CS-PRO-C80(V3)

METEOR SCREEN 2

What is our GOAL for this MODULE?

Completed the meteor screen that gives us information about the 5 most threatful meteors passing near Earth in the next 7 days. We built an interactive UI for it.

What did we ACHIEVE in the class TODAY?

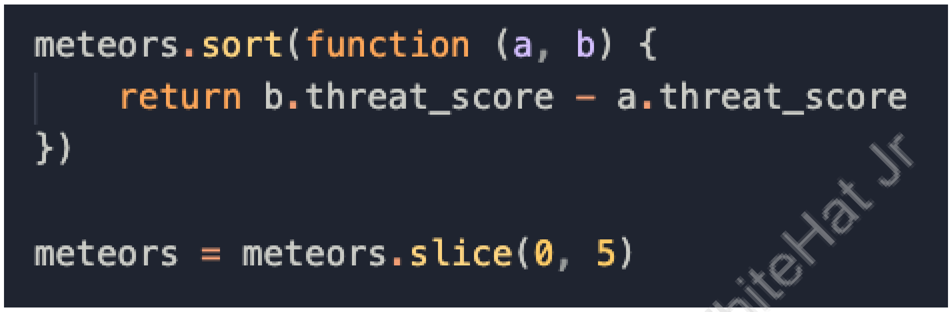
- We completed the Meteors Screen by displaying the meteor data using FlatList in carousel effect.
- We performed advanced styling to our components.

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Usage of API.
- Displaying data using **FlatList** in carousel effect

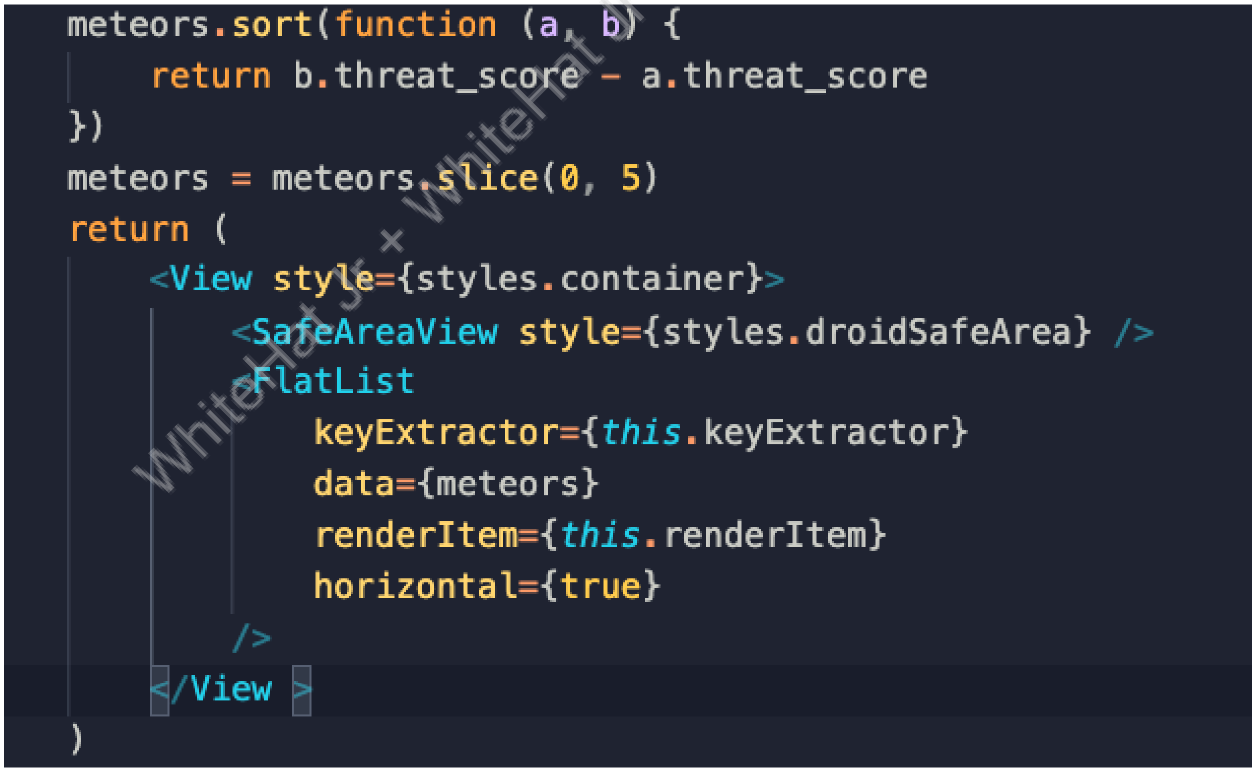
How did we DO the activities?

