AUTHENTICATING USERS

What is our GOAL for this MODULE?

In this class, you learned to create an authentication page for our users, so that only teachers whose email IDs and passwords are registered with us can log in and access our app. Others will not be able to use our application.

What did we ACHIEVE in the class TODAY?

- Created an authentication page for the user to authenticate.
- Used the firebase authentication service to authenticate a user.
- Modified security rules in the firebase database to allow only authenticated users to access the database.

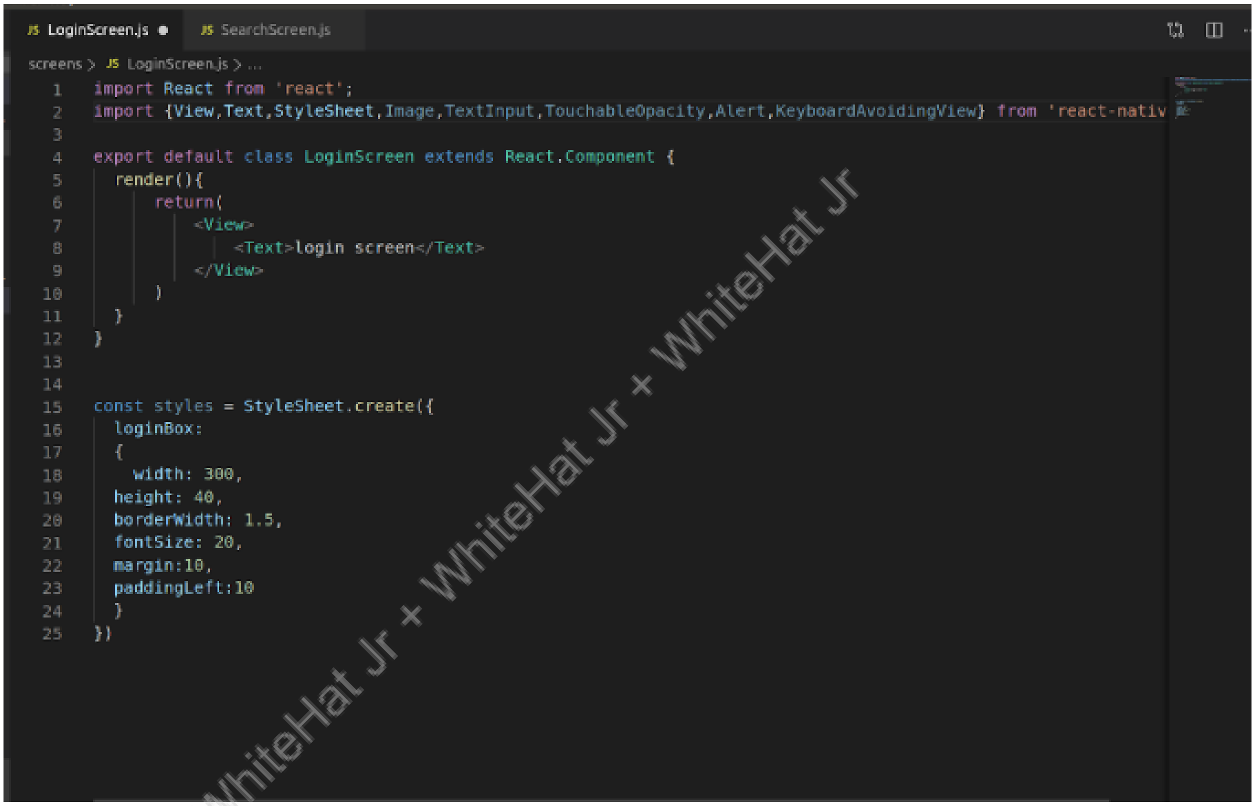
Which CONCEPTS/CODING BLOCKS did we cover today?

- Firebase Authentication
- Modify firebase rules

How did we DO the activities?

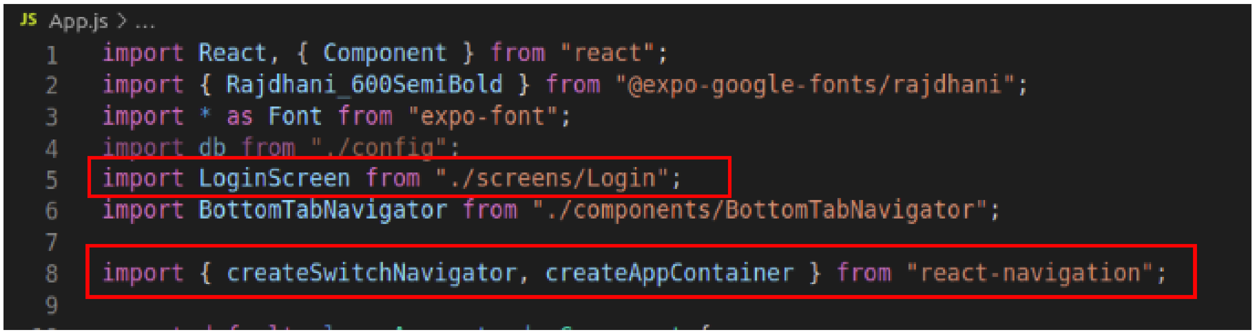
Currently, our app could be used by any user as the app did not have any authentication that only allows teachers to access it. To fix this you decided to add a login page that allows only the teacher to log in to the app.

