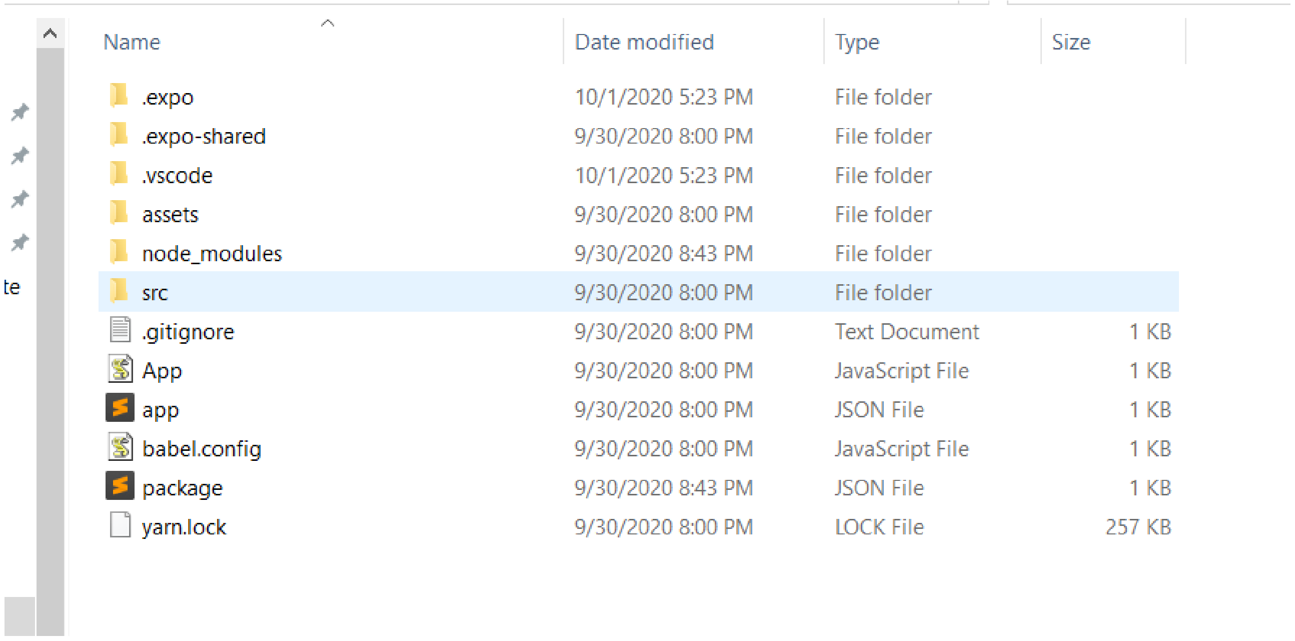
ชื่อ นายภูชิต รุ่งระวิ
รหัสนักศึกษา 61070166

Lab 6 : Core APIs

1. ให้นักศึกษาทำการสร้าง New Project โดยให้ Folder มีดังนี้ Mobile\<รหัสนักศึกษา>\Message

   Expo init <path\folder\StudentID\Project name>



Name	Date modified	Type	Size
.expo	10/1/2020 5:23 PM	File folder	
.expo-shared	9/30/2020 8:00 PM	File folder	
.vscode	10/1/2020 5:23 PM	File folder	
assets	9/30/2020 8:00 PM	File folder	
node_modules	9/30/2020 8:43 PM	File folder	
src	9/30/2020 8:00 PM	File folder	
.gitignore	9/30/2020 8:00 PM	Text Document	1 KB
App	9/30/2020 8:00 PM	JavaScript File	1 KB
app	9/30/2020 8:00 PM	JSON File	1 KB
babel.config	9/30/2020 8:00 PM	JavaScript File	1 KB
package	9/30/2020 8:43 PM	JSON File	1 KB
yarn.lock	9/30/2020 8:00 PM	LOCK File	257 KB

2. ให้นักศึกษาเขียนโปรแกรมเพื่อรับส่งข้อความและรูปภาพ (ดังแสดงตามรูปภาพข้างล่างนี้) โดยการปฏิบัติครั้งนี้ให้นักศึกษาทำการสร้าง NetInfo, StatusBar, MessageList, Alert

