**SPLASH SCREEN(Our Application’s BodyGuard)**

import 'package:flutter/material.dart';

import 'package:supabase\_flutter/supabase\_flutter.dart';

import 'home\_screen.dart';

import 'login\_screen.dart';

class SplashScreen extends StatefulWidget {

  const SplashScreen({super.key});

  @override

  State<SplashScreen> createState() => \_SplashScreenState(); }

class \_SplashScreenState extends State<SplashScreen> {

  @override

  void initState() {

    super.initState();

    \_redirect(); }

  Future<void> \_redirect() async {

    await Future.delayed(Duration.zero);

    if (!mounted) { return; }

    final session = Supabase.instance.client.auth.currentSession;

    if (session != null) {

      Navigator.of(context).pushReplacement(

        MaterialPageRoute(builder: (context) => const HomeScreen()),

      );

    } else {

      Navigator.of(context).pushReplacement(

        MaterialPageRoute(builder: (context) => const LoginScreen()),

      ); } }

  @override

  Widget build(BuildContext context) {

    return const Scaffold(body: Center(child: CircularProgressIndicator()));

  } }

**CODE BREAKDOWN:**

**📌 What is a Splash Screen?**

A **Splash Screen** is usually the first screen that appears when you open an app.  
It is like the **gatekeeper** — it decides whether to send you to the **Home Screen** (if you are logged in ✅) or to the **Login Screen** (if you are not logged in ❌).

In your case, this Splash Screen:

* Shows a **loading indicator** while checking user status.
* Then redirects the user based on their **authentication session**.

**📂 Code Breakdown**

**1. StatefulWidget Setup**

class SplashScreen extends StatefulWidget {

const SplashScreen({super.key});

@override

State<SplashScreen> createState() => \_SplashScreenState();

}

👉 Why StatefulWidget?  
Because the splash screen needs to **check authentication** and then update (redirect). A StatelessWidget cannot handle lifecycle methods like initState.

**2. initState() → The Starting Point**

@override

void initState() {

super.initState();

\_redirect();

}

* initState() runs **once** when the widget is first created.
* We immediately call \_redirect() to check if the user is logged in.

Think of it like: **When the splash gate opens, it immediately checks: Are you allowed in (Home)? Or do you need to log in first?**

**3. The \_redirect() Function**

Future<void> \_redirect() async {

await Future.delayed(Duration.zero);

if (!mounted) {

return;

}

final session = Supabase.instance.client.auth.currentSession;

if (session != null) {

Navigator.of(context).pushReplacement(

MaterialPageRoute(builder: (context) => const HomeScreen()),

);

} else {

Navigator.of(context).pushReplacement(

MaterialPageRoute(builder: (context) => const LoginScreen()),

);

}

}

🔍 Let’s break it into steps:

**(a) await Future.delayed(Duration.zero);**

* This waits for **just one event loop tick**.
* Why? Because Flutter sometimes needs to **finish building the widget tree first** before you use Navigator.
* It’s a trick to avoid errors like *“setState or Navigator called during build”*.

**(b) if (!mounted) return;**

* mounted means **is this widget still active in the widget tree?**
* If the widget is already disposed (destroyed), we stop here to avoid errors.

**(c) final session = Supabase.instance.client.auth.currentSession;**

* This checks if a user is already logged in.
* If session != null → user is logged in.
* If session == null → no one is logged in.

**(d) Redirect with Navigator.pushReplacement**

* pushReplacement removes the current page (Splash) and **replaces it** with the new one.
* If logged in → Go to HomeScreen().
* If not logged in → Go to LoginScreen().

Think of it like:

* 🔑 You already have the key → go inside the house (HomeScreen).
* 🚫 You don’t have the key → go to the guard (LoginScreen).

**4. The build() Method**

@override

Widget build(BuildContext context) {

return const Scaffold(

body: Center(child: CircularProgressIndicator()),

);

}

* While the redirect check is happening, the user just sees a **loading spinner**.
* This keeps the UI professional (instead of showing a blank screen).

**🔁 Flow of Execution**

1. App starts → **main.dart** shows SplashScreen.
2. initState() runs → calls \_redirect().
3. \_redirect() checks if the user has an active Supabase session.
4. If yes → Navigate to **HomeScreen**.
5. If no → Navigate to **LoginScreen**.
6. User only sees a loading spinner for a split second.

**🧠 Teacher’s Analogy**

Imagine you walk into a mall:

* First, there’s a **security gate (SplashScreen)**.
* The guard (Supabase check) looks at you:
  + If you already have a valid **membership card (session)** → you walk straight inside (HomeScreen).
  + If you don’t → you’re asked to go to the **reception (LoginScreen)** to get registered.
* While the guard checks, you’re waiting at the **security scanner (loading spinner)**.