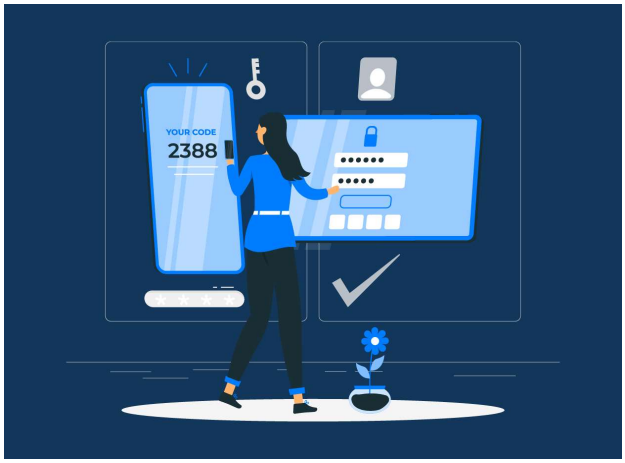
GOOGLE AUTHENTICATION AND DB INTEGRATION



What is our GOAL for this MODULE?

In this class, we continued to build the Storytelling application. We learned to implement Google login and integrate the App with the Firebase database using Firebase Authentication.

What did we ACHIEVE in the class TODAY?

Designed switch navigation to navigate from Loading screen to Login screen to the Dashboard screen.

Implemented Google Login for the App by setting up the credentials in our Firebase App on the Firebase console.

Integrated the App with Firebase using Firebase services for login authentication.

Which CONCEPTS/ CODING BLOCKS did we cover today?

Creating Loading, Login and Dashboard Screens

Function to decide which screen to display based on user's login status

Writing complex functions to implement Google authenticated login

How did we DO the activities?

1. Write code to create switch Navigator and App container using **createSwitchNavigator()** and **createAppContainer()** functions which would contain:
    - Loading Screen
    - Login Screen
    - Dashboard Screen
    - Call switch navigator **AppSwitchNavigator** and navigation container **AppNavigator** in App.js

```
import * as React from "react";
import { createSwitchNavigator, createAppContainer } from "react-
navigation";

import LoginScreen from "./screens/LoginScreen";
import LoadingScreen from "./screens/LoadingScreen";
import DashboardScreen from "./screens/DashboardScreen";

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

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

const AppSwitchNavigator = createSwitchNavigator({
          LoadingScreen: LoadingScreen,
          LoginScreen: LoginScreen,
          DashboardScreen: DashboardScreen
});

const AppNavigator = createAppContainer(AppSwitchNavigator);

export default function App() {
          return <AppNavigator />;
}
```

2. Create 3 files LoadingScreen.js, LoginScreen.js and DashboardScreen.js.

LoadingScreen.js

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

export default class LoadingScreen extends Component {
  render() {
    return (
      <View
        style={{
          flex: 1,
          justifyContent: "center",
          alignItems: "center"
        }}>
        <Text>Loading</Text>
      </View>
    )
  }
}
```

LoginScreen.js

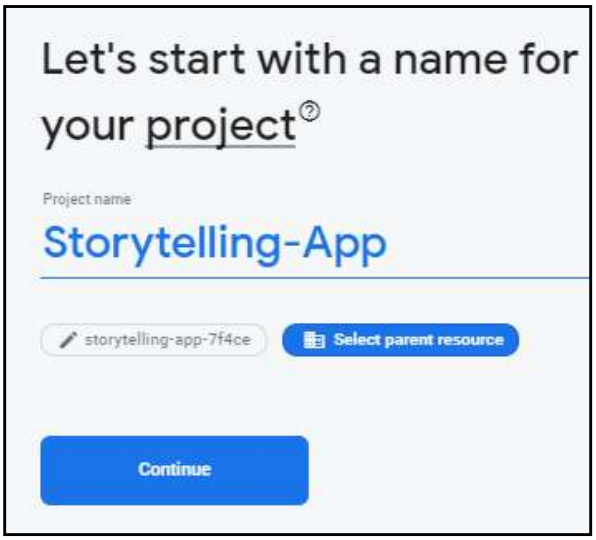
```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

export default class LoginScreen extends Component {
  render() {
    return (
      <View
        style={{
          flex: 1,
          justifyContent: "center",
          alignItems: "center"
        }}>
        <Text>LoginScreen</Text>
      </View>
```

```
        )
    }
}
```

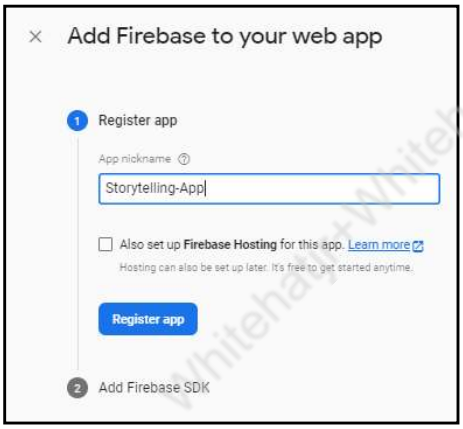
DashboardScreen.js

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

export default class DashboardScreen extends Component {
  render() {
    return (
      <View
        style={{
          flex: 1,
          justifyContent: "center",
          alignItems: "center"
        }}>
        <Text>DashboardScreen</Text>
      </View>
    )
  }
}
```

3. Create a new Firebase project and create configuration keys and save them in the config.js file of the project to use this database.
    - Create Firebase project.



