Topic	AUTHENTICATION AND DB INTEGRATION	
Class Description	**In this class, the students will implement user authentication and integrate their app with Firebase.**	
Class	**C85**	
Class time	**55 mins**	
Goal	• Using Email/Password Authentication to authenticate the users. • Registering a new user. • Integrating the Firebase database with the App.	
Resources Required	• Teacher Resources: ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ earphones with mic ○ notebook and pen • Student Resources: ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ earphones with mic ○ notebook and pen	
Class structure	**Warm-Up** **Teacher-Student Collaborative Activity** **Wrap-Up** **This class requires database configuration. Request students to live share VSC and perform activities to avoid writing the same code twice at both ends.*	**5 mins** **45 mins** **5 mins**
Credit:	**Code samples used for Firebase-Authentication**	

	are licensed under the Apache 2.0 License. **Expo documentation used from - https://expo.io**	

WARM-UP SESSION - 5 mins

Teacher starts slideshow from slides 1 to 12
Refer to speaker notes and follow the instructions on each slide.

Activity details	Solution/Guidelines
Run the presentation from slide 1 to slide 4. **The following are the warm-up session deliverables:** • Revision • Warm-Up Quiz Session	Click on the slide show tab and present the slides.

Continue the warm-up session

Activity details	Solution/Guidelines
Run the presentation from slide 4 to slide 12 to set the problem statement. **The following are the warm-up session deliverables:** • Introduce students to the coding environment - Workspace, blocks, and output. • Steps to write and run the code. • Introduce the concepts of Teacher-led Activity.	Narrate the story by using hand gestures and voice modulation methods to bring in more student interest.

- **Teacher ends slideshow**

- **Teacher-Student Collaborative Activity - 45 mins**

Teacher Initiates Screen Share		
CHALLENGE • **Login screen.** • **Integrating the Firebase database to the App.**		
Step 2: Teacher-led Activity (15 min)	Since we already have learnt user authentication using email/password in the wireless library app, we will start with. *(Ask the student to observe closely as all the changes should be made on both, the student's and teacher's codes.)* *(There are no separate Teacher and Student activities in this class.)* *Note - If the student and/or teacher is using the snack editor for these classes, please refer to the support document in Teacher Activity 6*	
	Let's download the boilerplate code. Let's start by installing Firebase and @react-navigation/native	*Student refers to Student Activity 1 and clones the boilerplate code.* *The student uses **live sharing** in **VSC** to share the code with the teacher.*

	npx expo install firebase@8.10.0 *(Teacher helps the student install Firebase and react-navigation.)* Remember that up until this point, we have installed specific navigation - Tab, Drawer and Stack. Now the idea for implementing Login is that we have 3 parts - 1. The Login screen where the user will Login from. 2. The register screen, if it's a new user. 3. The dashboard screen (or the Feed screen in our case) that the user will see once they are logged in. For this, we will be using the Stack Navigator. How many navigation methods have we implemented in our app so far? We already have the code for this provided to us in the boilerplate code	*The student installs Firebase and react-navigation.* **ESR:** 3 navigations - Stack, Drawer and Tab.

	that we just coded. Let's quickly go through it. Inside the file ***App.js -***	

If you remember, we discussed that it is important to have our navigation in the **<NavigationContainer>** component. We had already added that in **App.js**

We also need to import **createStackNavigator** to implement login navigation. We will also import the **LoginScreen** and the **Register** screens here.

Now let's see in our ***screens*** folder. There should be 2 files, that we just discussed -

1. **LoginScreen.js**
2. **Register.js**

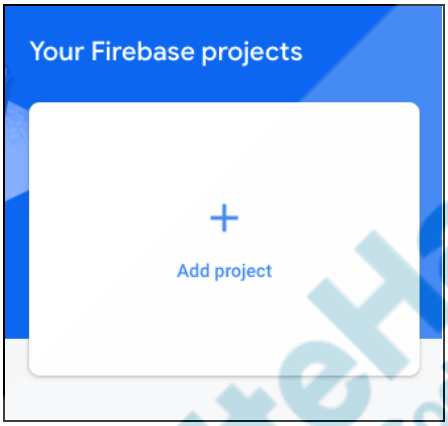
These 2 files have some boilerplate code added in them.

