />

            <View style={styles.titleBar}>
                <Text style={styles.titleText}>ISS Tracker App</Text>
            </View>

            <TouchableOpacity>
                <Text>Iss Location</Text>
            </TouchableOpacity>

            <TouchableOpacity>
                <Text>Meteors</Text>
            </TouchableOpacity>

        </View>
    )
}
}
```

We now have the buttons, but they don't look much like buttons, do they?

What can we do to make them look more like buttons?

As these buttons are going to be taking us to the different screens, let's name them **routeCards** for styling purposes.

In the stylesheet, we'll add the variable called **routeCards**. In the route card styles we'll:-

ESR:
No, they don't.

ESR:
We can add some styling to make it look like the buttons.

- First, add some flex to the button using the **flex** property. It could be anything generally, but since we have 3 screens to route along with our title, we will take a flex of **0.25.**
- Using the **justifyContent** and **alignItem** property we'll get the content aligned to the center of our screen.
- Add margin to the buttons using **marginLeft**, **marginRight**, and **marginTop** properties.
- Create circular borders using **borderRadius** and background color.

<The teacher codes to add the styles to the buttons>

```jsx
render() {
    return (
        <View style={styles.container}>
            <SafeAreaView style={styles.droidSafeArea} />

            <View style={styles.titleBar}>
                <Text style={styles.titleText}>ISS Tracker App</Text>
            </View>

            <TouchableOpacity style={styles.routeCard}>
                <Text>Iss Location</Text>
            </TouchableOpacity>

            <TouchableOpacity style={styles.routeCard}>
                <Text>Meteors</Text>
            </TouchableOpacity>

        </View>
    )
}
}
```

```
routeCard: {
    flex: 0.25,
    marginLeft: 50,
    marginRight: 50,
    marginTop: 50,
    borderRadius: 30,
    backgroundColor: 'white'
},
```

You may notice that we have added the **backgroundColor** as "**white**"!

Well, we've done that because we will be adding a background image to this screen later, and white would really look good on that! For now, we can add an extra styling in "**routeCard**" for the boxes to be visible to us -

< *The teacher can also add **borderWidth: 2** in **routeCards** to show students the borders around buttons, which will be removed later*>

The buttons look good, but the text inside the buttons is not proper yet, right?

Let's add some styles to it as well.

We called the styles on the button as the **routeCards**, similarly, we'll call the styles for text inside the button as **routeText**.

To style the text we can add some **fontWeight**, **fontSize**, and **color** to the text.

-

We now add styling to the text in the buttons.

Let's add some styling to the Title of the app as well.
Let's call the styling on the title as **titleText**.

To style the title we'll add some **fontWeight**, **fontSize**, and **color** to the text as well.

```
titleText: {
    fontSize: 40,
    fontWeight: "bold",
    color: "white"
},
routeText: {
    fontSize: 35,
    fontWeight: "bold",
    color: "black",
    marginTop: 75,
    paddingLeft: 30
},
```

```
render() {
    return (
        <View style={styles.container}>
            <SafeAreaView style={styles.droidSafeArea} />

            <View style={styles.titleBar}>
                <Text style={styles.titleText}>ISS Tracker App</Text>
            </View>

            <TouchableOpacity style={styles.routeCard}>
                <Text style={styles.routeText}>Iss Location</Text>
            </TouchableOpacity>

            <TouchableOpacity style={styles.routeCard}>
                <Text style={styles.routeText}>Meteors</Text>
            </TouchableOpacity>