- Create App and configuration keys and copy the SDK code from step 2; *(it will be visible once you click on Register App).*



4. Save **config.js file** in **.gitignore** so that our Firebase project will not be blocked by Google, and paste SDK code in 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"
  };
```

5. Import **firebase** and **firebaseConfig** into **App.js** and write an **if** statement to initialize firebase.

```
import LoginScreen from "./screens/LoginScreen";
import LoadingScreen from "./screens/LoadingScreen";
import DashboardScreen from "./screens/DashboardScreen";

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

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

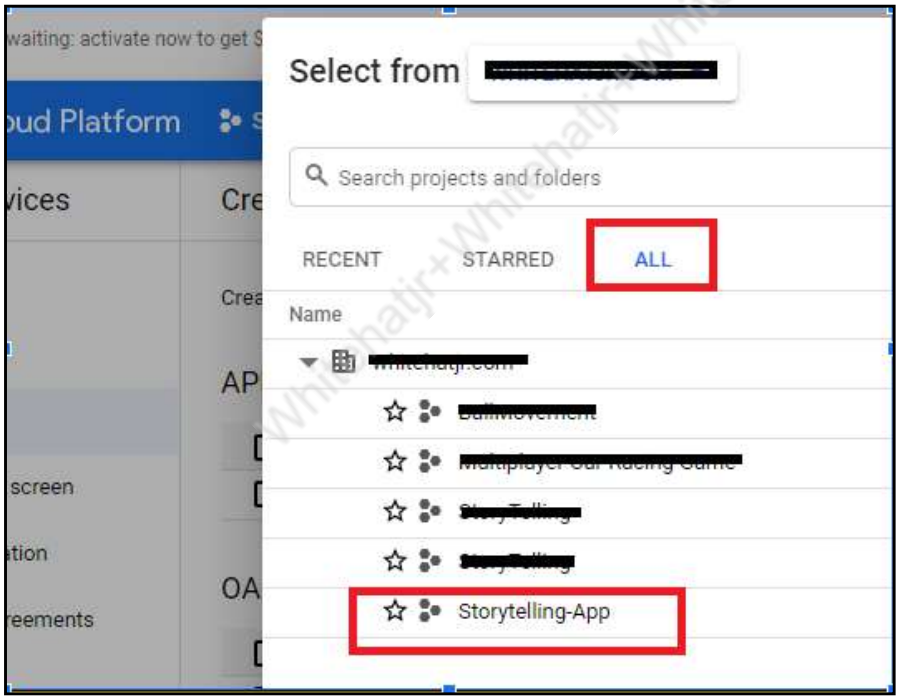
6. Write function **checkIfLoggedIn()** and call it in **componentDidMount()** in **loginScreen.js** to load the dashboard screen if the user is already logged in otherwise display the login screen.