1. Create a basic login screen component using react components.

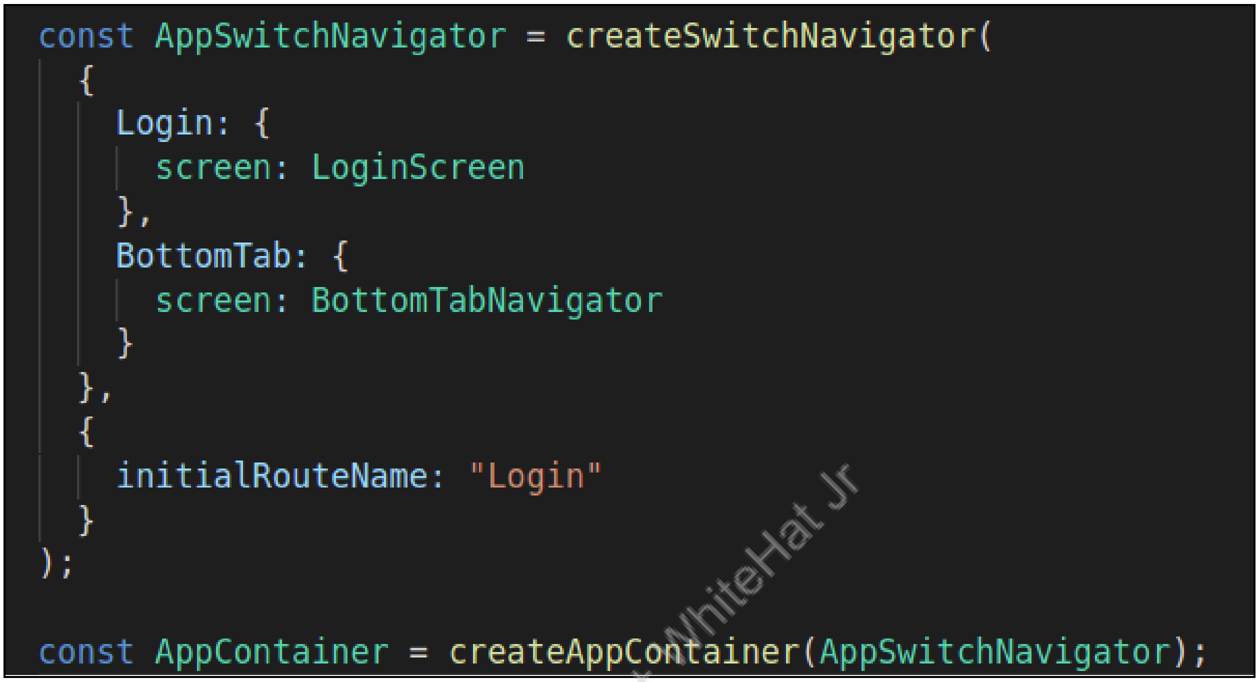


```
import React from 'react';
import {View,Text,StyleSheet,Image,TextInput,TouchableOpacity,Alert,KeyboardAvoidingView} from 'react-nativ

export default class LoginScreen extends React.Component {
  render(){
    return(
      <View>
        <Text>login screen</Text>
      </View>
    )
  }
}


const styles = StyleSheet.create({
  loginBox:
  {
    width: 300,
  height: 40,
  borderWidth: 1.5,
  fontSize: 20,
  margin:10,
  paddingLeft:10
  }
})
```

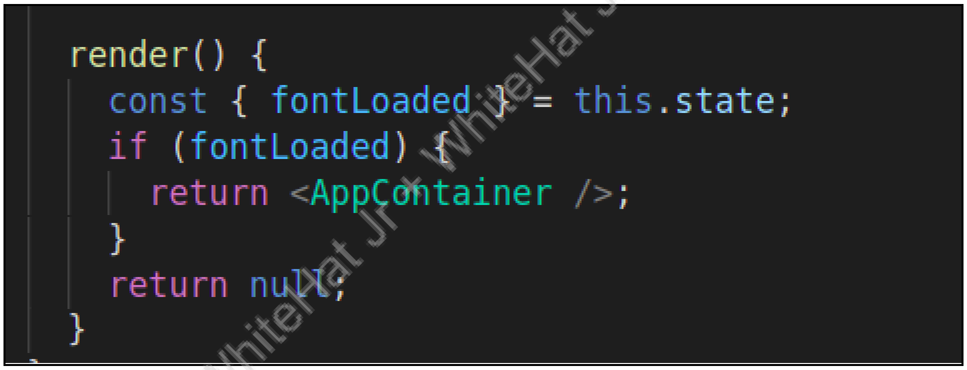
2. Create a **SwitchNavigator** containing **TabNavigator** and a Login Screen.



```
import React, { Component } from "react";
import { Rajdhani_600SemiBold } from "@expo-google-fonts/rajdhani";
import * as Font from "expo-font";
import db from "./config";
import LoginScreen from "./screens/Login";
import BottomTabNavigator from "./components/BottomTabNavigator";

import { createSwitchNavigator, createAppContainer } from "react-navigation";

```

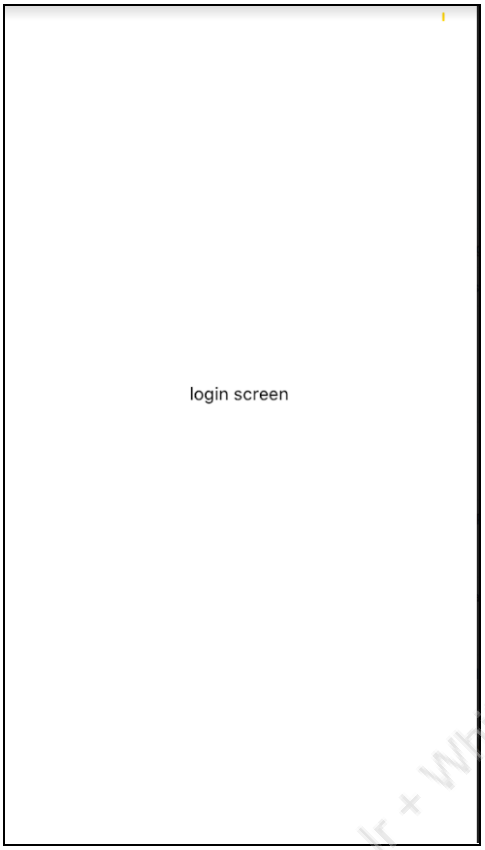


```
const AppSwitchNavigator = createSwitchNavigator(
  {
    Login: {
      screen: LoginScreen
    },
    BottomTab: {
      screen: BottomTabNavigator
    }
  },
  {
    initialRouteName: "Login"
  }
);

const AppContainer = createAppContainer(AppSwitchNavigator);
```



