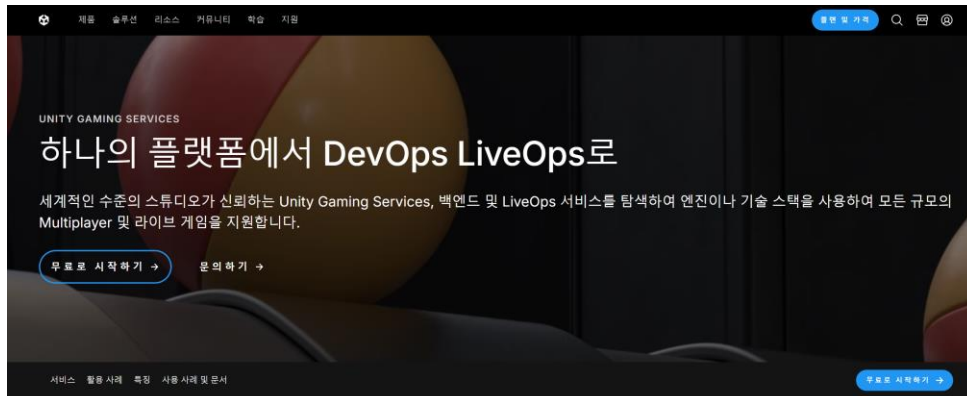


Unity Gaming Service (UGS)



라이브 게임을 위한 완벽한 서비스 생태계



기본 구축

계정

플랫폼 전반 플레이어 로그인 및 Authentication 시스템을 활성화합니다.

멀티플레이어

확장성과 성능의 Multiplayer 게임 호스팅을 얻으십시오.

콘텐츠 관리

게임 콘텐츠를 배포하고 관리합니다.

DevOps

게임 개발을 위한 Version Control 및 Build Automation 탐색합니다.



플레이어 참여 유도

Analytics

Analytics 도구를 사용하여 데이터 기반 의사 결정을 추진합니다.

플레이어 참여

실시간 및 장기간 도구로 플레이어 경험을 최적화합니다.

커뮤니티

확장 가능한 음성 및 텍스트 채팅을 통해 플레이어들을 연결하고 안전하게 유지하십시오.

크래시 리포트

중독 보고 및 조사 도구를 사용하여 게임 안정성을 향상시킵니다.



게임 성장시키기

수익화

게임 내 광고로 수익 창출

사용자 확보

고객의 성장에 적합한 사용자 찾기.

광고 중재

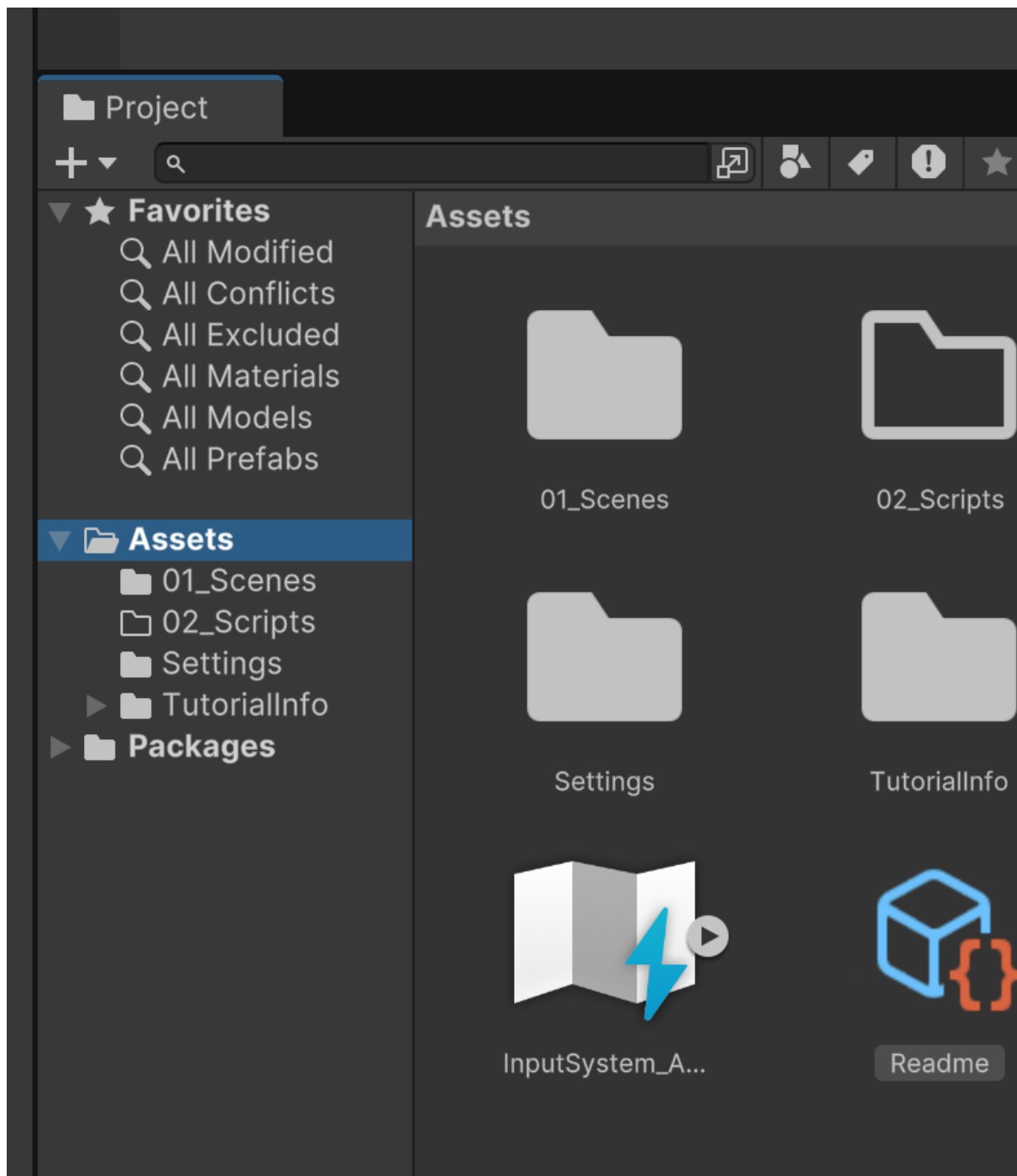
광고 수요를 높이고 게임에서 더 많은 수익을 창출하십시오.

퍼블리싱

모바일 게임 아이디어를 비즈니스로 구축합니다.

게임 Economy

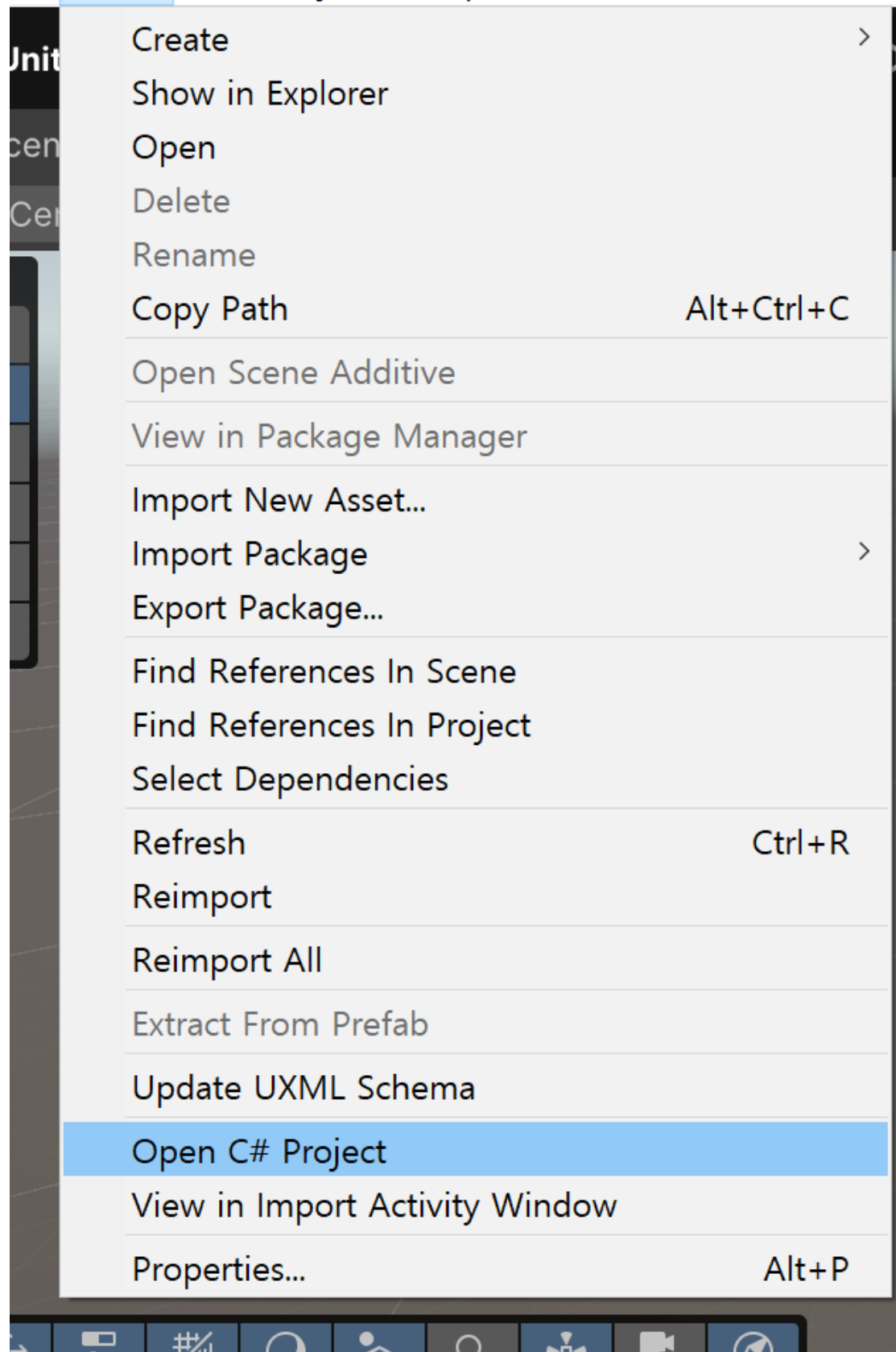
게임 내 Economy 인프라를 설계하고 앱 내 구매를 추가합니다.