   Use **firebase.auth ().onAuthStateChanged ()** to check user exists or not

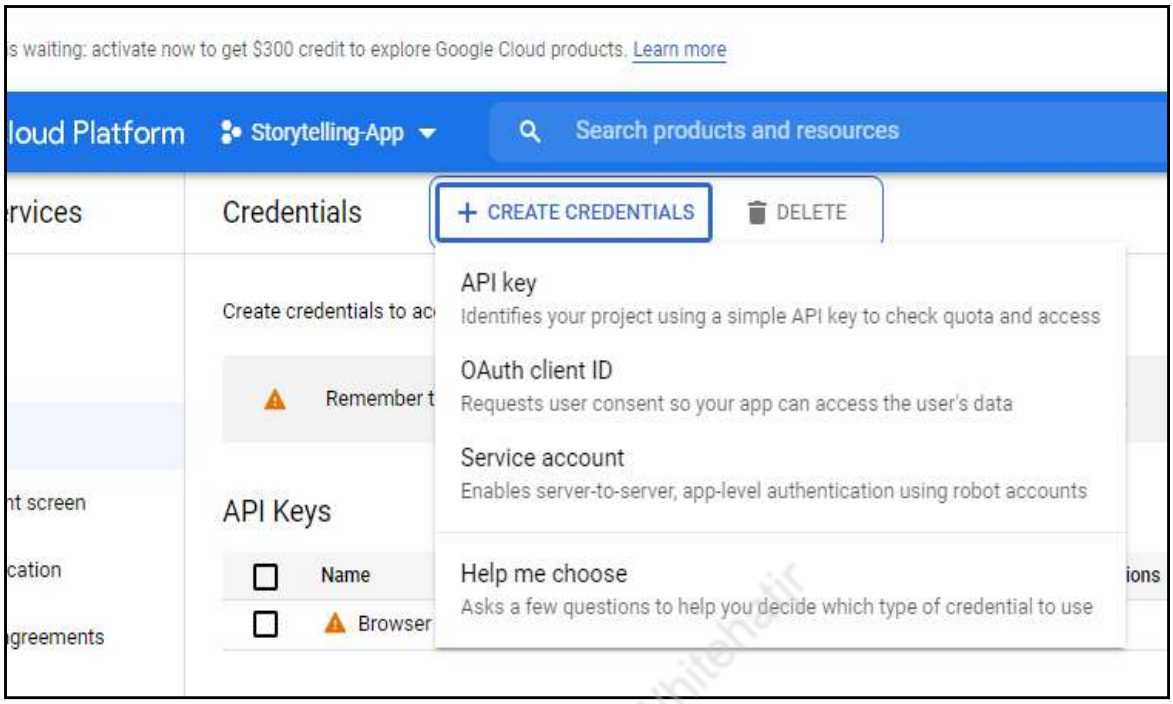
```
import React, { Component } from "react";
import {
  View,
  ActivityIndicator
} from "react-native";
import firebase from "firebase";
```

```
export default class LoadingScreen extends Component {

  componentDidMount() {
    this.checkIfLoggedIn()
  }

  checkIfLoggedIn = () => {
    firebase.auth().onAuthStateChanged((user) => {
      if (user) {
        this.props.navigation.navigate('DashboardScreen')
      } else {
        this.props.navigation.navigate('LoginScreen')
      }
    })
  }

  render() {
    return (
      <View
        style={{
          flex: 1,
          justifyContent: "center",
          alignItems: "center"
        }}
      >
        <ActivityIndicator size="large" />
      </View>
    )
  }
}
```

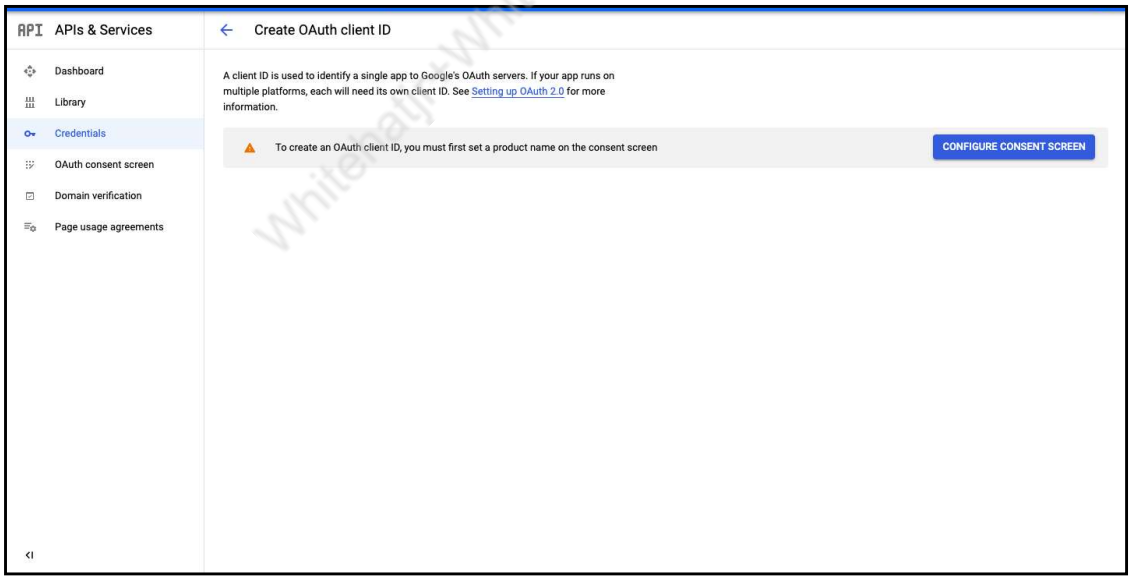
7. Implement Google login using [expo's documentation]() for Google Sign in.
   Click on the Credentials page.
   Select the Firebase project "Storytelling-App".

