Topic	**Blueprints and Stack Navigator**	
Class Description	**The student discusses the blueprints for the new app. Students learn about the Stack Navigator**	
Class	**C76**	
Class time	**45 mins**	
Goal	● Discuss and create the blueprints for the ISS Tracker app. ● Create multiple Screens and add them to the Stack Navigator.	
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**

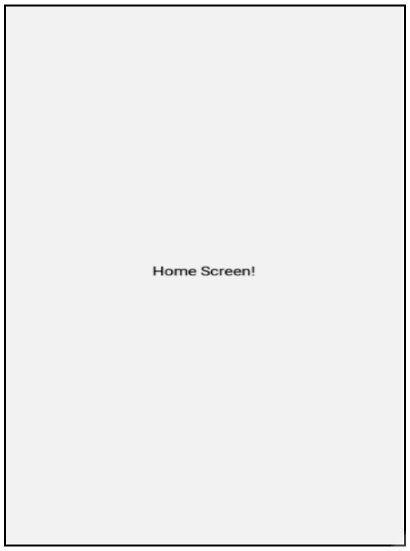
● WARM-UP SESSION - 10 mins

The teacher starts slideshow from slides 1 to 8

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

Teacher Action	**Student Action**

Hey <student's 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 2** **Following are the WARM-UP session deliverables:** ● **Greet the student.** ● **Revision of previous class activities.** ● **Quizzes**	**ESR**: Hi, thanks, Yes I am excited about it! Click on the slide show tab and present the slides
QnA Session	
Question	
Select the correct block of code which helps in passing Homescreen in Stack.Navigator. A. `Stack.Screen name:"Home" component:{HomeScreen} />` B. `<Stack.Screen name="Home" component={HomeScreen} />` C. `<Stack.Screen component="Home" name={HomeScreen} />` D. `<Stack.Screen name="Home" component="HomeScreen" />`	**B**
Select the correct block of code for importing the package used in creating Stack Navigator.	**A**

Home Screen! A. `import { createStackNavigator } from '@react-navigation/stack';` B. `import { createStackNavigation } from '@react-navigation/stack';` C. `import { StackNavigator } from '@react-navigation/stack';` D. `import { createStack } from '@react-navigation/stack';`	

Continue the warm-up session

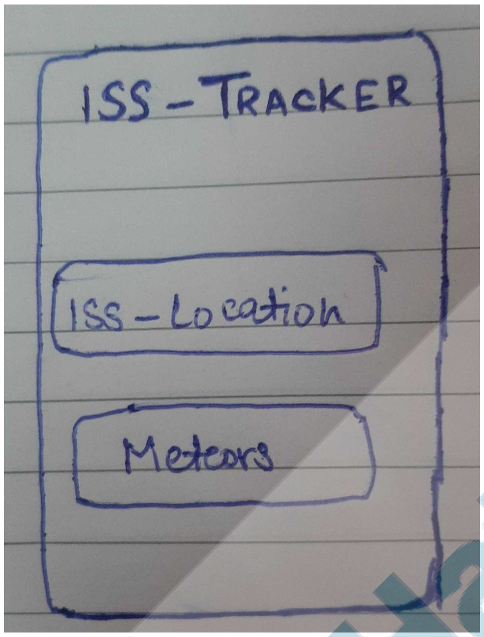
Teacher Action	Student Action
Run the presentation from slide 3 to slide 8 to set the problem statement. **Following are the warm-up session deliverables:** ● **International Space Station (ISS)** ● **about meteors**	Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students

WhiteHat Jr
Live online Coding for Kids

Teacher ends slideshow
TEACHER-LED ACTIVITY 1 (10 - 15 mins)
• **Teacher Initiates Screen Share**
ACTIVITY • **Discuss the flow of the App and create a blueprint of it.** • **Create an App Stack Navigator**

Teacher Action	**Student Action**
Hi Do you remember what we learned in the last class?	*<The student revises the concept learned in the previous classes.>*
Have you seen a meteor on the television or in the newspaper? Won't it be interesting to know when the next meteor will pass from near the earth and will be visible to us? Can you think of a way by which we can do that? Let's create an app which will give us the information of all the meteors which are going to pass near the earth,	**ESR:** Varied! **ESR:** YES!! **ESR:** Varied
Sounds exciting? Let's get started then.	**ESR:** Yes
So first we'll need to think about how our app will look and what information it will show.	

We'll think and plan about how the app will look. Talking about the things that we want to show in our app, we can display - ● The live location of the **ISS** (**International Space Station**) ● Meteors, when they visit earth and details about them.	
Based on the things that we want to display in our app, can you guess how many screens it should have? Alright, let's get a pen and paper and design the UI for the app. *<The teacher asks the student to get the pen and paper and design the UI for the app>*	**ESR:** Our app will have 3 screens. 1. The main screen, that will have buttons to navigate to different screens. 2. ISS tracker screen to display the live location of ISS. 3. Screen for meteors and information about them.
	<The student gets the pen and paper and designs his/her own UI for the app>



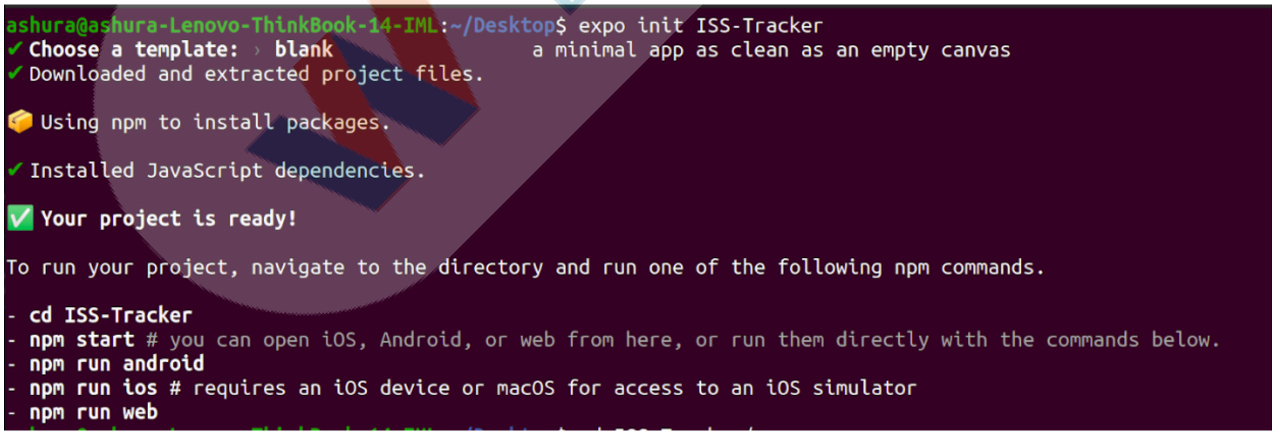
Now let's set up a project called the ISS-Tracker.

How do we set up a new project?

We'll do it using the command "***expo init ISS-Tracker***"

<The teacher codes to set up the project>

ESR:
varied



```
ashura@ashura-Lenovo-ThinkBook-14-IML:~/Desktop$ expo init ISS-Tracker
✔ Choose a template: › blank                 a minimal app as clean as an empty canvas
✔ Downloaded and extracted project files.

📦 Using npm to install packages.

✔ Installed JavaScript dependencies.

✅ Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

- cd ISS-Tracker
- npm start # you can open iOS, Android, or web from here, or run them directly with the commands below.
- npm run android
- npm run ios # requires an iOS device or macOS for access to an iOS simulator
- npm run web
```

Ok, so now that we have created an outline for how our app will look, let's start creating the screens for our app.

Let's first create a folder called **screens** which will contain all our screens. *<The teacher opens the folder in VS Code and creates the folder called as **screens**>* The first screen will be the **Home** screen. For now, we'll just be creating simple screens with sample text in them. The idea is that we want to learn how Stack Navigation really works in React Native!	
To create the Home screen we'll:- Create a file called **Home.js** in the **screens** folder. Inside this file, use the following code to import **React** and **Component**: **import React, { Component } from 'react';** Use the following lines of code to import **Text** and **View** from React Native: **import { Text, View } from 'react-native';** We'll create a class called **HomeScreen** which extends the home screen component and export it to be used on different screens. Use the following lines of code for it: **export default class HomeScreen extends Component { }**	The student observes and asks questions.

