

Topic	THE STORYTELLING APP	
Class Description	In this class, the students will start creating a new app. This will be a storytelling app with social media features where the students will learn how to integrate tab and drawer navigation and integrate them together.	
Class	C81	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> Discuss the structure and blueprint of the storytelling app. Learn tab and drawer navigation. 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources: <ul style="list-style-type: none"> Visual Studio Code Editor laptop with internet connectivity earphones with mic notebook and pen Student Resources: <ul style="list-style-type: none"> 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
Credits	We are using icons that are open source, MIT licensed and built by Ionic .	
WARM-UP SESSION - 5 mins		

CONTEXT

- Discuss the flow of the app and create blueprints accordingly.



Teacher starts slideshow from slides 1 to 10

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

Activity details	Solution/Guidelines
<p><i>Hey <student name>. How are you? It's great to see you! Are you excited to learn something new today?</i></p> <p>Run the presentation from slide 1 to slide 3.</p> <p>Following are the warm up session deliverables:</p> <ul style="list-style-type: none"> • Connecting students to the previous class. 	<p>ESR: Hi, thanks, yes I am excited about it!</p> <p>Click on the slide show tab and present the slides.</p>
QnA Session	
Question	Answer
<p>Based on the threat score, which of the following meteors is the least dangerous to Earth?</p> <p>A. threat_score above 75 but less than 150</p> <p>B. threat_score below 75</p> <p>C. threat_score even below 30</p> <p>D. threat_score above 150</p>	C
<p>Which navigation did we use in the ISS Tracker App?</p> <p>A. Drawer Navigation</p> <p>B. Tab Navigation</p> <p>C. Stack Navigation</p> <p>D. Switch Navigation</p>	C

Continue the warm up session		
Activity details		Solution/Guidelines
<p>Run the presentation from slide 4 to slide 10 to set the problem statement.</p> <p>Following are the warm up session deliverables:</p> <ul style="list-style-type: none"> • Explain the storytelling app, home screen, navigating screens. 		Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
<p><u>CHALLENGE</u></p> <ul style="list-style-type: none"> • Discuss and design the blueprint of the app. • Build the bottom tab navigation. 		
Step 2: Teacher-led Activity (15 min)	<p>Alright, so let's start by setting up our app with Expo -</p> <p><i>expo init storytelling-app</i></p> <p><i>Teacher sets up a new expo project.</i></p>	
	<p>Since our app will be using different navigations, let's install them.</p> <p>Don't forget to toggle into your newly created project folder before running the following commands:</p>	

	<p> <code>expo install</code> <code>react-native-gesture-handler</code> <code>react-native-reanimated</code> <code>react-native-screens</code> <code>react-native-safe-area-context</code> <code>@react-native-community/masked-view</code> <code>@react-navigation/native</code> <code>@react-navigation/bottom-tabs</code> </p> <p>And finally, for react-native-reanimated, add Reanimated's babel plugin to your babel.config.js</p> <p>Read more about react-native-reanimated installation Student Activity 3.</p> <p>Here, the <code>@react-navigation/native</code> <code>@react-navigation/bottom-tabs</code> <code>@react-navigation/drawer</code> modules depend on <code>react-native-gesture-handler</code>, <code>react-native-reanimated</code> and <code>react-native-screens</code> modules.</p>	
--	--	--

B babel.config.js ✕

B babel.config.js > <unknown> > exports

```

1  module.exports = function(api) {
2    ...
3    api.cache(true);
4    return {
5      presets: ['babel-preset-expo'],
6      plugins: ['react-native-reanimated/plugin']
7    };
8  };

```

Let's think about the app!

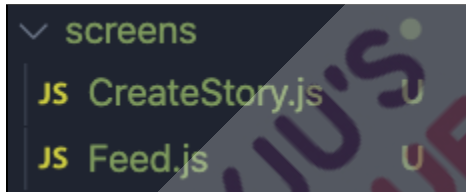
We want our storytelling app to be like a social media app.

In our app, we will require the following screens -


1. Story feed screen to display a list of all the stories.
2. A screen with a form to create stories.
3. A story screen where the user can read a particular story.
4. A profile screen where the user can set the theme of their app.

Let's not worry about the user logging in and logging out at this phase.