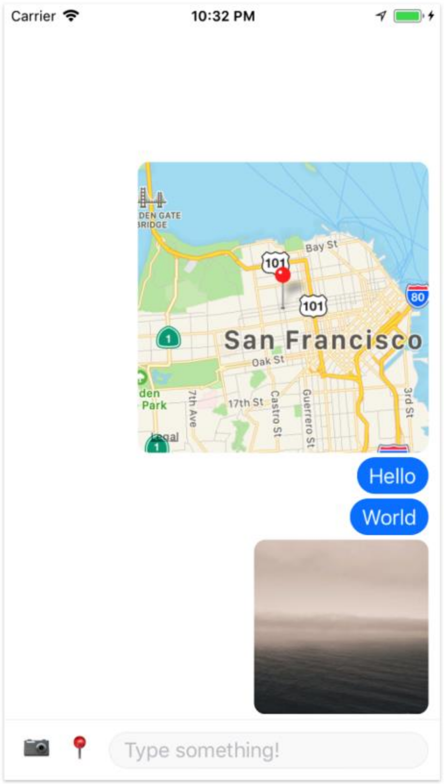
   Hint: (Component) View, Text, Image

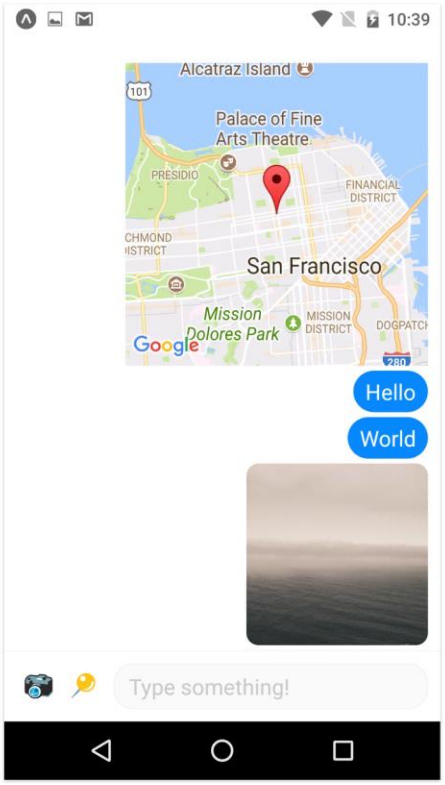
   (APIs) NetInfo, MessageList, Statusbar, Alert

ชื่อ นายภูชิต รุ่งระวิ

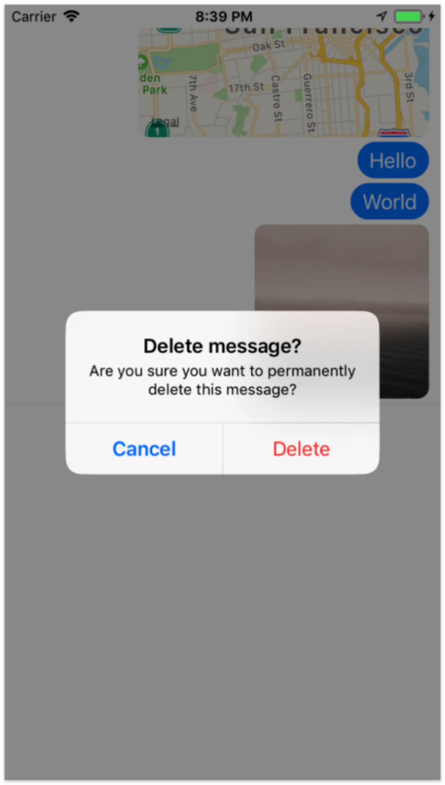
รหัสนักศึกษา 61070166

iOS **Android**





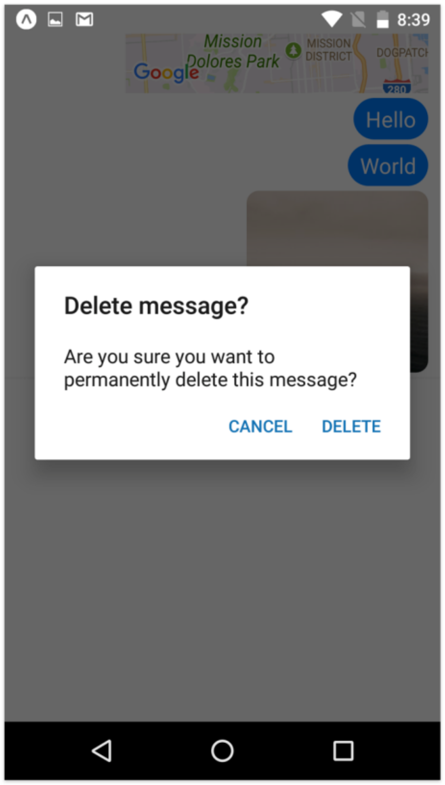
iOS **Android**



Delete message?

Are you sure you want to permanently delete this message?