Inside the **HomeScreen** component, we'll have the **render()** function.

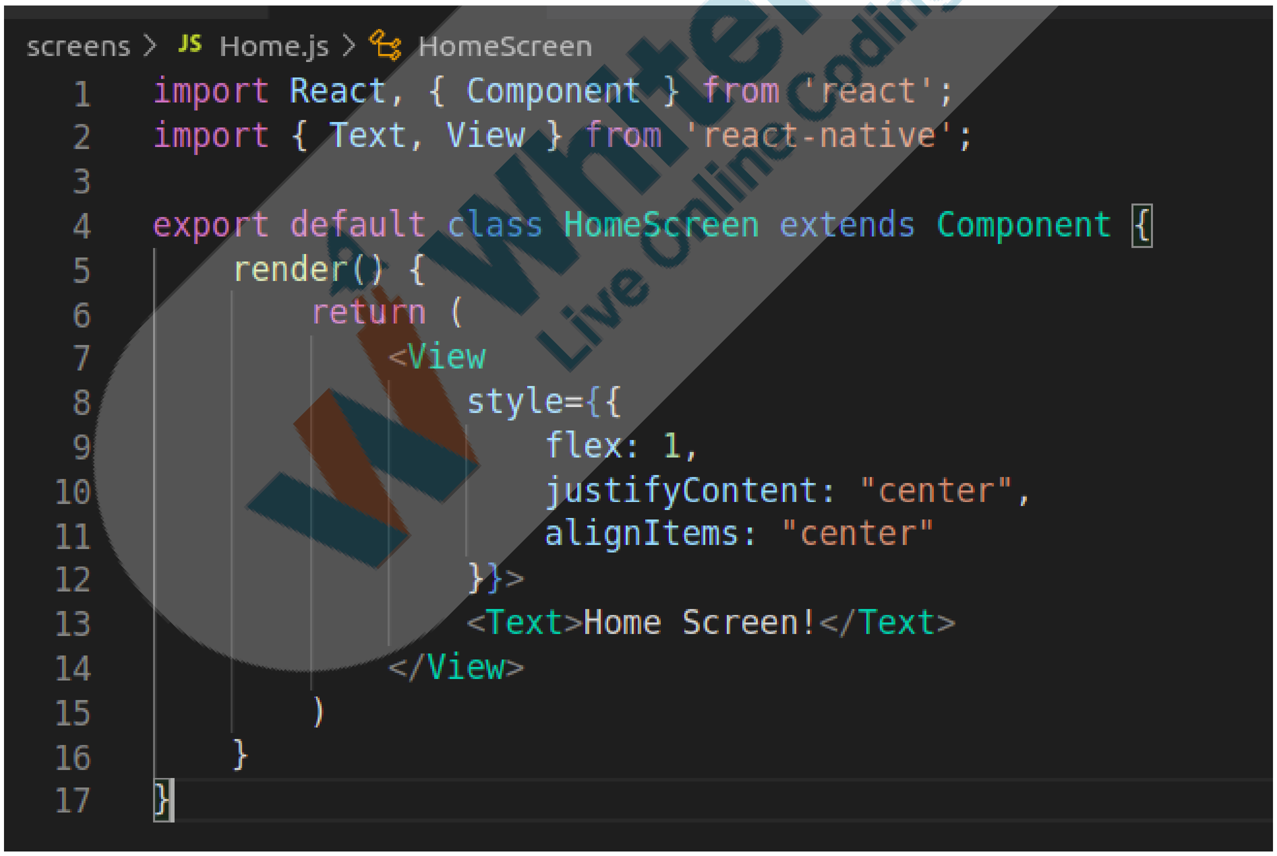
Inside the **render()** function we'll have the **return()** function.

Inside the **return()** function using the **View** and the **Text** component we'll display **Home Screen** text.

We'll also add the styles to the **View** component using the inline styles.

<The teacher codes to create the Home Screen>

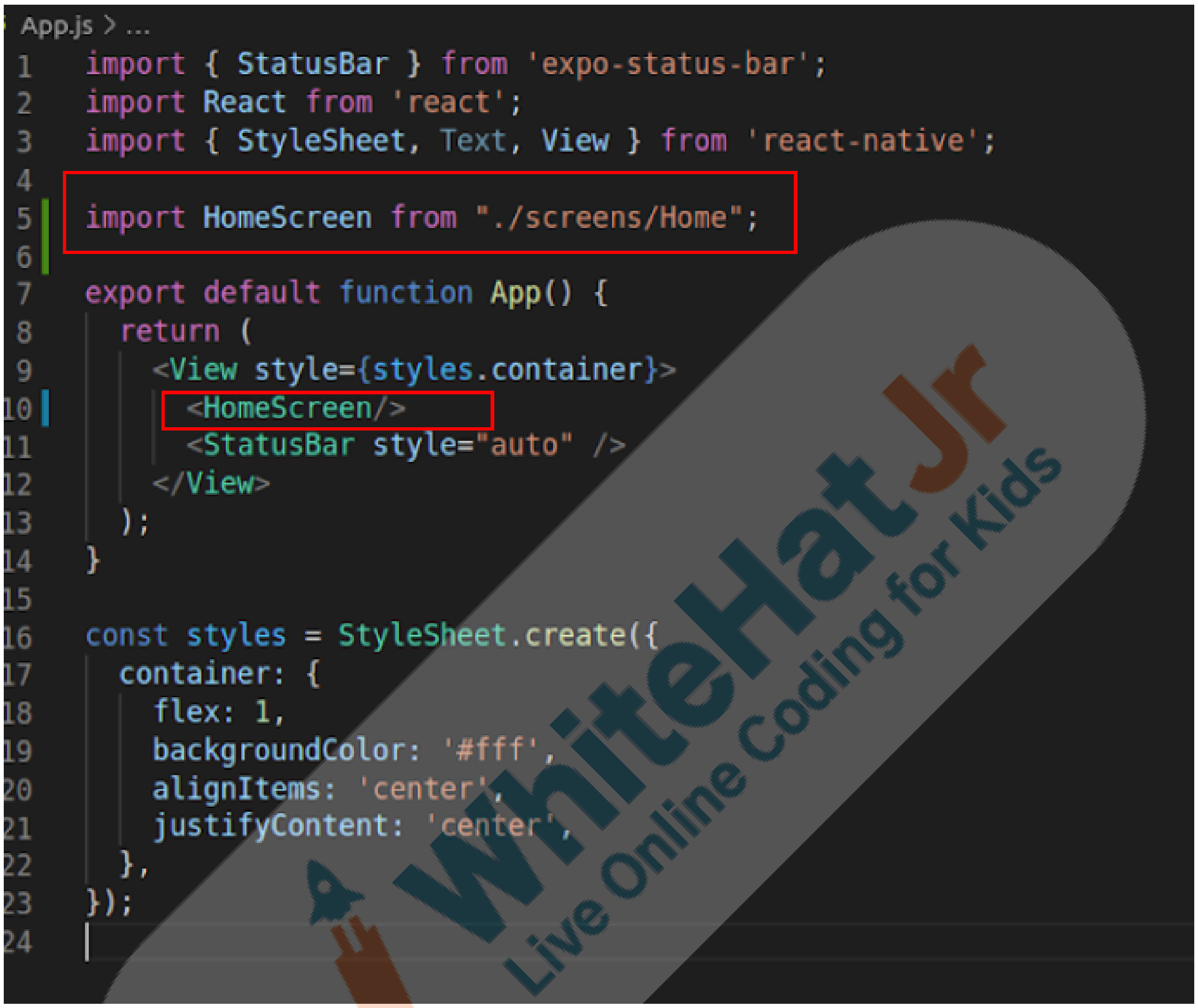
<The code shall look something like below for Home.js>



