Building a login flow with .NET MAUI

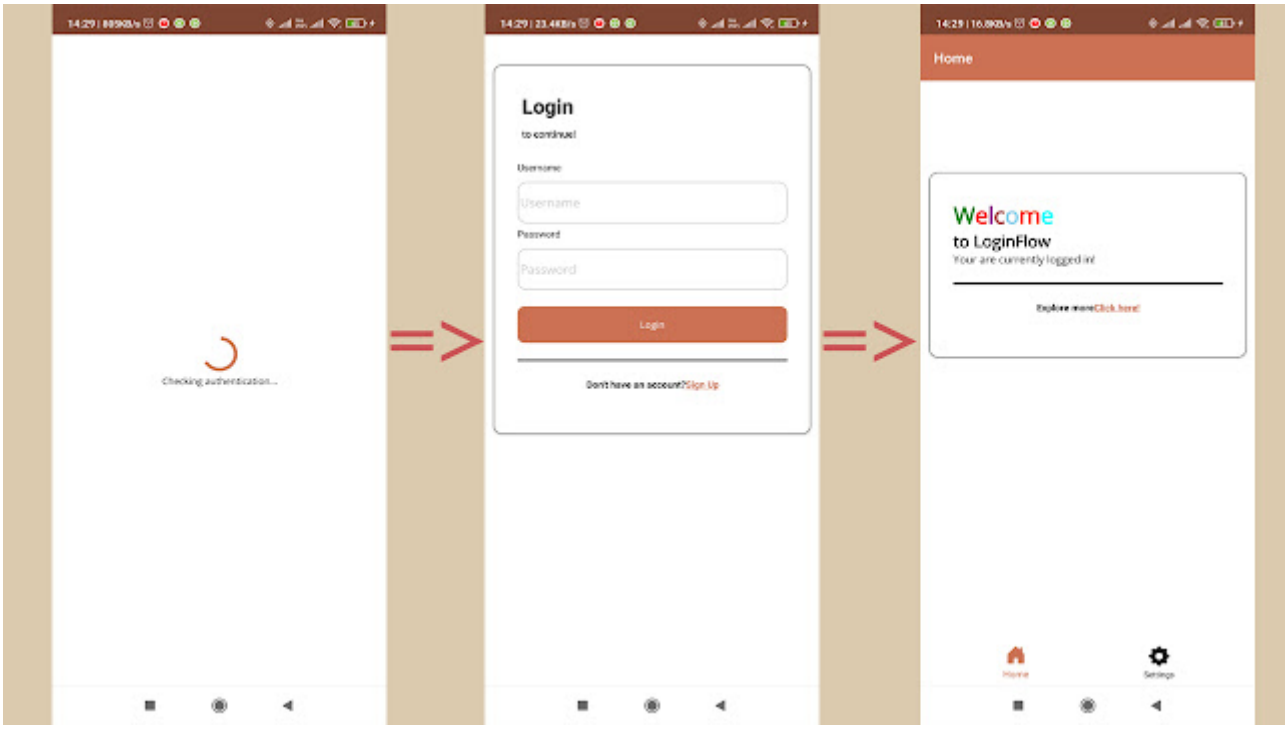
Pritish Ranjan Jan 16, 2023 44.8k 1 2

In this blog, let's build a Login Flow with .NET MAUI with Shell.

Authentication in any mobile app is very common. Let's get started with this. It's obvious that it should ask for login only if it isn't authenticated. We will check for authentication if not there we will move to Login page. If login is successful we will move to the Home page. For this example, we will override the back button pressed event to quit the application but you can customize it accordingly as per your need.

For this post, I am using a simple authentication but you can use JWT or any method you want.

Here is an example of the login flow:



All the pages that have to be used need to be registered with Shell.

If you are a bit familiar with the Shell navigation the first content page is the one that is displayed after startup. So we need to structure the shell accordingly in order.