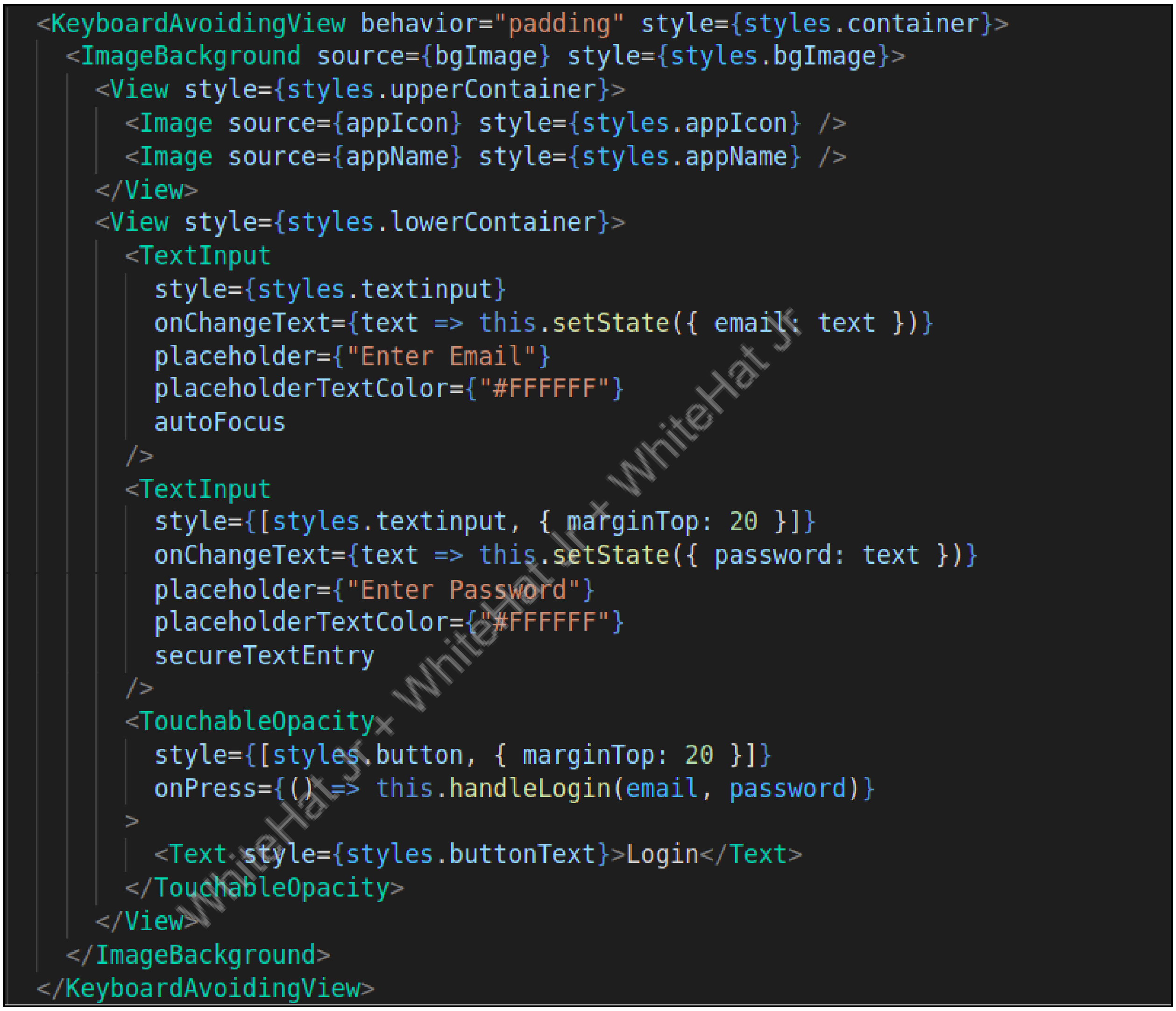
```
  render() {
    const { fontLoaded } = this.state;
    if (fontLoaded) {
      return <AppContainer />;
    }
    return null;
  }
```