screens > Home.js > HomeScreen

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

export default class HomeScreen extends Component {
    render() {
        return (
            <View
                style={{
                    flex: 1,
                    justifyContent: "center",
                    alignItems: "center"
                }}>
                <Text>Home Screen!</Text>
            </View>
        )
    }
}
```

Now we have created the file so let's see how the content of our Screen looks. To do so we'll first need to import the screen in the **App.js** file and pass it in the **render()** function. *<The teacher codes to import the **HomeScreen** from screens folder and pass it in the **return()** function>* *<The teacher runs the app using the **'expo start'** command in the terminal. Opens the expo app and scans the QR code>*	

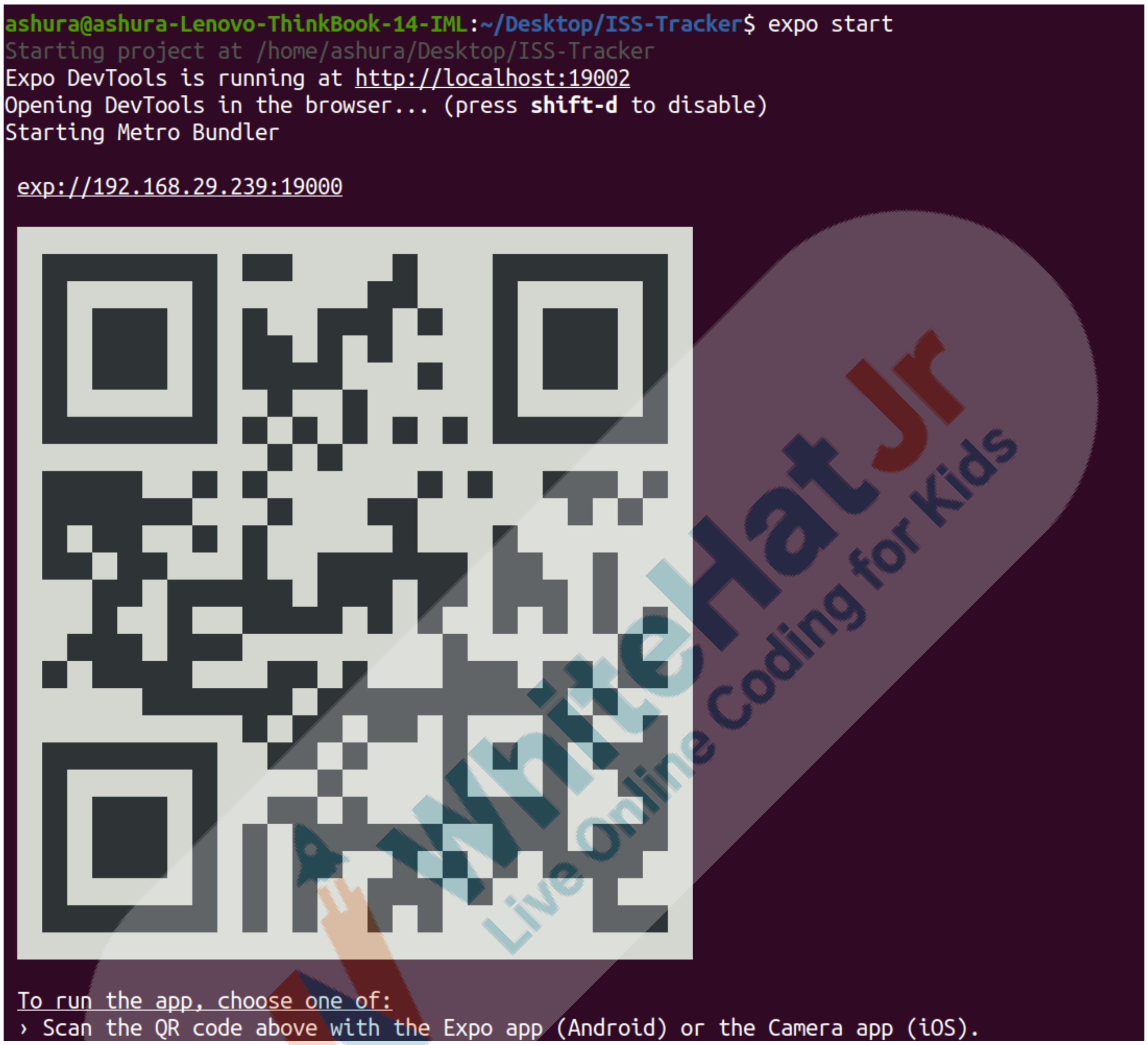


App.js > ...

```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

import HomeScreen from "./screens/Home";

export default function App() {
  return (
    <View style={styles.container}>
      <HomeScreen/>
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

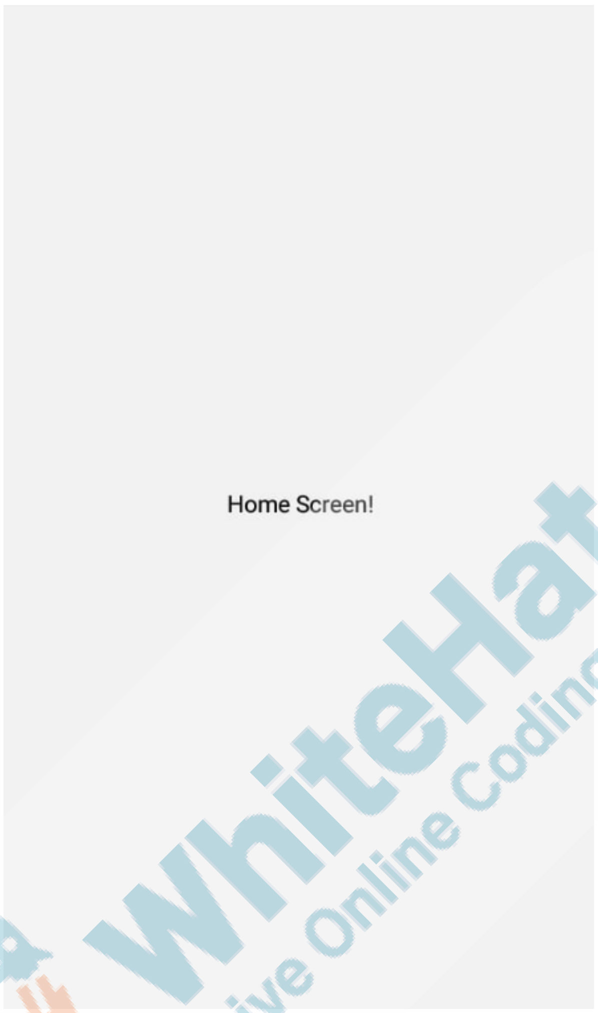
Here, in our **App.js**, we are rendering the **HomeScreen** component that we just created.

We can now see the output of our app by running the **expo start** command on the terminal/command prompt.