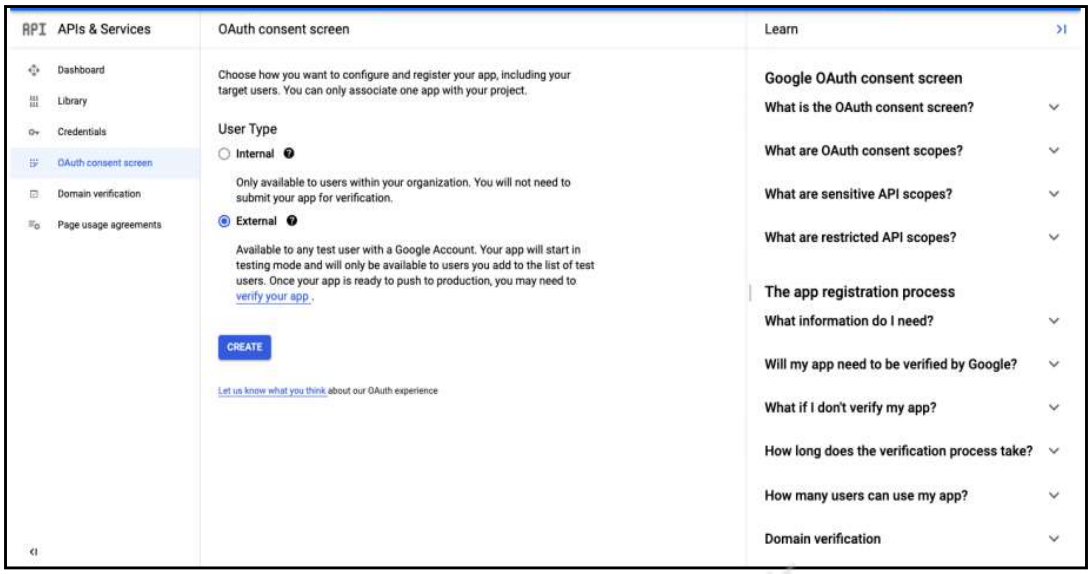


8. Create the credentials for the app.

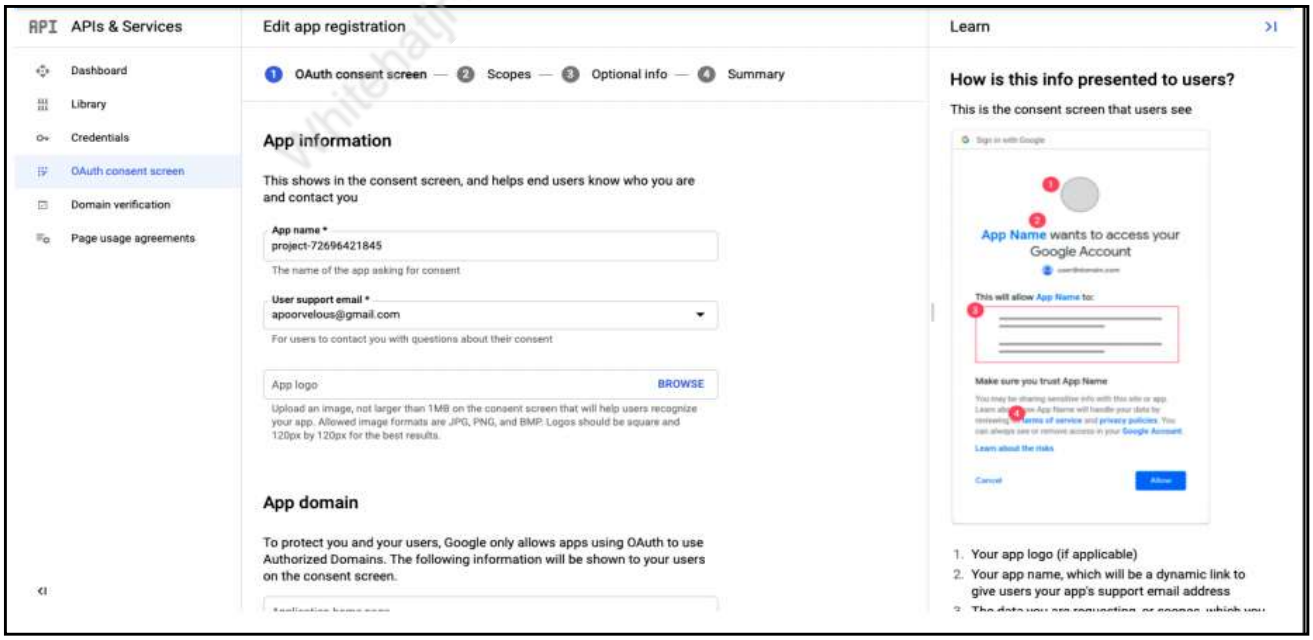