3. Output:

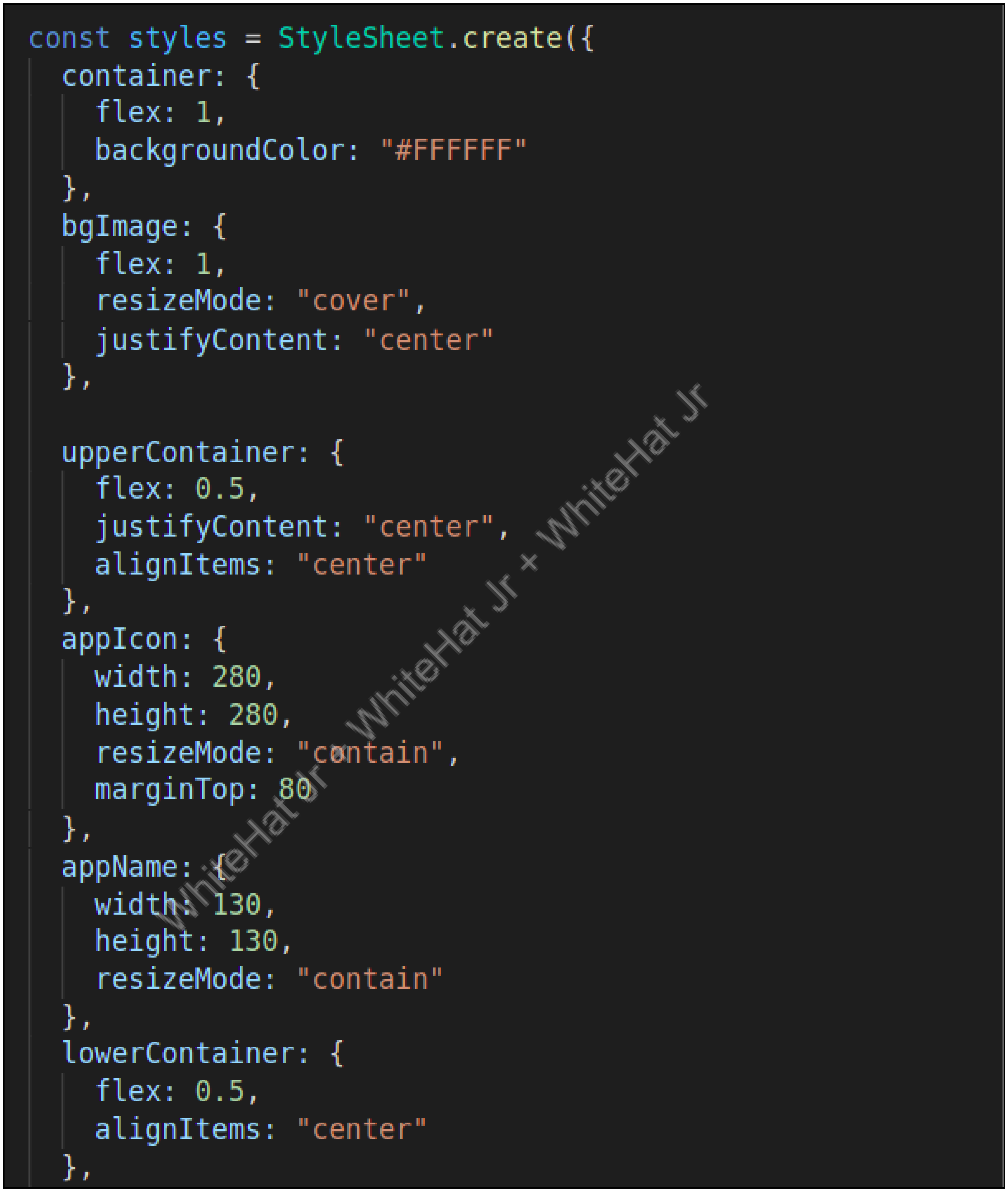


login screen

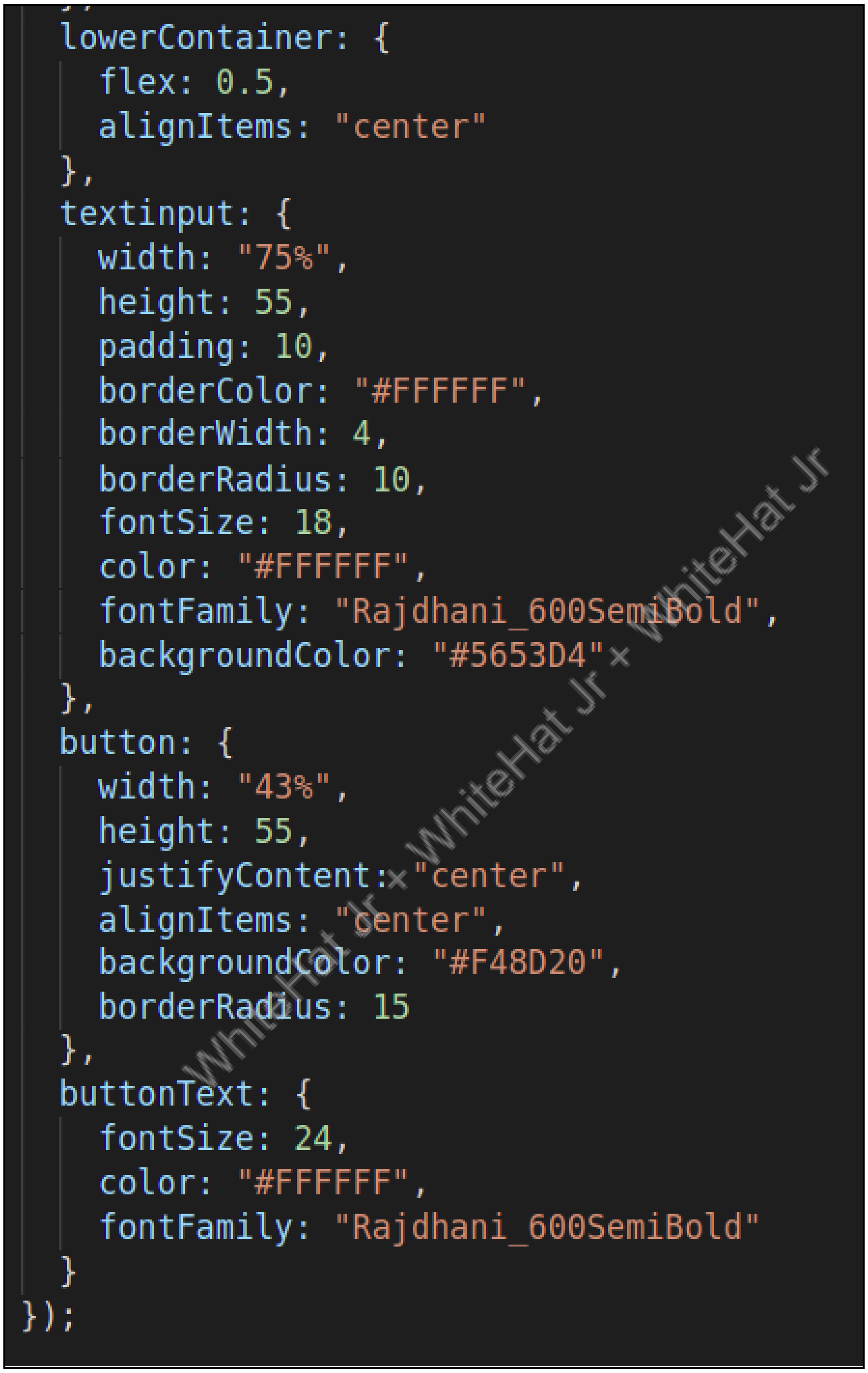
4. Add two text inputs for email, password, and a touchable opacity button. Add styles to these inputs.



```
<KeyboardAvoidingView behavior="padding" style={styles.container}>
  <ImageBackground source={bgImage} style={styles.bgImage}>
    <View style={styles.upperContainer}>
      <Image source={appIcon} style={styles.appIcon} />
      <Image source={appName} style={styles.appName} />
    </View>
    <View style={styles.lowerContainer}>
      <TextInput
        style={styles.textinput}
        onChangeText={text => this.setState({ email: text })}
        placeholder={"Enter Email"}
        placeholderTextColor={"#FFFFFF"}
        autoFocus
      />
      <TextInput
        style={[styles.textinput, { marginTop: 20 }]}
        onChangeText={text => this.setState({ password: text })}
        placeholder={"Enter Password"}
        placeholderTextColor={"#FFFFFF"}
        secureTextEntry
      />
      <TouchableOpacity
        style={[styles.button, { marginTop: 20 }]}
        onPress={() => this.handleLogin(email, password)}
      >
        <Text style={styles.buttonText}>Login</Text>
      </TouchableOpacity>
    </View>
  </ImageBackground>
</KeyboardAvoidingView>
```

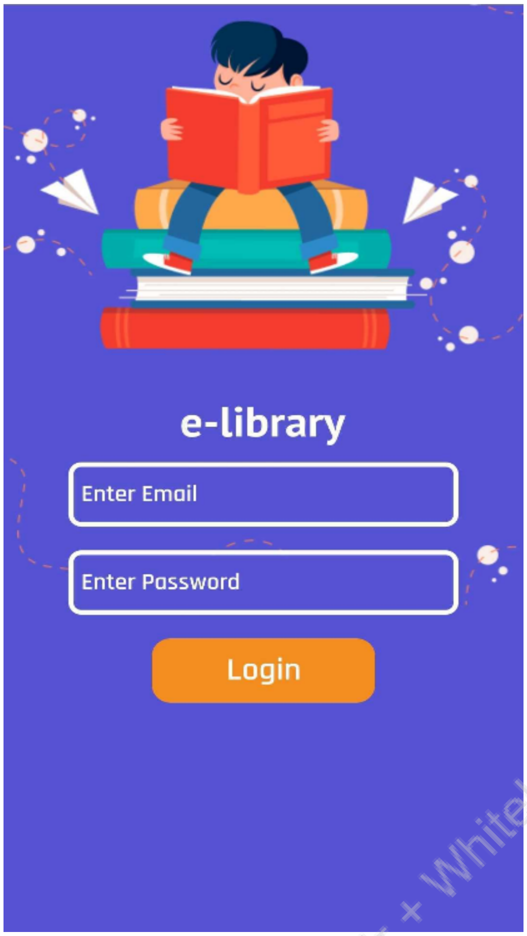


```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#FFFFFF"
  },
  bgImage: {
    flex: 1,
    resizeMode: "cover",
    justifyContent: "center"
  },

  upperContainer: {
    flex: 0.5,
    justifyContent: "center",
    alignItems: "center"
  },
  appIcon: {
    width: 280,
    height: 280,
    resizeMode: "contain",
    marginTop: 80
  },
  appName: {
    width: 130,
    height: 130,
    resizeMode: "contain"
  },
  lowerContainer: {
    flex: 0.5,
    alignItems: "center"
  },
```



```
  },
  lowerContainer: {
    flex: 0.5,
    alignItems: "center"
  },
  textinput: {
    width: "75%",
    height: 55,
    padding: 10,
    borderColor: "#FFFFFF",
    borderWidth: 4,
    borderRadius: 10,
    fontSize: 18,
    color: "#FFFFFF",
    fontFamily: "Rajdhani_600SemiBold",
    backgroundColor: "#5653D4"
  },
  button: {
    width: "43%",
    height: 55,
    justifyContent: "center",
    alignItems: "center",
    backgroundColor: "#F48D20",
    borderRadius: 15
  },
  buttonText: {
    fontSize: 24,
    color: "#FFFFFF",
    fontFamily: "Rajdhani_600SemiBold"
  }
});
```