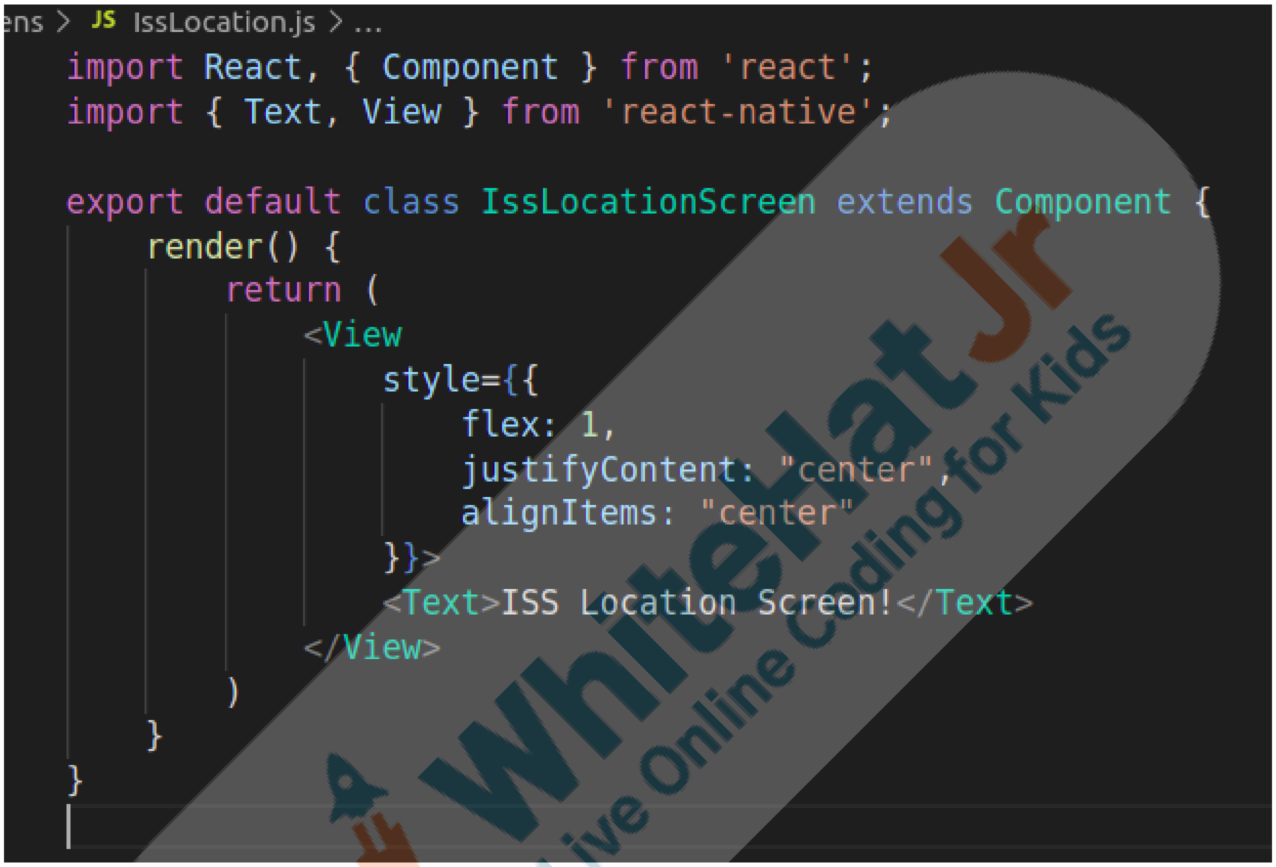
```
ashura@ashura-Lenovo-ThinkBook-14-IML:~/Desktop/ISS-Tracker$ expo start
Starting project at /home/ashura/Desktop/ISS-Tracker
Expo DevTools is running at http://localhost:19002
Opening DevTools in the browser... (press shift-d to disable)
Starting Metro Bundler

 exp://192.168.29.239:19000

 To run the app, choose one of:
 › Scan the QR code above with the Expo app (Android) or the Camera app (iOS).
```

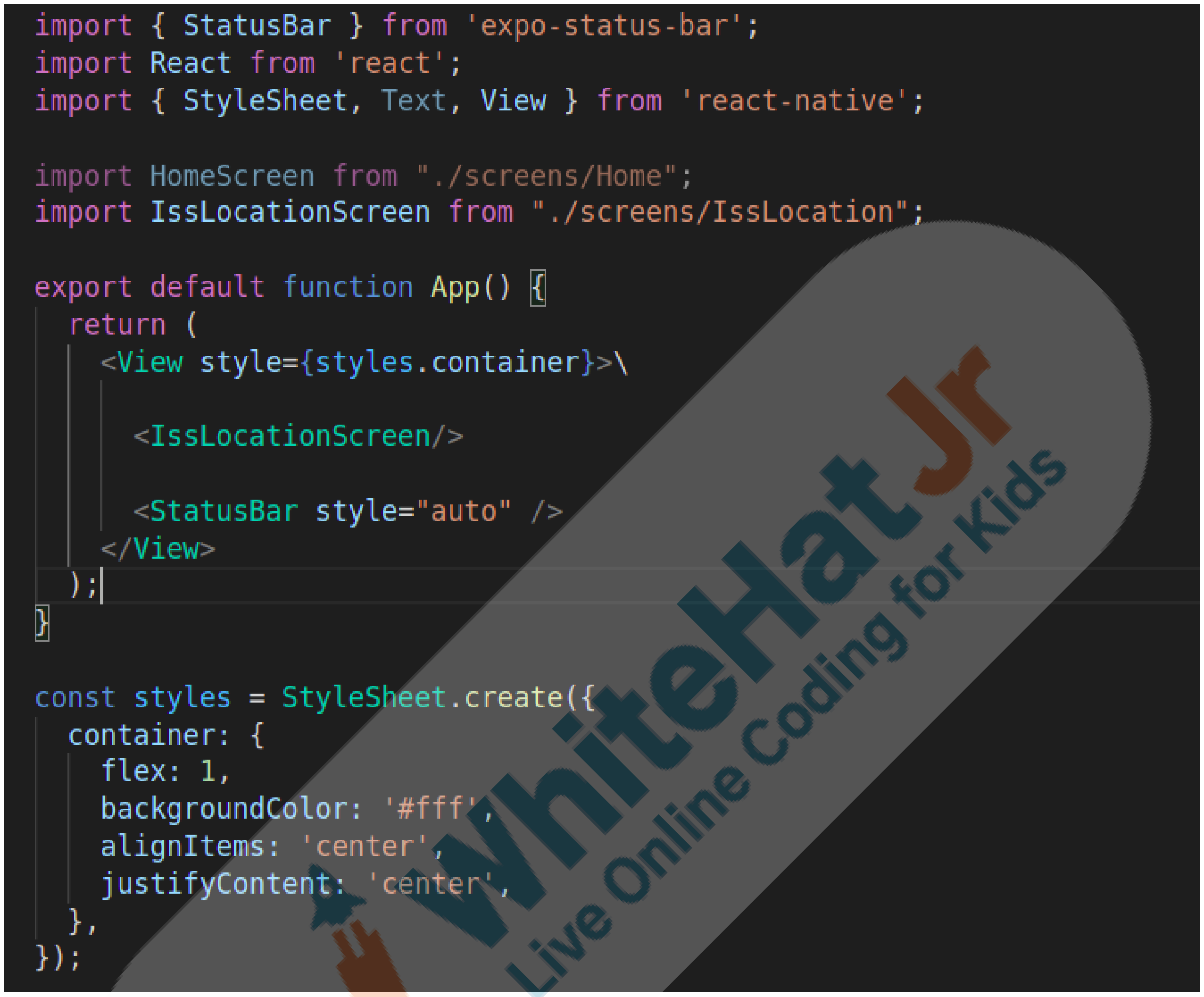
We can see the output for our **HomeScreen** component:



Similarly, we'll code to create the **IssLocationScreen** and **MeteorScreen**. *<The teacher codes to create **IssLocationScreen**>* *<The teacher imports **IssLocation** screen from screens folder into **App.js** screen and passes the screen as a component to the **render()** function>* ***Note**: The teacher can copy the code for all the screens as it is the same code, just change the name of the classes.*	*The student observes and asks questions*

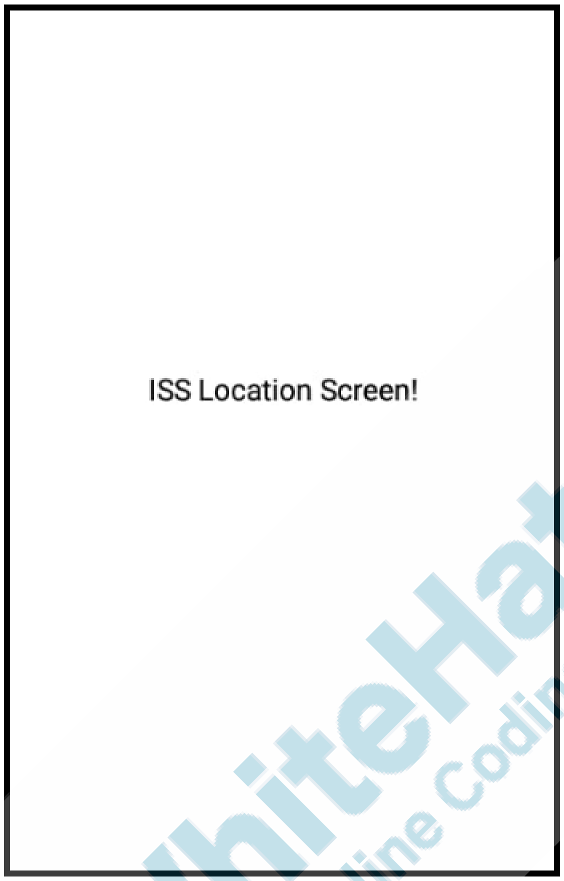
Also Note*: Teacher can test different screen by changing the component in the App.js*	