폴더 정리

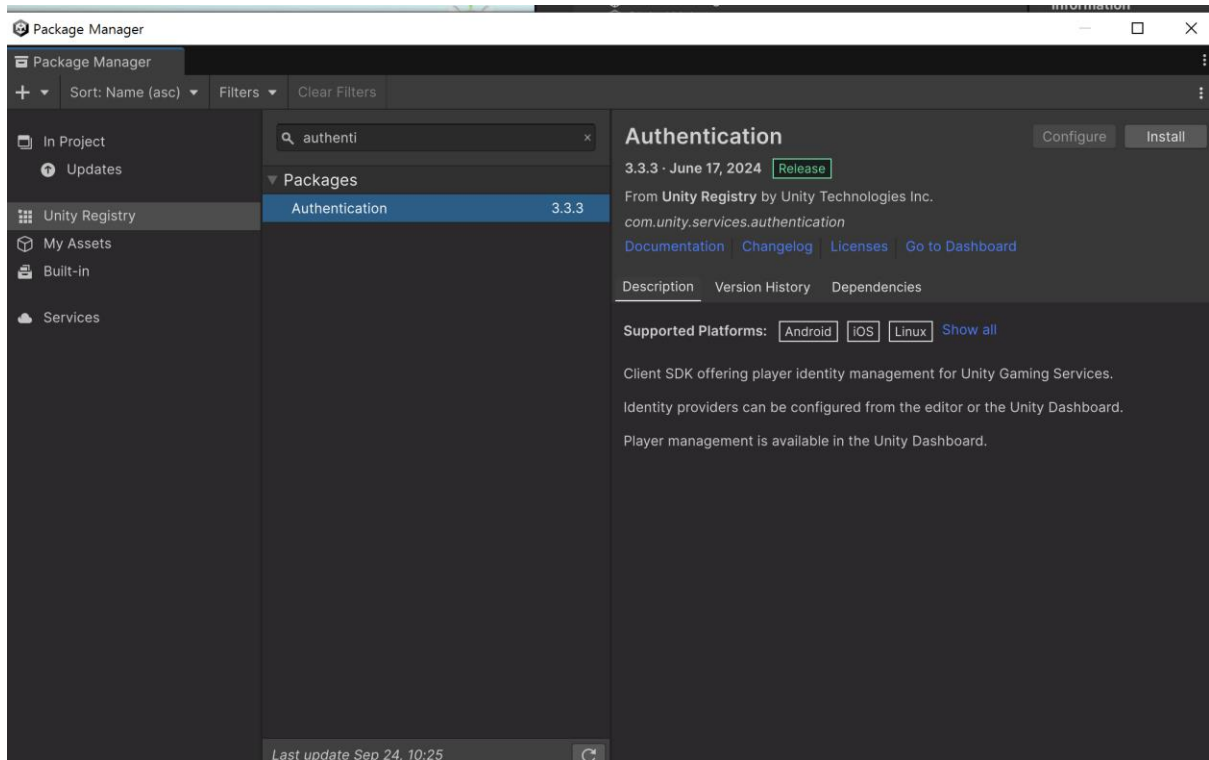
GDemo - 01_Anymouse - Windows, Mac, Linux - Unity 6

dit **Assets** GameObject Component Services Jobs Wind

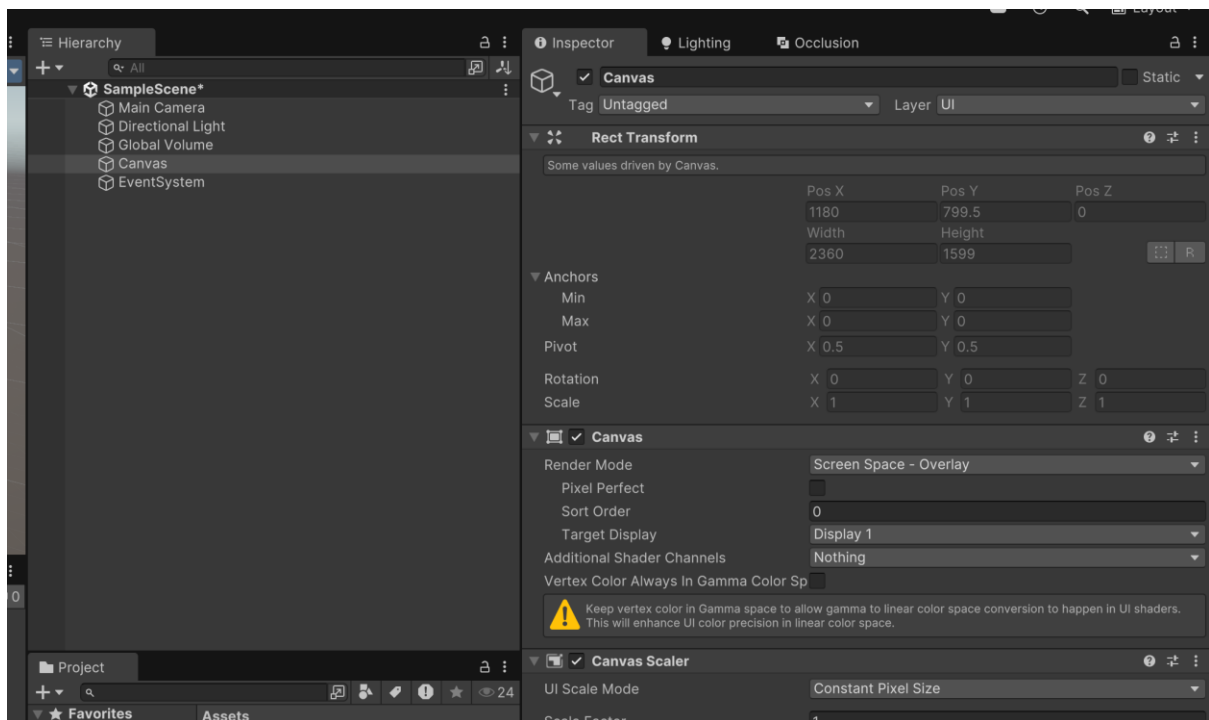
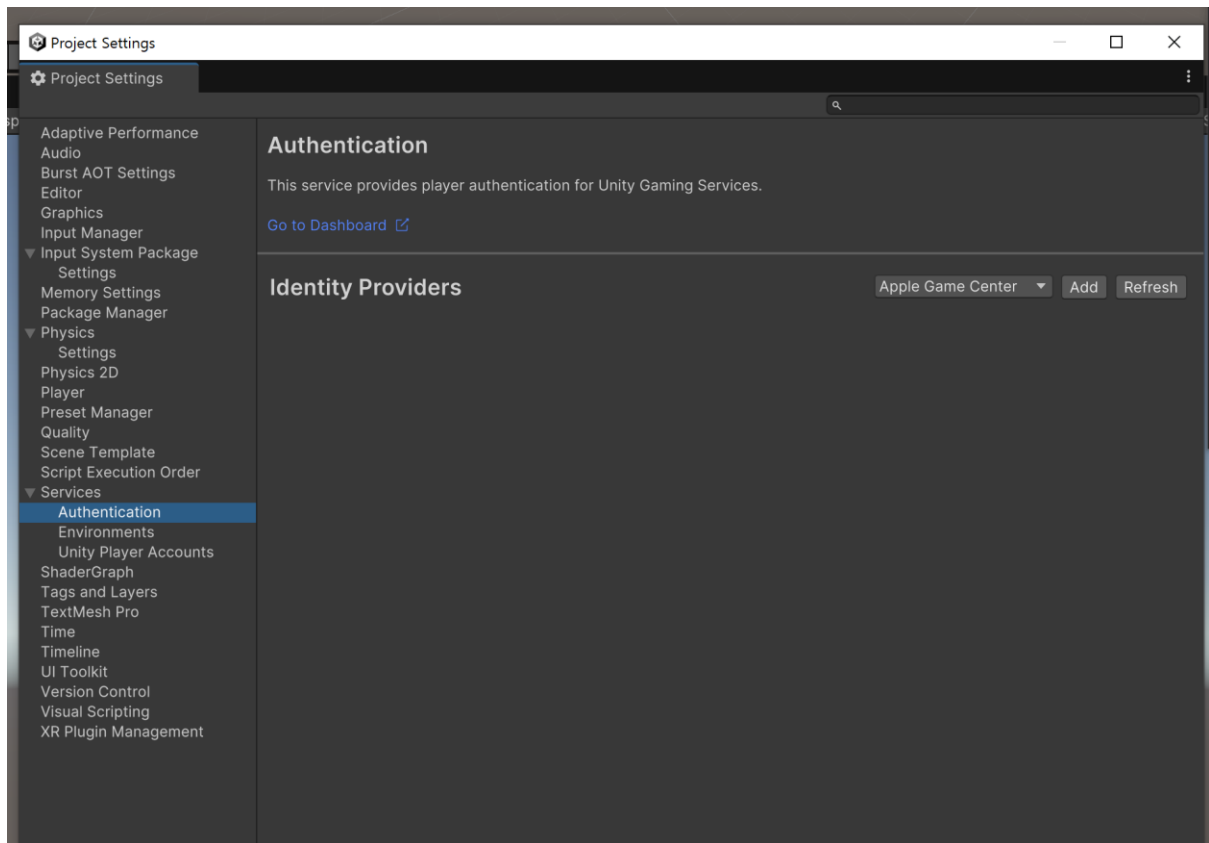