Output:



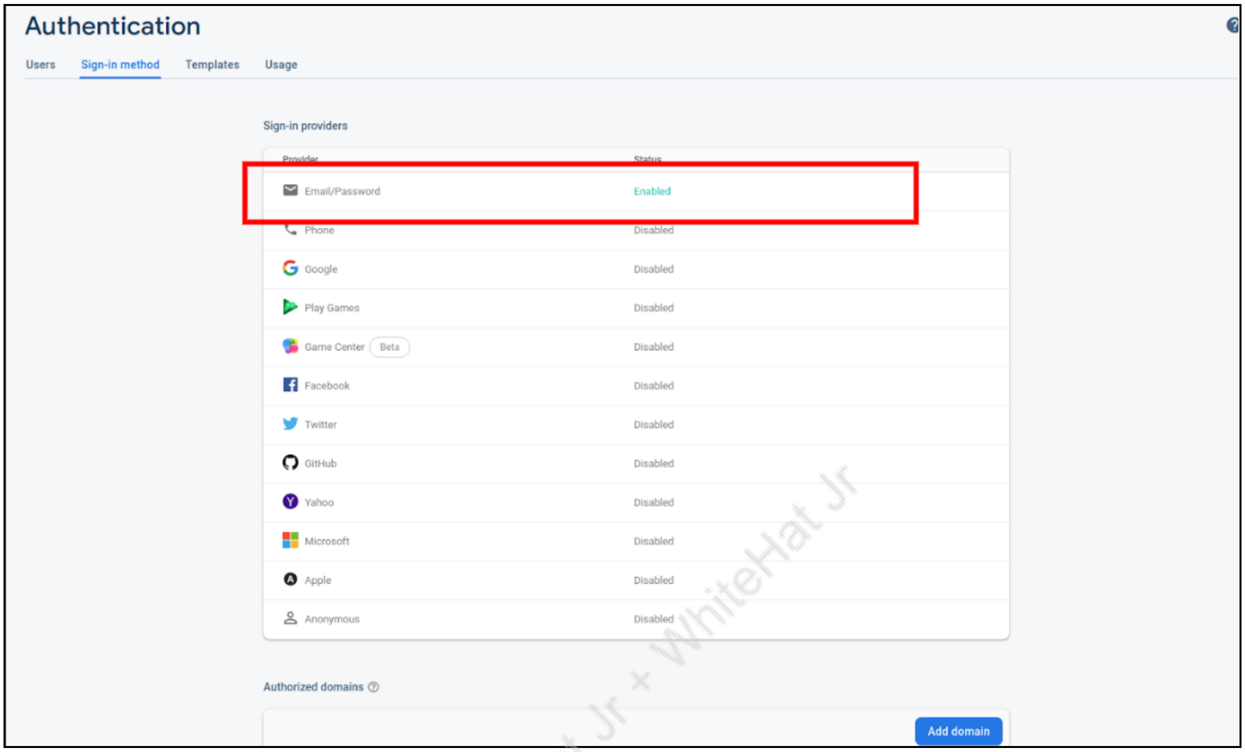
e-library

Enter Email

Enter Password

Login

5. Enable the **firebase** authorization service for email sign-ins.



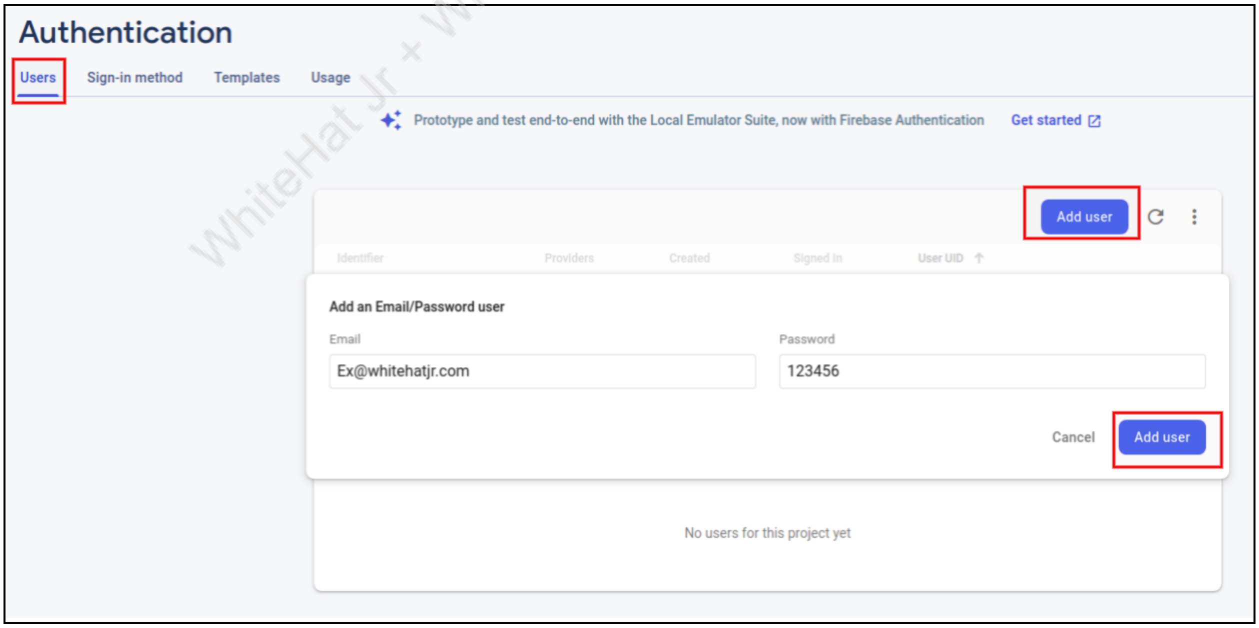
Authentication

Sign-in providers

Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center Beta	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Yahoo	Disabled
Microsoft	Disabled
Apple	Disabled
Anonymous	Disabled