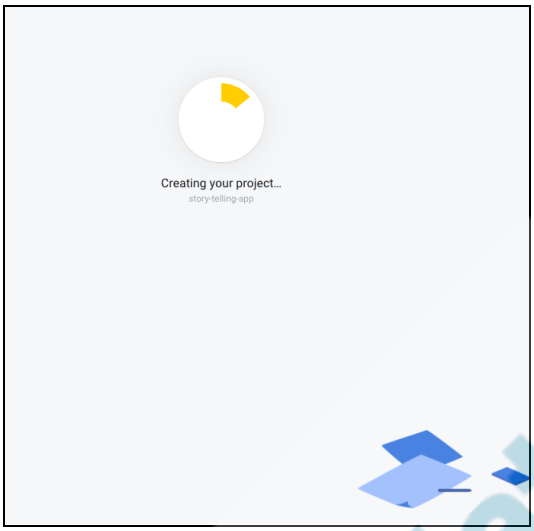
```
JS App.js > ...
1  import * as React from "react";
2  import { NavigationContainer } from '@react-navigation/native';
3
4  import { createStackNavigator } from "@react-navigation/stack";
5  import LoginScreen from "./screens/LoginScreen";
6  import RegisterScreen from "./screens/Register";
7
8  import DrawerNavigator from "./navigation/DrawerNavigator";
```

In the Stack Navigator, we will add **LoginScreen, RegisterScreen** and **DrawerNavigator.** Note that we don't have the complete screens yet, so we'll create them next.

	Let's set up a new Firebase DB before we proceed with the next steps. *Teacher refers to [Teacher Activity 2]*	*Student refers to [Student Activity 2].*

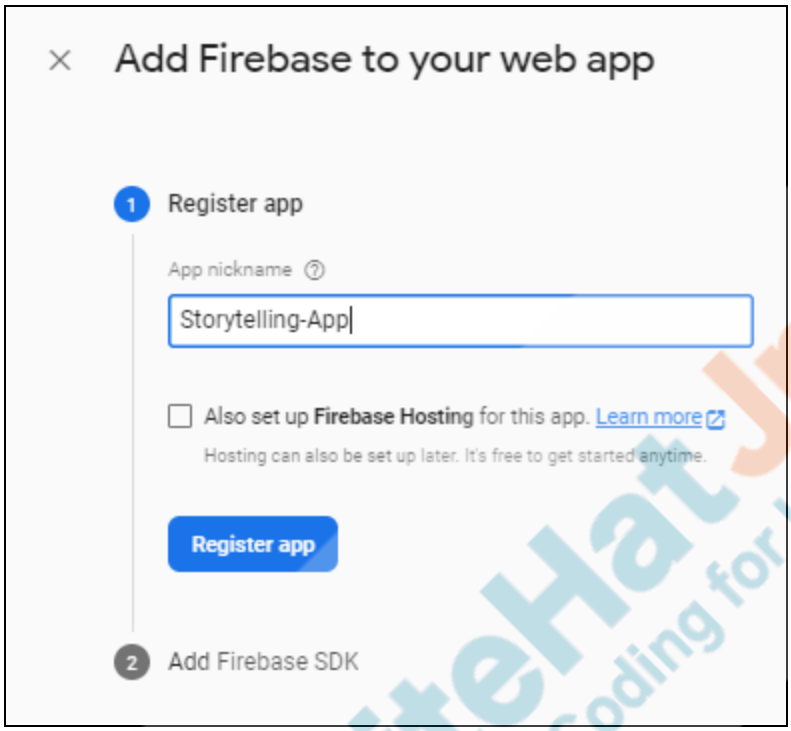
	Teacher tells the student to open the Firebase console. Here, let's create a new project by clicking on the **Add project** button.	*Student creates a new Firebase project.*





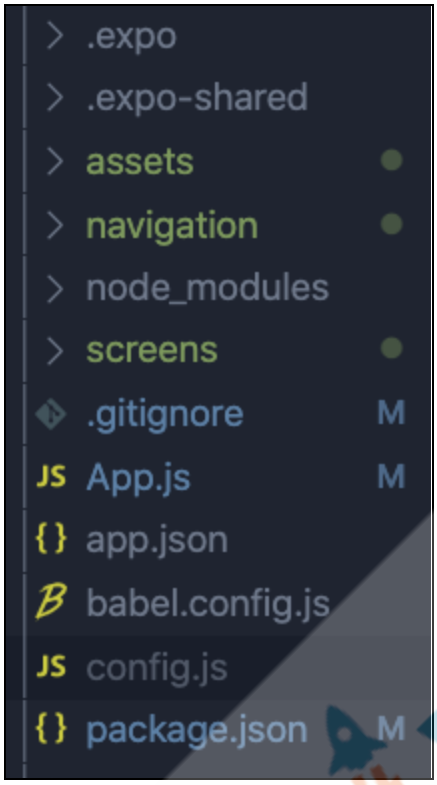
Great! Now that we have our project ready, we need to create our configuration keys to use this database.