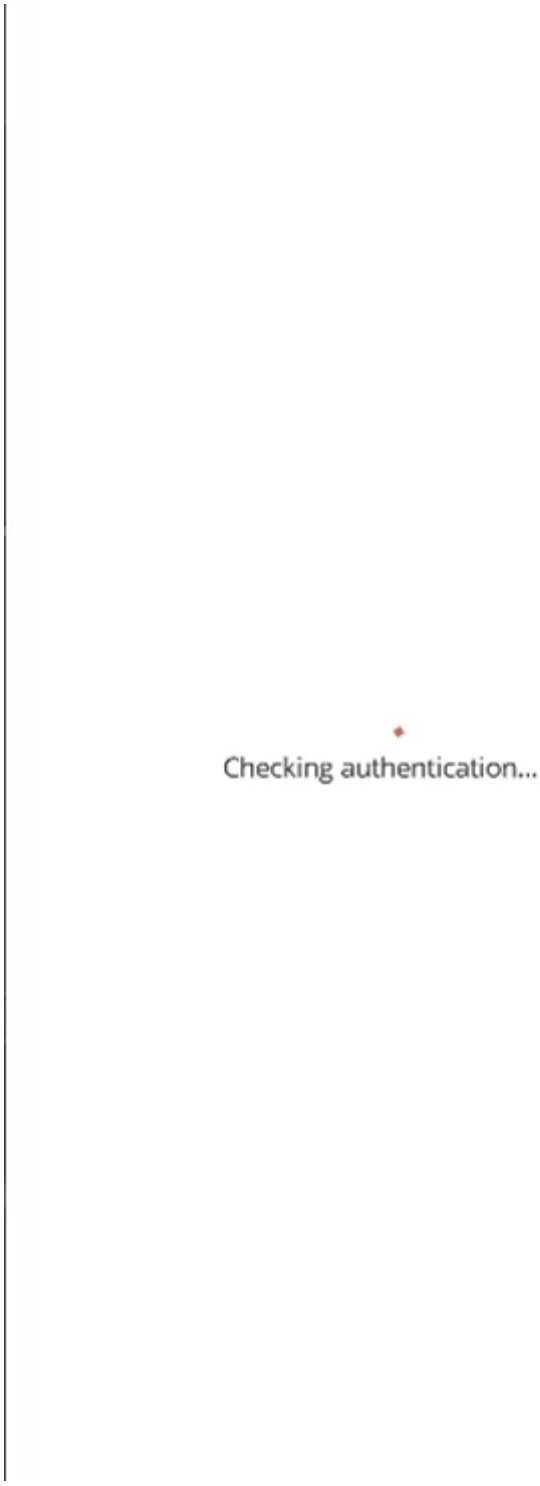
The pages we are using here for the example:

Here is the **AppShell.xaml**

```
<?xml version="1.0" encoding="UTF-8" ?>
<Shell
    x:Class="LoginFlow.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:LoginFlow"
    xmlns:views="clr-namespace:LoginFlow.Views"
    Shell.FlyoutBehavior="Disabled">

    <!--Loading Page-->
    <ShellContent
        Title="Home"
        ContentTemplate="{DataTemplate views:LoadingPage}"
        Route="loading" />

    <!--Login Page-->
    <ShellContent
        Title="Login"
        ContentTemplate="{DataTemplate views:LoginPage}"
        Route="login"/>
    <!--Main Page-->
    <TabBar>
        <Tab Title="Home" Icon="house_door_fill.svg">
            <ShellContent
                Icon="house_door_fill.svg"
        Title="Home"
        ContentTemplate="{DataTemplate views:HomePage}"
        Route="home" />

        </Tab>
        <Tab Title="Settings" Icon="gear_fill.svg">
            <ShellContent
                Icon="house_door_fill.svg"
        Title="Settings"
        ContentTemplate="{DataTemplate views:SettingsPage}"
        Route="settings" />
```

Register all the routes

```
public partial class AppShell : Shell
{
    public AppShell()
    {
        InitializeComponent();
        //Register all routes
        Routing.RegisterRoute("login", typeof(LoginPage));
        Routing.RegisterRoute("main", typeof(MainPage));
        Routing.RegisterRoute("home", typeof(HomePage));
        Routing.RegisterRoute("settings", typeof(SettingsPage));
    }
}
```



The loading screen checks if the user is authenticated or not. In this example, we will check for a key in SecureStorage. If we found data for that key we will authenticate it and return true if not we will return false.

