CS-PRO-C84(V3)

WhiteHat Jr
Live Online Coding for Kids

STACK NAVIGATION

What is our GOAL for this MODULE?

In this class, we continued to build a Storytelling application. We learned to add **Stack Navigation** to the Storytelling App. Plus, we implemented a text-to-speech conversion feature.

What did we ACHIEVE in the class TODAY?

- Installed **Stack Navigation** with the following command and created a new file - **StoryScreen.js.**
- Created a new file - **StackNavigator.js** inside our navigation folder.
- Accessed the StoryScreen from our Feed screen.
- Installed **expo-speech** *using* ***expo install expo-speech.***
- Wrap our icon within a **TouchableOpacity** to be able to add an **onPress** event to it!
- Created the **initiateTTS()** function.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- integrate stack navigation on the app
- create the story screen
- add text-to-speech

How did we DO the activities?

1. Install stack navigation with the following command -
   yarn add @react-navigation/stack

2. Create a new file - *StoryScreen.js*.

```
JS StoryScreen.js ×
84t > screens > JS StoryScreen.js > ...
1   import React, { Component } from 'react';
2   import { Text, View } from 'react-native';
3
4   export default class StoryScreen extends Component {
5       render() {
6           return (
7               <View
8                   style={{
9                       flex: 1,
10                      justifyContent: "center",
11                      alignItems: "center"
12                  }}>
13                  <Text>Story Screen</Text>
14              </View>
15          )
16      }
17  }
```

3. Create a new file—StackNavigator.js inside our navigation folder.

```
import React from "react";
import { createStackNavigator } from "@react-navigation/stack";
import TabNavigator from "./TabNavigator";
import StoryScreen from "../screens/StoryScreen";

const Stack = createStackNavigator();

const StackNavigator = () => {
  return (
    <Stack.Navigator initialRouteName="Home" screenOptions={{
```

```
        headerShown: false
      }}>
        <Stack.Screen name="Home" component={TabNavigator} />
        <Stack.Screen name="StoryScreen" component={StoryScreen} />
      </Stack.Navigator>
  );
};

export default StackNavigator;
```

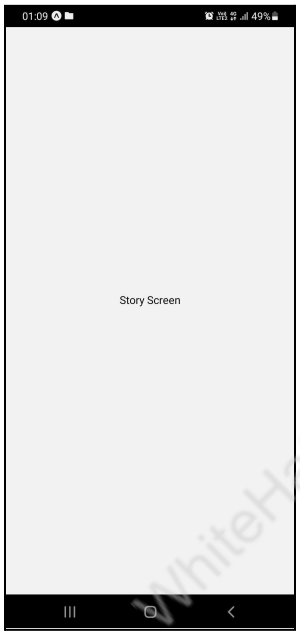
4. **TabNavigator** is our default view. If we add this **Stack Navigator** now in our **DrawerNavigator**, we will still see the **TabNavigator** by default, and have the ability to toggle to our StoryScreen.

```
import React from "react";
import { createDrawerNavigator } from "@react-navigation/drawer";
import StackNavigator from "./StackNavigator";
import Profile from "../screens/Profile";

const Drawer = createDrawerNavigator();

const DrawerNavigator = () => {
  return (
    <Drawer.Navigator>
      <Drawer.Screen name="Home" component={StackNavigator} />
      <Drawer.Screen name="Profile" component={Profile} />
    </Drawer.Navigator>
  );
};

export default DrawerNavigator;
```

5. Access the StoryScreen from our Feed screen, but our cards are in **StoryCard.js.** Access to the navigation props in the Feed Screen, but we want the navigation to happen in the **StoryCard**; therefore we will have to pass it to the component.

```
renderItem = ({ item: story }) => {
    return <StoryCard story={story} navigation={this.props.navigation} />
};
```

6. Use this navigation in our **<StoryCard>** component. We will have to use a **<TouchableOpacity>** component to wrap our card contents inside it and perform the navigation on the onPress event of our **<TouchableOpacity>** component.



7. Add 2 new states, **speakerColor** set to **gray** and **speakerIcon** set to **'volume-high-outline'** in StoryScreen.js

```
const current_color = this.state.speakerColor;
this.setState({
  speakerColor: current_color === "gray" ? "#ee8249" : "gray"
});  if (current_color === "gray") {

  Speech.speak(`${title} by ${author}`);
  Speech.speak(story);
```

```
    Speech.speak("The moral of the story is!");
    Speech.speak(moral);
  } else {
    Speech.stop();
  }
}
```

OUTPUT:

8. Install **expo-speech** *using* **expo *install* expo-speech**.

9. Wrap the **icon** within a **TouchableOpacity** to be able to add an **onPress** event to it. Now, for the text-to-speech, we want it to relay the:
   - title
   - name of the author
   - story
   - moral of the story

   For this, we can call an **initiateTTS()** function on the onPress event and pass these values to it.

```
<TouchableOpacity
  onPress={() =>
    this.initiateTTS(
      this.props.route.params.story.title,
      this.props.route.params.story.author,
      this.props.route.params.story.story,
      this.props.route.params.story.moral
    )
  }
>
  <Ionicons
    name={this.state.speakerIcon}
    size={RFValue(30)}
    color={this.state.speakerColor}
    style={{ margin: RFValue(15) }}
  />
</TouchableOpacity>
```

10. Create the **initiateTTS()** function. To check the current state of speakerColor in the constant current_color and based on its value set the state.

    - Check if the current_color is gray. If the current color is gray, that means that the user enabled text-to-speech. Note that the current_color here is the color of the icon before the user pressed it.

```
async initiateTTS(title, author, story, moral) {
const current_color = this.state.speakerColor;
this.setState({
  speakerColor: current_color === "gray" ? "#ee8249" : "gray"
});
if (current_color === "gray") {
  Speech.speak(`${title} by ${author}`);
  Speech.speak(story);
  Speech.speak("The moral of the story is!");
  Speech.speak(moral);
} else {
  Speech.stop();
}
}
```

OUTPUT

What's next?

In the next class, we will be implementing Google Authentication and integrating the app with Firebase.

Expand your knowledge

1. Learn more about text-to-speech conversion using react native: https://www.npmjs.com/package/react-native-tts