For that, we will click on the web button, register our app by giving it a name and copy the config keys from there -



Copy the SDK code from step 2; *(It will be visible once you click on Register App.)*

	Now, to save these config keys, let's create a new file ***config.js*** in our root folder of the project. Also don't forget to enter the ***config.js*** file in ***.gitignore***, or else your Firebase project will be blocked. It is a poor practice in software development to expose your authentication keys on github, and opens a door for hackers to access and view sensitive information from your database.	

	*Teacher copies the config in **config.js** and adds the filename in **.gitignore.***	*Student copies the config in **config.js** and adds the file name in **.gitignore.***
		

```
85t > .gitignore
*.jks
*.p8
*.p12
*.key
*.mobileprovision
*.orig.*
web-build/

# macOS
.DS_Store

config.js
```

Create config.js

```
export const firebaseConfig = {
    apiKey: "AIzaSyDce_gGywAiuJEftp4Ccbt9odCV5y7rZiI",
    authDomain: "storytelling-app-cab54.firebaseapp.com",
    projectId: "storytelling-app-cab54",
    storageBucket: "storytelling-app-cab54.appspot.com",
    messagingSenderId: "843153669971",
    appId: "1:843153669971:web:05101931886d9498266ba6"
  };
```

Here, please note that we are using **export const** for our config keys, since we want to export it as a constant in our app.

	Great! Now our Firebase database will be available to our app.	

	Now let's refer back to our ***App.js*** We have to add a few lines of code in **App.js** to import Firebase.	

```
App.js > ...
import * as React from "react";
import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from "@react-navigation/stack";
import LoginScreen from "./screens/LoginScreen";
import RegisterScreen from "./screens/Register";

import DrawerNavigator from "./navigation/DrawerNavigator";

import * as firebase from "firebase";
import { firebaseConfig } from "./config";


if (!firebase.apps.length) {
  firebase.initializeApp(firebaseConfig);
} else {
  firebase.app();
}
```

We are importing the Firebase database and our config to **App.js**. Then, we are initializing the app with it. We have the **if-else** condition to check if we already have the Firebase app initialized. If not, we are initializing the Firebase app otherwise, we are using the already initialized app.

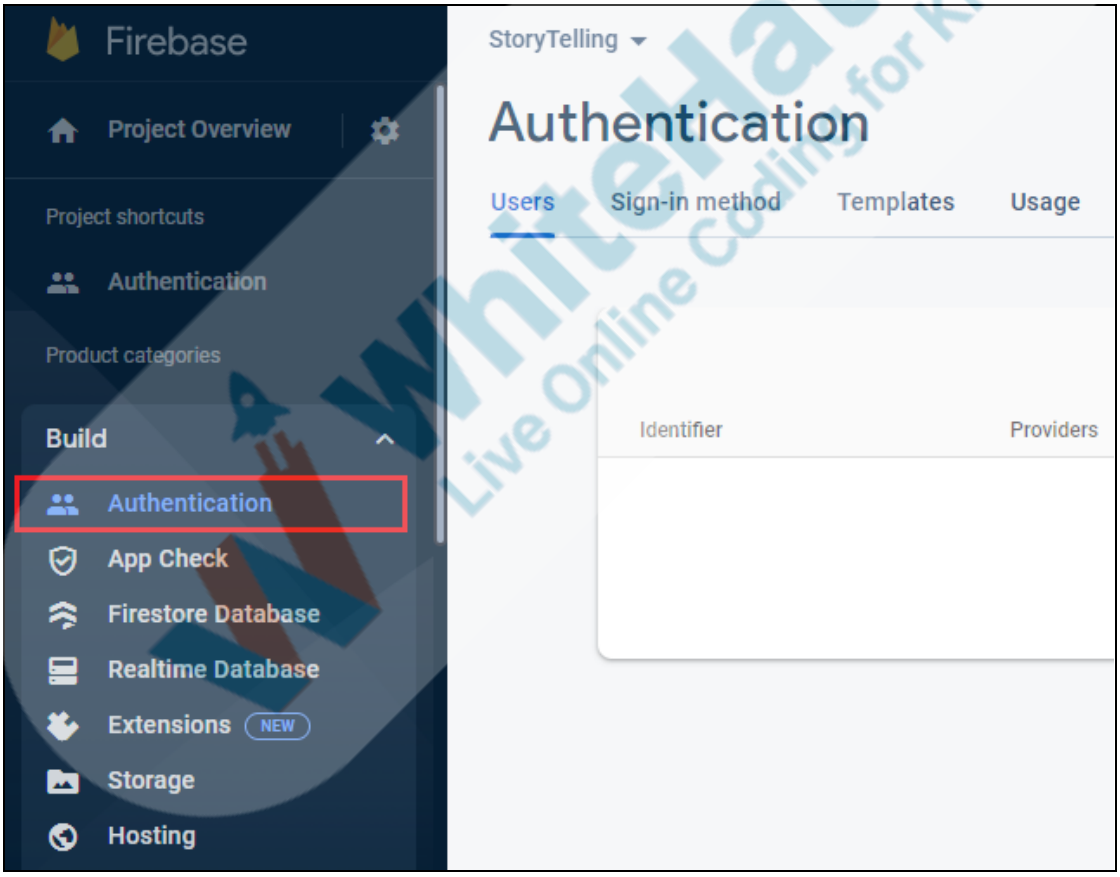
	Now we are pretty much halfway through.	
	Let's add the Stack Navigator in **App.js** which will hold our screens.	

```
10  import * as firebase from "firebase";
11  import { firebaseConfig } from "./config";
12
13
14  if (!firebase.apps.length) {
15    firebase.initializeApp(firebaseConfig);
16  } else {
17    firebase.app();
18  }
19
20  const Stack = createStackNavigator();
21
22  const StackNav = () => {
23    return(
24    <Stack.Navigator initialRouteName="Login"  screenOptions={{
25      headerShown: false,
26      gestureEnabled: false
27    }}>
28      <Stack.Screen name="Login" component={LoginScreen} />
29      <Stack.Screen name="RegisterScreen" component={RegisterScreen} />
30      <Stack.Screen name="Dashboard" component={DrawerNavigator} />
31    </Stack.Navigator>)
32  }
33
34  export default function App() {
35    return (
36      <NavigationContainer>
37        <StackNav/>
38      </NavigationContainer>)
39
40  }
41
42
```