C# 프로젝트 열기

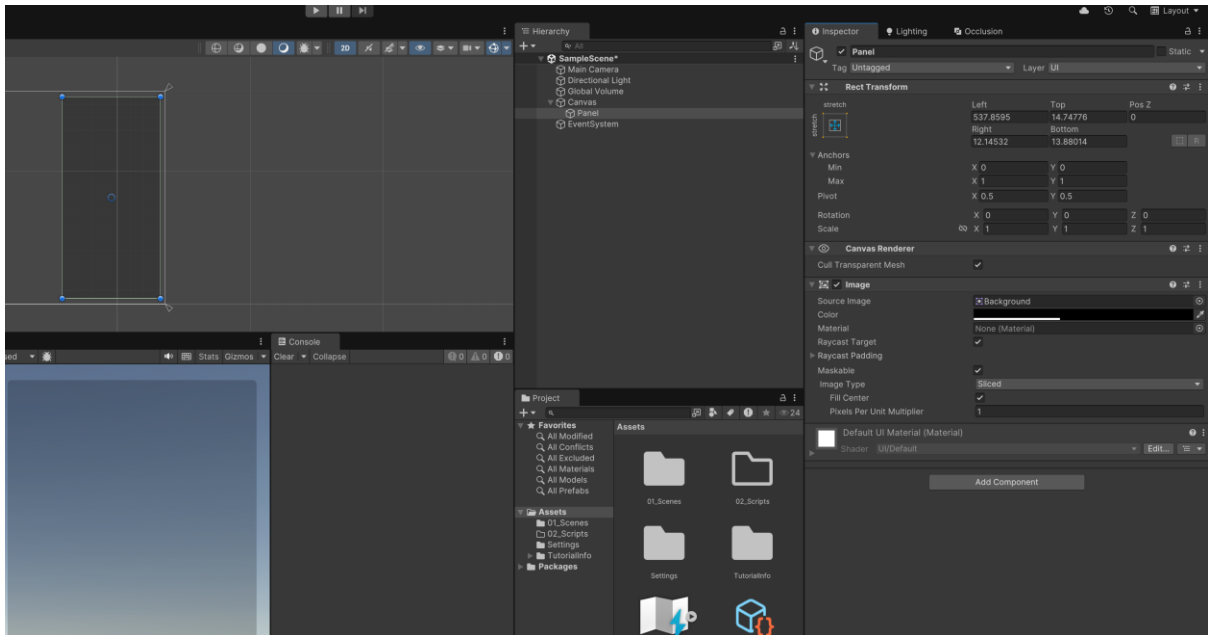
.gitignore, .gitattributes 설정



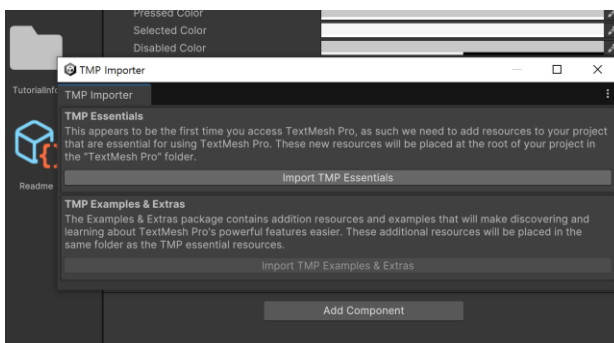
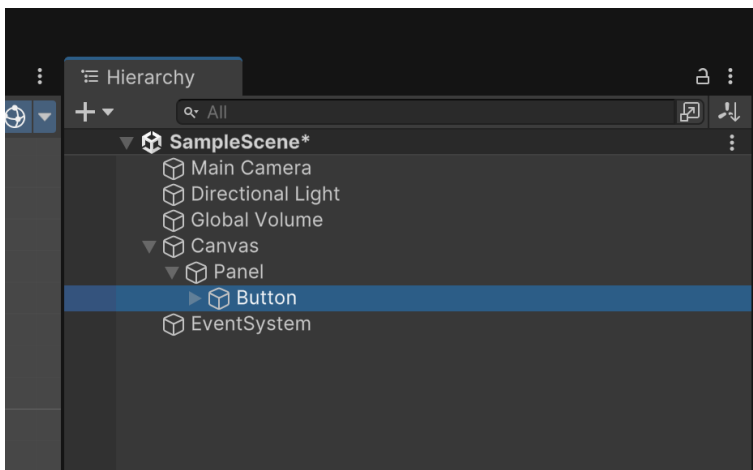
Authentication install



