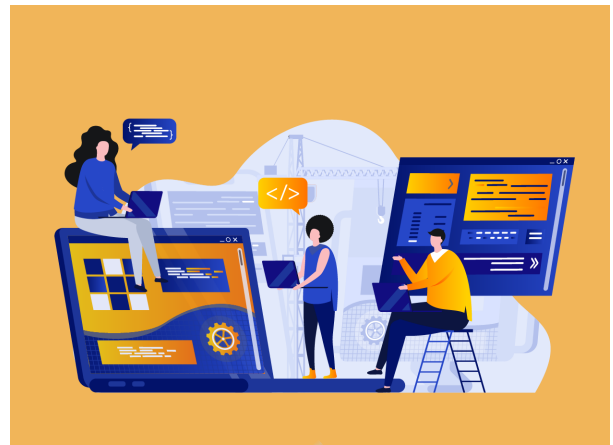


LOGOUT AND PROFILE SCREEN



What is our GOAL for this MODULE?

In this class, we completed the Logout Screen and created a Profile Screen for our App.

What did we ACHIEVE in the class TODAY?

- Added an option for the user to Logout of our App.
- Added the fonts to the screen.
- Imported firebase and wrote a function to fetch the user.
- Created a fetchUser() function which we are calling in componentDidMount() function.
- Created a function toggleSwitch().

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Logging out of an app.
- UI for the Profile screen.

How did we DO the activities?

1. Add an option for the user to **Logout** of our App, for which we create a **Logout.js** in our **screens** folder.



```

JS Logout.js
85t > screens > JS Logout.js > Logout
1  import React, { Component } from 'react';
2  import { Text, View } from 'react-native';
3  import firebase from "firebase";
4
5  export default class Logout extends Component {
6    componentDidMount() {
7      firebase.auth().signOut();
8    }
9    render() {
10     return (
11       <View
12         style={{
13           flex: 1,
14           justifyContent: "center",
15           alignItems: "center"
16         }}>
17         <Text>Logout</Text>
18       </View>
19     )
20   }
21
  
```

2. Create a **Logout.js** file in which the **componentDidMount()** function will use **firebase.auth().signOut()**, which will sign out the user from our firebase app. Therefore, the user will then not see the screen and will directly be logged out and navigated to the Login Screen instead.

Import and add **Logout.js** inside the Drawer Navigation.



```

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

const Drawer = createDrawerNavigator();
  
```

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

export default DrawerNavigator;
```

Now, to create the UI for the Profile screen:

- Fetch the user's details from **firebase** to display their name, profile image, and their preferred theme (dark by default) as follows:
 - Import **firebase** and write a function to fetch the user.

```
import firebase from "firebase";
```

- Create **constructor()** to declare the properties. Create a property **isEnabled** in the constructor, for creating a toggle switch to toggle between themes. Also, have **light_theme** set to **true** by default.

```
export default class Profile extends Component {
  constructor(props) {
    super(props);
    this.state = {
      fontsLoaded: false,
      isEnabled: false,
      light_theme: true,
      name: ""
    };
  }
}
```

- Create a **fetchUser()** function which call in our **componentDidMount()** function. This function fetches the user information using **firebase.auth().currentUser.uid**.

```

componentDidMount() {
  this._loadFontsAsync();
  this.fetchUser();
}

async fetchUser() {
  let theme, name, image;
  await firebase
    .database()
    .ref("/users/" + firebase.auth().currentUser.uid)
    .on("value", function (snapshot) {
      theme = snapshot.val().current_theme;
      name = `${snapshot.val().first_name} ${snapshot.val().last_name}`;
    });
  this.setState({
    light_theme: theme === "light" ? true : false,
    isEnabled: theme === "light" ? false : true,
    name: name
  });
}

```

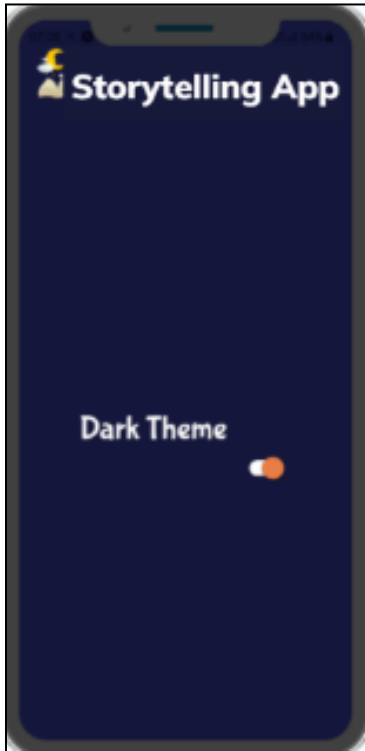
5. Use the profile image and name of the user. Use the **<Switch>** component to toggle between themes. In this component, the attribute **trackColor** is for the color of the **trackbackground** when it's true or false. Similarly, **thumbColor** is the color of the circle on the switch. In the **<Switch>**, use a function **toggleSwitch()** for the **onPress** event.

```

toggleSwitch() {
  const previous_state = this.state.isEnabled;
  const theme = !this.state.isEnabled ? "dark" : "light"
  var updates = {}
  updates["/users/" + firebase.auth().currentUser.uid + "/current_theme"]
= theme
  firebase.database().ref().update(updates);
  this.setState({ isEnabled: !previous_state, light_theme: previous_state
  })
};

```

Output:



What's next?

In the next class, we will be building and then integrating the light theme into our app, so that the user can choose between the themes they prefer.

Expand your knowledge:

Bookmark the following link: to know more about App-Loading

<https://docs.expo.io/versions/latest/sdk/app-loading/>