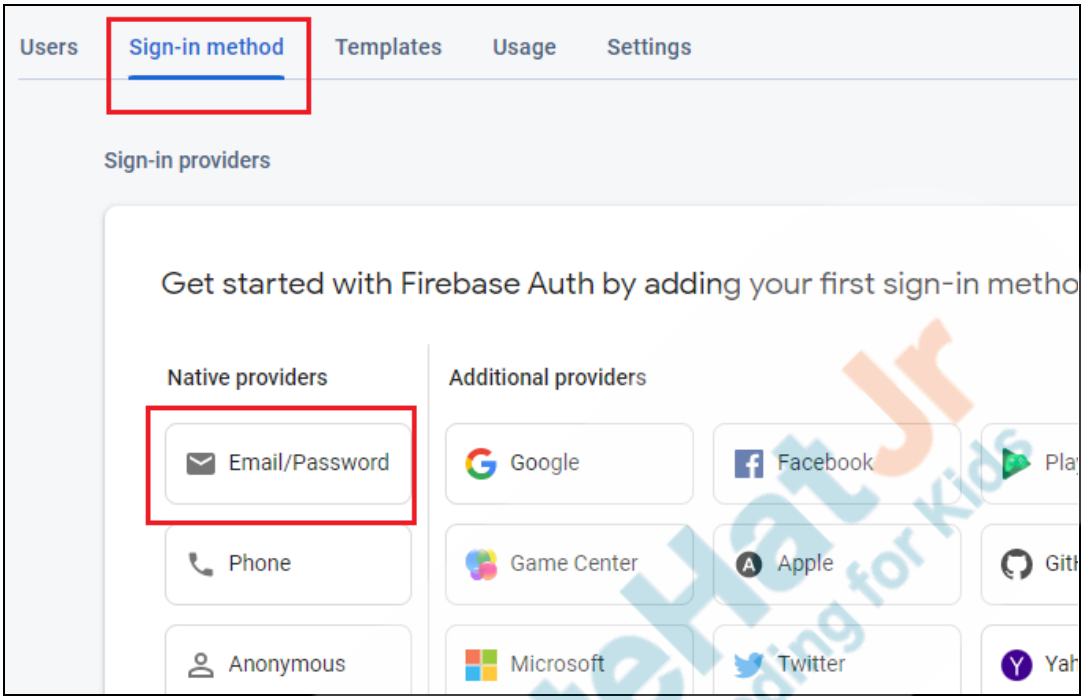
	With this, we have a lot of our functionality ready.	

