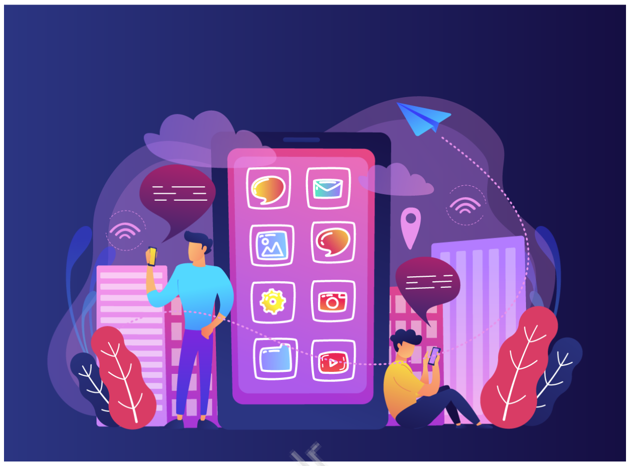
CS-PRO-C82(V3)

MATERIAL UI AND FEED SCREEN



What is our GOAL for this MODULE?

In this class, we continued to build a Storytelling application. We also explored the material UI with the help of which, we styled the bottom tab navigation and the feed screen.

What did we ACHIEVE in the class TODAY?

- Customized the appearance of bottom tab navigation using material UI.
- Designed the interface of the feed screen.

Which CONCEPTS/ CODING BLOCKS did we cover today?

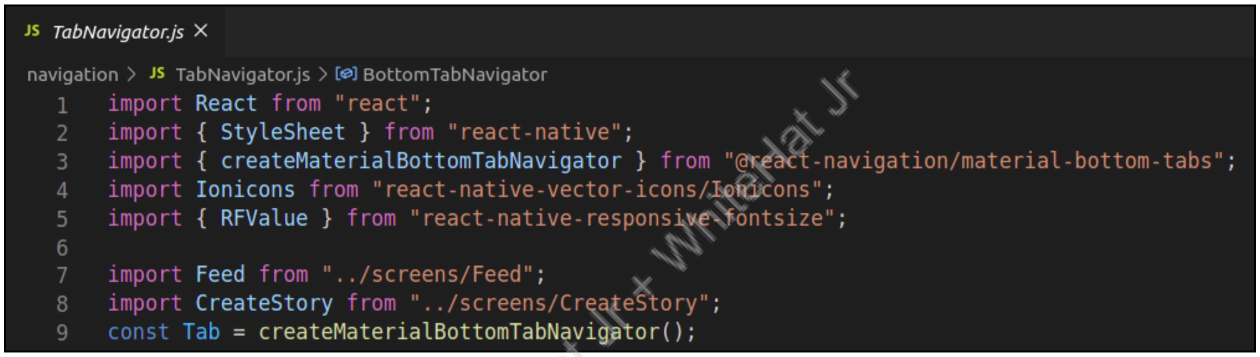
- Feed screen designing
- Bottom Tab Customization

How did we DO the activities?

1. Start with installing the required dependency:
   yarn add @react-navigation/material-bottom-tabs react-native-paper

2. Edit the TabNavigator.js in the navigation folder to add styling.

3. Import **createMaterialBottomTabNavigator()** which comes with exciting styling options.



TabNavigator.js

navigation > TabNavigator.js > BottomTabNavigator

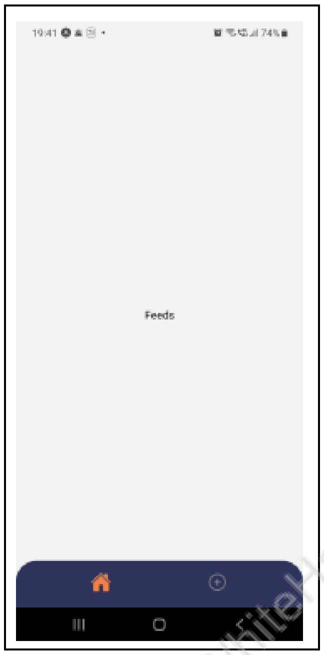
```
import React from "react";
import { StyleSheet } from "react-native";
import { createMaterialBottomTabNavigator } from "@react-navigation/material-bottom-tabs";
import Ionicons from "react-native-vector-icons/Ionicons";
import { RFValue } from "react-native-responsive-fontsize";

import Feed from "../screens/Feed";
import CreateStory from "../screens/CreateStory";
const Tab = createMaterialBottomTabNavigator();
```