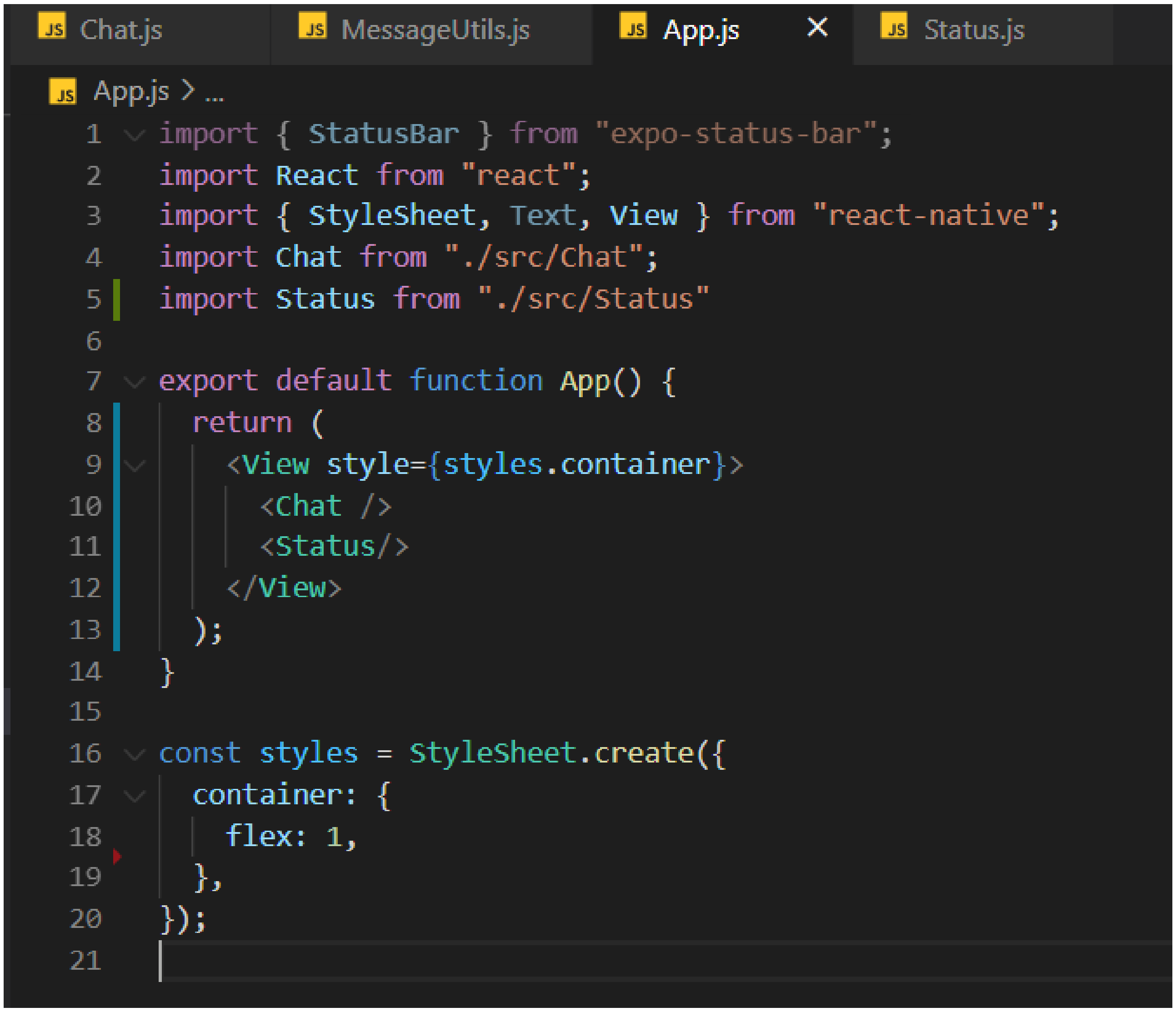
Cancel | Delete



Delete message?

Are you sure you want to permanently delete this message?