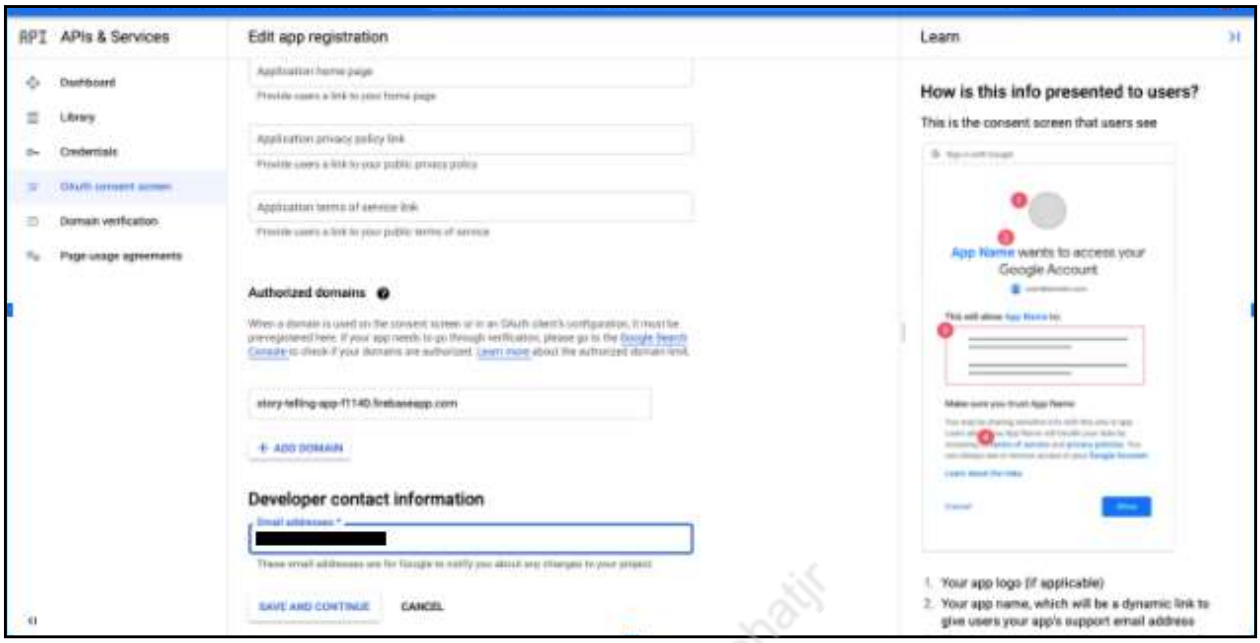
9. Create **OAuth client ID** and configure the consent screen by selecting **External** option.