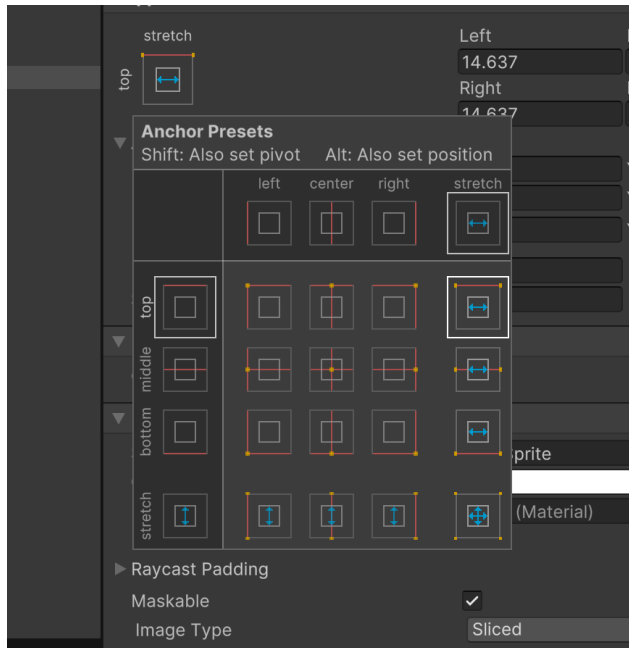
캔버스 추가



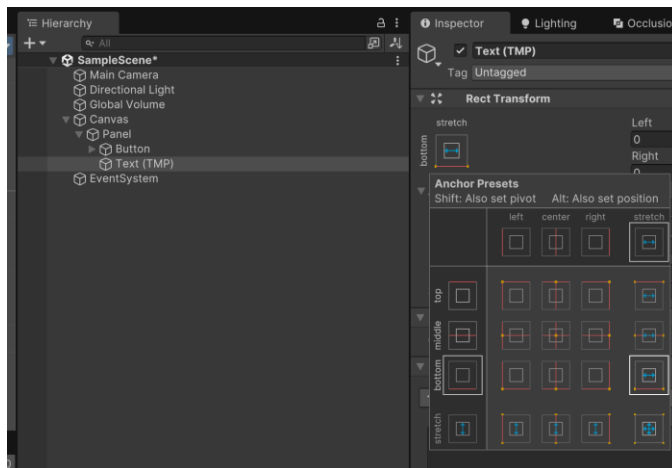
버튼 추가



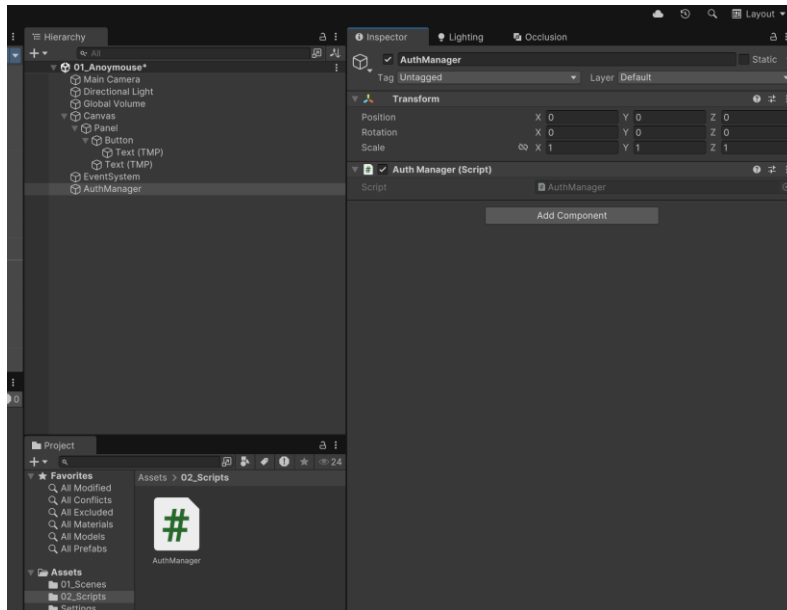
임포트



앵커포인트 조절

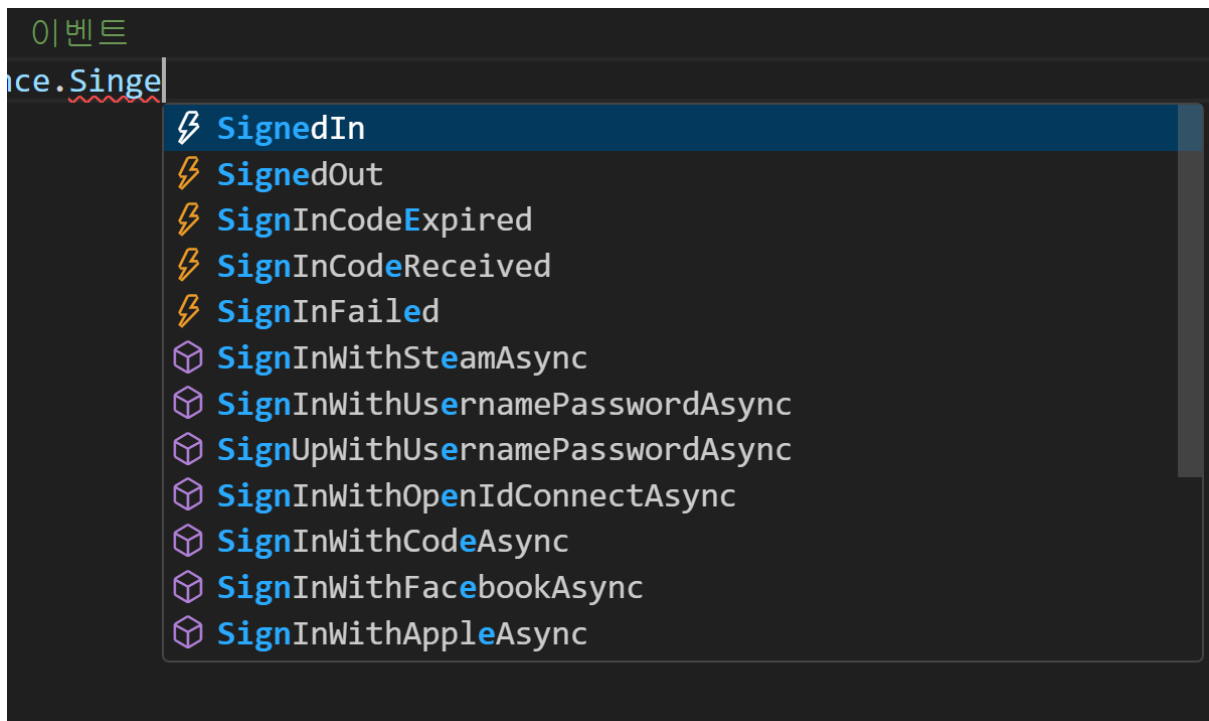


텍스트 추가 후 바텀스트래치

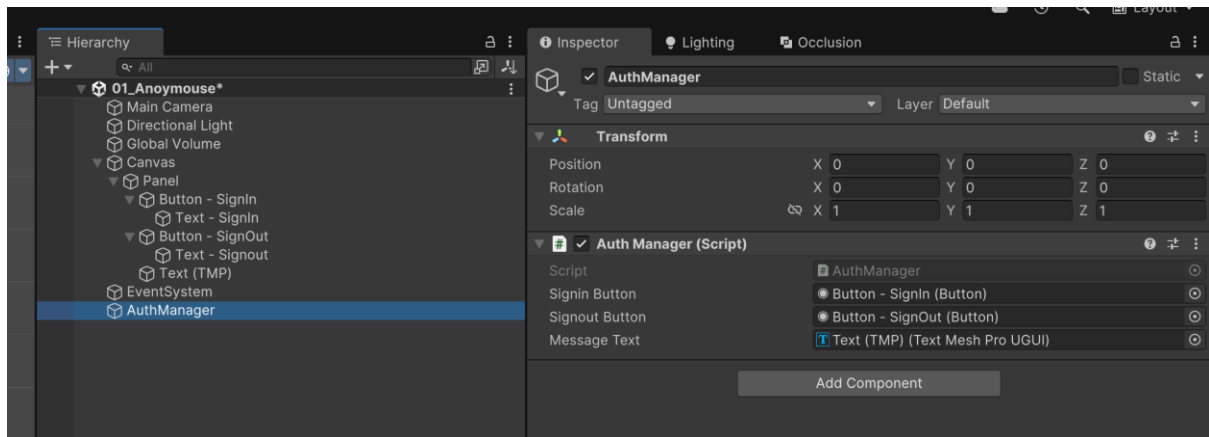


스크립트와 빈게임오브젝트로 AuthManager 생성 후 추가

이후에 코딩



번개표시 - 이벤트



Signout 버튼 추가하기

```
private async void Awake()
{
    // Awake 함수를 멀티쓰레드로 사용하겠다.
    // USG 초기화
    await UnityServices.InitializeAsync(); // 관례상 async 함수들은 접미사로 다 붙인다

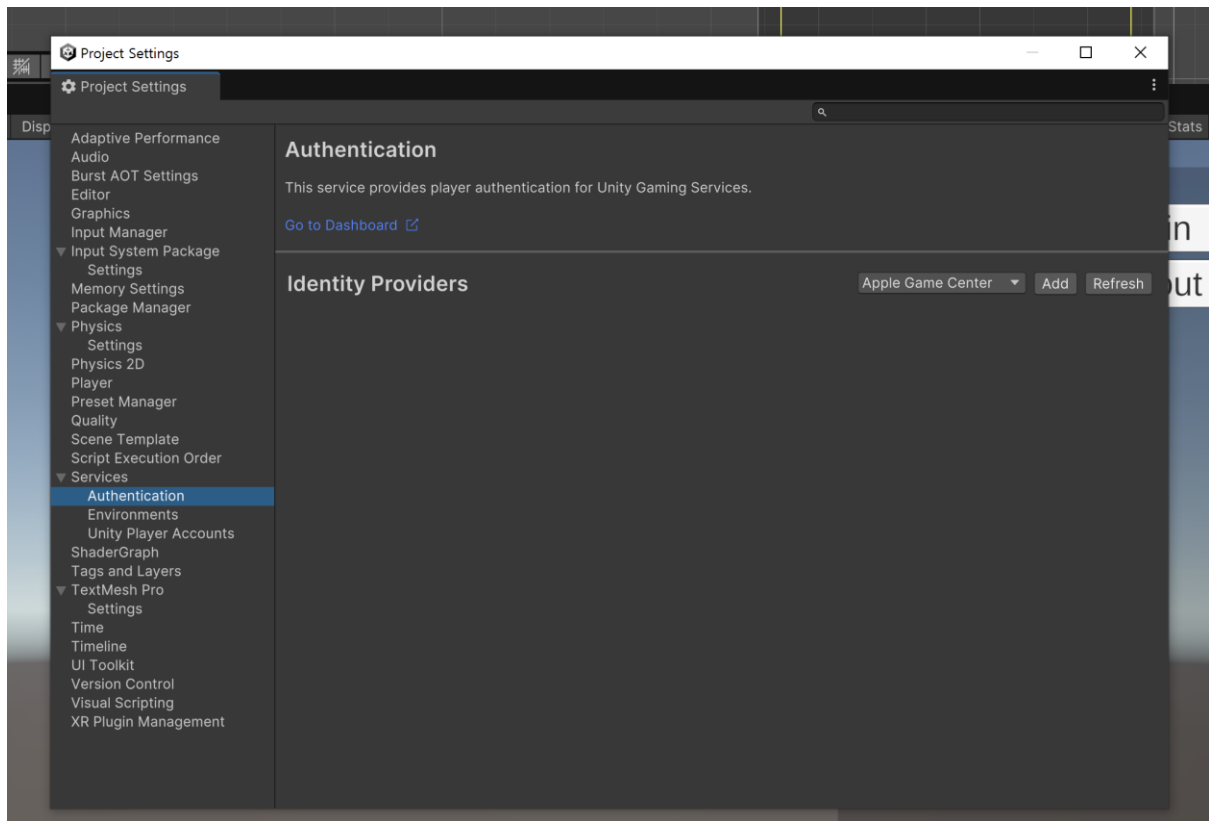
    // 이벤트 초기화
    EventConfig();

    // 버튼 이벤트 연결
    signinButton.onClick.AddListener(async () =>
    {
        // 익명 로그인
        await SignInAsync();
        // 아래에 SignInAsync가 async 로 되어있어서 async 로 되어있는 람다식을 받겠다
    });

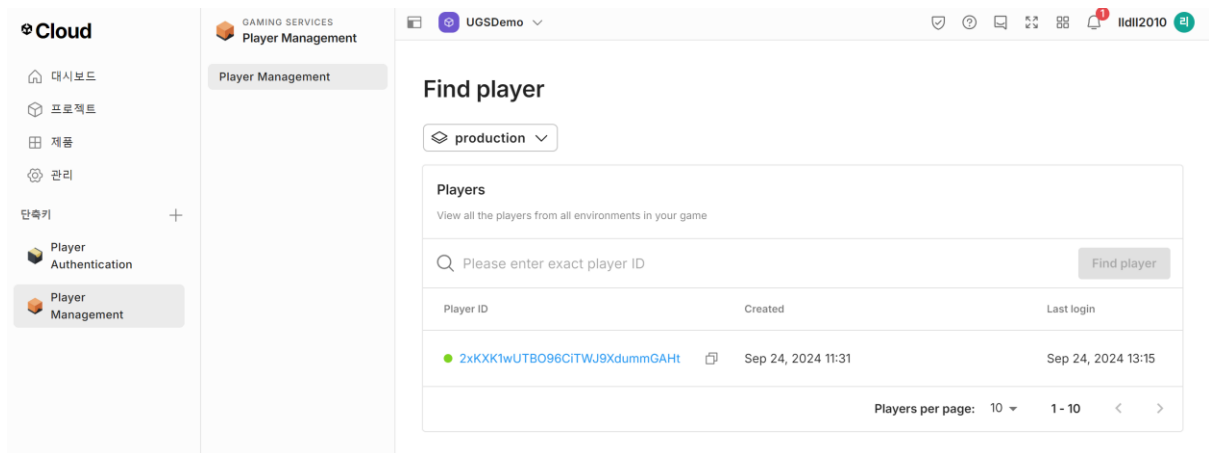
    // 로그아웃버튼 이벤트
    signoutButton.onClick.AddListener(() =>
    {
        // 로그아웃
        AuthenticationService.Instance.SignOut();
    });
}
```

You, 1초 전 • Uncommitted changes

함수 추가



Go to dash board



한 번 접속하고 나면 로그 남는다

playerName Inputfield 만들고 코딩

```

91
92     private async Task SetPlayerNameAsync(string playerName)
93     {
94         try
95         {
96             await AuthenticationService.Instance.UpdatePlayerNameAsync(playerName);
97         }
98         catch (AuthenticationException e)
99         {
100             Debug.Log(e.Message);
101         }
102     }
103 }
104
105
106

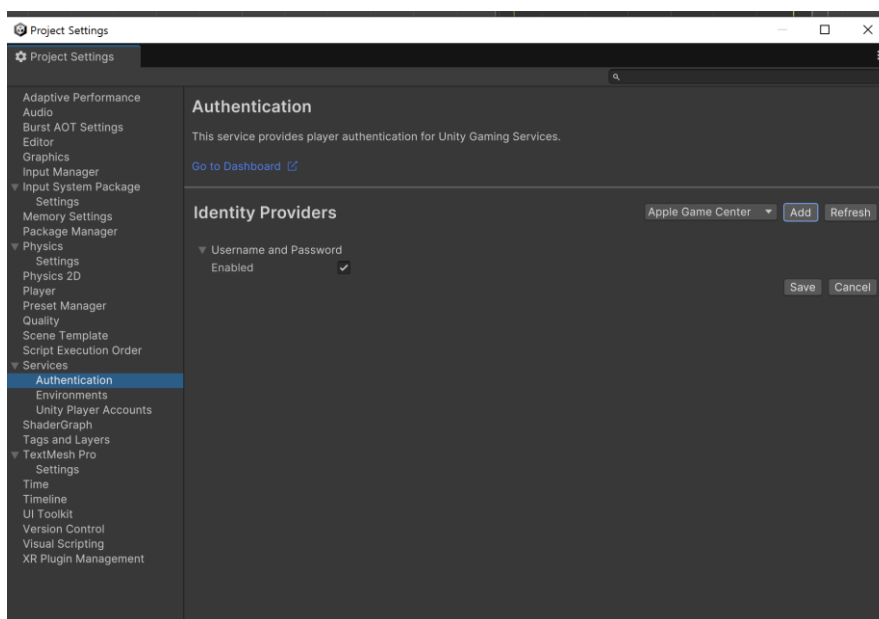
```

```

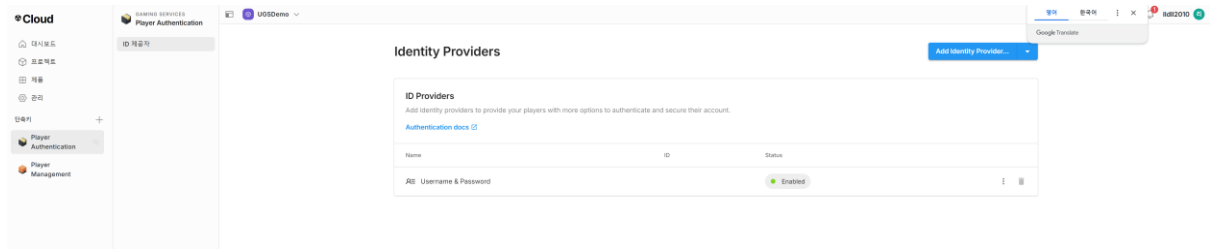
// 로그인 후
AuthenticationService.Instance.SignOut();
});
// 플레이어 이름 변경 버튼 이벤트 연결
/*
    50자
    공백 x
    Zack#1234 아이디 뒤에 4자리 난수 발생
*/
playerNameSaveButton.onClick.AddListener(async () =>
{
    await SetPlayerNameAsync(playerNameIF.text);
});
}

