# Managing login sessions in flutter

### Overview

Let's take the **authentication scenario** in an application .

When the user first logins to an application, then the application manages his session, which means that even when the user closes and opens the application his session is still there so that he is directed to the dashboard of that application.

This is **essential for good user experience**, because the user will get frustrated for logging in every time to use that application.

Let's see a step by step guide on how to **manage persistent sessions** 

#### Store auth token

Initially when the user installs and opens the application we shall route him to the login page to create a session.

Here when the login api is called we should store the **Authentication Token** (or) User Object that comes from the Login response to the Local Storage. We could use **shared\_preferences** in this case.

Here **shared\_preferences** is the local storage package used to store **key value pairs** in flutter .

## How to use shared\_preferences

After adding the package shared\_preferences to your pubspec.yaml file in your flutter project, you can now fetch a value that is stored in local using its key. If there isn't any value corresponding the key then the package will return a null. Similary for storing a value, you can mention the key to store a value.

### Code example

```
// Inititate the SharedPreferences
final prefs = await
SharedPreferences.getInstance();
// Store a value
prefs.setString('token', 'your_auth_token');
// Retrieve a value
prefs.getString('token');
```

## Check session when user opens the application

Since the **authentication token** is stored in your local storage, you now check whether **there is a active session**. You can **route the user** accordingly. This function can be performed in **initState of Splash Screen**.

### Code example

```
String? token = prefs.getString('token');

if(token == null){
    // No Session Found
    // Go to login page
}else{
    // Session found
    // Go to dashboad page
}
```

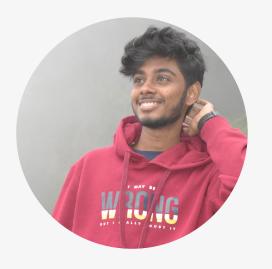
### Check if token is valid

Once if you find an active session you can further check whether the **token is still valid** .

Because the every authentication token might **have a expiry time** depending on the backend implementation. It is a must to check the token's validity using an **seperate API** provided by your backend team.

### Summary

- Check for any Authentication token stored in local storage when the user opens the application
- If there is no token stored then route the user to the login page
- Once the login API is successful then store the Authentication token to your local storage.
- If there is already an Authentication token available in local storage, then **check its validity** using an API
- If the token is valid, route to the dashboard page or else route to the login page.
- Don't forgot to clear your Authentication tokens from local storage when the user logs out the application.



## Was This Helpful?

Be sure to save this post so you can come back to it later!