1. Start by sorting the meteors in descending order and take the top 5 most threatful meteors in an array -



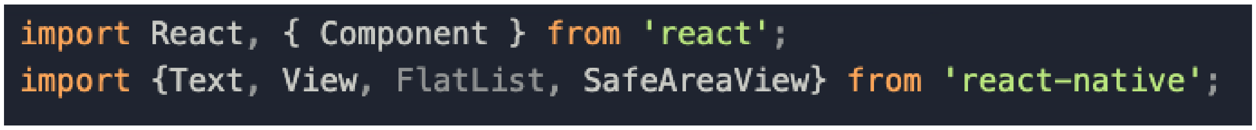
```
meteors.sort(function (a, b) {
    return b.threat_score - a.threat_score
})

meteors = meteors.slice(0, 5)
```

2. Add a **FlatList** in the return statement to render meteor data -



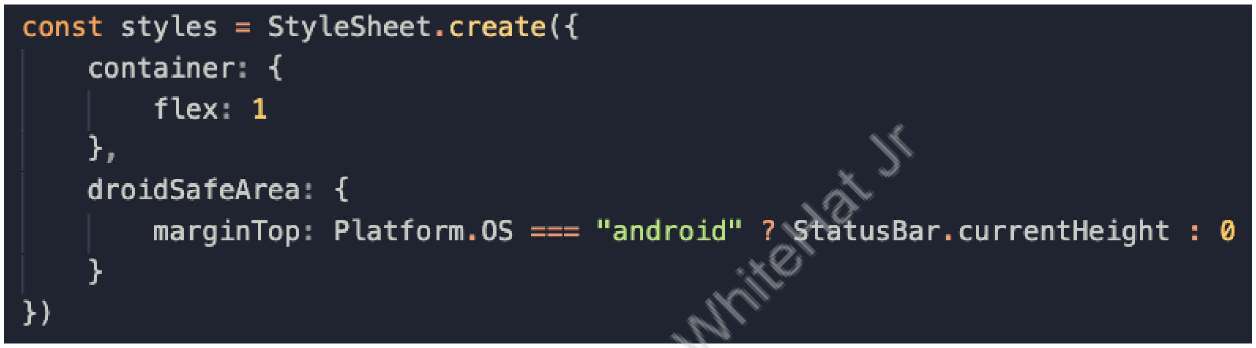
```
meteors.sort(function (a, b) {
    return b.threat_score - a.threat_score
})
meteors = meteors.slice(0, 5)
return (
    <View style={styles.container}>
        <SafeAreaView style={styles.droidSafeArea} />
        <FlatList
            keyExtractor={this.keyExtractor}
            data={meteors}
            renderItem={this.renderItem}
            horizontal={true}
        />
    </View >
)
```