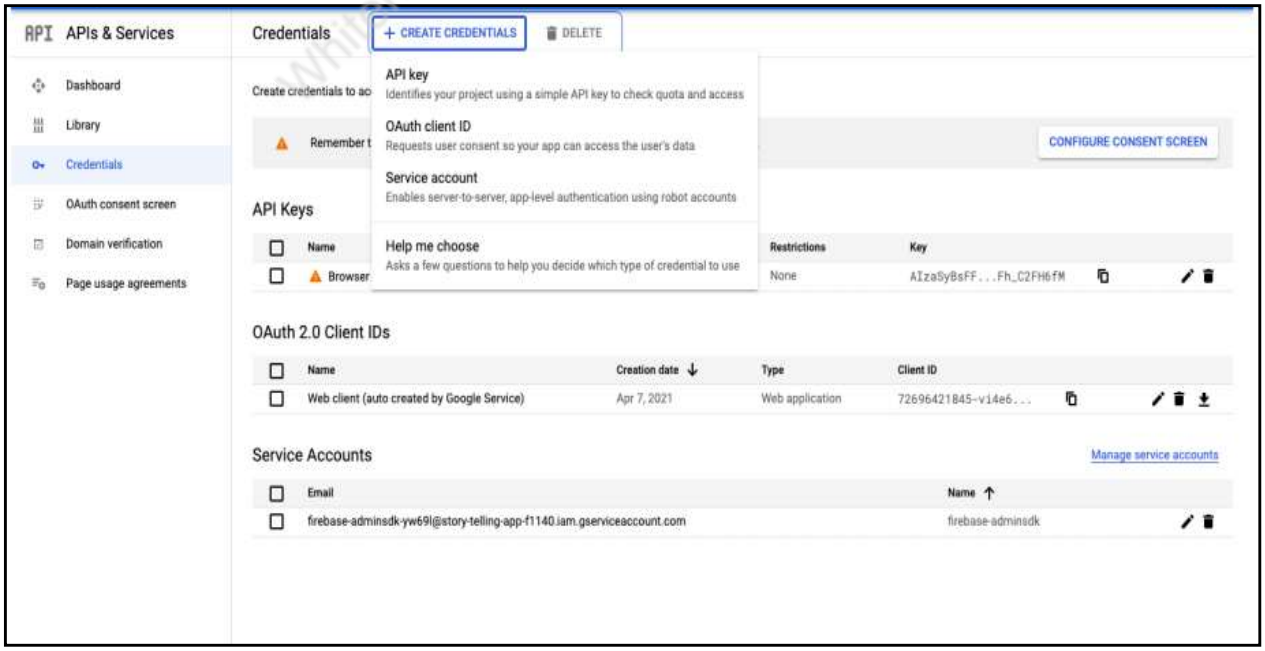


10. Enter your email ID for user support along with our developer information.





11. Go back to the dashboard and click on OAuth Client ID from Create Credentials.



12. Follow the instructions to generate OAuth IDs for both Android and iOS.

- **Create an iOS OAuth Client ID**
  - Select "iOS Application" as the Application Type. Give it a name if you want (e.g. "iOS Development").
  - Use `host.exp.exponent` as the bundle identifier.
  - Click "Create"
  - You will now see a modal with the client ID.
  - The client ID is used in the `iosClientId` option for `Google.loginAsync` (see code example below).
- **Create an Android OAuth Client ID**
  - Select "Android Application" as the Application Type. Give it a name if you want (maybe "Android Development").
  - Run `openssl rand -base64 32 | openssl sha1 -c` in your terminal, it will output a string that looks like `A1:B2:C3` but longer. Copy the output to your clipboard.
  - Paste the output from the previous step into the "Signing-certificate fingerprint" text field.
  - Use `host.exp.exponent` as the "Package name".
  - Click "Create"
  - You will now see a modal with the Client ID.
  - The client ID is used in the `androidClientId` option for `Google.loginAsync` (see code example below).

Here are it's installation steps -

To check if OpenSSL is installed in your system or not, run the following in a terminal / command prompt -

openssl --version

If it is not installed, which is highly unlikely, then see the following steps -

MacOS -

In a new terminal, run the following command -

export ACL_OPENSSL_VERSION=11

After that, run the following command -

brew install openssl

Ubuntu -

Run the following commands in a new Terminal -

sudo apt-get install libssl-dev

Windows -

Navigate to ***C:\Program Files\Git\usr\bin***

Run the command - ***openssl rand -base64 32 | openssl sha1 -c***

This will generate device fingerprint required to create OAuth ID for Android.

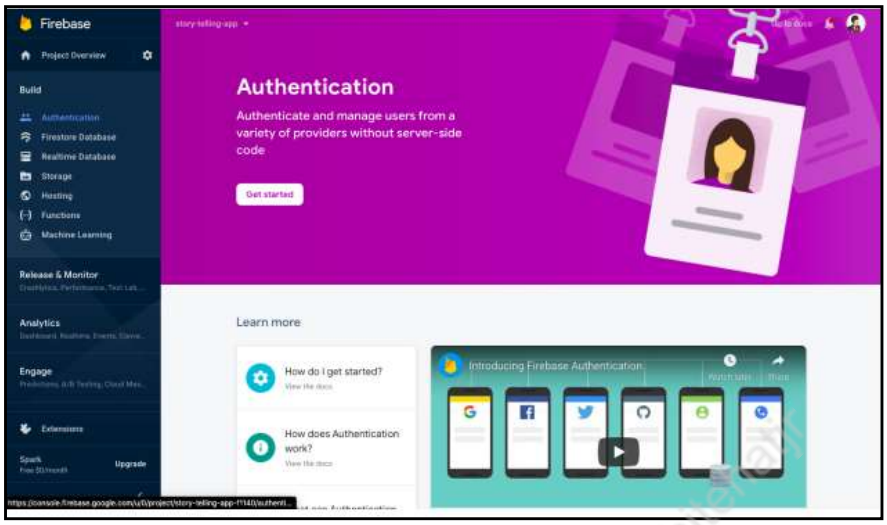
13. Add a **signInWithGoogleAsync()** function in login screen class, and update **androidClientID** and **iosClientId.**

```
import React, { Component } from 'react';
import { View, Button } from 'react-native';

export default class LoginScreen extends Component {
  signInWithGoogleAsync = async () => {
    try {
      const result = await Google.logInAsync({
        behaviour: "web",
        androidClientId: "512716633783-
rkrc8tpbrknrpnt0akjbqqo0v4ukafkc.apps.googleusercontent.com",
        iosClientId: "512716633783-
n6qco4smqeh9g64ffalebfrsepq0nvlf.apps.googleusercontent.com",
        scopes: ['profile', 'email'],
      });
```