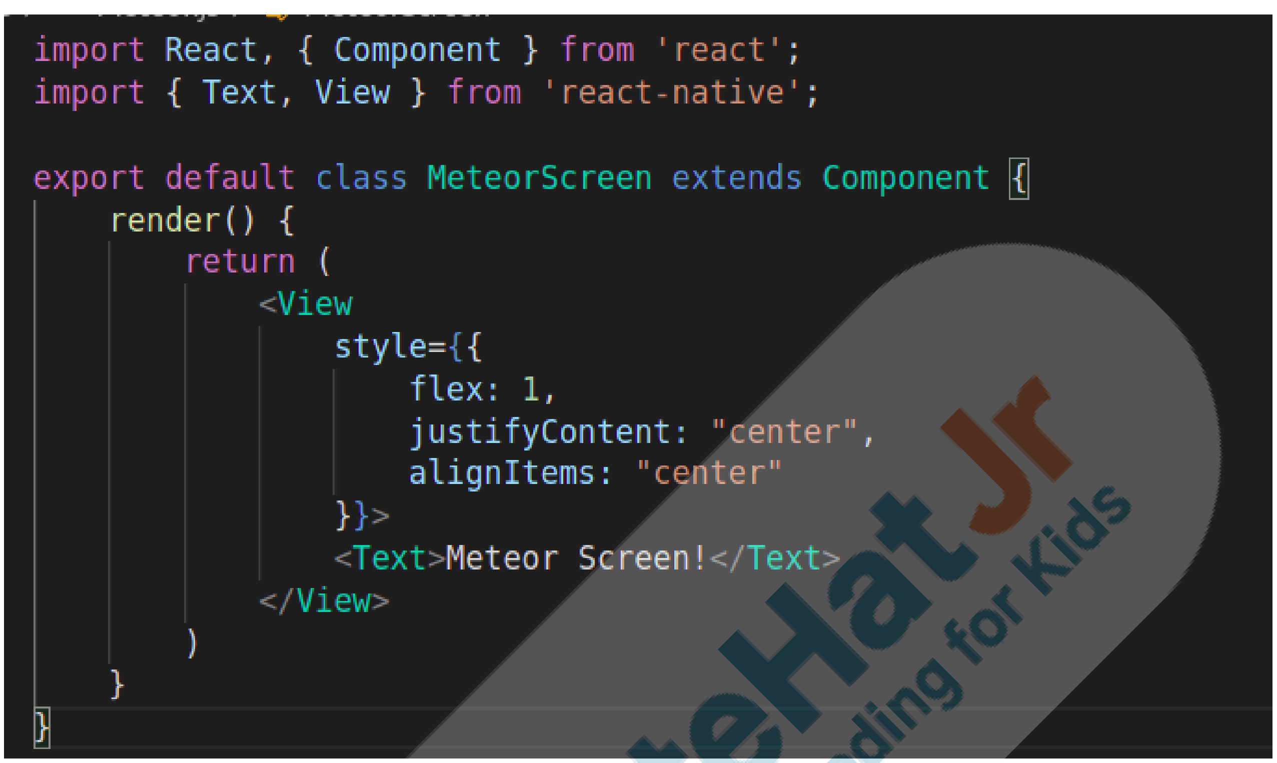
 ens > JS IssLocation.js > ...
`import React, { Component } from 'react';`
`import { Text, View } from 'react-native';`
 	
`export default class IssLocationScreen extends Component {`
`    render() {`
`        return (`
`            <View`
`                style={{`
`                    flex: 1,`
`                    justifyContent: "center",`
`                    alignItems: "center"`
`                }}>`
`                <Text>ISS Location Screen!</Text>`
`            </View>`
`        )`
`    }`
`}`
 	
Code for **ISSLocationScreen** component:	



```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

import HomeScreen from "./screens/Home";
import IssLocationScreen from "./screens/IssLocation";

export default function App() {
  return (
    <View style={styles.container}>\

      <IssLocationScreen/>

      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

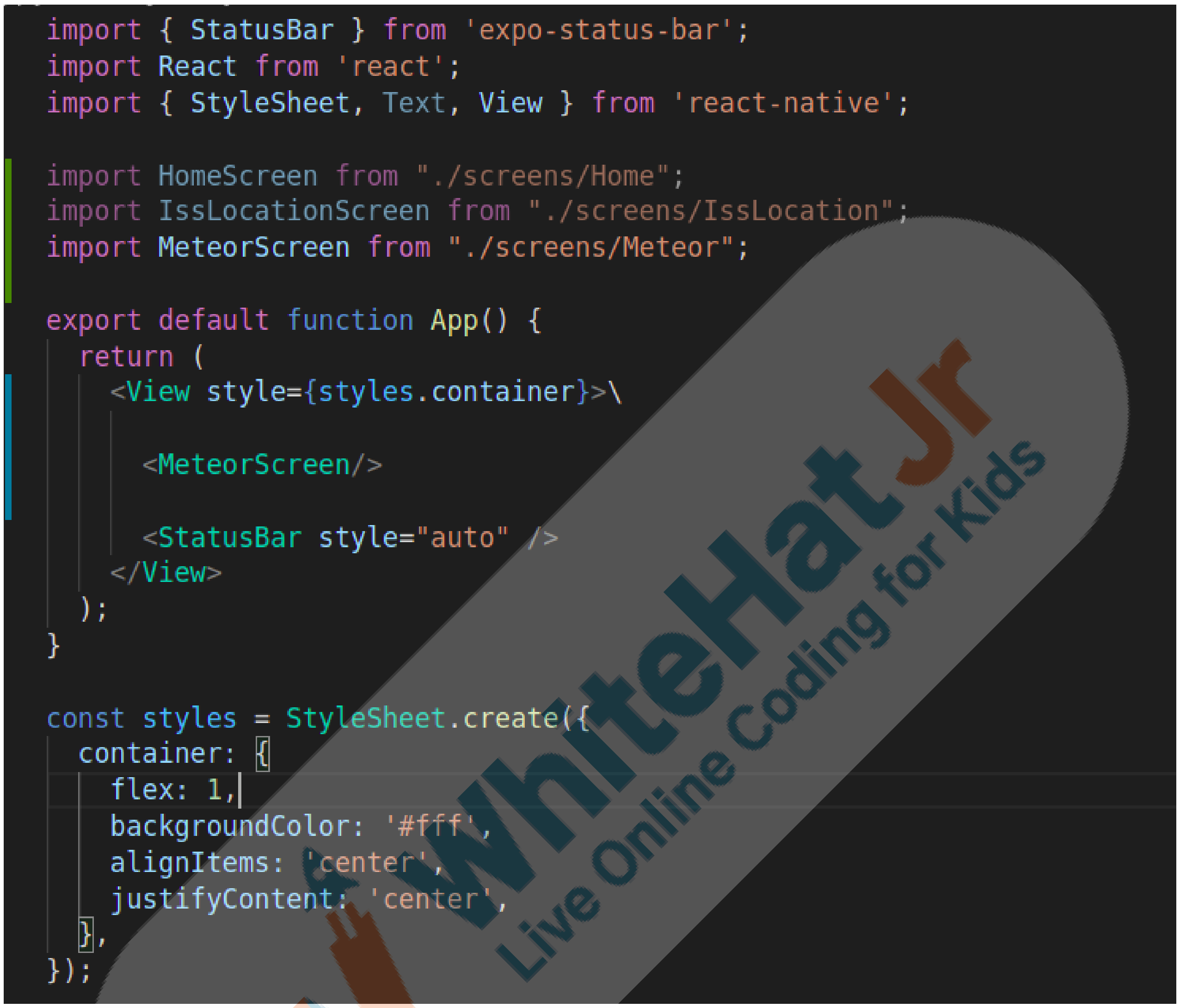
Using the **IssLocationScreen** in **App.js** to check its output:

 ISS Location Screen!	
<The teacher codes to create Meteor screen> *<The teacher imports Meteor screen from **screens** folder into **app.js** screen and passes the screen as a component to the **render()** function>*	*The student observes and asks questions*



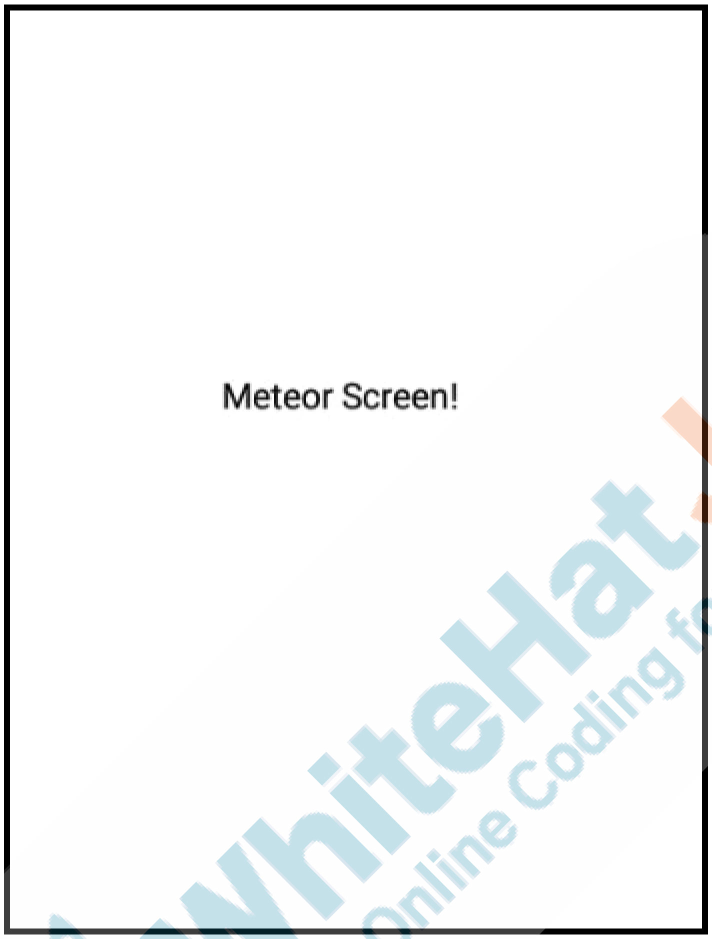
```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