        </View>
    )
}
}
```

Output:

ISS Tracker App

Iss Location

Meteors
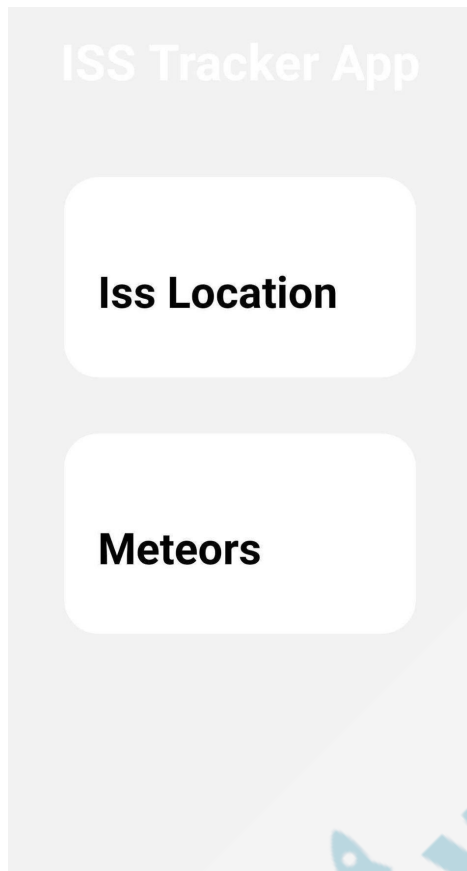
|	
We have buttons ready but what's missing?	**ESR:** we can't navigate to the different screens when we press the buttons. We can also add some images to the background or the buttons to make it look visually appealing.
Would you like to add navigation and images to make the screen look much more awesome? Awesome! Let's get you started then.	**ESR**: Yes.

Teacher Stops Screen Share	
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.**
- **The teacher gets into Fullscreen.**

ACTIVITY
- **Add images to the different elements of the screen.**
- **Add navigation on the button press.**

Teacher starts slideshow **for slide 12 & 13**

Refer to speaker notes and follow the instructions on each slide.

Teacher Action	Student Action
<The teacher guides the student to update their code same as covered during the Teacher Activity>	*<The student writes code to Add title and buttons to the screen with styling>*
First, we can start by adding images to the background and the buttons. *<The teacher guides the student to download the images from Student Activity 1 and adds them as assets to the **project** folder>*	*<The student downloads the images from the Student Activity 1 link and adds them as assets to the **project** folder>*
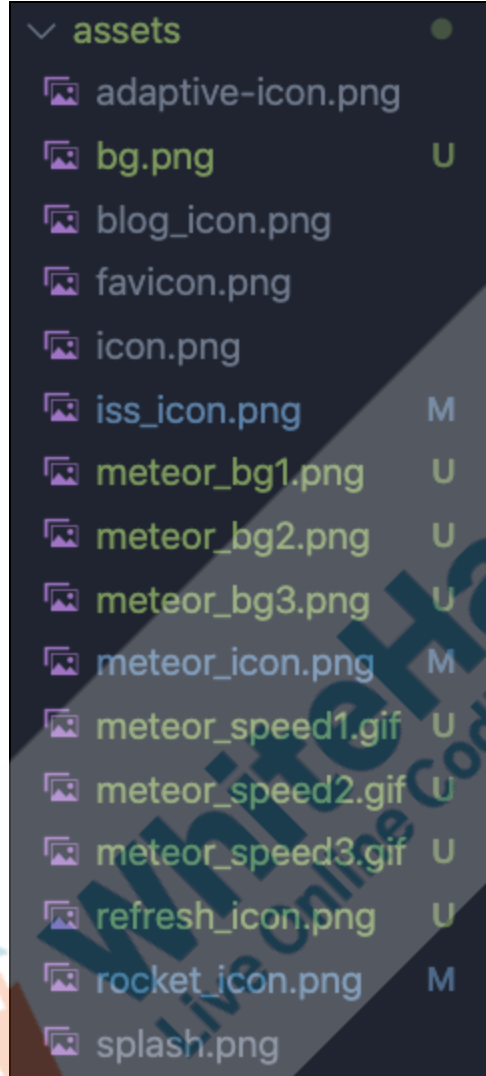
React Native has a component called **ImageBackground** which will help us to add images to our background. Import the **ImageBackground** component first.	*<The student imports the **ImageBackground** from the React Native>*

```
import {
    View,
    Text,
    StyleSheet,
    SafeAreaView,
    TouchableOpacity,
    Platform,
    StatusBar,
    ImageBackground
} from "react-native";
```

	The **ImageBackground** component has a **source** property where we'll provide the location of the assets. Set the location of the background image in the **source** property of the **ImageBackground** component.	*<The student sets the location of the background image in the source property of the **ImageBackground** component>*

```
render() {
    return (
        <View style={styles.container}>
            <SafeAreaView style={styles.droidSafeArea} />
            <ImageBackground source={require('../assets/bg.png')} style={styles.backgroundImage}>
                <View style={styles.titleBar}>
                    <Text style={styles.titleText}>ISS Tracker App</Text>
                </View>

                <TouchableOpacity style={styles.routeCard}>
                    <Text style={styles.routeText}>Iss Location</Text>
                </TouchableOpacity>