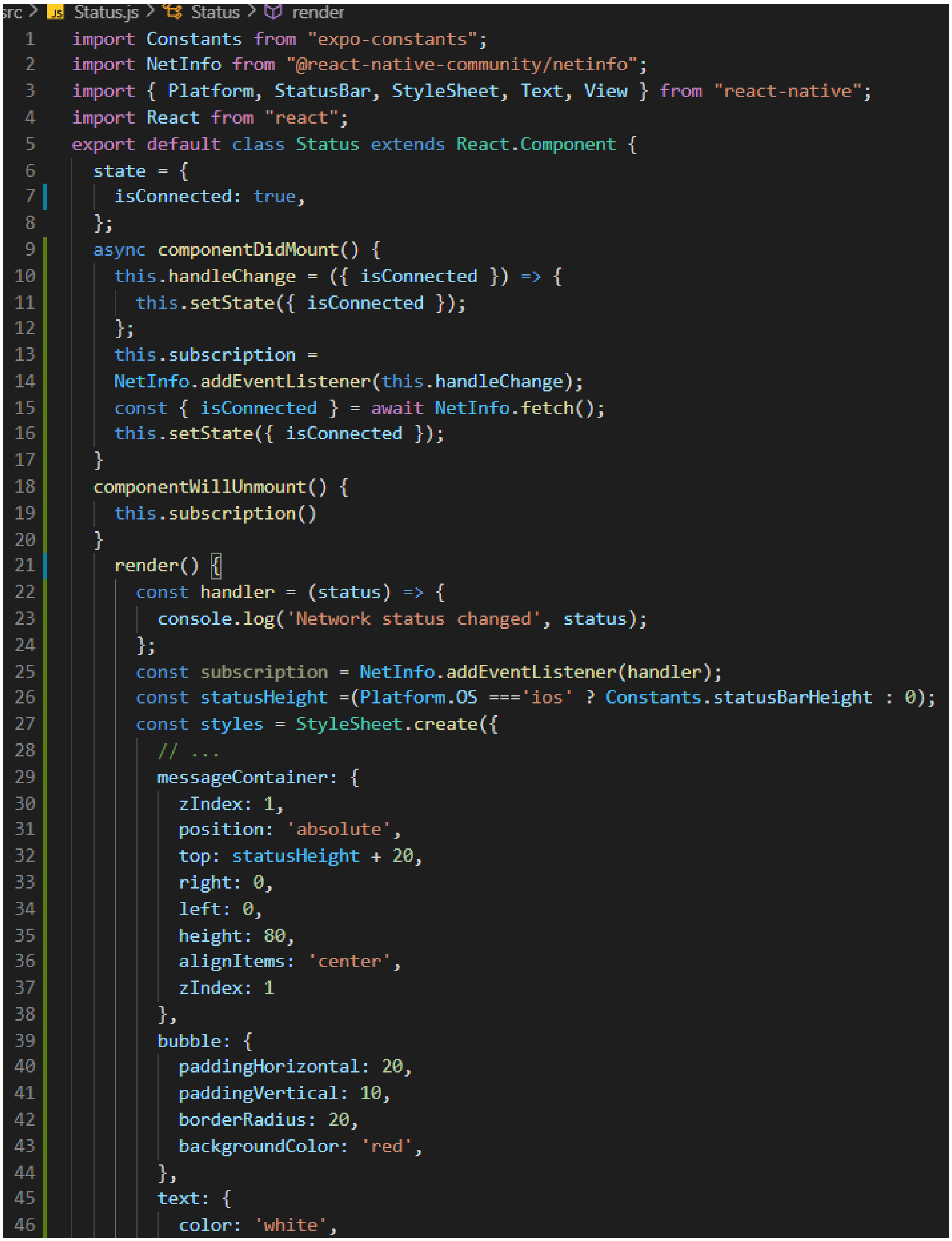
CANCEL DELETE



Chat.js | MessageUtils.js | App.js | Status.js

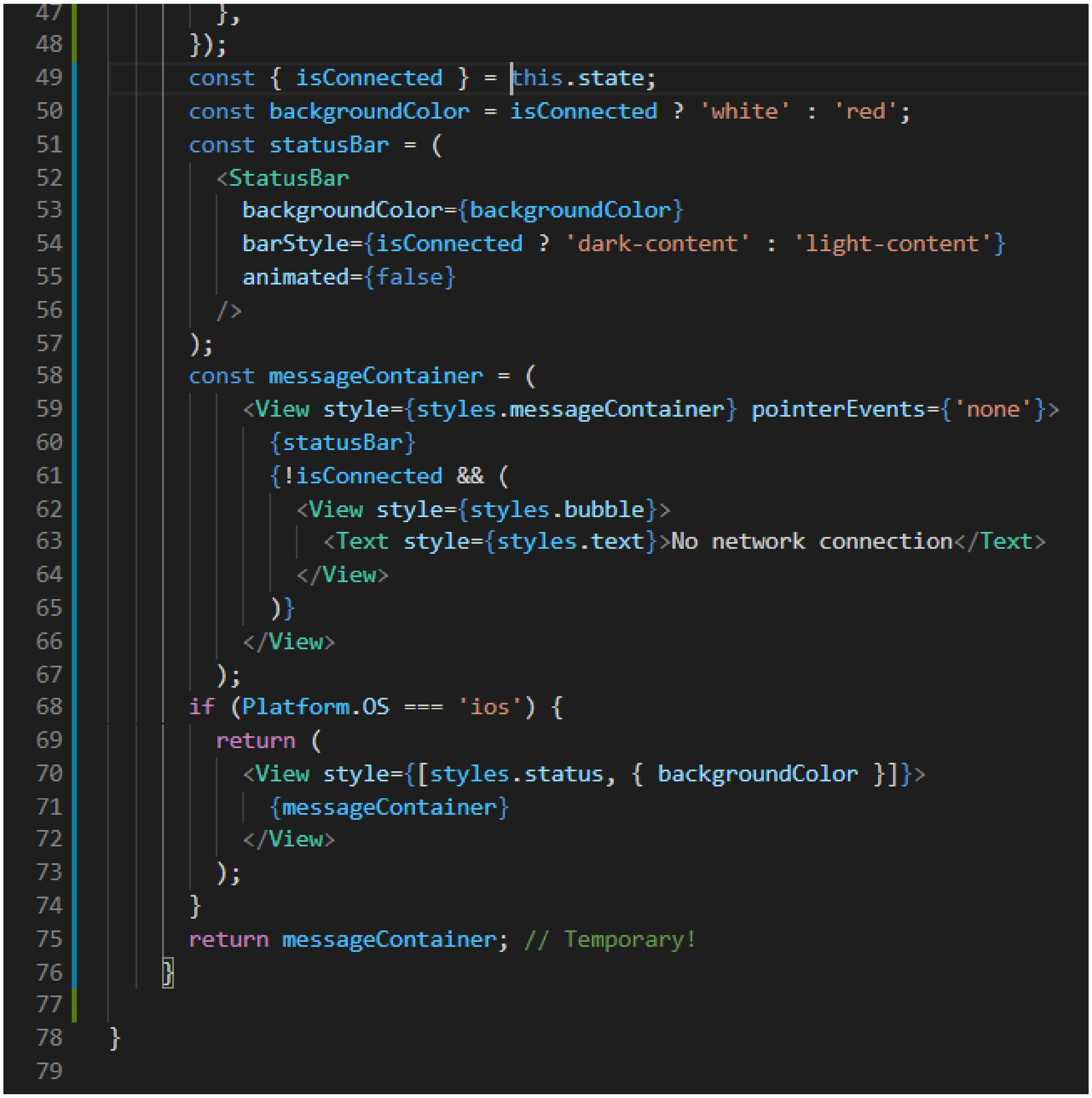
App.js > ...

```
import { StatusBar } from "expo-status-bar";
import React from "react";
import { StyleSheet, Text, View } from "react-native";
import Chat from "./src/Chat";
import Status from "./src/Status"

export default function App() {
  return (
    <View style={styles.container}>
      <Chat />
      <Status/>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
  },
});