Authorized domains

6. Create a registered email ID and password for testing.



Authentication

Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication Get started

Identifier	Providers	Created	Signed In	User UID

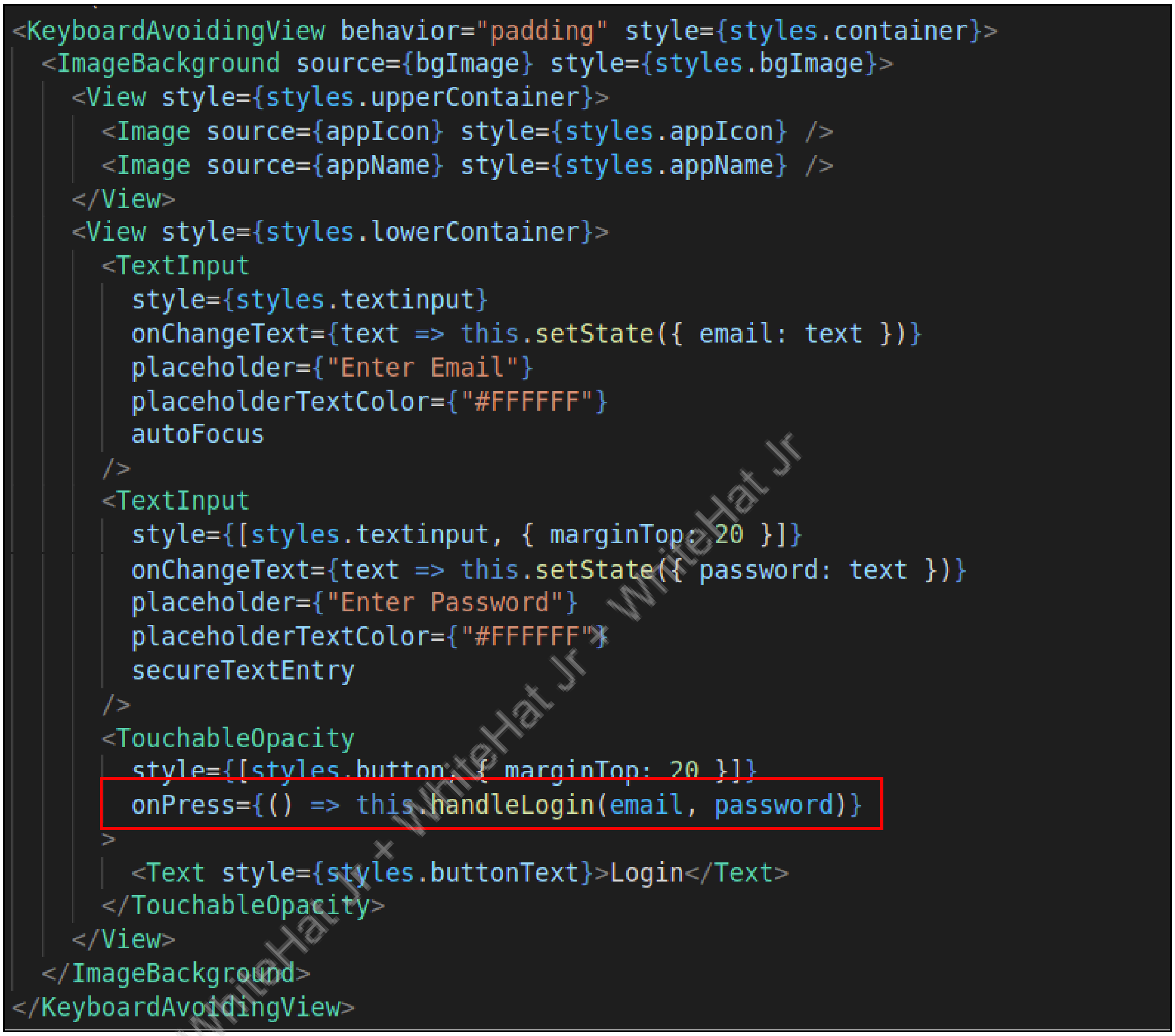
Add an Email/Password user

Email: Ex@whitehatjr.com

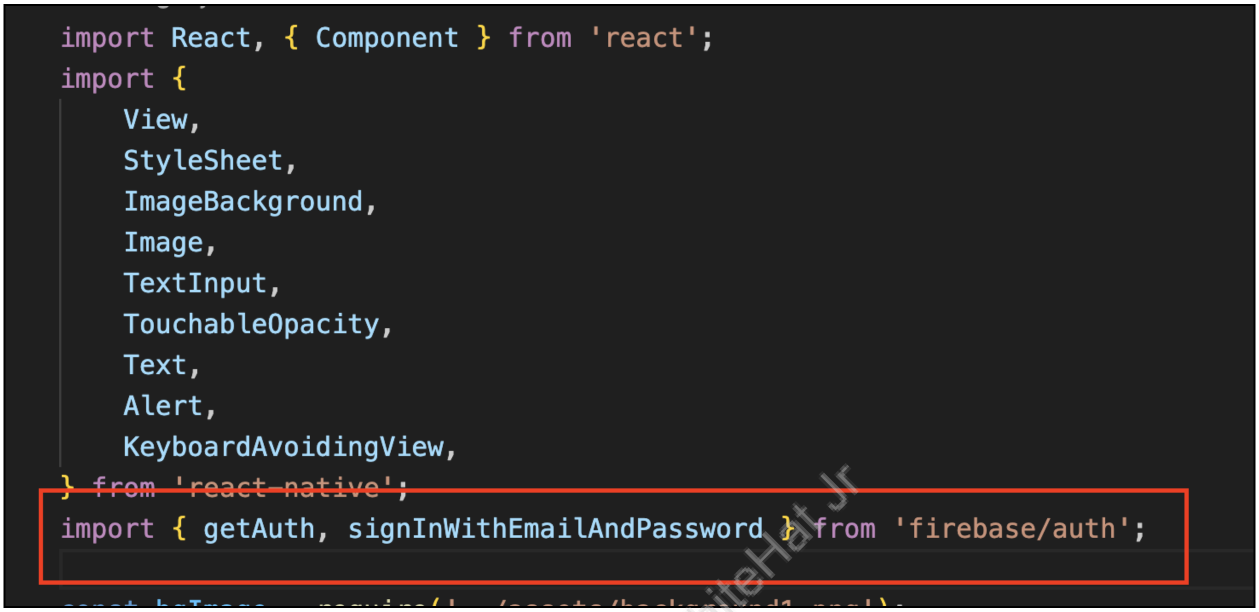
Password: 123456

No users for this project yet

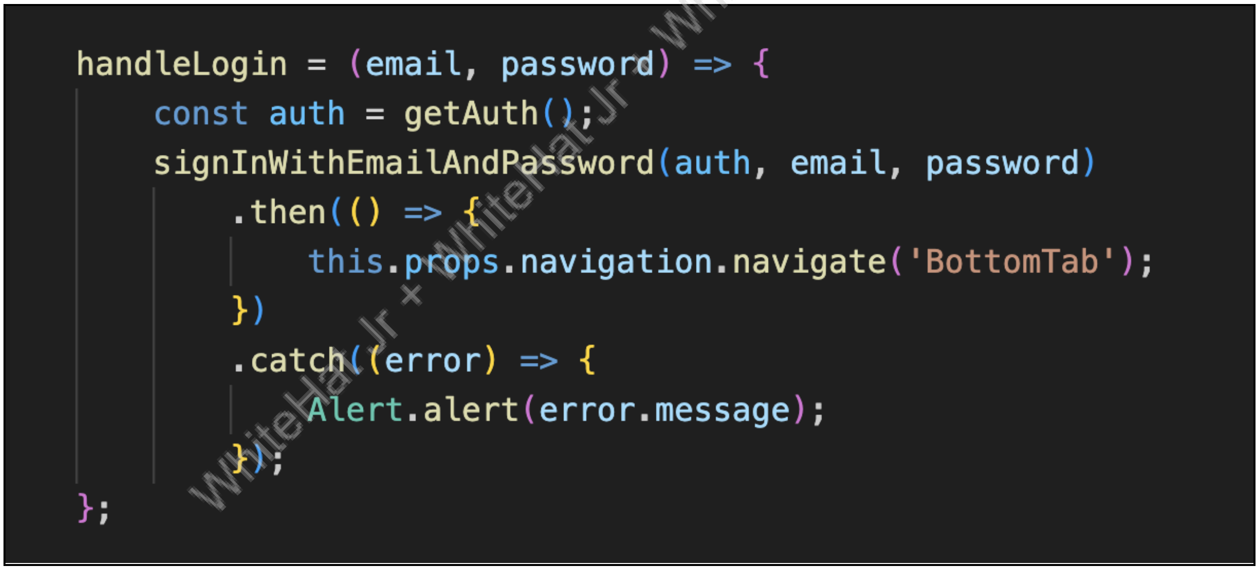
7. Write the code for login functions that are called when a user presses the login button.



```
<KeyboardAvoidingView behavior="padding" style={styles.container}>
  <ImageBackground source={bgImage} style={styles.bgImage}>
    <View style={styles.upperContainer}>
      <Image source={appIcon} style={styles.appIcon} />
      <Image source={appName} style={styles.appName} />
    </View>
    <View style={styles.lowerContainer}>
      <TextInput
        style={styles.textinput}
        onChangeText={text => this.setState({ email: text })}
        placeholder={"Enter Email"}