```

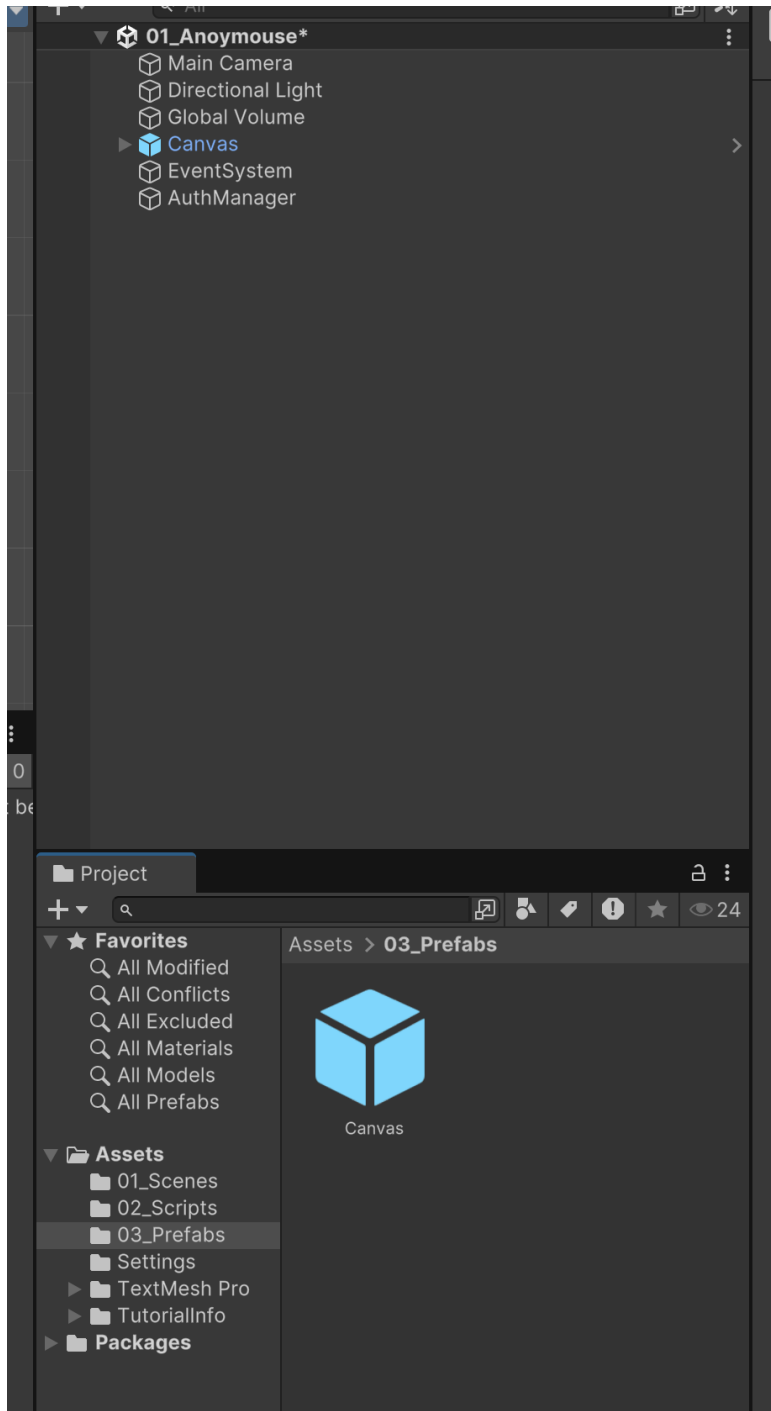
아이디, 비밀번호 입력해서 회원가입하게 만드는 것



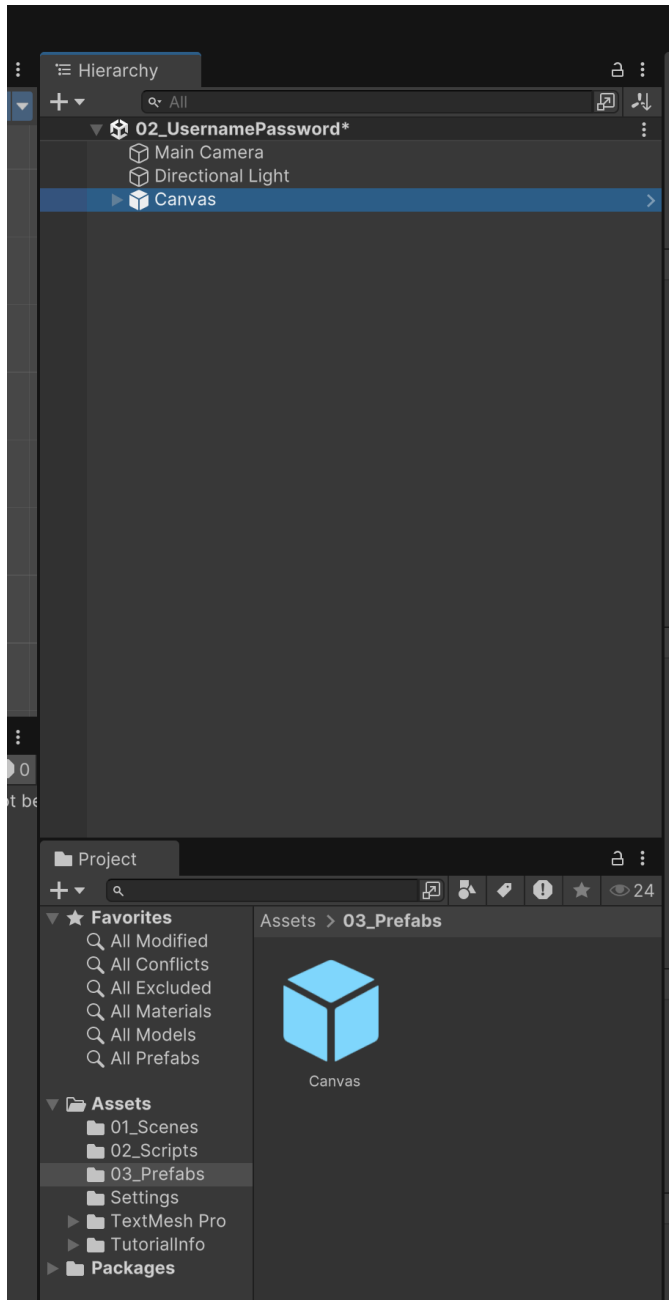
Add 해서 username and password 선택, Save 하고 Go to dashboard