	Now comes the part where we will be implementing our Login feature. We have already done this in the Wily app. *Teacher refers to Teacher Activity 2.*	

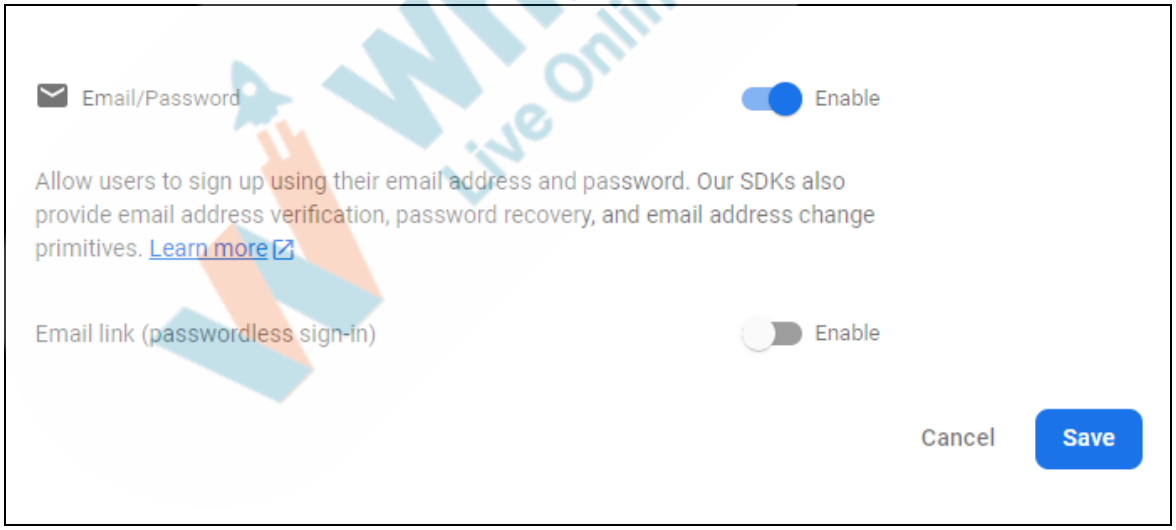
1. Go to the **StoryTelling** project that you just created in the firebase console → go to the left hand side of your screen and expand the **build** option → Click on **Authentication.**