4. Use **MaterialBottomTabNavigator** component for a variety of different attributes such as **labeled**, **barStyle, screenOptions, activeColor and inactiveColor.**

```
const BottomTabNavigator = () => {
 return (
     <Tab.Navigator
        labeled={false}
        barStyle={styles.bottomTabStyle}
        screenOptions={({ route }) => ({
           tabBarIcon: ({ focused, color, size }) => {
              let iconName;
              if (route.name === 'Feed') {
                 iconName = focused
                    ? 'home'
                    : 'home-outline';
              } else if (route.name === 'Create Story') {
                 iconName = focused ? 'add-circle' : 'add-circle-outline';
              }
              return <Ionicons name={iconName} size={30} color={color} style={{
width: 30 }} />;
           },
      }}}
      activeColor={'#ee8249'}
      inactiveColor={'gray'}
    >
      <Tab.Screen name="Feed" component={Feed} />
      <Tab.Screen name="Create Story" component={CreateStory} />
    </Tab.Navigator>
  );
}
```

Output:



5. Replace default font with custom fonts available inside the asset folder.

6. Install the following dependencies:
   expo install expo-font
   expo install expo-splash-screen

7. Create a temporary JSON file containing an array of objects with story data. Use this to create a UI. This will be added in the new file temp_stories.json in the screens folder.

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


import * as Font from 'expo-font';
import * as SplashScreen from 'expo-splash-screen';
SplashScreen.preventAutoHideAsync();



let customFonts = {
  'Bubblegum-Sans': require('../assets/fonts/BubblegumSans-Regular.ttf'),
};

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

8. Load custom fonts in the Feed screen using the following code blocks:

```
async _loadFontsAsync() {
     await Font.loadAsync(customFonts);
     this.setState({ fontsLoaded: true });
  }
componentDidMount() {
     this._loadFontsAsync();
 }
 render() {
     if (this.state.fontsLoaded) {
        SplashScreen.hideAsync();
        return (
           <View
              style={{
                 flex: 1,
                 justifyContent: "center",
                 alignItems: "center"
              }}>
              <Text>Feeds</Text>
```

```
            </View>
        )
    }
  }
}
```

9. Render FlatList on screen by using the following code block:

```
return (
            <View style={styles.container}>
               <SafeAreaView style={styles.droidSafeArea} />
               <View style={styles.appTitle}>
                  <View style={styles.appIcon}>
                     <Image source={require("../assets/logo.png")} style={{ width: 60,
height: 60, resizeMode: 'contain', marginLeft: 10 }}></Image>
                  </View>
                  <View style={styles.appTitleTextContainer}>
                     <Text style={styles.appTitleText}>
                        Storytelling App
                     </Text>
                  </View>
               </View>
               <View style={styles.cardContainer}>
                  <FlatList
                     keyExtractor={this.keyExtractor}
                     data={stories}
                     renderItem={this.renderItem}
                  />
               </View>
            </View>
        )
```