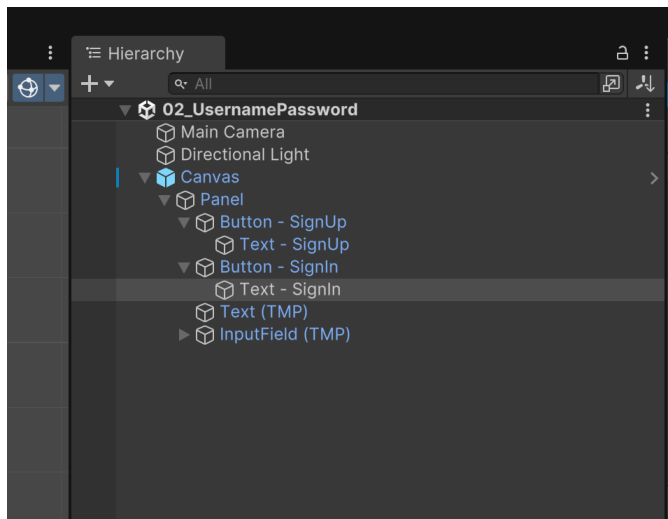
확인할 수 있음



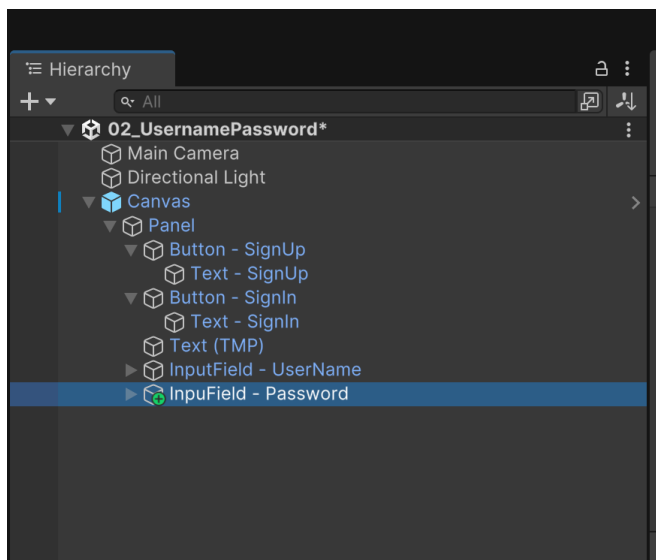
03_Prefabs 폴더 만든 후, canvas 옮겨서 프리팹화



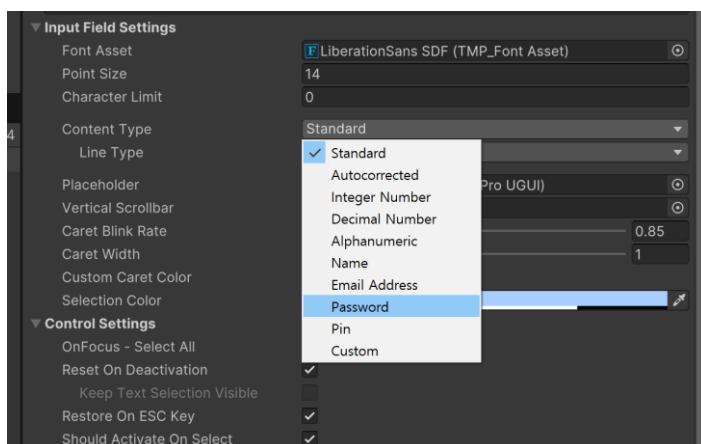
새 씬 만든 후에 프리팹 Canvas 끌고오기

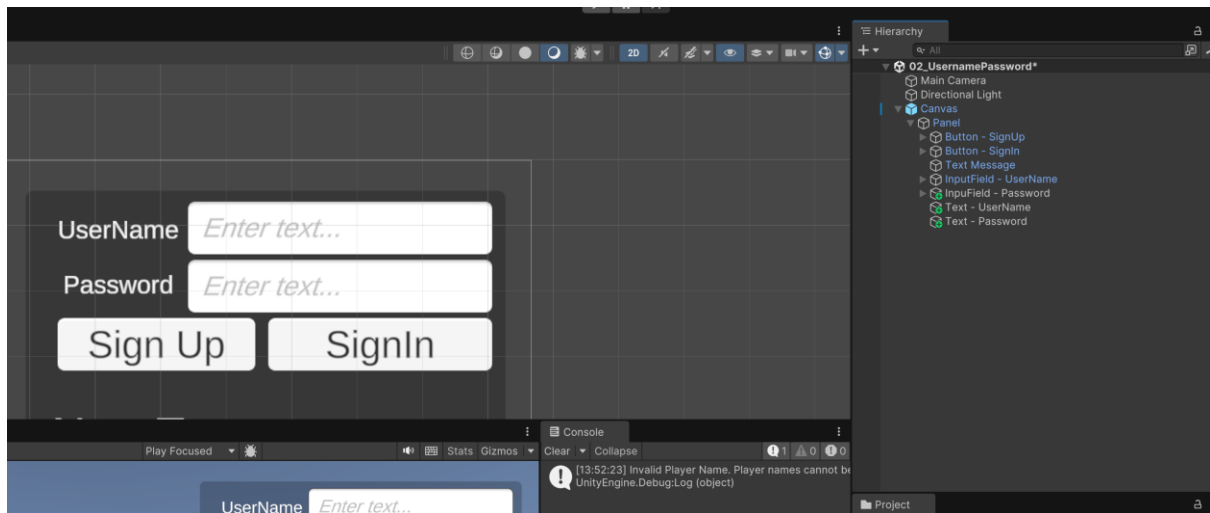


변경

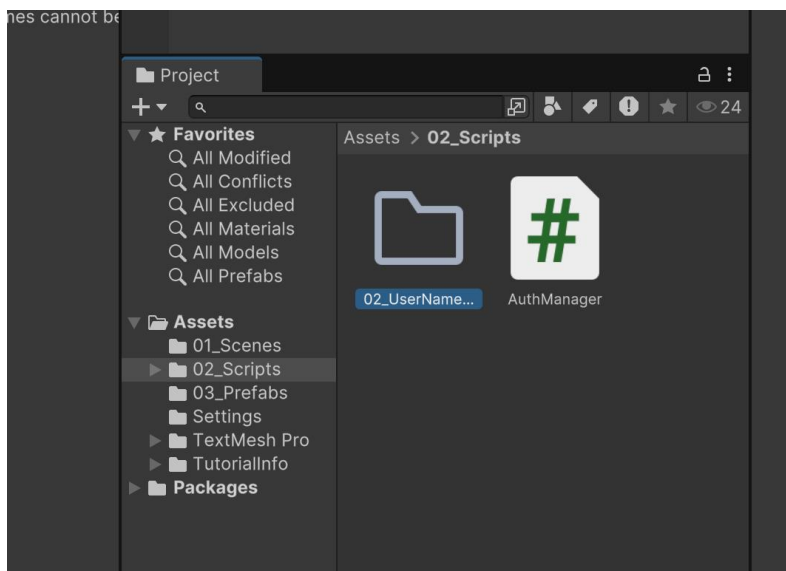


비밀번호 ***** 찍히게 하는 옵션

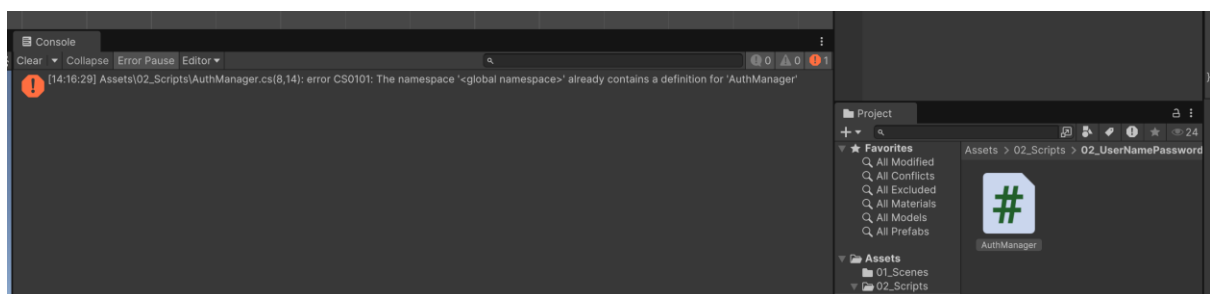




텍스트 추가 및 정리



AuthManager 또 하나 만들건데 이미 한 개 있기 때문에 따로 폴더를 정해서 하나 더 작성하기

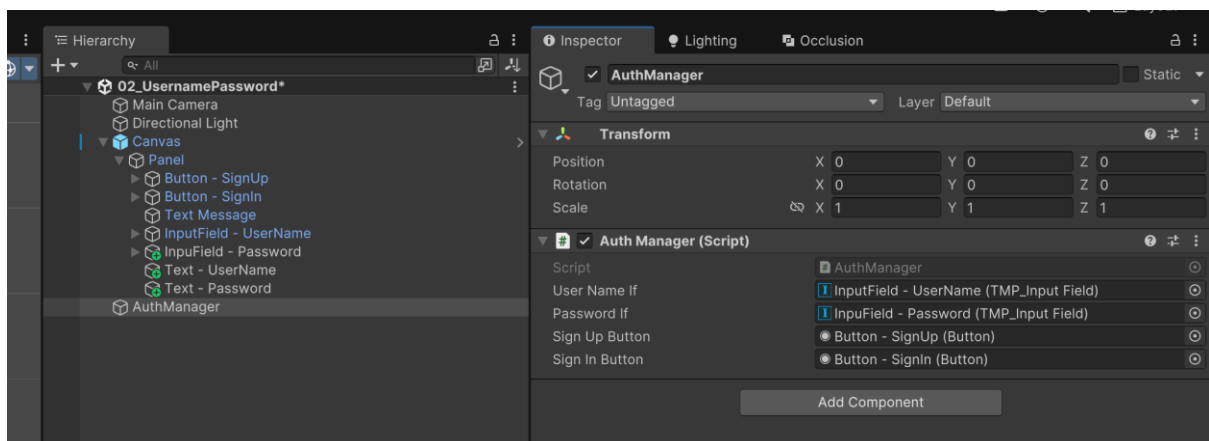


에러뜸

```
Assets > 02_Scripts > 02_UserNamePassword > AuthManager.cs > AuthManager > signInButton

1  using TMPro;
2  using UnityEngine;
3  using UnityEngine.UI;
4
5  namespace AuthUserNamePassword
6  {
7      public class AuthManager : MonoBehaviour
8      {
9          // 파라미터 2개를 동시에 선언
10         [SerializeField] private TMP_InputField userNameIf, passwordIf;
11         [SerializeField] private Button signUpButton, signInButton;
12     }
13 }
```

네임스페이스 namespace AuthUserNamePassword {}로 분리를 하면 동일한 이름의 스크립트를 사용할 수 있다.



버튼 연결해주기

```
// 회원가입
private async Task SignUpAsync(string userName, string password)
{
    try
    {
        await
AuthenticationService.Instance.SignUpWithUsernamePasswordAsync(userName,
password);

        Debug.Log("회원가입 완료");
    }
    // catch 2 번. 예외에 걸렸을 때
    catch (AuthenticationException e)
    {
        Debug.LogException(e);
    }
}
```



```

        // 로그인에 실패했을 때
        catch (RequestFailedException e)
        {
            Debug.LogException(e);
        }
    }
}

```

버튼연결

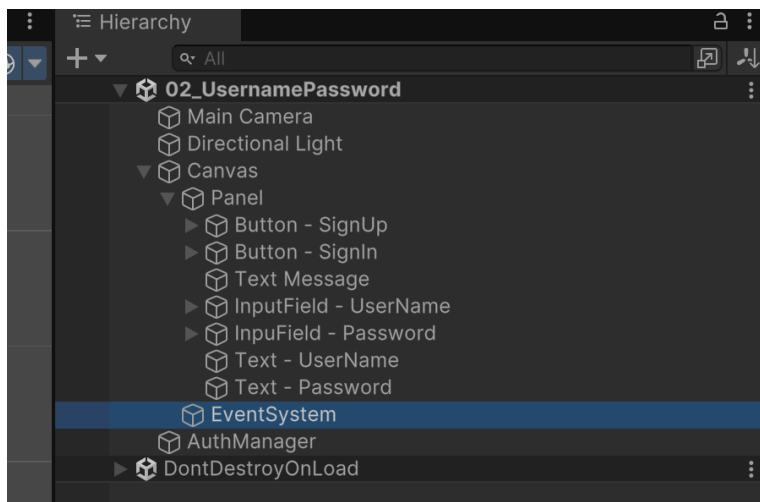
```

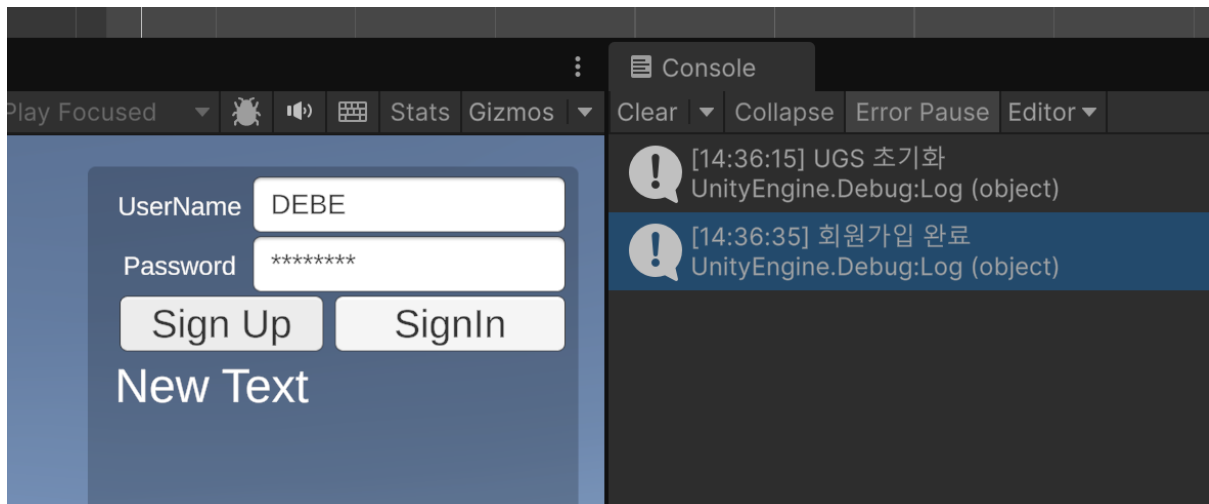
private async void Awake()
{
    // 제일 처음에 UGS 초기화를 위해서 await 쓰기
    await UnityServices.InitializeAsync();
    Debug.Log("UGS 초기화");

    signUpButton.onClick.AddListener(async () =>
    {
        await SignUpAsync(userNameIf.text, passwordIf.text);
    });
}

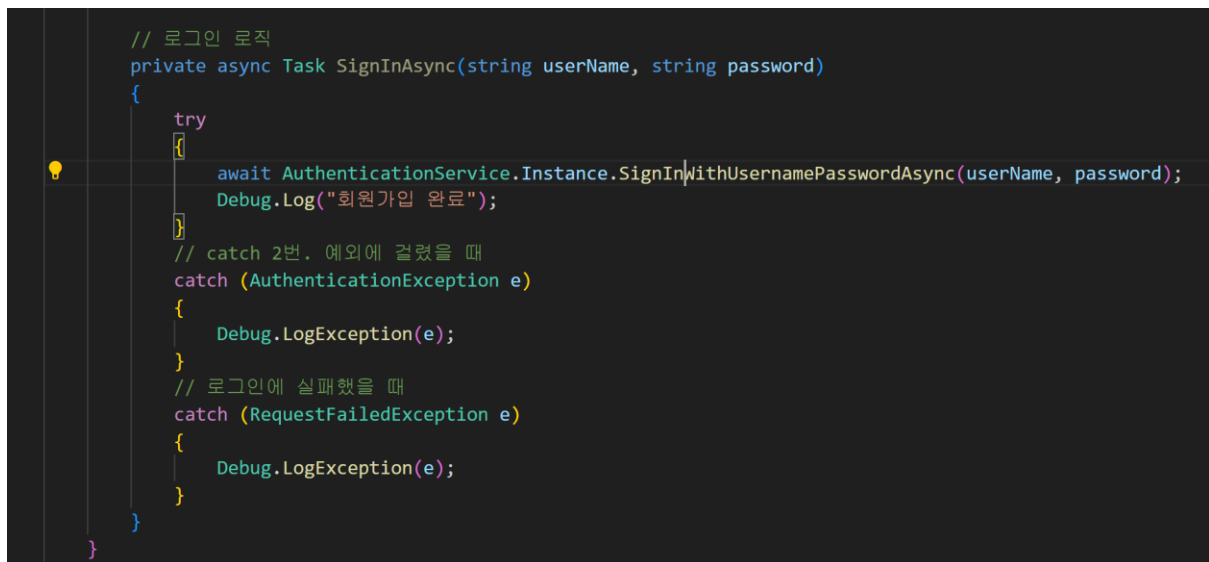
```

