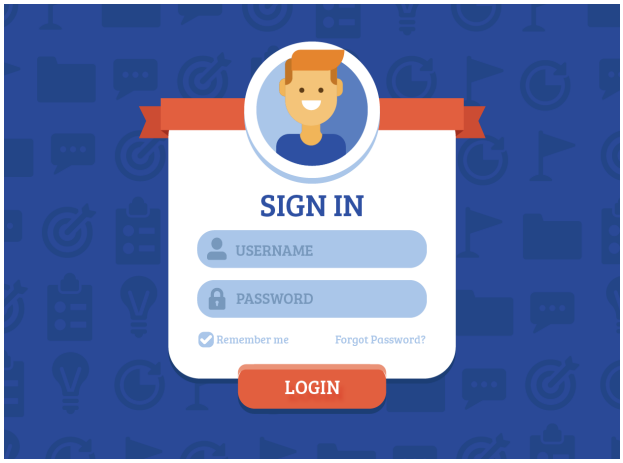
LOGIN AND PROFILE SCREEN



What is our GOAL for this MODULE?

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

What did we ACHIEVE in the class TODAY?

- Improved the User Interface(UI) of the Login Screen.
- Worked on the render() function.
- 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?

- UI for the Login Screen.
- UI for the Profile screen.

How did we DO the activities?

1. To create the UI of Login Screen as follows:
    - Import fonts.

```
import AppLoading from 'expo-app-loading';
import * as Font from 'expo-font';
```

    - Create a variable **customFonts** to hold the required fonts.

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

    - Create a **constructor()** to define the state property, make sure to pass **props** as an argument; as we are using the navigator.

```
constructor(props) {
    super(props);
    this.state = {
        fontsLoaded: false
    };
}

async _loadFontsAsync() {
    await Font.loadAsync(customFonts);
    this.setState({ fontsLoaded: true });
}

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

    - Use the **If-else** condition to load the font and output.

```
render() {
    if (!this.state.fontsLoaded) {
        return <AppLoading />;
    } else {
        return (
            <View
                style={{
                    flex: 1,
                    justifyContent: "center",
                    alignItems: "center"
                }}>
                <Button
                    title="Sign in with Google"
                    onPress={() => this.signInWithGoogleAsync()}></Button>
            </View>
        )
    }
}
```

2. Work on the **render()** function to make it look like the UI we want. We have the **<TouchableOpacity>** component containing our button's code with Google's Icon and Text. We have called our **signInWithGoogleAsync()** function we created in the previous class on the **onPress** event of our **<TouchableOpacity>**.
   Next, have our cloud image inside a container with styles **cloudContainer**.

```
return (
        <View style={styles.container}>
          <SafeAreaView style={styles.droidSafeArea} />
          <View style={styles.appTitle}>
            <Image
              source={require("../assets/logo.png")}
              style={styles.appIcon}
            ></Image>
            <Text style={styles.appTitleText}>{`Story
Telling\nApp`}</Text>
          </View>
          <View style={styles.buttonContainer}>
            <TouchableOpacity
              style={styles.button}
```

```
            onPress={() => this.signInWithGoogleAsync()}
          >
            <Image
              source={require("../assets/google_icon.png")}
              style={styles.googleIcon}
            ></Image>
            <Text style={styles.googleText}>Sign in with
Google</Text>
          </TouchableOpacity>
        </View>
        <View style={styles.cloudContainer}>
          <Image
            source={require("../assets/cloud.png")}
            style={styles.cloudImage}
          ></Image>
        </View>
      </View>
    );
```

Output::

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

4. 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:

5. 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.

```
constructor(props) {
    super(props);
    this.state = {
        fontsLoaded: false,
        isEnabled: false,
        light_theme: true,
        profile_image: "",
        name: ""
    };
}
```

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

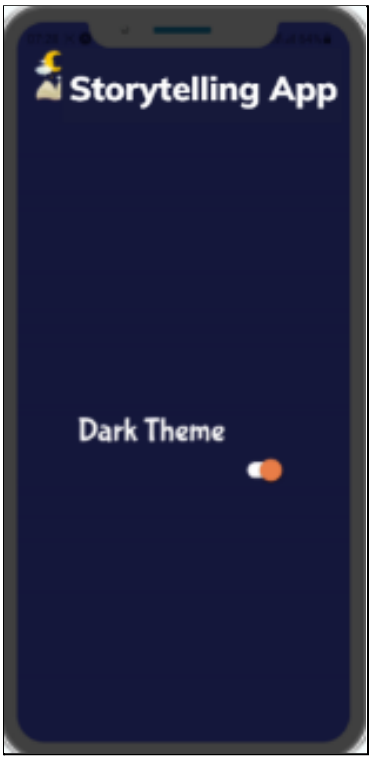
```
componentDidMount() {
    this._loadFontsAsync();
    this.fetchUser();
}

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}`
            image = snapshot.val().profile_picture
        })
    this.setState({
        light_theme: theme === "light" ? true : false,
        isEnabled: theme === "light" ? false : true,
        name: name,
        profile_image: image
    })
}
```

7. 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/