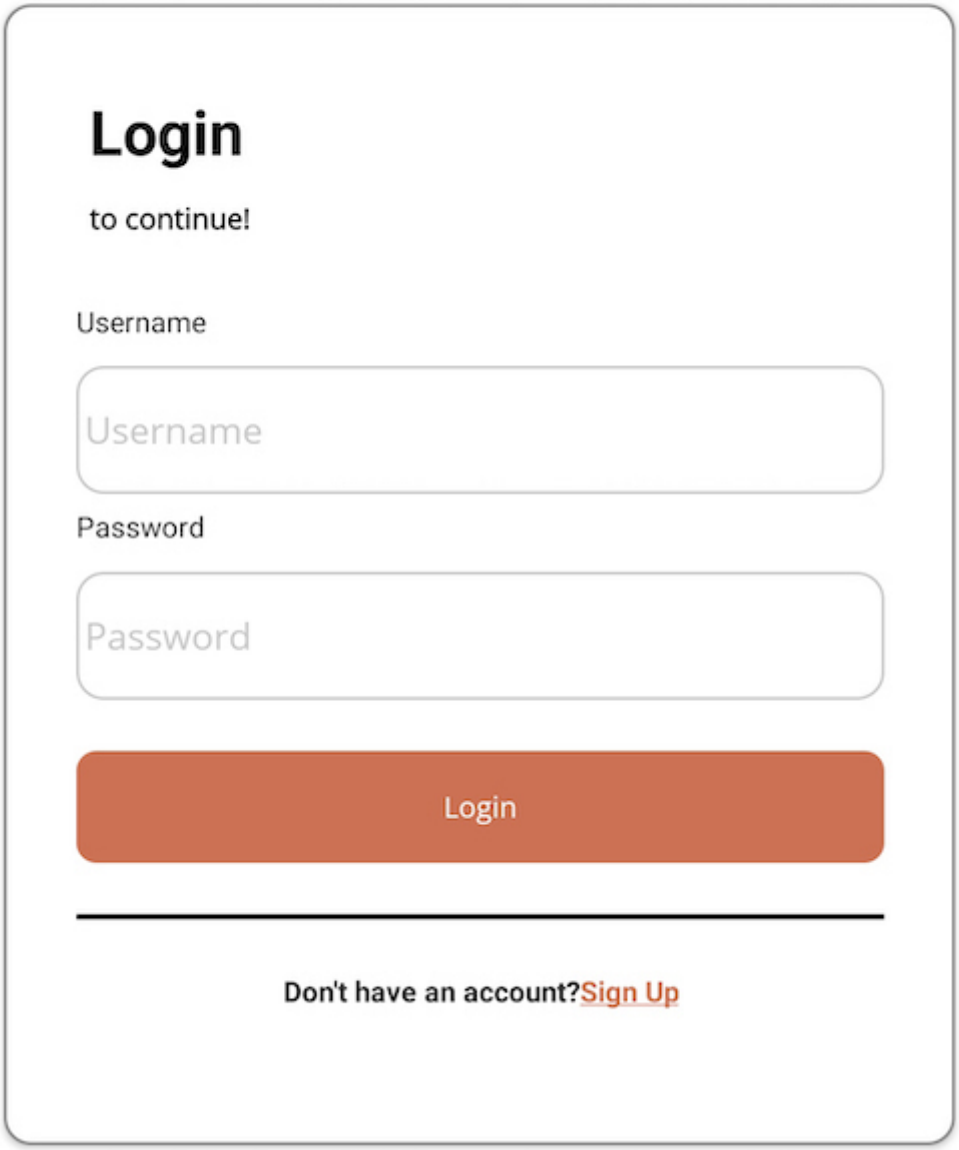
So we can override the OnNavigatedTo method to check if the app is authenticated or not. Here is the code which can be written in the page or in the view model.

```
    }
    else
    {
        await Shell.Current.GoToAsync("login");
    }
    base.OnNavigatedTo(args);
}

async Task<bool> isAuthenticated()
{
    await Task.Delay(2000);
    var hasAuth = await SecureStorage.GetAsync("hasAuth");
    return !(hasAuth == null);
}
```

The back button pressed event

```
protected override bool OnBackButtonPressed()
{
   Application.Current.Quit();
   return true;
}
```

The Login page



When the app isn't authenticated it goes to the login page. I haven't placed any validation for the controls but it can be implemented. I have hardcoded the username and password to authenticate for simulating the login flow.

Here is the login page with one button and two entry controls for Username and Password.

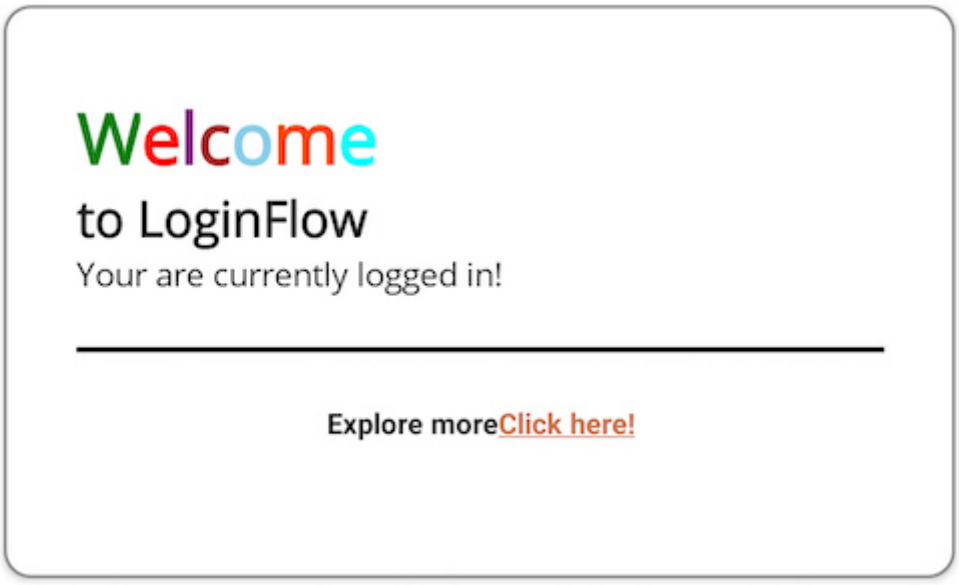
Here is the click event:

```
    private async void LoginButton_Clicked(object sender, EventArgs e)
    {
        if (IsCredentialCorrect(Username.Text,Password.Text))
        {
```