2. Now, click on the **Sign-in method** tab → select **Email/Password**.

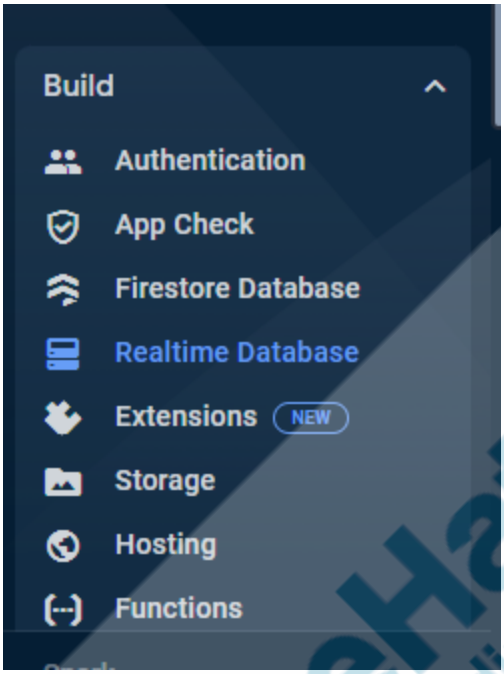


3. Enable the Email/Password option and click on **Save**.

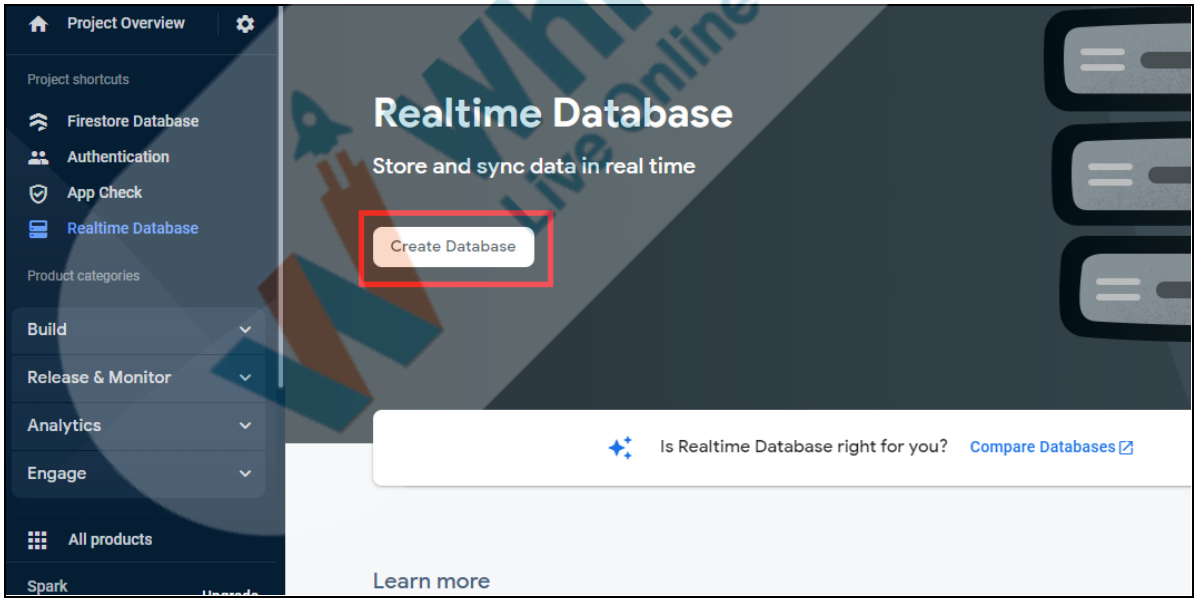


Create a **Realtime Database.**

- go to the left hand side of the screen and expand the **build** option → Click on the **Realtime Database** option.



- Click on the **Create Database** button.



	Let's now go over the code in ***LoginScreen.js*** to understand what we have done there -	

1. In the **LoginScreen.js**, we already have the code which creates the basic structure of the screen and styles.
   <Teacher helps the student to understand the code>

2. Now, define the **signIn()** function which will help the user to sign in . We will use the **signInWithEmailAndPassword()** method to do this.

```
signIn = async (email, password) => {
  firebase
    .auth()
    .signInWithEmailAndPassword(email, password)
    .then(() => {
      this.props.navigation.replace("Dashboard");
    })
    .catch(error => {
      Alert.alert(error.message);
    });
};
```

3. Notice that we have used the **.replace()** method instead of **.navigate()** method. This will make sure that once we login, we cannot go back to the login page by just clicking the back button.

```
signIn = async (email, password) => {
  firebase
    .auth()
    .signInWithEmailAndPassword(email, password)
    .then(() => {
      this.props.navigation.replace("Dashboard");
    })
    .catch(error => {
      Alert.alert(error.message);
    });
};
```

4. We have also added the **.catch()** function to return error messages.

```
signIn = async (email, password) => {
  firebase
    .auth()
    .signInWithEmailAndPassword(email, password)
    .then(() => {
      this.props.navigation.replace("Dashboard");
    })
    .catch(error => {
      Alert.alert(error.message);
    });
};
```

5. Now, call the **signIn()** function when the login button is clicked.

```
<TouchableOpacity
  style={[styles.button, { marginTop: 20 }]}
  onPress={() => this.signIn(email, password)}
>
  <Text style={styles.buttonText}>Login</Text>
</TouchableOpacity>
```

6. Let's add another **TouchableOpacity** which will let the user go to **RegisterScreen** when it's a new user.

```
<TouchableOpacity
  onPress={() => this.props.navigation.navigate("RegisterScreen")}
>
  <Text style={styles.buttonTextNewUser}>New User ?</Text>
</TouchableOpacity>
```

	Let's build the **Register.js** screen now.	

1. In the **Register.js**, we already have some of the code which creates the basic structure of the screen and styles.
   <Teacher helps the student to understand the code>

2. We already have states which store the email and password. We need to add three new states- **first_name**, **last_name** and **confirm_password.**

3. We already have two **TextInputs** which take the **email** and **password** as input. We need three more **TextInputs** to input the **first_name**, **last_name** and **confirm_password** for the registration process.

   When we register on any app, we enter the password twice to make sure that the password is correct during the registration process.

```
<View style={styles.container}>
  <SafeAreaView style={styles.droidSafeArea} />

    <Text style={styles.appTitleText}>Register</Text>

    <TextInput
      style={styles.textinput}
      onChangeText={text => this.setState({ first_name: text })}
      placeholder={"First name"}
      placeholderTextColor={"#FFFFFF"}

    />
    <TextInput
      style={styles.textinput}
      onChangeText={text => this.setState({ last_name: text })}
      placeholder={"Last name"}
      placeholderTextColor={"#FFFFFF"}

    />
    <TextInput
      style={styles.textinput}
      onChangeText={text => this.setState({ email: text })}
      placeholder={"Enter Email"}
      placeholderTextColor={"#FFFFFF"}

    />
```

```
<TextInput
  style={styles.textinput}
  onChangeText={text => this.setState({ email: text })}
  placeholder={"Enter Email"}
  placeholderTextColor={"#FFFFFF"}

/>
<TextInput
  style={styles.textinput}
  onChangeText={text => this.setState({ password: text })}
  placeholder={"Enter Password"}
  placeholderTextColor={"#FFFFFF"}
  secureTextEntry
/>
<TextInput
  style={styles.textinput}
  onChangeText={text => this.setState({ confirmPassword: text })}
  placeholder={"Re-enter Password"}
  placeholderTextColor={"#FFFFFF"}
  secureTextEntry
/>
```