export default class MeteorScreen extends Component {
    render() {
        return (
            <View
                style={{
                    flex: 1,
                    justifyContent: "center",
                    alignItems: "center"
                }}>
                <Text>Meteor Screen!</Text>
            </View>
        )
    }
}
```

Code for **MeteorScreen** component:



```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

import HomeScreen from "./screens/Home";
import IssLocationScreen from "./screens/IssLocation";
import MeteorScreen from "./screens/Meteor";

export default function App() {
  return (
    <View style={styles.container}>\

      <MeteorScreen/>

      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

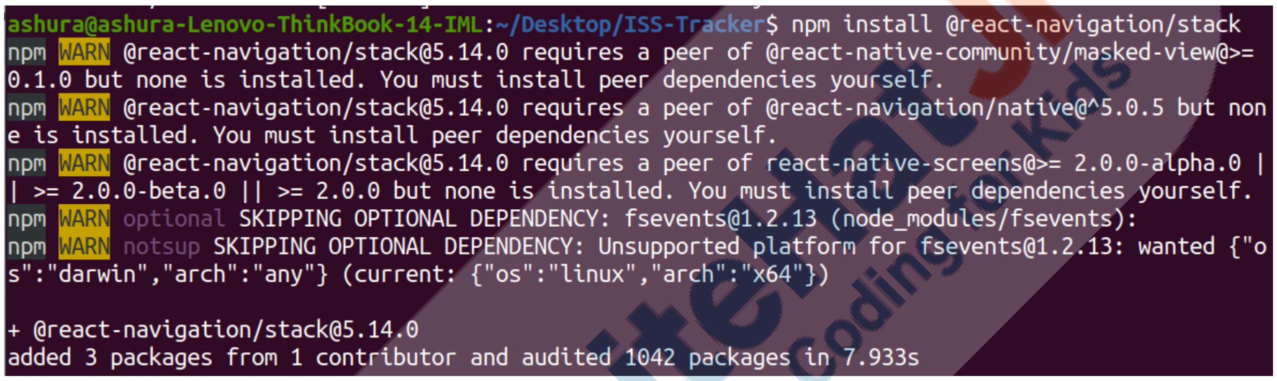
The output using the **MeteorScreen** in **App.js**:



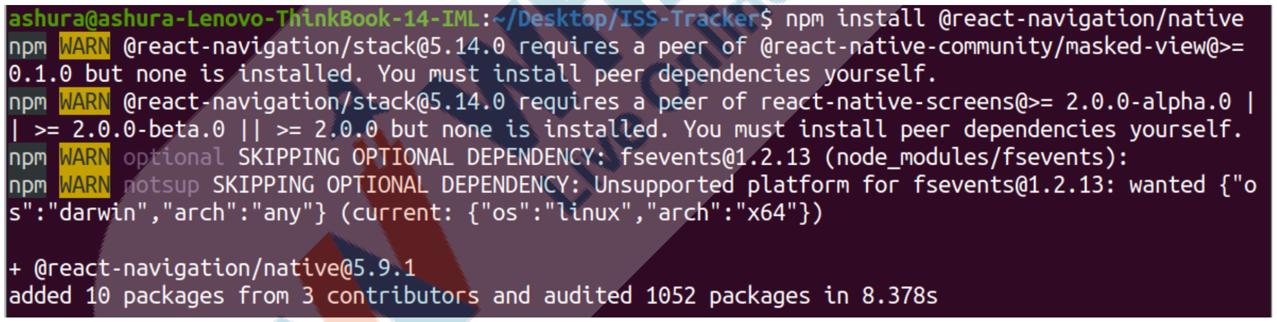
We have the screens ready but we have no means of going from one screen to another. How do you think we can do it? Yes!! React- Native provides us with a library called **createStackNavigator**, which will help us to add the navigation to the screens. Will you like to try using this library to create navigation between the screens?	**ESR:** We can have a navigation system to navigate between different screens. **ESR:** Yes!!
Let's get started then.	
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 ● **Create an App Stack Navigator to add navigation to the app**	
The teacher starts slideshow 🖥 **for slide 9 to 11** ● **Refer to speaker notes and follow the instructions on each slide.**	
Teacher Action	**Student Action**
<The teacher guides the student to download the code from Student Activity 1> Before using the library we'll first need to install it. *Note: Make sure the student is in the same project directory in the terminal while installing the packages.*	*<The student clones the code from Student Activity 1>* ***[Student Activity 1]***
You can add the library using the command - "**yarn add @react-navigation/stack**" or **npm install @react-navigation/stack** and "**yarn add @react-navigation/native**" or **npm install @react-navigation/native** Yarn is a faster and better tool than NPM, and you can install Yarn if you don't have it installed with the following command - **"npm install -g yarn"**	*<The student installs the library using the terminal commands>*

We will be using Yarn instead of NPM a lot in the upcoming classes! We need to add the **react-navigation/native** library as it provides us the container for stack navigation. we'll see the usage of containers as we proceed. *<The teacher guides the student to install the library using the terminal commands>*	