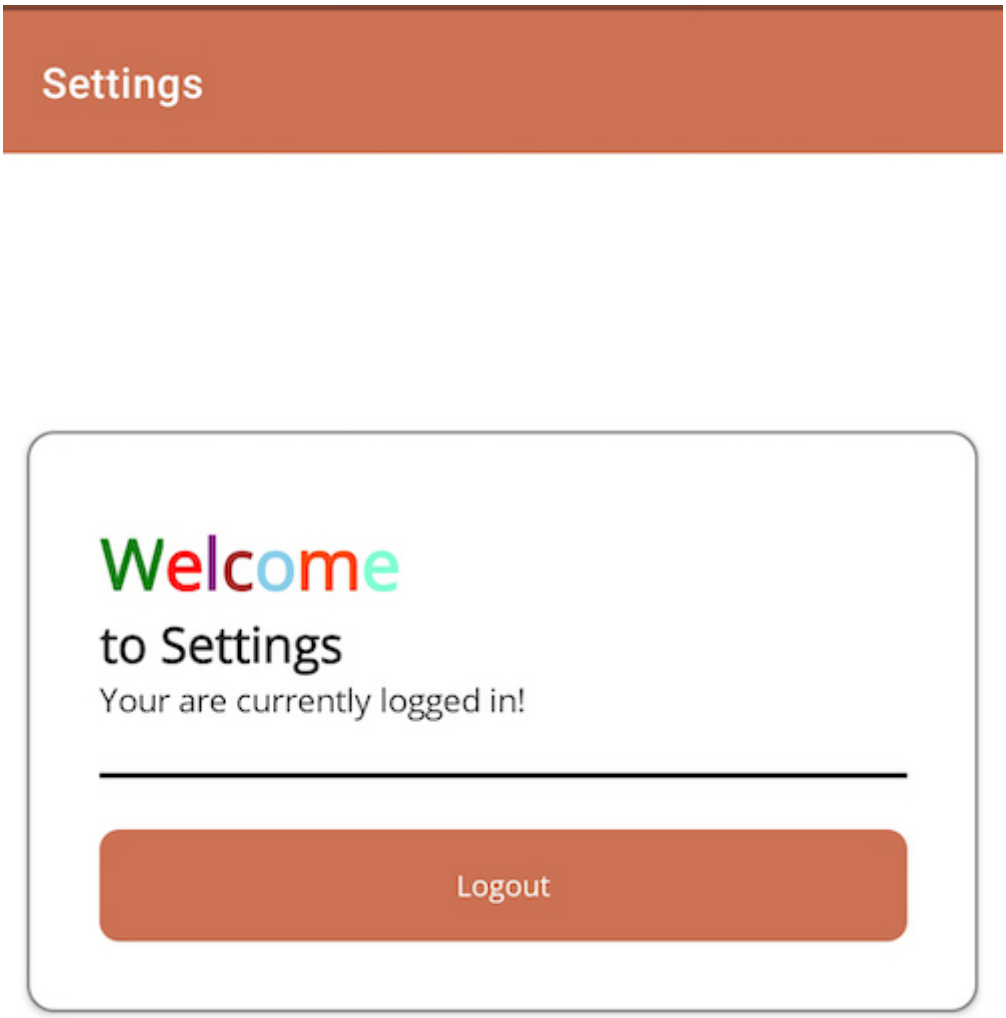
```
            await DisplayAlert("Login failed", "Uusername or password if inv
        }
    }
```

If the username and password are correct we can navigate to HomePage.

The Home Page



The settings page



In the settings page, we can implement the logout logic. In the logout button click we can write like this.

```
private async void LogoutButton_Clicked(object sender, EventArgs e)
{
    if (await DisplayAlert("Are you sure?", "You will be logged out.", "
    {
        SecureStorage.RemoveAll();
        await Shell.Current.GoToAsync("///login");
    }
}
```

Conclusion

It is a very small example for building a login flow in .NET MAUI. You can check out the repository on GitHub

How To Change Status Bar Color In .NET MAUI

Pritish Ranjan

.NET enthusiastic. Data Engineer. https://www.pritishranjan.com

https://www.pritishranjan.com

2007 **72k**

View All Comments

1

Type your comment here and press Enter Key (Minimum 10 characters)