4. Write the code to register a user.
    a. define a method named **registerUser.** It will have 5 parameters: **email**, **password**, **confirm_password**, **first_name** and **last_name**.

```
registerUser = (email, password,confirmPassword,first_name,last_name) => {

};
```

    b. Now, we will add a conditional statement which will check if the **password** and **confirm_password** matches or not.

```
registerUser = (email, password,confirmPassword,first_name,last_name) => {
 if(password==confirmPassword){

   }else{
     Alert.alert("Passwords don't match!");
   }
 };
```

c. Then, if the password matches, we need to write the code to register the user with the firebase method **createUserWithEmailAndPassword()**

```
registerUser = (email, password,confirmPassword,first_name,last_name) => {
 if(password==confirmPassword){
   firebase
     .auth()
     .createUserWithEmailAndPassword(email, password)
     .then((userCredential) => {
       Alert.alert("User registered!!");
     })
     .catch(error => {
       Alert.alert(error.message);
     });
   }else{
     Alert.alert("Passwords don't match!");
   }
 };
```

d. The above code will register the user with **email** and **password**. Everytime a new user is created, the new user is assigned a unique **uid**. We can fetch this **uid** -

```
registerUser = (email, password,confirmPassword,first_name,last_name) => {
 if(password==confirmPassword){
   firebase
     .auth()
     .createUserWithEmailAndPassword(email, password)
     .then(userCredential => {
       Alert.alert("User registered!!");
       console.log(userCredential.user.uid)

     })
     .catch(error => {
       Alert.alert(error.message);
     });
   }else{
     Alert.alert("Passwords don't match!");
   }
 };
```

e. We will use the **uid** to store our users data in the database. Here, we will store the:
   i. Email ID
   ii. First Name
   iii. Last Name
   iv. Current Theme

```
registerUser = (email, password,confirmPassword,first_name,last_name) => {
 if(password==confirmPassword){
   firebase
     .auth()
     .createUserWithEmailAndPassword(email, password)
     .then((userCredential) => {
       Alert.alert("User registered!!");
       console.log(userCredential.user.uid)
       this.props.navigation.replace("Login");
       firebase.database().ref("/users/" + userCredential.user.uid)
                .set({
                  email: userCredential.user.email,
                  first_name: first_name,
                  last_name: last_name,
                  current_theme: "dark"
                })
     })
     .catch(error => {
       Alert.alert(error.message);
     });
   }else{
     Alert.alert("Passwords don't match!");
   }
 };

```

this is how the database will look like, when a user registers -

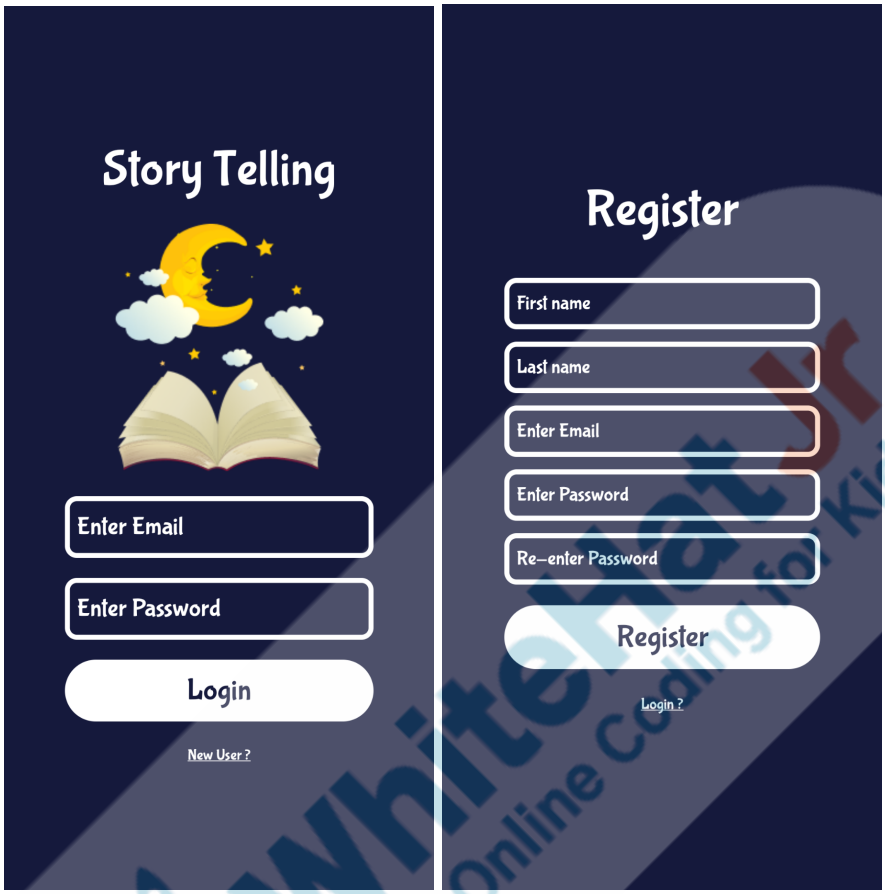
5. After this is done, we will call the **registerUser** method in the **Register TouchableOpacity**.

```
<TouchableOpacity
  style={[styles.button, { marginTop: 20 }]}
  onPress={() => this.registerUser(email, password, confirmPassword,first_name,last_name)}
>
  <Text style={styles.buttonText}>Register</Text>
</TouchableOpacity>
```