                <TouchableOpacity style={styles.routeCard}>
                    <Text style={styles.routeText}>Meteors</Text>
                </TouchableOpacity>
            </ImageBackground>
        </View>
    )
}
}
```

```
const styles = StyleSheet.create({
    container: {
        flex: 1
    },
    droidSafeArea: {
        marginTop: Platform.OS === "android" ? StatusBar.currentHeight : 0
    },
    backgroundImage: {
        flex: 1,
        resizeMode: 'cover',
    },
    routeCard: {
        flex: 0.2,
        justifyContent: "center",
        alignItems: "center",
        marginLeft: 10,
        marginRight: 10,
        marginTop: 5,
        borderRadius: 10,
        backgroundColor: 'rgba(52, 52, 52, 0.5)'
    },
```

Similarly, we'll use the **Image** component to add the images to the buttons and some text to make it look good.

We'll also add the relevant styling for what we're trying to achieve.

Now, import the **Image** component from React Native and add images to the buttons.

<The student imports the Image component from React Native and adds images to the buttons>

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

And then add the images to the buttons as shown in the following code snippet:

```
<TouchableOpacity style={styles.routeCard}>
    <Text style={styles.routeText}>ISS Location</Text>
    <Text style={styles.knowMore}>{"Know More --->"}</Text>
    <Text style={styles.bgDigit}>1</Text>
    <Image source={require("../assets/iss_icon.png")} style={styles.iconImage}></Image>
</TouchableOpacity>

<TouchableOpacity style={styles.routeCard}>
    <Text style={styles.routeText}>Meteors</Text>
    <Text style={styles.knowMore}>{"Know More --->"}</Text>
    <Text style={styles.bgDigit}>2</Text>
    <Image source={require("../assets/meteor_icon.png")} style={styles.iconImage}></Image>
</TouchableOpacity>
```
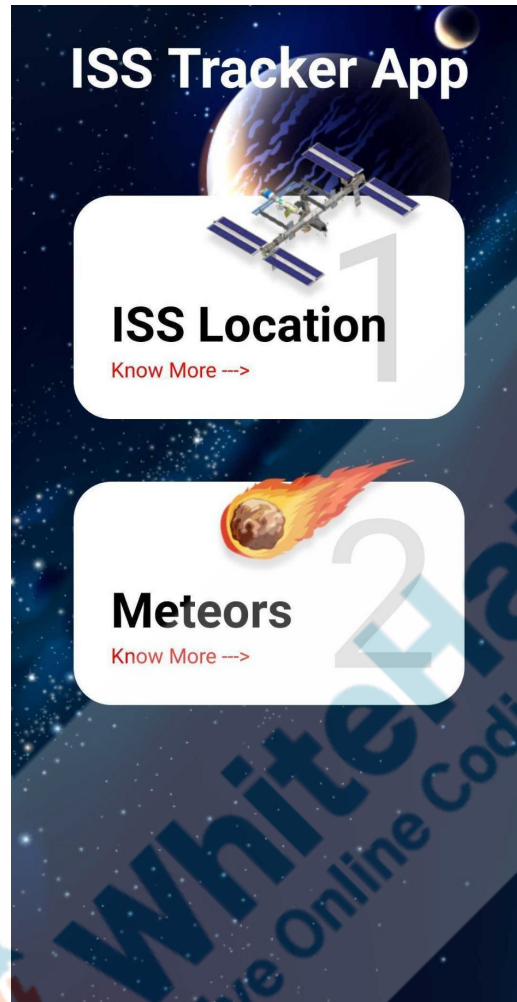
```
knowMore: {
    paddingLeft: 30,
    color: "red",
    fontSize: 15
},
bgDigit: {
    position: "absolute",
    color: "rgba(183, 183, 183, 0.5)",
    fontSize: 150,
    right: 20,
    bottom: -15,
    zIndex: -1
},
iconImage: {
    position: "absolute",
    height: 200,
    width: 200,
    resizeMode: "contain",
    right: 20,
    top: -80
}
```

Now we just have to navigate to the screens when the buttons are pressed.

The **TouchableOpacity** has an **onPress** property using which we can navigate to the screens.

In the **onPress** property, we have to pass **this.props.navigation.navigate("name of screen to navigate").**

Pass the **this.props.navigation.navigate("IssLocation")** in the **onPress** property of the **TouchableOpacity**.	*\<The student passes **this.props.navigation.navigate("IssLocation")** in the **onPress** property of the **Touchableopacity**>*