EventSystem 없으니까 추가해줘야 canvas 가 제대로 작동함

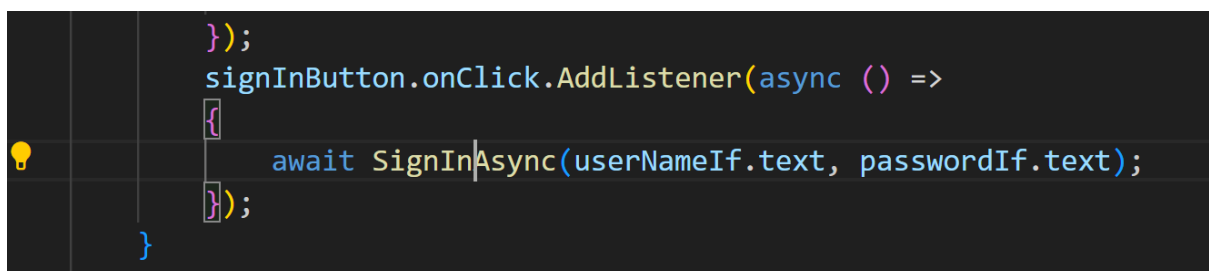




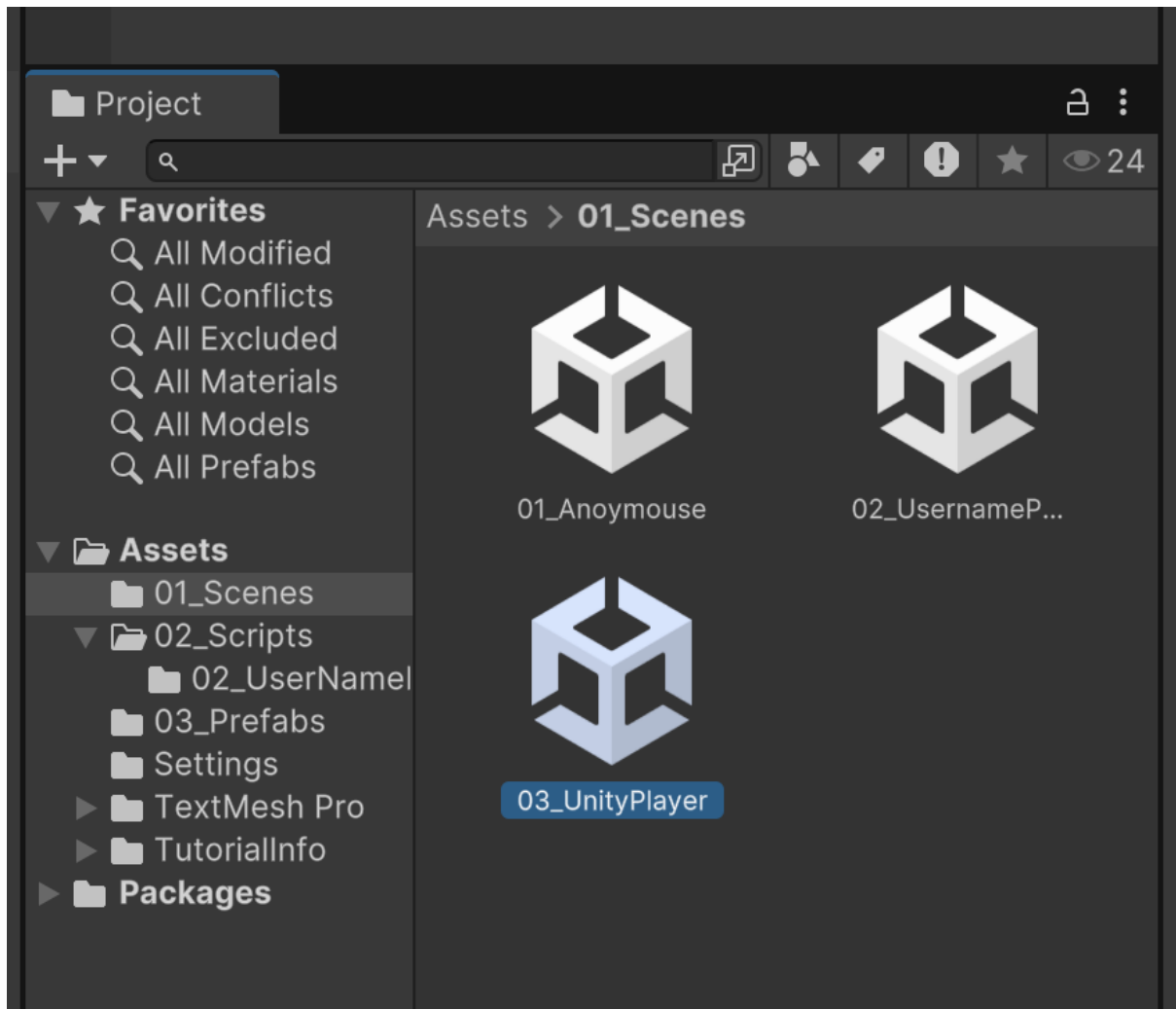
안되면 UGS 연결 되어있는지 확인하기



위에 SignUp 에서 복사해와서, SignIn 으로만 바꾸면 됨

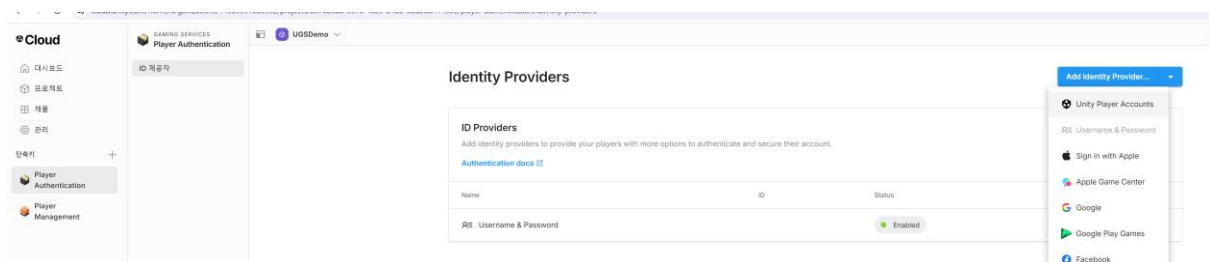


버튼도 동일하게




유니티에서 제공하는 로그인 통합서비스 만들기

03_UnityPlayer 씬 추가하기



Unity 선택

ie & Password

Add  Unity Player Accounts

OAuth client name ⓘ

Platform *


☐ iOS / Android


☐ PC









Links ⓘ

Cancel [Add provider](#)

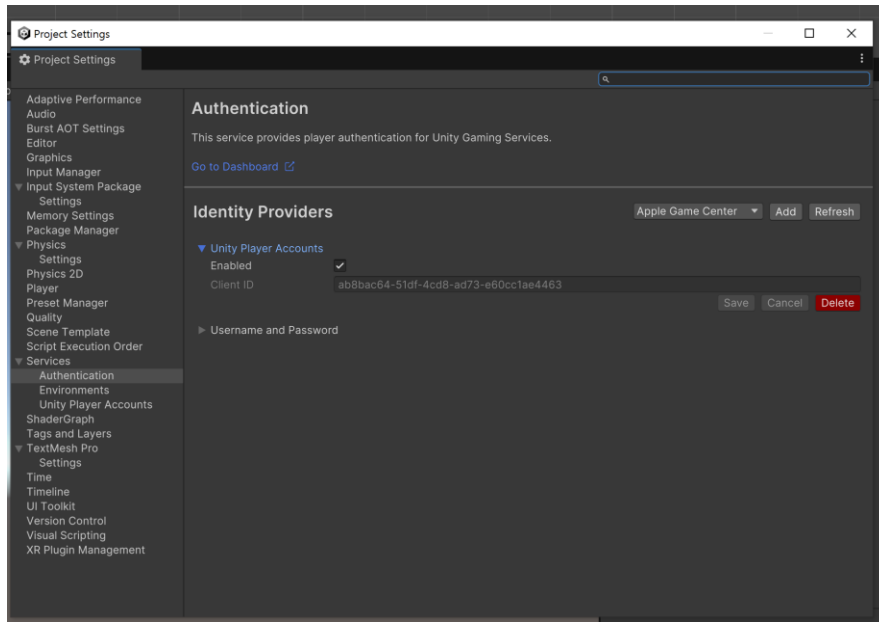
Platform 등 선택 후 add provider

 **Missing links**
Your Unity Identity Provider configuration is missing a Privacy Policy link, which is required for launch. Adding a Terms of Service link is recommended.

←  Unity Player Accounts [Delete provider](#)

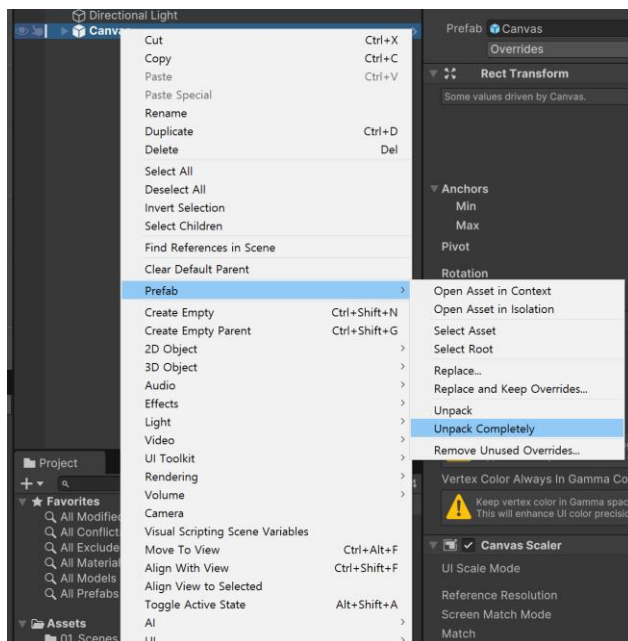
Provider details Unity Player Accounts setup documentation	OAuth client name ⓘ	UGSDemo	
	Client ID	ab8bac64-51df-4cd8-ad73-e60cc1ae4463	
	Platform	PC, iOS / Android	
	Terms of Service link ⓘ		
	Privacy Policy link ⓘ		
	Status	● Enabled	