        placeholderTextColor={"#FFFFFF"}
        autoFocus
      />
      <TextInput
        style={[styles.textinput, { marginTop: 20 }]}
        onChangeText={text => this.setState({ password: text })}
        placeholder={"Enter Password"}
        placeholderTextColor={"#FFFFFF"}
        secureTextEntry
      />
      <TouchableOpacity
        style={[styles.button, { marginTop: 20 }]}
        onPress={() => this.handleLogin(email, password)}
      >
        <Text style={styles.buttonText}>Login</Text>
      </TouchableOpacity>
    </View>
  </ImageBackground>
</KeyboardAvoidingView>
```

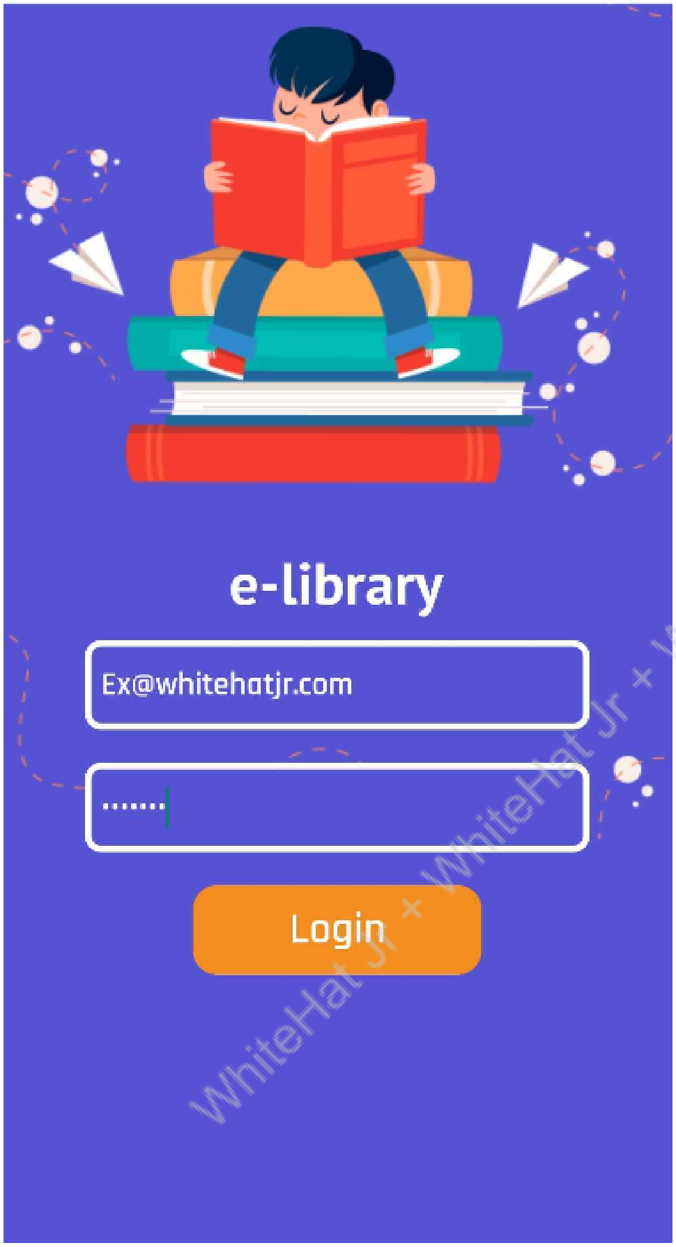


```
import React, { Component } from 'react';
import {
    View,
    StyleSheet,
    ImageBackground,
    Image,
    TextInput,
    TouchableOpacity,
    Text,
    Alert,
    KeyboardAvoidingView,
} from 'react-native';
import { getAuth, signInWithEmailAndPassword } from 'firebase/auth';

const bgImage = require('../assets/background1.png');
```