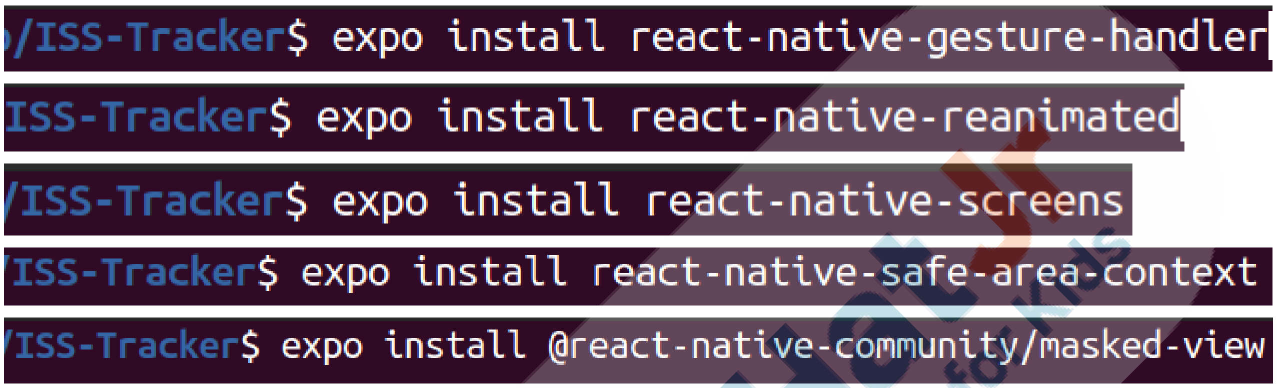
```
ashura@ashura-Lenovo-ThinkBook-14-IML:~/Desktop/ISS-Tracker$ npm install @react-navigation/stack
npm WARN @react-navigation/stack@5.14.0 requires a peer of @react-native-community/masked-view@>=
0.1.0 but none is installed. You must install peer dependencies yourself.
npm WARN @react-navigation/stack@5.14.0 requires a peer of @react-navigation/native@^5.0.5 but non
e is installed. You must install peer dependencies yourself.
npm WARN @react-navigation/stack@5.14.0 requires a peer of react-native-screens@>= 2.0.0-alpha.0 |
| >= 2.0.0-beta.0 || >= 2.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"o
s":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})

+ @react-navigation/stack@5.14.0
added 3 packages from 1 contributor and audited 1042 packages in 7.933s
```



```
ashura@ashura-Lenovo-ThinkBook-14-IML:~/Desktop/ISS-Tracker$ npm install @react-navigation/native
npm WARN @react-navigation/stack@5.14.0 requires a peer of @react-native-community/masked-view@>=
0.1.0 but none is installed. You must install peer dependencies yourself.
npm WARN @react-navigation/stack@5.14.0 requires a peer of react-native-screens@>= 2.0.0-alpha.0 |
| >= 2.0.0-beta.0 || >= 2.0.0 but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"o
s":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})

+ @react-navigation/native@5.9.1
added 10 packages from 3 contributors and audited 1052 packages in 8.378s
```

There are few dependencies that this library has we'll also install those using the following expo command: **expo install react-native-gesture-handler** **expo install react-native-reanimated** **expo install react-native-screens** **expo install react-native-safe-area-context** **expo install @react-native-community/masked-view**	*<The student installs the dependencies using the expo commands>*

*<**expo install** and **npm install** is same, the only difference is npm installs the latest version of library whereas **expo install** installs compatible version of library>*
*<The teacher guides the student to install the library using the **expo command**>*



```
/ISS-Tracker$ expo install react-native-gesture-handler
ISS-Tracker$ expo install react-native-reanimated
/ISS-Tracker$ expo install react-native-screens
/ISS-Tracker$ expo install react-native-safe-area-context
/ISS-Tracker$ expo install @react-native-community/masked-view
```

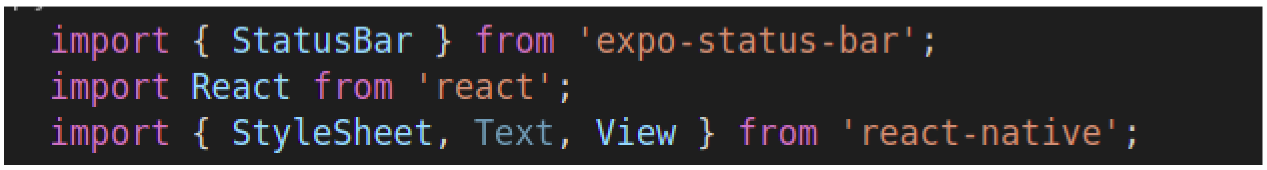
We have all the dependencies installed, now let's code to add the screens to the navigator.

In which file will we set up the stack navigator?
As you can see we already have some code in the **App.js** file, we don't need some of the code from there such as code for the status bar and the styles sheet.

ESR:
In the App.js file

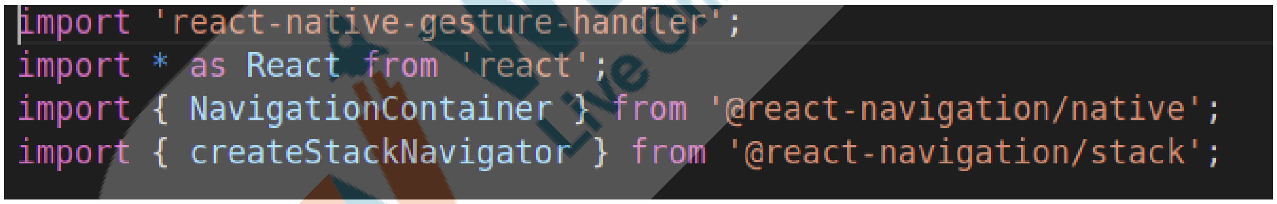
<The teacher guides the student to remove the code for the status bar and the style sheet>

<The student removes the code for the status bar and the style sheet>

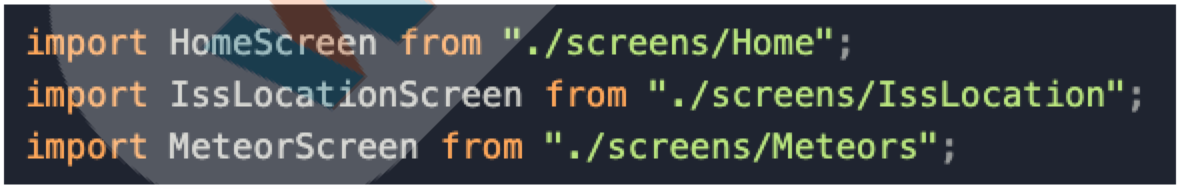


```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';
```

First, we'll start by importing the necessary components from the libraries and dependencies that we installed earlier. We'll import **react-native-gesture-handler** *It is recommended for use with react-navigation because it enhances the touch experiences of the user and gives nice interactivity to the app. The clicks would feel real, intuitive, etc.* We'll import **Navigationcontainer** from **@react-navigation/native.** We'll import **createStackNavigator** from **@react-navigation/stack** We have already imported our screens. Now we just need to create a stack navigator and pass the screen to it.	

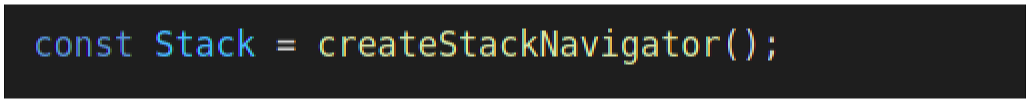


```
import 'react-native-gesture-handler';
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
```



```
import HomeScreen from "./screens/Home";
import IssLocationScreen from "./screens/IssLocation";
import MeteorScreen from "./screens/Meteors";
```