10. FlatList props data will be given like this:

```
let stories = require("./temp_stories.json");
```

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

keyExtractor = (item, index) => index.toString();
```

11. Add styling for app header.

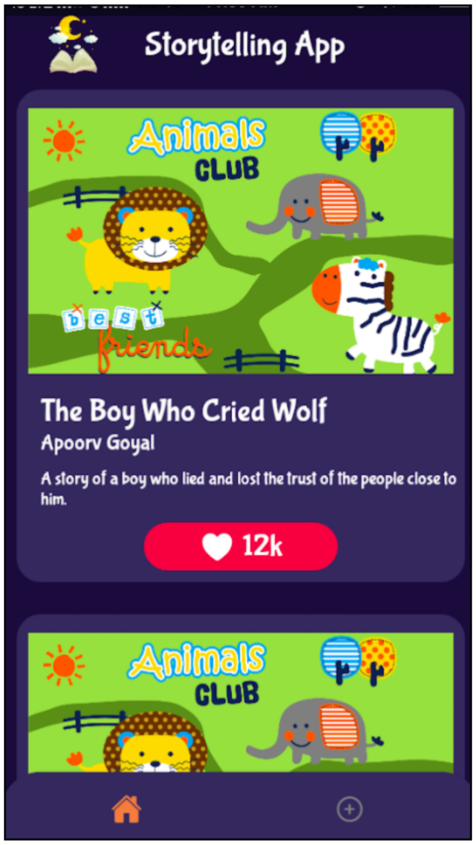
```
const styles = StyleSheet.create({
droidSafeArea: {
    marginTop: Platform.OS === "android" ? StatusBar.currentHeight : 0
  },
  cardContainer: {
    marginTop: -20,
    marginBottom: 20,
    marginLeft: 20,
    marginRight: 20,
    backgroundColor: "#2f345d",
    borderRadius: 20,
    height: undefined,
    padding: 10
  },
  titleContainer: {
    flexDirection: "row"
  },
  titleTextContainer: {
    flex: 1
  },
  storyTitleText: {
    fontFamily: "Bubblegum-Sans",
    fontSize: 25,
    color: "white"
  },
  storyAuthorText: {
    fontFamily: "Bubblegum-Sans",
    fontSize: 18,
    color: "white"
  },
  descriptionContainer: {
    marginTop: 5
  },
  descriptionText: {
    fontFamily: "Bubblegum-Sans",
    fontSize: 13,
    color: "white"
  },
  actionContainer: {
    marginTop: 10,
    justifyContent: "center",
    alignItems: "center"
  },
  likeButton: {
    backgroundColor: "#eb3948",
    borderRadius: 30,
    width: 160,
    height: 40,
    flexDirection: "row"
  },
  likeText: {
    color: "white",
    fontFamily: "Bubblegum-Sans",
    fontSize: 25,
    marginLeft: 25,
    marginTop: 6
  }
});
```

12. In the StoryCard.js file, update the render function and add the styling.

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

```
                    <View style={styles.storyImage}>
                        <Image source={require("../assets/story_image_1.png")} style={{
resizeMode: 'contain', width: Dimensions.get('window').width - 60, height: 250,
borderRadius: 10 }}></Image>
                    </View>
                    <View style={styles.titleContainer}>
                        <View style={styles.titleTextContainer}>
                            <View style={styles.storyTitle}>
                                <Text
style={styles.storyTitleText}>{this.props.story.title}</Text>
                            </View>
                            <View style={styles.storyAuthor}>
                                <Text
style={styles.storyAuthorText}>{this.props.story.author}</Text>
                            </View>
                        </View>
                    </View>
                    <View style={styles.descriptionContainer}>
                        <Text style={styles.descriptionText}>
                            {this.props.story.description}
                        </Text>
                    </View>
                    <View style={styles.actionContainer}>
                        <View style={styles.likeButton}>
                            <View style={styles.likeIcon}>
                                <Ionicons name={"heart"} size={30} color={"white"} style={{
width: 30, marginLeft: 20, marginTop: 5 }} />
                            </View>
                            <View>
                                <Text style={styles.likeText}>12k</Text>
                            </View>
                        </View>
                    </View>
                </View>
            </View>
        )
```

Final Output:



The Boy Who Cried Wolf
Apoorv Goyal
A story of a boy who lied and lost the trust of the people close to him.
12k

What's next?

We will work on the CreateStory Screen in the next class and add functionality to submit the stories.

Expand your knowledge:

1. Explore and experiment with styling the **TabNavigator:** https://reactnavigation.org/docs/tab-based-navigation/#customizing-the-appearance