All the social media apps that are available have a generic structure which is to have icons at the bottom

	<p>of the screen to toggle between different pages.</p> <p>Let's start by creating a screens folder where we will place all our screens, just like how we did in the previous project.</p> <p>In this screens folder, we will have 2 files:</p> <ol style="list-style-type: none"> 1. Feed.js 2. CreateStory.js  <p>Again, we don't have to build these screens at this point in time.</p> <p>We can simply just use screen names as placeholders for the time being -</p>	

JS Feed.js ×

81t > screens > JS Feed.js >  Feed

```
1  import React, { Component } from 'react';
2  import { Text, View } from 'react-native';
3
4  export default class Feed extends Component {
5    render() {
6      return (
7        <View
8          style={{
9            flex: 1,
10           justifyContent: "center",
11           alignItems: "center"
12         }}>
13          <Text>Feeds</Text>
14        </View>
15      )
16    }
17  }
```

JS CreateStory.js ×

81t > screens > JS CreateStory.js > CreateStory

```

1  import React, { Component } from 'react';
2  import { Text, View } from 'react-native';
3
4  export default class CreateStory extends Component {
5    render() {
6      return (
7        <View
8          style={{
9            flex: 1,
10           justifyContent: "center",
11           alignItems: "center"
12         }}>
13          <Text>Create Story</Text>
14        </View>
15      )
16    }
17  }

```

Alright! Now in the previous project, we used stack navigation to navigate between different screens.

Here, we will be using the bottom tab navigation. Our navigation code would again be inside *App.js* file, and it would look something like this -

```

import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import Icons from 'react-native-vector-icons/Ionicons';

```



```
import Feed from "../screens/Feed";
import CreateStory from "../screens/CreateStory";

const Tab = createBottomTabNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator
        screenOptions={({ route }) => ({
          tabBarIcon: ({ focused, color, size }) => {
            let iconName;
            if (route.name === 'Feed') {
              iconName = focused
                ? 'book'
                : 'book-outline';
            } else if (route.name === 'CreateStory') {
              iconName = focused ? 'create' : 'create-outline';
            }
            return <Ionicons name={iconName} size={size} color={color} />;
          },
        })
        tabBarOptions={{
          activeTintColor: 'tomato',
          inactiveTintColor: 'gray',
        }}
      >
        <Tab.Screen name="Feed" component={Feed} options={{headerShown:false}} />
        <Tab.Screen name="CreateStory" component={CreateStory}
options={{headerShown:false}} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```

Let's understand this code. First, we are importing all the modules we require for bottom tab navigation along with the screens that we've created.

Next, we created a **bottom tab navigator** called **Tab** with the help of **createBottomTabNavigator()** function.

Inside the app's code, we are then using the **<NavigationContainer>** to wrap our navigation screens. Remember that no matter whatever type of navigation you're using, it will always be enclosed within the **<NavigationContainer>** component.

Inside this component, we have our **<Tab.Navigator>** component. This is our bottom tab navigator where with the help of attributes we are defining the specifications of our bottom tab navigation menu.

We have passed an attribute **screenOptions** inside which we are defining icons for all the **routes**. In our case, these routes will be **<Feed/>** and **<CreateStory/>** components that we created.

With the help of the if-else condition, we are setting different icons for our different screens. We are also checking if the current route is focused or not.

What this means is that if a user has selected the **Feed** screen, then the screen is focused and it will display the **filled book icon**. If not, it will display the **outline book icon**.

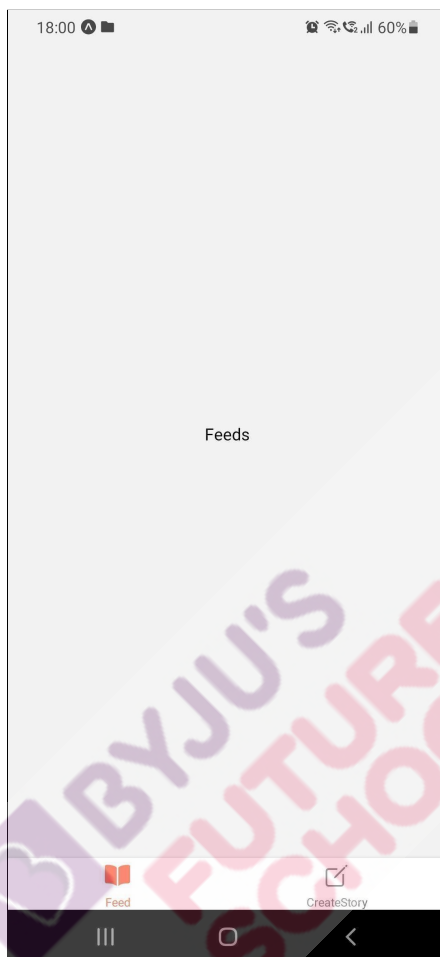
You can check out the icons that are available with ionicons by referring to [Teacher Activity 1](#) (<[Student Activity 1](#)> for students).

We also have another attribute, **tabBarOptions** where we have specified that the active icon (the screen on which the user has clicked) should be in **tomato** color while the inactive icon should be in **gray** color.

We finally use our two screens in the **<Tab.Screen>** component in the navigator and set the name by which we want to refer to our screen and the component that should be used for our screen.

	By far, if we run our app, our output should look something like this -	
--	---	--





Awesome! Now let's add drawer navigation and some more screens to this.

Teacher Stops Screen Share

Now it's your turn. Please share your screen with me.

STUDENT-LED ACTIVITY - 20 mins

Teacher starts slideshow



from slide 11 to slide 12

- 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

- Create the Profile Screen.
- Integrate the drawer navigation.

Step 3: Student-Led Activity (15 mins)	<p>Please refer to <Student Activity 2> to clone the boilerplate code.</p> <p>Now before adding drawer navigation to the project, let's first install it with the following command -</p> <pre>yarn add @react-navigation/drawer</pre>	<p><i><Student refers to Student Activity 2 to clone the boilerplate code.></i></p>
	<p>Great! Now let's start by creating a <i>Profile.js</i> in the <i>screens</i> folder -</p>	<p><i>Student creates Profile.js.</i></p>

JS Profile.js ×

81t > screens > JS Profile.js > Profile