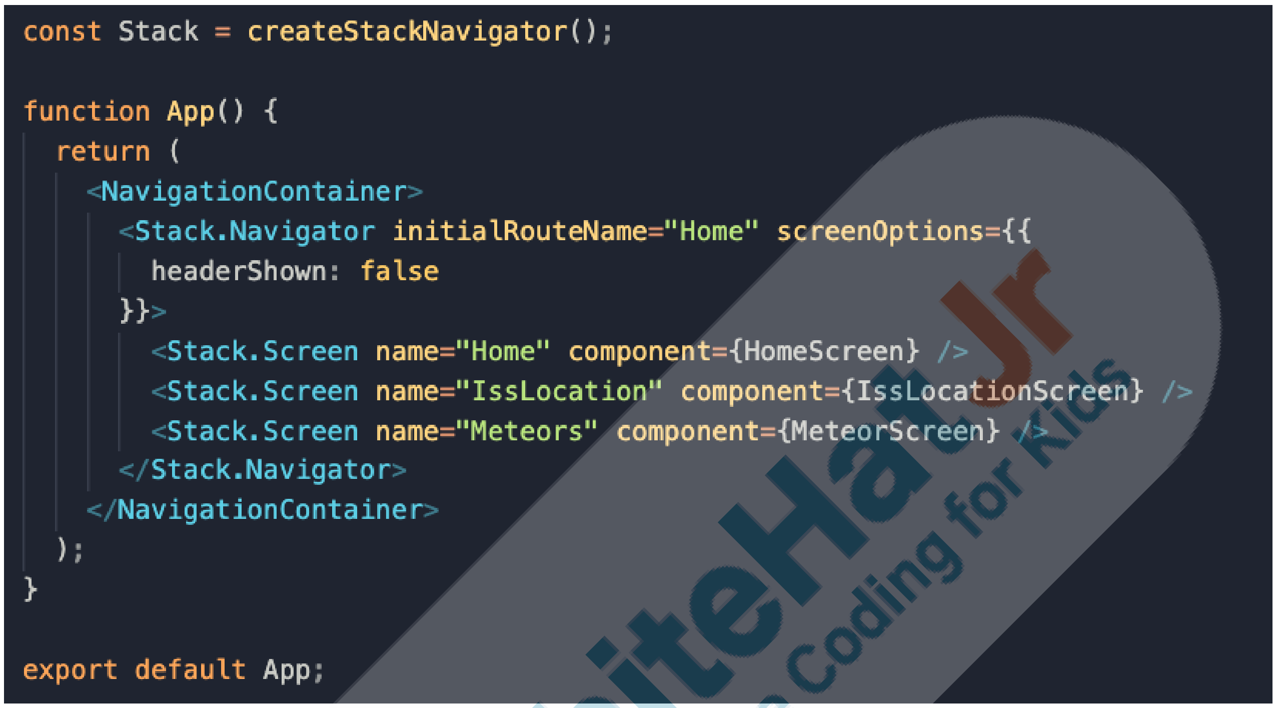
We'll create a **createStackNavigator()** function and store it in a constant variable called **Stack**. *<The teacher guides the student to create a **createStackNavigator()** function and stores it in a constant variable called **Stack**.>*	*<The student codes to create a **createStackNavigator()** function and store in a constant variable called **Stack**.>*



```
const Stack = createStackNavigator();
```

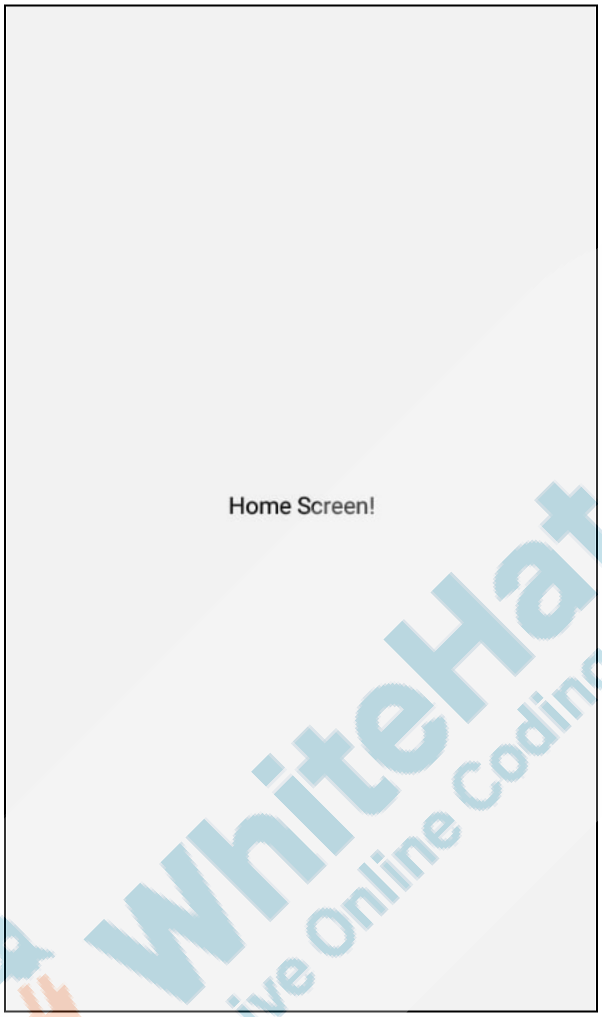
Inside the **return()** function we'll use the **NavigationContainer** component and inside this component, we'll add the screens to the stack navigator as the components. *<The teacher opens the documentation from Teacher Activity-2 link and goes through the usage of Navigation container>* Which is the main screen that we want to see when our app opens up? And to do so **stackNavigator** has a property called as **initialRouteName**. Using this property we can set the name of the screen that we want to see by default. So using this property we'll set the **Home** screen as the initial screen and in the screen options, we'll set the **headerShown** as **false** as we don't need it. **screenOptions** is a property of the **stackNavigator**. Using the **Stack.Screen** we'll get the screens from the Stack navigator that we have created earlier. *<The teacher guides the student to create the **NavigationContainer** component and inside the component add the screens.>*	**ESR:** We want to see the Home screen when we open the app. *<The student creates the **NavigationContainer** component and inside*

	container component add the screens>



```
const Stack = createStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home" screenOptions={{
        headerShown: false
      }}>
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="IssLocation" component={IssLocationScreen} />
        <Stack.Screen name="Meteors" component={MeteorScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default App;
```

Awesome work! Now let's run and test the output. Now we only see the **Home** screen on our screen. That is because we have only added the screens to the stack navigators and not decided how we'll move from one screen to another.	*<The student runs the code to check the output>*



Teacher Guides Student to Stop Screen Share	
• WRAP UP SESSION - 5 mins	
The teacher starts slideshow for slide 13 to 23 Refer to speaker notes and follow the instructions on each slide.	
Activity details	**Solution/Guidelines**

Run the presentation from slide 13 to slide 22. **Following are the wrap-up session deliverables:** • **Explain the facts and trivia.** • **Next class challenge.** • **Project for the day.** • **Additional Activity.**	***Guide the student to develop the project and share it with us.***
QnA Session	
Question	**Answers**
To install any library using npm we need to use the ______ command. A. install npm library_name B. npm.install library_name C. npm install library_name D. library_name.install	**C**
Which React-Native library is used to implement navigation in today's class? A. React navigation B. Navigation Container C. createStackNavigator D. Stack.Screen	**C**
Considering we have a screen HomeScreen.js in which we have a class HomeScreen, to use it App.js, what needs to be done? A. Import it in App.js B. Export it in app.js C. No need to import, we can directly use it D. None of the above	**A**

WhiteHat Jr
Live online Coding for Kids

End the quiz panel	
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!	*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-1** **Goal of the Project:** In Class 76, you learned how to create new screens. We also learned to create a Stack Navigator and add the screens to it. In the project, set up a new project folder. Create different screens for the app and add them to the Stack Navigator. **Story:**	**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.**

Jeff is an astrophysicist and he loves to look at the stars. Over the years of work, he has gained a lot of knowledge and he wants to share the knowledge with people. So he seeks your help in creating an app that will help him relay the information to other people. 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 ACTIVITIES	
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	Teacher Reference code	https://github.com/React-Native-Frontier/PRO-C76-ISS-Tracker
Teacher Activity 2	Navigation Container documentation	https://reactnavigation.org/docs/navigation-container/
Teacher Activity 3	Teacher Aid	https://drive.google.com/file/d/1WA1BQff4dmgv5BInU3f_imk4vlpvAyMa/view?usp=sharing
Student Activity 1	Boilerplate code	https://github.com/React-Native-Frontier/PRO-C76-ISS-Tracker-SA-boilerplate
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC-PRO-V3-C76-With+Cues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/43edb142-d2d5-459b-94f8-f04ab90860be.pdf
Project Solution	Stellar Stage-1	https://github.com/pro-whitehatjr/Stellar-Stage-1