```

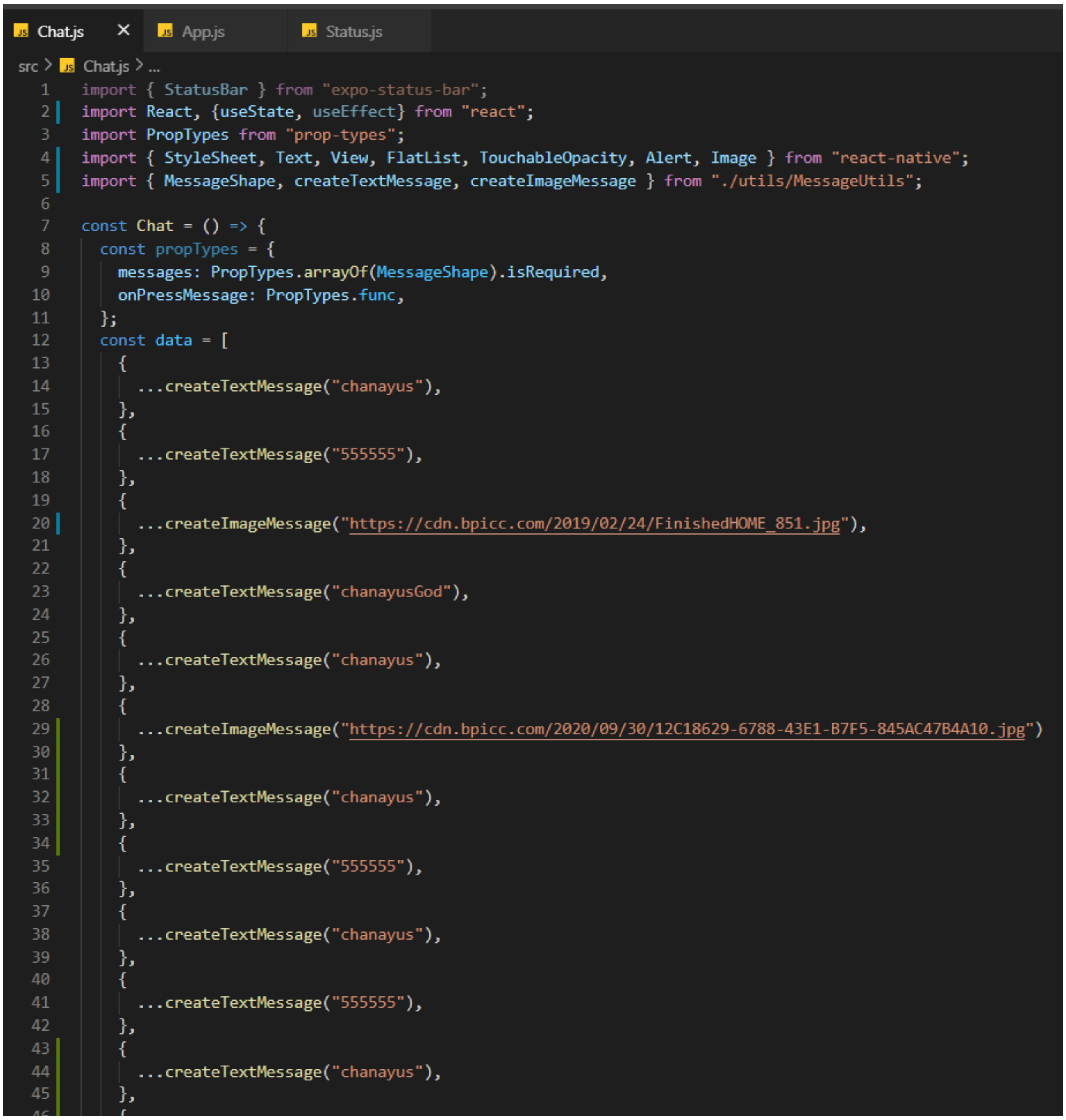


src > Status.js > Status > render

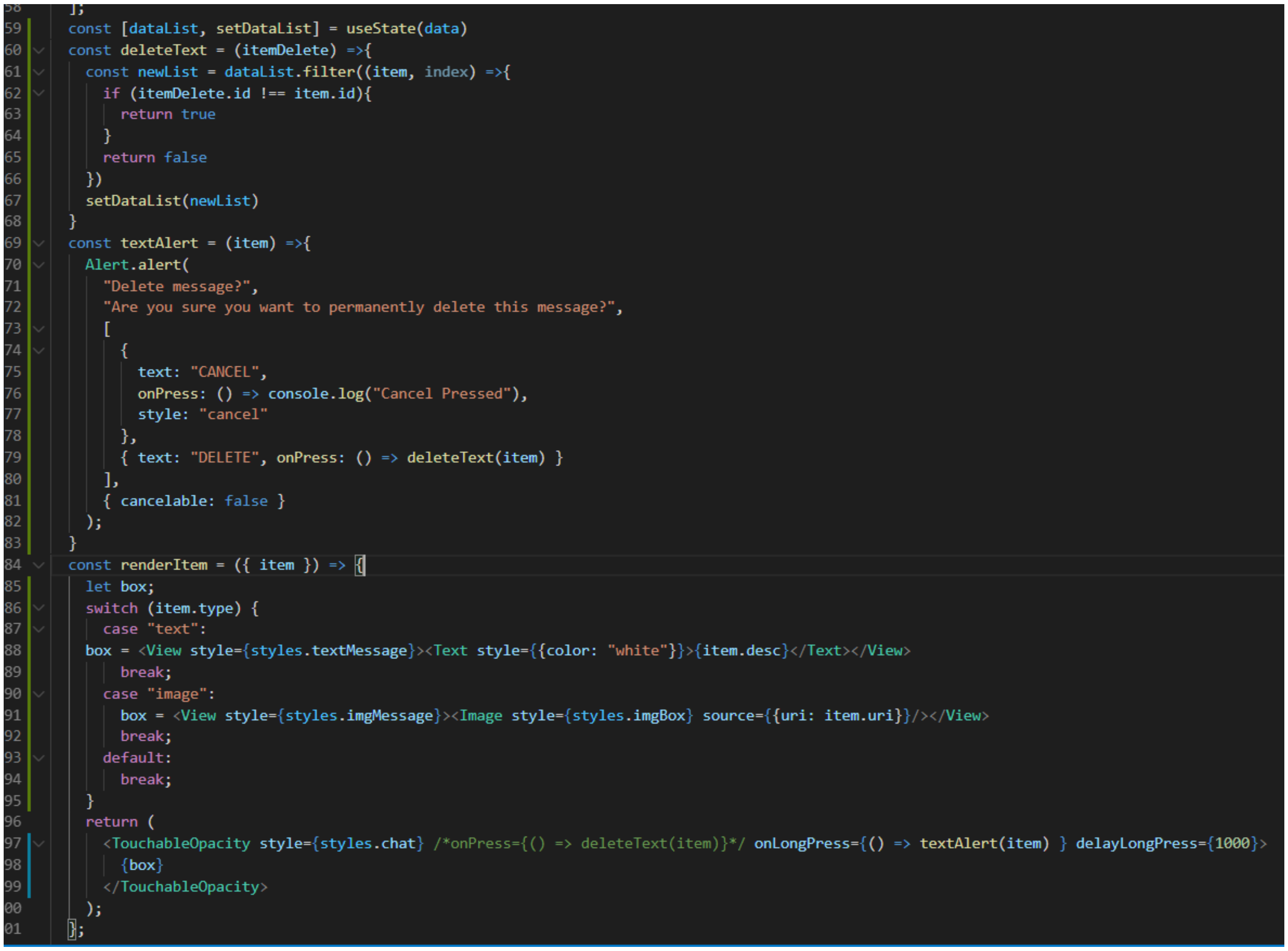
```
import Constants from "expo-constants";
import NetInfo from "@react-native-community/netinfo";
import { Platform, StatusBar, StyleSheet, Text, View } from "react-native";
import React from "react";
export default class Status extends React.Component {
  state = {
    isConnected: true,
  };
  async componentDidMount() {
    this.handleChange = ({ isConnected }) => {
      this.setState({ isConnected });
    };
    this.subscription =
    NetInfo.addEventListener(this.handleChange);
    const { isConnected } = await NetInfo.fetch();
    this.setState({ isConnected });
  }
  componentWillUnmount() {
    this.subscription()
  }
   render() {
     const handler = (status) => {
       console.log('Network status changed', status);
     };
     const subscription = NetInfo.addEventListener(handler);
     const statusHeight =(Platform.OS ==='ios' ? Constants.statusBarHeight : 0);
     const styles = StyleSheet.create({
       // ...
       messageContainer: {
         zIndex: 1,
         position: 'absolute',
         top: statusHeight + 20,
         right: 0,
         left: 0,
         height: 80,
         alignItems: 'center',
         zIndex: 1
       },
       bubble: {
         paddingHorizontal: 20,
         paddingVertical: 10,
         borderRadius: 20,
         backgroundColor: 'red',
       },
       text: {
         color: 'white',
```