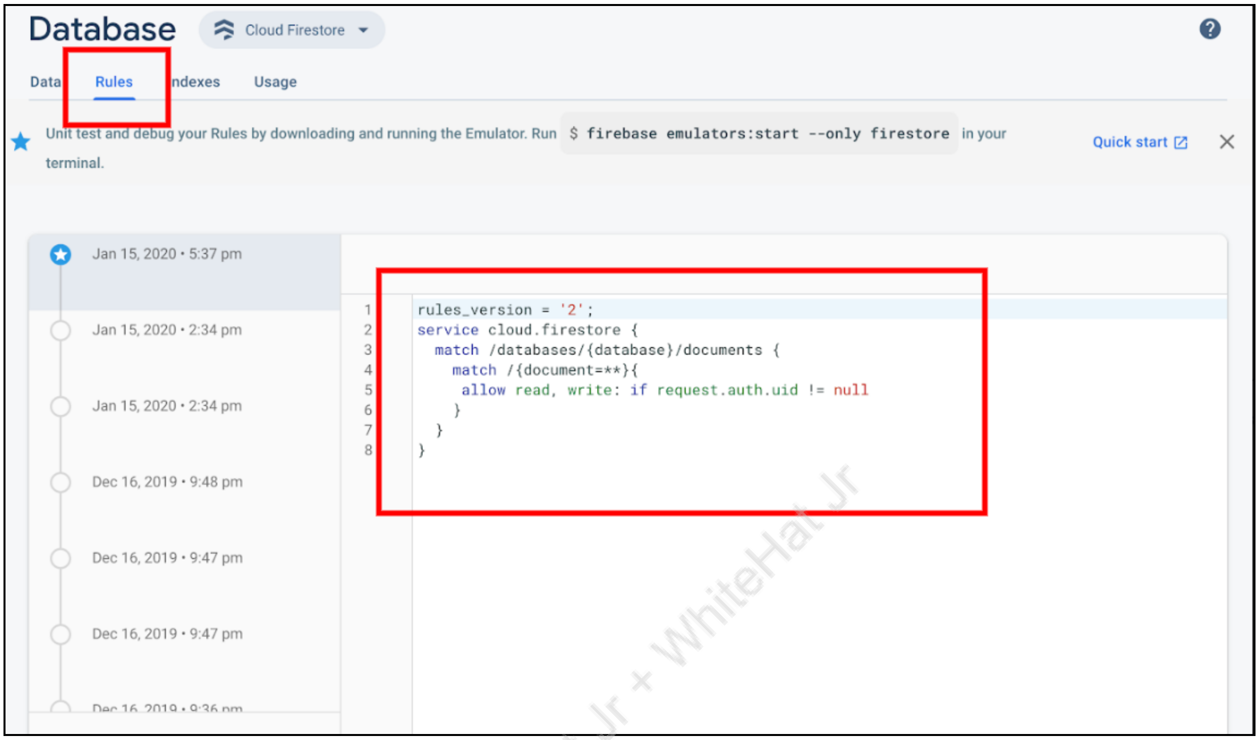


```
handleLogin = (email, password) => {
    const auth = getAuth();
    signInWithEmailAndPassword(auth, email, password)
        .then(() => {
            this.props.navigation.navigate('BottomTab');
        })
        .catch((error) => {
            Alert.alert(error.message);
        });
};
```

8. Output:



9. Modify the security rules to allow only authenticated users to access the database.

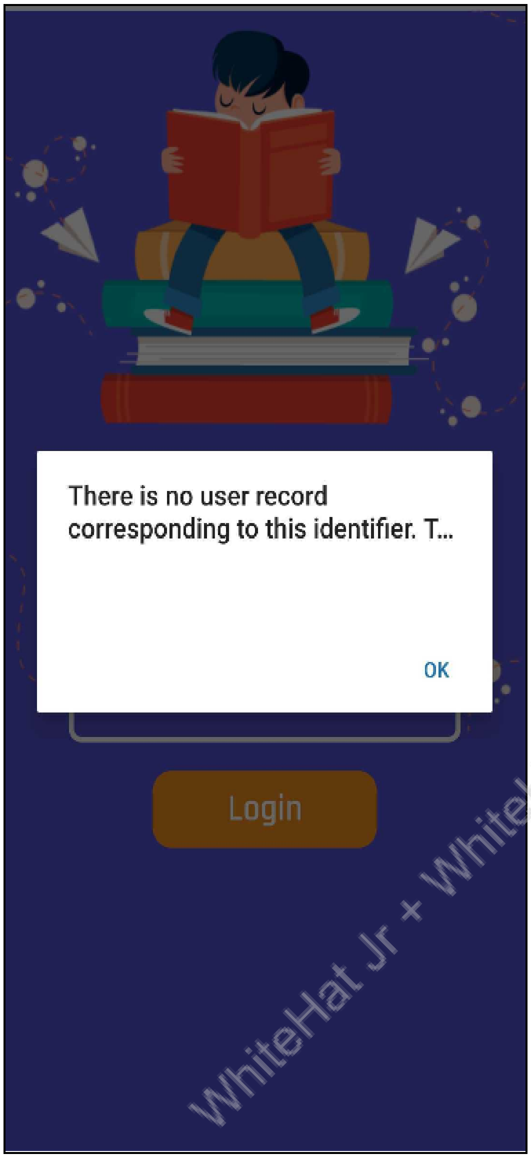


Unit test and debug your Rules by downloading and running the Emulator. Run `$ firebase emulators:start --only firestore` in your terminal.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**}{
     allow read, write: if request.auth.uid != null
    }
  }
}
```

Similarly, the student can change the security rules for their application.

10. Try logging in using an unauthorized email and password to get the following output:



There is no user record corresponding to this identifier. T...

What's NEXT?

In the next class, we will be working on creating a space-related application called the ISS Tracker.

EXTEND YOUR KNOWLEDGE

You can refer to the Firebase authentication document to learn more about the authentication process from here.