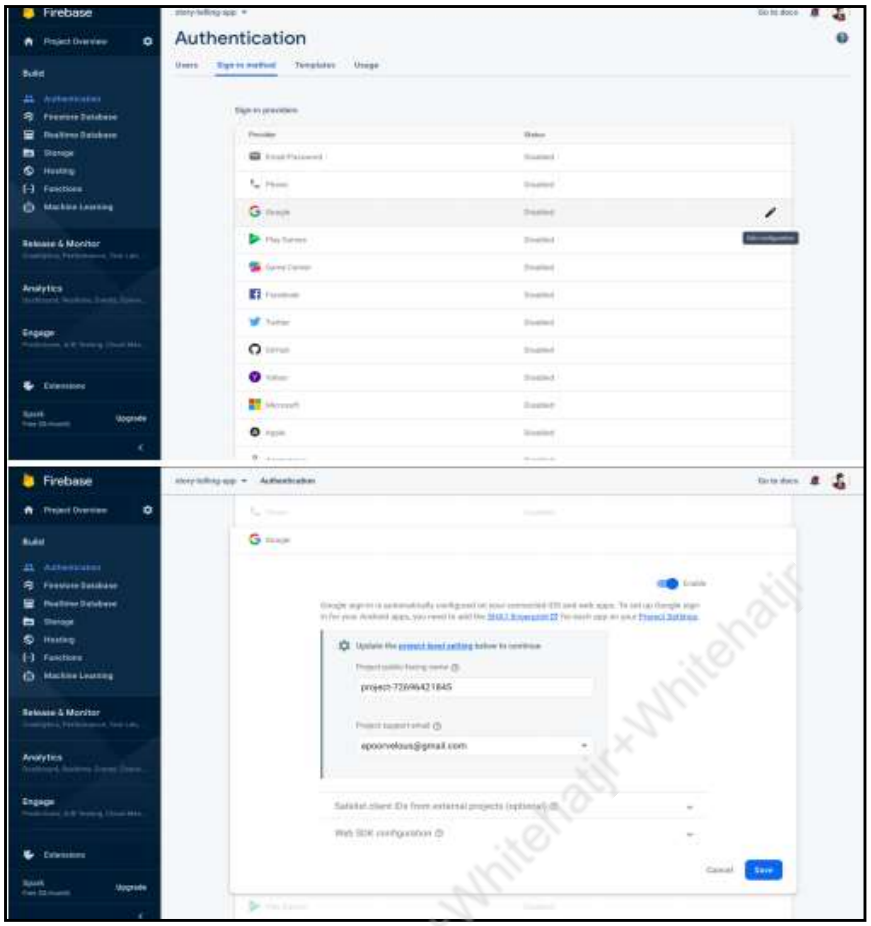
14. Create a button in the Login screen. On pressing on this button call **signInWithGoogleAsync**() function.

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

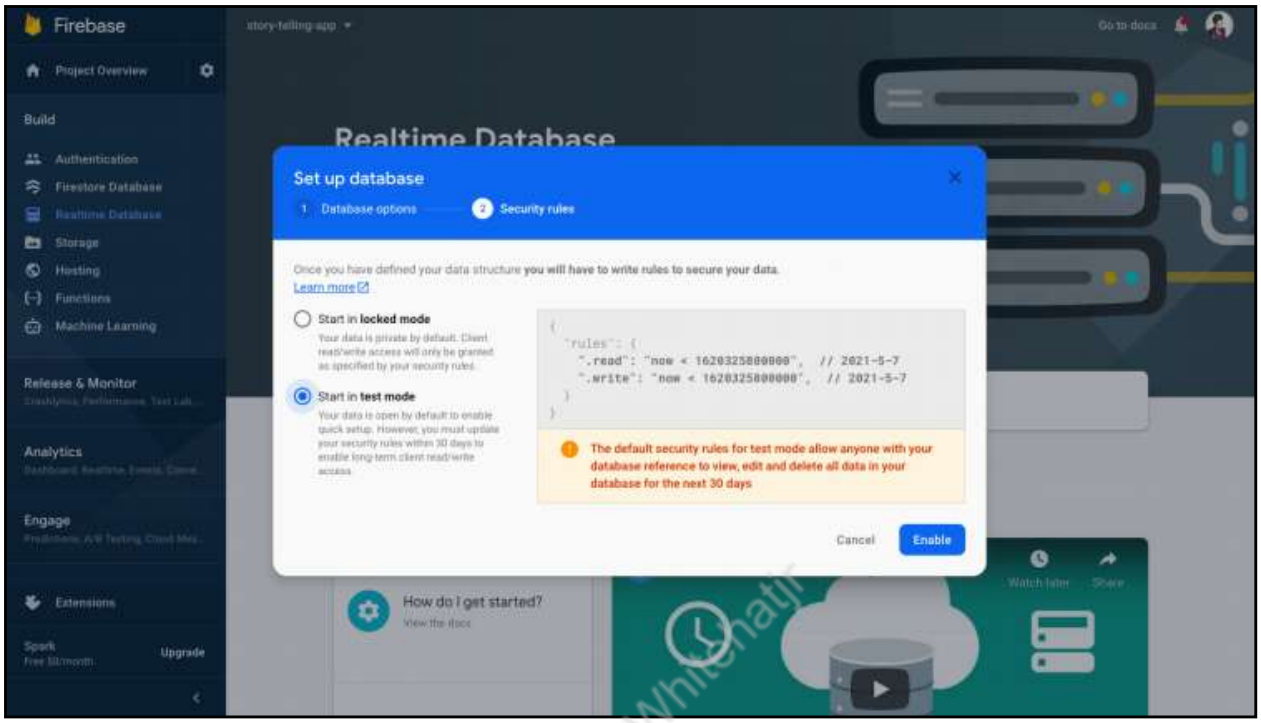
15. Fetch the user's data from the database and change the settings in Authentication of Firebase.