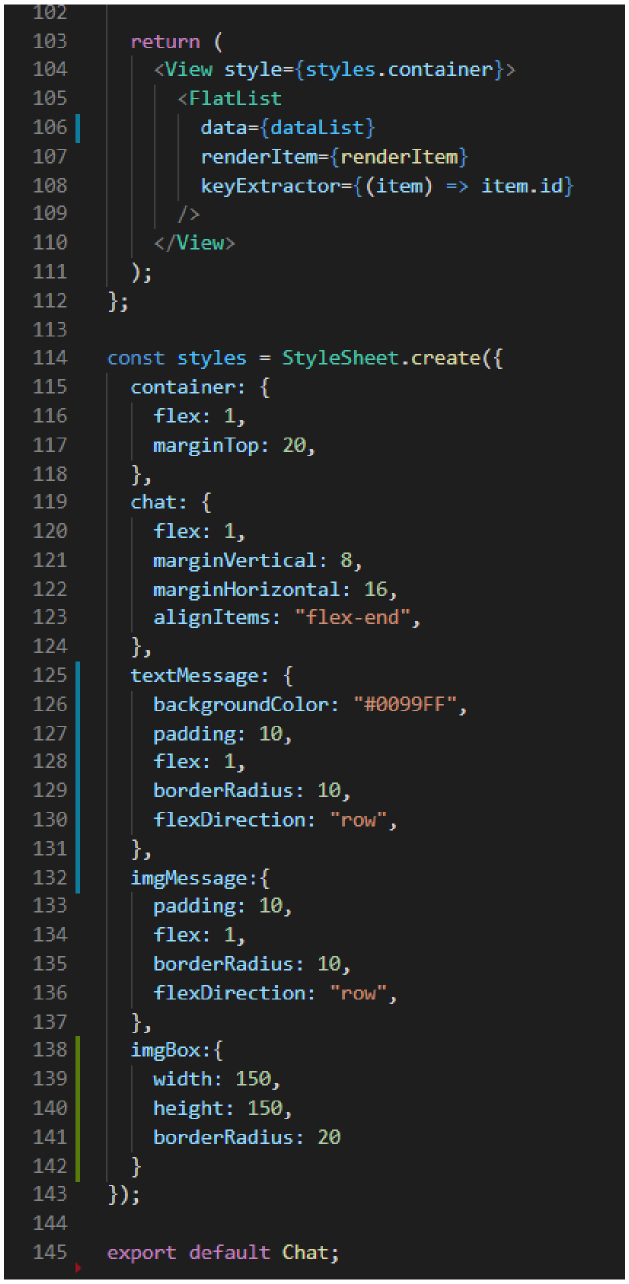
```
        },
      });
      const { isConnected } = this.state;
      const backgroundColor = isConnected ? 'white' : 'red';
      const statusBar = (
        <StatusBar
          backgroundColor={backgroundColor}
          barStyle={isConnected ? 'dark-content' : 'light-content'}
          animated={false}
        />
      );
      const messageContainer = (
        <View style={styles.messageContainer} pointerEvents={'none'}>
          {statusBar}
          {!isConnected && (
            <View style={styles.bubble}>
              <Text style={styles.text}>No network connection</Text>
            </View>
          )}
        </View>
      );
      if (Platform.OS === 'ios') {
        return (
          <View style={[styles.status, { backgroundColor }]}>
            {messageContainer}
          </View>
        );
      }
      return messageContainer; // Temporary!
    }

}

```


Chat.js × App.js Status.js
src > Chat.js > ...