```
<TouchableOpacity style={styles.routeCard} onPress={() =>
    this.props.navigation.navigate("IssLocation")
}>
    <Text style={styles.routeText}>ISS Location</Text>
    <Text style={styles.knowMore}>{"Know More --->"}</Text>
    <Text style={styles.bgDigit}>1</Text>
    <Image source={require("../assets/iss_icon.png")} style={styles.iconImage}></Image>
</TouchableOpacity>
```

	We'll do the same for the other buttons. Repeat the same for the **Meteors** screen.	*\<The student passes **this.props.navigation.navigate("Meteors")** in the **onPress** property of the **Touchableopacity** for **Meteor** button>*

```
<TouchableOpacity style={styles.routeCard} onPress={() =>
    this.props.navigation.navigate("Meteors")
}>
    <Text style={styles.routeText}>Meteors</Text>
    <Text style={styles.knowMore}>{"Know More --->"}</Text>
    <Text style={styles.bgDigit}>2</Text>
    <Image source={require("../assets/meteor_icon.png")} style={styles.iconImage}></Image>
</TouchableOpacity>
```

Now let's run and check the output.	*\<The student runs the code and checks the output>*

With this, our Home Screen is completed with all the navigation added!

WRAP-UP SESSION - 5 mins

Teacher starts slideshow **for slide 14 to 24.**

Refer to speaker notes and follow the instructions on each slide.

Teacher Action	Student Action

Run the presentation from slide 14 to slide 24.
 	
Following are the wrap-up session deliverables:
 	
  ● **Explain the facts and trivias.**
  ● **Next class challenge.**
  ● **Project for the day.**
  ● **Additional Activity**	

QnA Session

Question	Answers
What does the following piece of code do? ```\n },\n androidSafeArea: {\n marginTop: Platform.OS === "android" ? StatusBar.currentHeight : 0\n },\n});\n``` A. The status bar only works for the IOS devices, so we'll need to check which operating system the user is using and then decide the height of the status bar. B. The status bar only works for android devices, so we'll need to check which operating system the user is using and then decide the height of the status bar. C. It is providing marginTop to the status bar irrespective of the platform D. None of the above	**B**

By default, the title hides behind the status bar, to avoid this what could be used?
  A. andriodSafeArea
  B. SafeAreaView
  C. StatusBar
	B

D. platform	

To create circular buttons which CSS property is used?
   A. borderRadius
   B. borderWidth
   C. borderThickness
   D. borderCircle
	<u>**A**</u>

<div align="center">

End the quiz panel

</div>

FEEDBACK
- **Appreciate the student for their efforts in the class.**
- **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**

Teacher Action	Student Action
You get Hats Off for your excellent work! Awesome! In the next class, we will be working on designing the ISS location screen.	*Make sure you have given at least 2 Hats Off during the class for:* Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
* This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class. **Project Overview** **Stellar Stage-2** **Goal of the Project:**	**Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.**

In Class 77, we have designed the home screen by adding the buttons for different screens and adding the background image to the screen. You will have to add a background image, and style different components of the Home Screen of the Stellar App created in the last class.

*This is a continuation project of Project-76, make sure to complete that one before attempting this one.

Story:

Jeff liked the layout of the App created by you in the last project. However, he feels it can be made more attractive by adding styles and relevant images to it.
Can you help him?

I am very excited to see your project solution and I know you will do really well.

Bye Bye!

Teacher ends slideshow	
Teacher Clicks ✖ End Class	
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?
  ○ Describe what happened.
  ○ The code I wrote.
	The student uses the Markdown editor to write their reflections in a reflection journal.

● 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?	

Links:

Activity	Activity Name	Links
Teacher Activity 1	Previous class code	https://github.com/React-Native-Frontier/PRO-C76-ISS-Tracker
Teacher Activity 2	Reference code	https://github.com/React-Native-Frontier/PRO-C77-ISS-Tracker
Teacher Activity 3	Teacher Aid	https://drive.google.com/file/d/1WA1BQff4dmgv5BInU3f_imk4vlpvAyMa/view?usp=sharing
Student Activity 1	Assets	https://curriculum.whitehatjr.com/PRO+Asset/ISS-assets.zip
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC-PRO-V3-C77-withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/0c1bd2d2-58c5-4a67-b536-0879957a9d2a.pdf
Project Solution	Stellar Stage-2	https://github.com/pro-whitehatjr/Stellar-Stage-2