```

1  import React, { Component } from 'react';
2  import { Text, View } from 'react-native';
3
4  export default class Profile extends Component {
5    render() {
6      return (
7        <View
8          style={{
9            flex: 1,
10           justifyContent: "center",
11           alignItems: "center"
12         }}>
13          <Text>Profile</Text>
14        </View>
15      )
16    }
17  }

```

Next, since we have Tab Navigator already and we want to add Drawer Navigator as well, let's create a new folder *navigation* and move our code of Bottom Tab Navigator to a file called *TabNavigator.js*.

We are doing this so that it's easier for us to use our navigation as components. We'll see why in just a minute.

Student creates a new folder "navigation" and moves the code of the bottom tab navigator to a new file TabNavigator.js inside this folder.

▼ navigation ●
JS TabNavigator.js U

```
import React from 'react';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import Ionicons from 'react-native-vector-icons/Ionicons';
import Feed from "../screens/Feed";
import CreateStory from "../screens/CreateStory";

const Tab = createBottomTabNavigator();

const BottomTabNavigator = () => {
  return (
    <Tab.Navigator
      screenOptions={({ route }) => ({
        tabBarIcon: ({ focused, color, size }) => {
          let iconName;
          if (route.name === 'Feed') {
            iconName = focused
              ? 'book'
              : 'book-outline';
          } else if (route.name === 'CreateStory') {
            iconName = focused ? 'create' : 'create-outline';
          }
          return <Ionicons name={iconName} size={size} color={color} />;
        },
      })}
      tabBarOptions={{
        activeTintColor: 'tomato',
        inactiveTintColor: 'gray',
      }}
    >
    <Tab.Screen name="Feed" component={Feed} />
    <Tab.Screen name="CreateStory" component={CreateStory} />
  </Tab.Navigator>
);
```

```
}  
  
export default BottomTabNavigator
```

Alright, now we do know that this code in the **TabNavigator.js** renders our **Feed Screen and CreateStory Screen**.

What if we use this **TabNavigation.js** as a screen in our Drawer Navigation? Will we be able to work it out?

Let's create a **DrawerNavigator.js** in the **navigation** folder and add the following code to it -

ESR:
Varied.

<Student listens and writes the code as the teacher recites.>

```
import React from "react";  
import { createDrawerNavigator } from "@react-navigation/drawer";  
import TabNavigator from "../TabNavigator";  
import Profile from "../screens/Profile";  
  
const Drawer = createDrawerNavigator();  
  
const DrawerNavigator = () => {  
  return (  
    <Drawer.Navigator screenOptions={{headerShown:false}}>  
      <Drawer.Screen name="Home" component={TabNavigator} />  
      <Drawer.Screen name="Profile" component={Profile} />  
    </Drawer.Navigator>  
  );  
};  
  
export default DrawerNavigator;
```


Here, similar to what we have done with our Tab Navigation, we have created a Drawer Navigator and added the screens to it. Do note that we are not using **<NavigationContainer>** component here, since we will use it in our *App.js*.

Now here, as you can see, we have called our **TabNavigator** as the **Home** screen and we have also added our **Profile** screen here.

Now in our *App.js*, let's just simply call the **<DrawerNavigator/>** component that we created inside a **<NavigationContainer>** component.

Student writes the code.