3. Include **FlatList** at the top while importing -



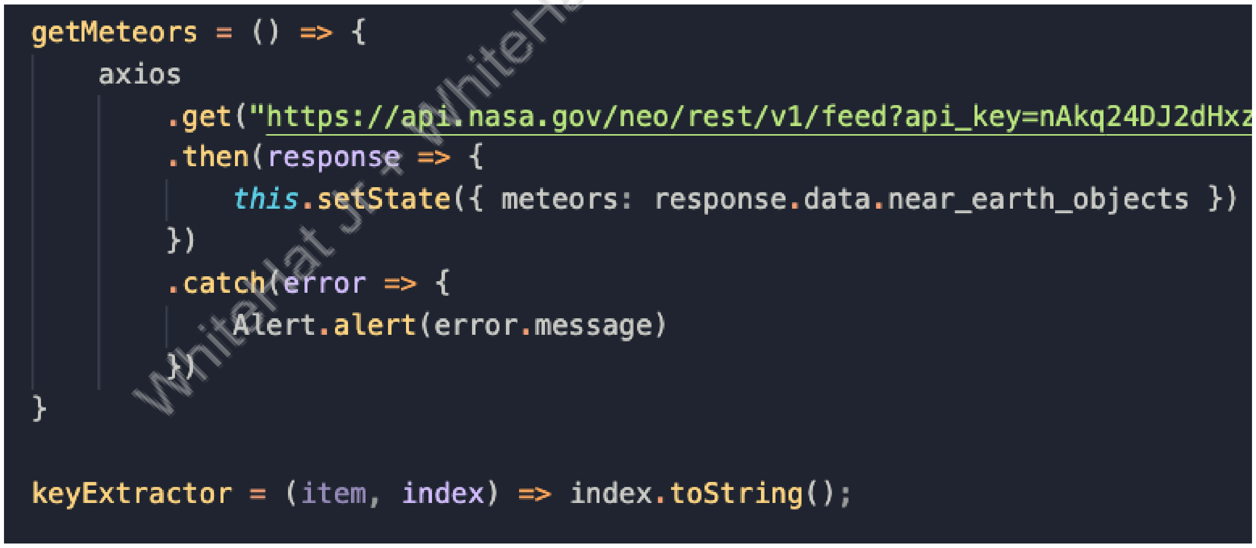
```
import React, { Component } from 'react';
import {Text, View, FlatList, SafeAreaView} from 'react-native';
```

4. Add the relevant styles -



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

5. Add a **keyExtractor** function for our **FlatList** -



```
getMeteors = () => {
    axios
        .get("https://api.nasa.gov/neo/rest/v1/feed?api_key=nAkq24DJ2dHxz
        .then(response => {
            this.setState({ meteors: response.data.near_earth_objects })
        })
        .catch(error => {
            Alert.alert(error.message)
        })
}

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

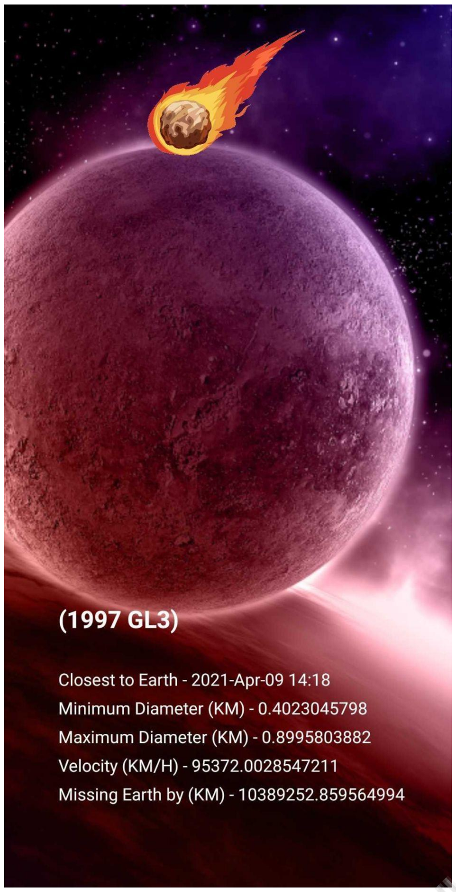
6. Create the **renderItem** function -