16. Enable Google Sign In.



17. Create a Realtime Database in test mode.



18. Install **expo install expo-google-app-auth** dependency.
19. Write code for function **OnSignIn** .
    - Check if the App is already signed in with the correct user using **firebase.auth().onAuthStateChanged().**
    - Use **isUserEqual()** to check the same user signed in.
    - Create the credentials for the user so they can use our app as a signed in user with **firebase.auth().GoogleAuthProvider.credential()** function. For setting up the credential, use the **id_token** and the **access_token** of our **googleUserSign** in with credentials from the Google user.

```
onSignIn = googleUser => {
    console.log('Google Auth Response', googleUser);
    // We need to register an Observer on Firebase Auth to make sure auth is
initialized.
    var unsubscribe = firebase.auth().onAuthStateChanged((firebaseUser) => {
        unsubscribe();
        // Check if we are already signed-in Firebase with the correct user.
        if (!this.isUserEqual(googleUser, firebaseUser)) {
            // Build Firebase credential with the Google ID token.
            var credential = firebase.auth.GoogleAuthProvider.credential(
```

```
            googleUser.idToken,
            googleUser.accessToken
        );

        // Sign in with credential from the Google user.
        firebase
            .auth()
            .signInWithCredential(credential)
            .then(function (result) {
                if (result.additionalUserInfo.isNewUser) {
                    firebase
                        .database()
                        .ref("/users/" + result.user.uid)
                        .set({
                            gmail: result.user.email,
                            profile_picture: result.additionalUserInfo.profile.picture,
                            locale: result.additionalUserInfo.profile.locale,
                            first_name: result.additionalUserInfo.profile.given_name,
                            last_name: result.additionalUserInfo.profile.family_name,
                            current_theme: "dark"
                        })
                        .then(function (snapshot) {

                        })
                }
            })
            .catch((error) => {
                // Handle Errors here.
                var errorCode = error.code;
                var errorMessage = error.message;
                // The email of the user's account used.
                var email = error.email;
                // The firebase.auth.AuthCredential type that was used.
                var credential = error.credential;
                // ...
            });
    } else {
        console.log('User already signed-in Firebase.');
    }
});
```

```
    }

    signInWithGoogleAsync = async () => {
      try {
        const result = await Google.logInAsync({
          behaviour: "web",
          androidClientId: "512716633783-
rkrc8tpbrknrpnt0akjbqqo0v4ukafkc.apps.googleusercontent.com",
          iosClientId: "512716633783-
n6qco4smqeh9g64ffalebfrsepq0nvlf.apps.googleusercontent.com",
          scopes: ['profile', 'email'],
        });

        if (result.type === 'success') {
          this.onSignIn(result)
          return result.accessToken;
        } else {
          return { cancelled: true };
        }
      } catch (e) {
        return { error: true };
      }
    }

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

20. Sign in the user with the Firebase authentication system using **firebase.auth().signInWithCredential()** function.

    Use a **.then()** function on it in which we are checking if the user is a new user or not with **result.additionalUserInfo.isNewUser.**

```
// Sign in with credential from the Google user.
firebase
  .auth()
  .signInWithCredential(credential)
  .then(function (result) {
    if (result.additionalUserInfo.isNewUser) {
      firebase
        .database()
        .ref("/users/" + result.user.uid)
        .set({
          gmail: result.user.email,
          profile_picture: result.additionalUserInfo.profile.picture,
          locale: result.additionalUserInfo.profile.locale,
          first_name: result.additionalUserInfo.profile.given_name,
          last_name: result.additionalUserInfo.profile.family_name,
          current_theme: "dark"
        })
        .then(function (snapshot) { });
    }
  })
```

21. Call **onSignIn()** function inside **signInWithGoogleAsync ()** function on finding the user. Use **Drawer Navigation** in the Dashboard screen to display the main screen.

```
import * as React from "react";
import { NavigationContainer } from "@react-navigation/native";
import DrawerNavigator from "../navigation/DrawerNavigator";

export default function DashboardScreen() {
  return (
    <NavigationContainer>
      <DrawerNavigator />
    </NavigationContainer>
```

```
            );
        }
```

22. Test the App by running the app on Mobile and login into it using a Gmail Id.. It should register the Email ID in the database.

OUTPUT:

What's next?

In the next class, we will complete the UI of the Login screen and allow the user to set the theme for the profile screen.

Expand your knowledge:

1. To explore more options for creating Authentication in Firebase: https://firebase.google.com/docs/auth?authuser=0