```
81t > JS App.js > App
1  import * as React from 'react';
2  import { NavigationContainer } from '@react-navigation/native';
3  import DrawerNavigator from "../navigation/DrawerNavigator";
4
5  export default function App() {
6    return (
7      <NavigationContainer>
8      | <DrawerNavigator />
9      | </NavigationContainer>
10
11    );
12  }
```

If we run our app now, we will see the following results.

With this, our basic blueprint and the structure of the app is complete.



Now we have the basic structure of the app ready. *Teacher can ask the student how the class was and prompt the student to discuss what next can be done in the app.*

Teacher Guides Student to Stop Screen Share

Teacher starts slideshow




from slide 13 to slide 23

Activity details	Solution/Guidelines
<p>Run the presentation from slide 13 to slide 23.</p> <p>Following are the warm up session deliverables:</p> <ul style="list-style-type: none"> ● Explain the facts and trivias ● Next class challenge 	<p>Guide the student to develop the project and</p>

<ul style="list-style-type: none"> ● Project for the day ● Additional Activity 	share with us.
Quiz time - Click on in-class quiz	
Question	Answer
<p>What is the use of NavigationContainer?</p> <ul style="list-style-type: none"> A. It creates the navigation. B. It wraps the navigation screens. C. It creates the screens for us. D. It creates the bottom navigation. 	B
<p>What does line name = "Feed" and name = "CreateStory" in the following snippet of code do?</p> <pre><Tab.Screen name="Feed" component={Feed} /> <Tab.Screen name="CreateStory" component={CreateStory} /></pre> <ul style="list-style-type: none"> A. We are setting different icons for our different screens. We are also checking if the current route is focused or not. B. We have specified that the active icon (the screen on which the user has clicked) should be in tomato color while the inactive icon should be in gray color. C. We are setting the name by which we want to refer to our screen and the component that should be used for our screen. D. We are navigating between the components "Feed" and "CreateStory". 	C
<p>Which of the following can be used to create a drawer navigator?</p> <ul style="list-style-type: none"> A. createNavigator() B. createDrawerNavigator() C. createNavigatorDrawer() D. createAppDrawerNavigator() 	B
End the quiz panel	
<p align="center"><u>FEEDBACK</u></p> <ul style="list-style-type: none"> ● Appreciate the student for their attentiveness in the class. 	

● Get them to play around with different ideas		
	<p>Amazing work today! You get a “hats-off”.</p> <p>In the next class, we’ll be working on creating the full-fledged Feed screen where the user will be able to see different stories.</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div> <div>Creatively Solved Activities +10</div> <div>Great Question +10</div> <div>Strong Concentration +10</div> </div>
<p>Project Overview</p> <p>Spectagram Stage 1</p> <p>Goal of the Project:</p> <p>In Class 81, we learned to create the blueprints for the storytelling app. We have also learned to create two different screens and add them to the tab navigator.</p> <p>In this project, you will practice the concepts learned in the class.</p> <p>Story:</p> <p>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.</p>		<p><i>Students engage with the teacher over the project.</i></p>

<p>Guide Jenny to set up the project and name it as spectagram and create the blueprints for the app.</p> <p>Bye Bye!</p>		
<p style="text-align: center;">Teacher ends slideshow</p> 		
<p style="text-align: center;">Teacher Clicks</p> <div style="text-align: center; background-color: red; color: white; padding: 5px; border-radius: 10px;"> ✕ End Class </div>		
<p>Additional Activities</p>	<p><i>Encourage the student to write reflection notes in their reflection journal using markdown.</i></p> <p>Use these as guiding questions:</p> <ul style="list-style-type: none"> • What happened today? <ul style="list-style-type: none"> ◦ 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? 	<p><i>The student uses the markdown editor to write their reflections in a reflection journal.</i></p>

Activity	Activity Name	Links
Teacher Activity 1	Ion Icons	https://ionicons.com/
Teacher Activity 2	Reference Code	https://github.com/pro-whitehatjr/ST-81-Solution
Teacher Activity 3	Teacher Aid	https://drive.google.com/file/d/1WA1

		BQff4dmgv5BInU3f_imk4vlpvAyMa/view?usp=sharing
Student Activity 1	Ion Icons	https://ionicons.com/
Student Activity 2	Boilerplate Code	https://github.com/pro-whitehatjr/ST-81-Boilerplate
Student Activity 3	react-native-reanimated installation	https://docs.swmansion.com/react-native-reanimated/docs/fundamentals/installation/
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_V3_C81_LITE_withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whitehatjr.online/aff7fc06-5e45-45c0-bac9-7a40c81e44c1.pdf