```
renderItem = ({ item }) => {
  let meteor = item
  let bg_img, speed, size;
  if (meteor.threat_score <= 30) {
     bg_img = require("../assets/meteor_bg1.png")
     speed = require("../assets/meteor_speed3.gif")
     size = 100
  } else if (meteor.threat_score <= 75) {
     bg_img = require("../assets/meteor_bg2.png")
     speed = require("../assets/meteor_speed3.gif")
     size = 150
  } else {
     bg_img = require("../assets/meteor_bg3.png")
     speed = require("../assets/meteor_speed3.gif")
     size = 200
  }
  return (
     <View>
        <ImageBackground source={bg_img} style={styles.backgroundImage}>
           <View styles={styles.gifContainer}>
              <Image source={speed} style={{ width: size, height: size, alignSelf: "center"
}}></Image>
              <View>
                 <Text style={[styles.cardTitle, { marginTop: 400, marginLeft: 50
}]}>{item.name}</Text>
                 <Text style={[styles.cardText, { marginTop: 20, marginLeft: 50 }]}>Closest to
Earth - {item.close_approach_data[0].close_approach_date_full}</Text>
                 <Text style={[styles.cardText, { marginTop: 5, marginLeft: 50 }]}>Minimum
Diameter (KM) - {item.estimated_diameter.kilometers.estimated_diameter_min}</Text>
                 <Text style={[styles.cardText, { marginTop: 5, marginLeft: 50 }]}>Maximum
Diameter (KM) - {item.estimated_diameter.kilometers.estimated_diameter_max}</Text>
                 <Text style={[styles.cardText, { marginTop: 5, marginLeft: 50 }]}>Velocity
(KM/H) - {item.close_approach_data[0].relative_velocity.kilometers_per_hour}</Text>
                 <Text style={[styles.cardText, { marginTop: 5, marginLeft: 50 }]}>Missing
Earth by (KM) - {item.close_approach_data[0].miss_distance.kilometers}</Text>
              </View>
           </View>
        </ImageBackground>
      </View>
  );
};
```

7. Add the relevant styling

```
const styles = StyleSheet.create({
  container: {
      flex: 1
  },
  droidSafeArea: {
      marginTop: Platform.OS === "android" ? StatusBar.currentHeight : 0
  },
  backgroundImage: {
      flex: 1,
      resizeMode: 'cover',
      width: Dimensions.get('window').width,
      height: Dimensions.get('window').height
  },
  titleBar: {
      flex: 0.15,
      justifyContent: "center",
      alignItems: "center"
  },
  titleText: {
      fontSize: 30,
      fontWeight: "bold",
      color: "white"
  },
  meteorContainer: {
      flex: 0.85
  },
  listContainer: {
      backgroundColor: 'rgba(52, 52, 52, 0.5)',
      justifyContent: "center",
      marginLeft: 10,
      marginRight: 10,
      marginTop: 5,
      borderRadius: 10,
      padding: 10
  },
  cardTitle: {
     fontSize: 20,
     marginBottom: 10,
     fontWeight: "bold",
     color: "white"
  },

  cardText: {
     color: "white"
  },
  threatDetector: {
     height: 10,
     marginBottom: 10
  },
  gifContainer: {
     justifyContent: "center",
     alignItems: "center",
     flex: 1
  },
  meteorDataContainer: {
     justifyContent: "center",
     alignItems: "center",

  }
});
```

8. Run the code to check the output.



(1997 GL3)

Closest to Earth - 2021-Apr-09 14:18
Minimum Diameter (KM) - 0.4023045798
Maximum Diameter (KM) - 0.8995803882
Velocity (KM/H) - 95372.0028547211
Missing Earth by (KM) - 10389252.859564994

What's NEXT?

In the next class, we will start working on a new app called the Storytelling App. It would be a social media like app for story sharing.

EXTEND YOUR KNOWLEDGE

1. Learn and experiment with FlatList - https://reactnative.dev/docs/flatlist