6. Let's add another TouchableOpacity which will take the user back to the **LoginScreen.**

```
<TouchableOpacity
  onPress={()=>this.props.navigation.replace("Login")}
>
  <Text style={styles.buttonTextNewUser}>Login ?</Text>
</TouchableOpacity>
```

	We are ready. Let's test our app now by trying to Log In. Run the following command- **npx expo start --tunnel** (*Teacher and student try to Login and see the Feed screen.*) If you check your Database, you will see that your email ID has been registered.	
<u>Reference Output:</u>		



Great! Now we have our Authentication all done and our app is integrated with Firebase.

*(**A note for teachers -** We know this class might be difficult to complete based on the machine speed and installation required. We have kept the next classes smaller to get time in the later classes so the process can be completed in next classes.)*

Teacher Stops Screen Share

WRAP-UP SESSION - 5 mins

FEEDBACK

- **Appreciate the student for their attentiveness in the class.**
- **Get them to play around with different ideas**

Teacher can show slideshow from slides 12 to 22

Refer to speaker notes and follow the instructions on each slide.

For the 'Wrap-Up' section, there will be slides on the panel as a visual aid to summarize what has been done in the class.

Activity details		Solution/Guidelines
	Amazing work today! You get a "hats-off". In the next class, we will add the logout feature and we will work on the profile screen. Alright. See you in the next class.	*Make sure you have given at least 2 Hats Off during the class for:* Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
Project Overview ***Teachers make sure to tell students to refer to documents used during class and Post class Summary to implement Authentication in the Project.*** **Spectagram Stage - 5** **Goal of the Project:**		*The students engage with the teacher over the project.*

In Class 85, we learned to implement Authentication and integrate the app with Firebase. In this project, you will practice the concepts learned in the class and implement Authentication and integrate the Spectagram app with Firebase. *This is a continuation project of 81, 82, 83 & 84; please make sure to finish that before attempting this one. **Story:** Jenny is a photographer. She wants to share pictures taken by her with others. At the same time, she wants to create a space for others to share their talent too. She has decided to create a social media app. She has asked for your help to create an app. Guide Jenny to implement Authentication and integrate the app with Firebase.	
Teacher ends slideshow	
Teacher Clicks ✖ End Class	
Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.* / *The student uses the markdown editor to write their reflections in a reflection journal.*

	Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	

Activity	Activity Name	Links
Teacher Activity 1	Boilerplate code	https://github.com/pro-whitehatjr/PRO-C85-boilerplate
Teacher Activity 2	Firebase Console	https://console.firebase.google.com/
Teacher Activity 3	Authenticate with Firebase	https://firebase.google.com/docs/auth/web/password-auth
Teacher Reference 1	Reference Code	https://github.com/pro-whitehatjr/PRO-C85-solution
Teacher Reference 2	Teacher Aid	https://drive.google.com/file/d/1WA1BQff4dmgv5BInU3f_imk4vlpvAyMa/view?usp=sharing
Teacher Reference 3	Snack Support Document	https://docs.google.com/document/d/11vq49uJQCfdxaUUzOoY7A65aau0kZqNMFhObZH-e71Y/edit?usp=sh

		aring
Teacher Reference 4	Visual aid link	https://s3-whjr-curriculum-uploads.whjr.online/1464af65-f3be-464f-b876-cb42e761b05d.html
Teacher Reference 5	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/8ad7a8be-a409-45b7-b7eb-19a840477245.pdf
Student Activity 1	Boilerplate code	https://github.com/pro-whitehatjr/PRO-C85-boilerplate
Student Activity 2	Firebase Console	https://console.firebase.google.com/
Student Activity 3	Authenticate with Firebase	https://firebase.google.com/docs/auth/web/password-auth