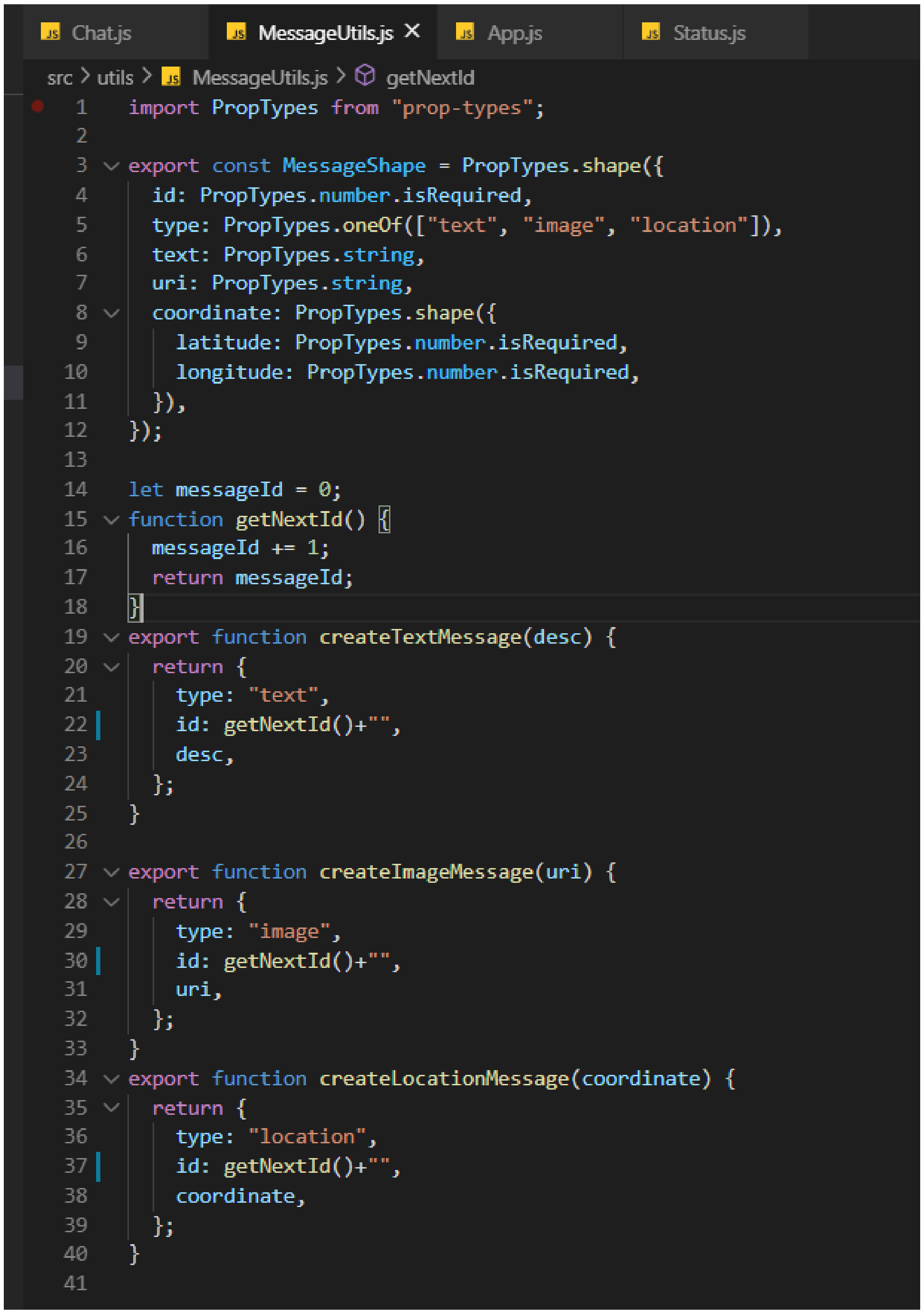
```
import { StatusBar } from "expo-status-bar";
import React, {useState, useEffect} from "react";
import PropTypes from "prop-types";
import { StyleSheet, Text, View, FlatList, TouchableOpacity, Alert, Image } from "react-native";
import { MessageShape, createTextMessage, createImageMessage } from "./utils/MessageUtils";

const Chat = () => {
  const propTypes = {
    messages: PropTypes.arrayOf(MessageShape).isRequired,
    onPressMessage: PropTypes.func,
  };
  const data = [
    {
      ...createTextMessage("chanayus"),
    },
    {
      ...createTextMessage("555555"),
    },
    {
      ...createImageMessage("https://cdn.bpicc.com/2019/02/24/FinishedHOME_851.jpg"),
    },
    {
      ...createTextMessage("chanayusGod"),
    },
    {
      ...createTextMessage("chanayus"),
    },
    {
      ...createImageMessage("https://cdn.bpicc.com/2020/09/30/12C18629-6788-43E1-B7F5-845AC47B4A10.jpg")
    },
    {
      ...createTextMessage("chanayus"),
    },
    {
      ...createTextMessage("555555"),
    },
    {
      ...createTextMessage("chanayus"),
    },
    {
      ...createTextMessage("555555"),
    },
    {
      ...createTextMessage("chanayus"),
    },
```



```
];
const [dataList, setDataList] = useState(data)
const deleteText = (itemDelete) =>{
  const newList = dataList.filter((item, index) =>{
    if (itemDelete.id !== item.id){
      return true
    }
    return false
  })
  setDataList(newList)
}
const textAlert = (item) =>{
  Alert.alert(
    "Delete message?",
    "Are you sure you want to permanently delete this message?",
    [
      {
        text: "CANCEL",
        onPress: () => console.log("Cancel Pressed"),
        style: "cancel"
      },
      { text: "DELETE", onPress: () => deleteText(item) }
    ],
    { cancelable: false }
  );
}
const renderItem = ({ item }) => {
  let box;
  switch (item.type) {
    case "text":
  box = <View style={styles.textMessage}><Text style={{color: "white"}}>{item.desc}</Text></View>
      break;
    case "image":
      box = <View style={styles.imgMessage}><Image style={styles.imgBox} source={{uri: item.uri}}/></View>
      break;
    default:
      break;
  }
  return (
    <TouchableOpacity style={styles.chat} /*onPress={() => deleteText(item)}*/ onLongPress={() => textAlert(item) } delayLongPress={1000}>
      {box}
    </TouchableOpacity>
  );
};
```



```
  return (
    <View style={styles.container}>
      <FlatList
        data={dataList}
        renderItem={renderItem}
        keyExtractor={(item) => item.id}
      />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    marginTop: 20,
  },
  chat: {
    flex: 1,
    marginVertical: 8,
    marginHorizontal: 16,
    alignItems: "flex-end",
  },
  textMessage: {
    backgroundColor: "#0099FF",
    padding: 10,
    flex: 1,
    borderRadius: 10,
    flexDirection: "row",
  },
  imgMessage:{
    padding: 10,
    flex: 1,
    borderRadius: 10,
    flexDirection: "row",
  },
  imgBox:{
    width: 150,
    height: 150,
    borderRadius: 20
  }
});

export default Chat;
```



Chat.js MessageUtils.js × App.js Status.js

src > utils > MessageUtils.js > getNextId

```
import PropTypes from "prop-types";

export const MessageShape = PropTypes.shape({
  id: PropTypes.number.isRequired,
  type: PropTypes.oneOf(["text", "image", "location"]),
  text: PropTypes.string,
  uri: PropTypes.string,
  coordinate: PropTypes.shape({
    latitude: PropTypes.number.isRequired,
    longitude: PropTypes.number.isRequired,
  }),
});

let messageId = 0;
function getNextId() {
  messageId += 1;
  return messageId;
}
export function createTextMessage(desc) {
  return {
    type: "text",
    id: getNextId()+"",
    desc,
  };
}

export function createImageMessage(uri) {
  return {
    type: "image",
    id: getNextId()+"",
    uri,
  };
}
export function createLocationMessage(coordinate) {
  return {
    type: "location",
    id: getNextId()+"",
    coordinate,
  };
}

```