ID 카피



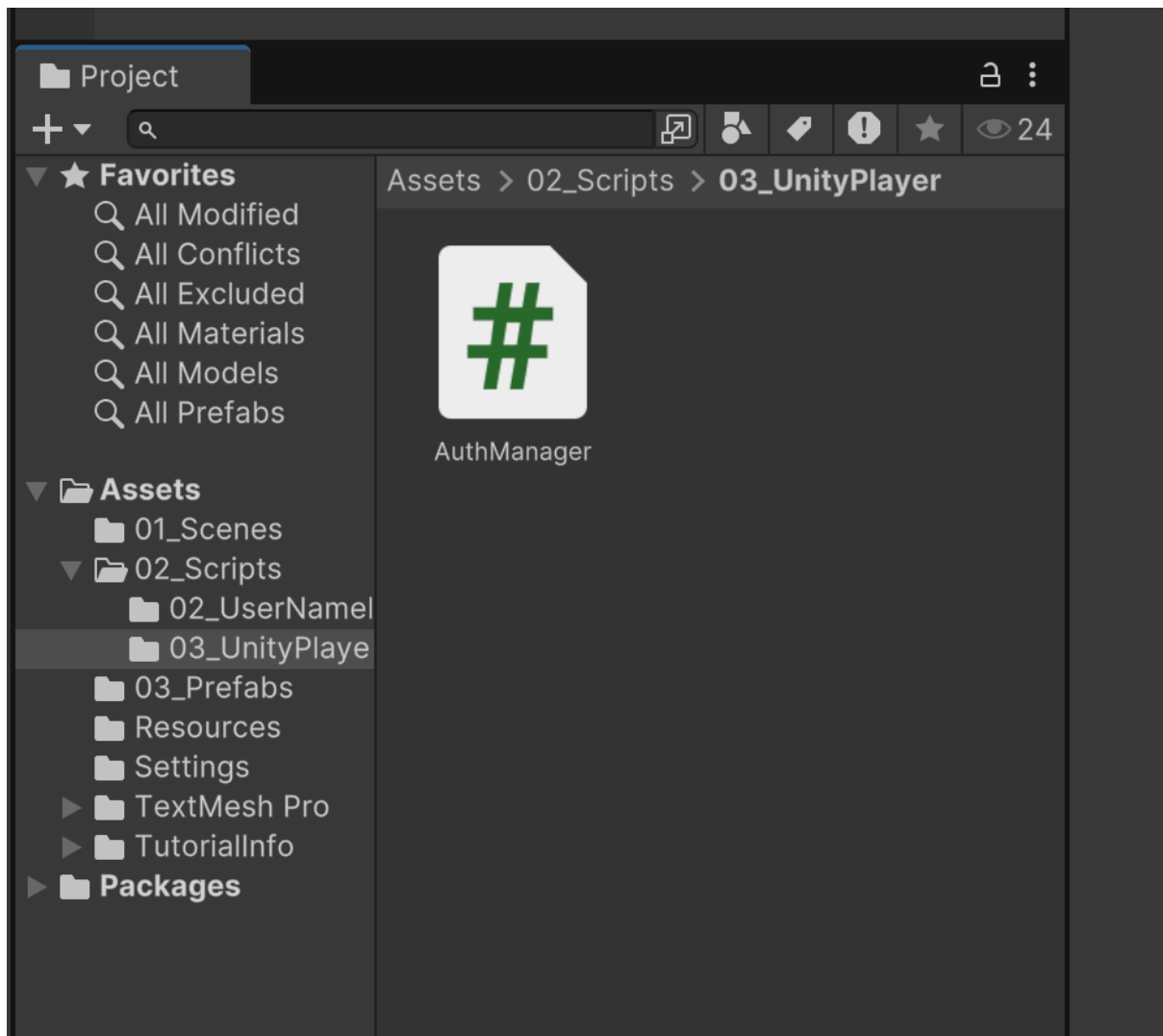
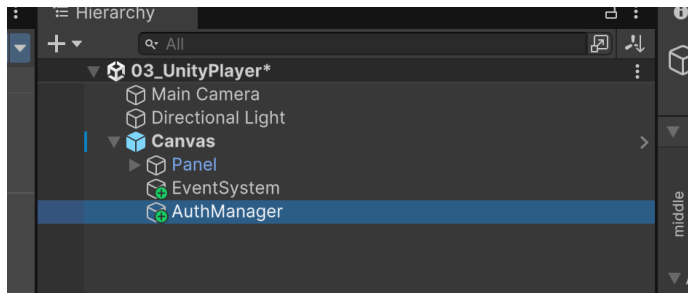
근데 미리 연결 다 되어있네...?

링크 워닝은 무시하기 배포하기 전에 추가하면 되는 것



Canvas 프리팹 들고와서 prefab unpack(프리팹을 끊어주기),

EventSystem 추가 , AuthManager 추가

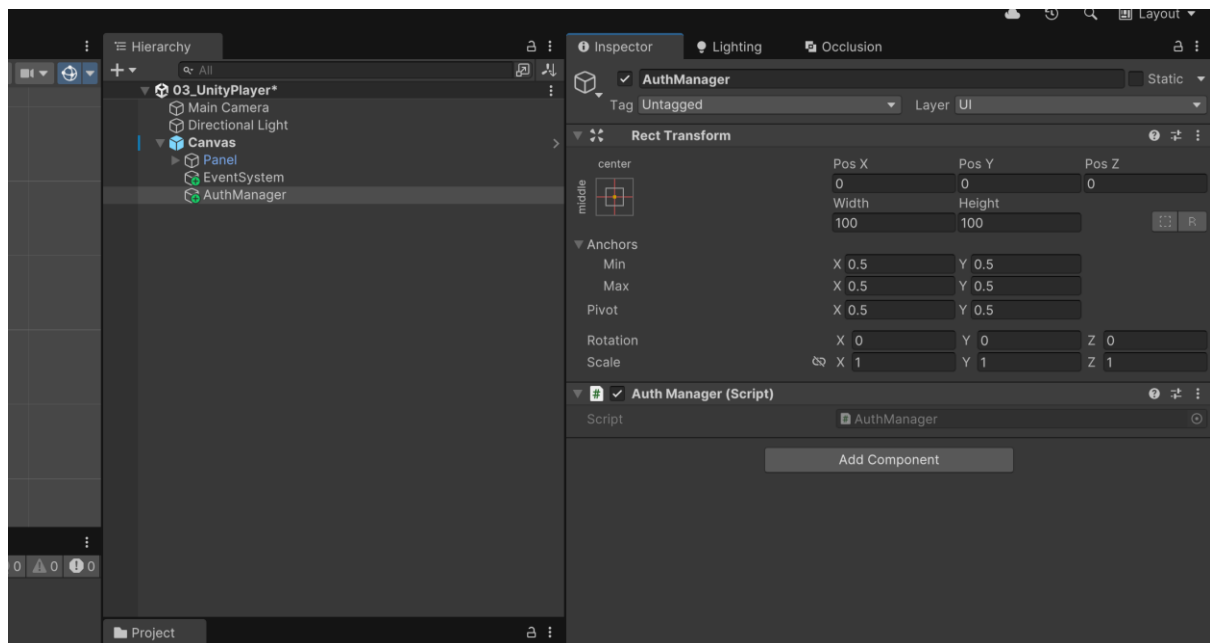


02_UserNamePassword 했던 것 처럼 똑같이 진행

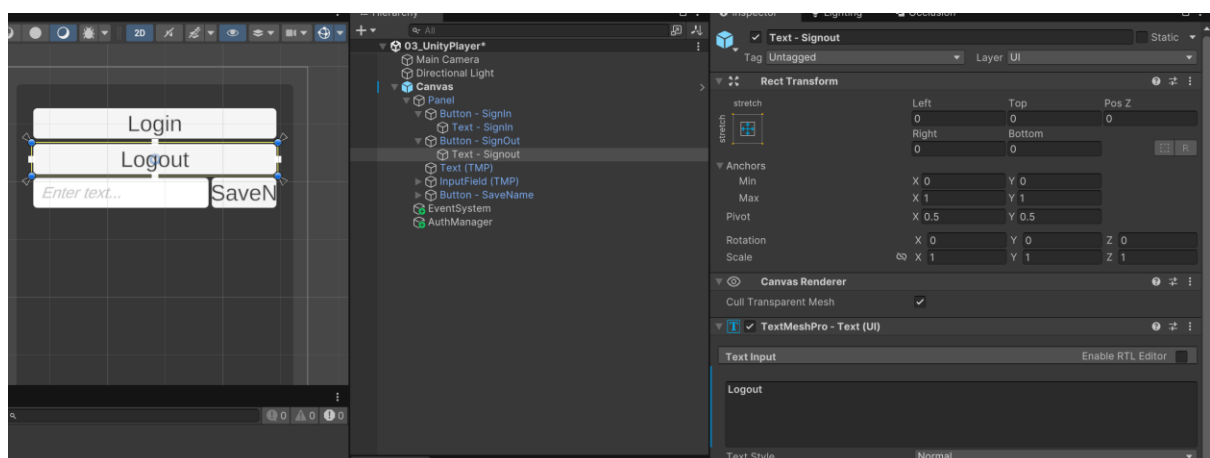
```

1 using UnityEngine;
2
3 namespace AuthUnityPlayer
4 {
5     public class AuthManager : MonoBehaviour
6     {
7         // Start is called once before the first execution of Update after the MonoBehaviour is created
8         void Start()
9         {
10
11         }
12
13         // Update is called once per frame
14         void Update()
15         {
16
17         }
18     }
19 }

```



스크립트 추가

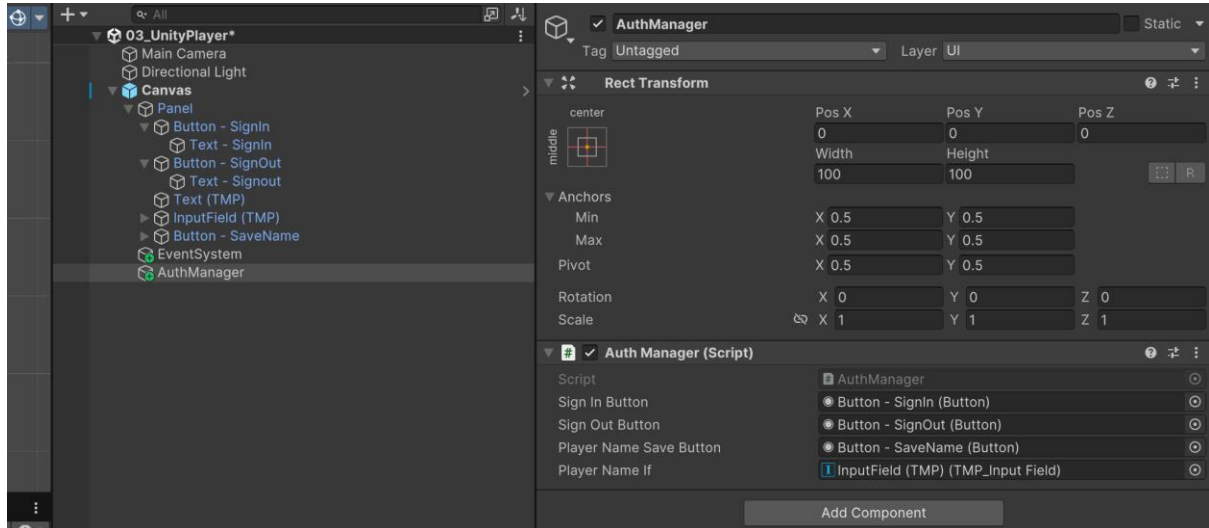


Text Login과 Logout 으로 변경

```

4
5 namespace AuthUnityPlayer
6 {
7     public class AuthManager : MonoBehaviour
8     {
9         [SerializeField] private Button signInButton, signOutButton, playerNameSaveButton;
10        [SerializeField] private TMP_InputField playerNameIf;
11    }
12 }

```



버튼들 연결하기

```

namespace AuthUnityPlayer
{
    public class AuthManager : MonoBehaviour
    {
        [SerializeField] private Button signInButton, signOutButton, playerNameSaveButton;
        [SerializeField] private TMP_InputField playerNameIf;

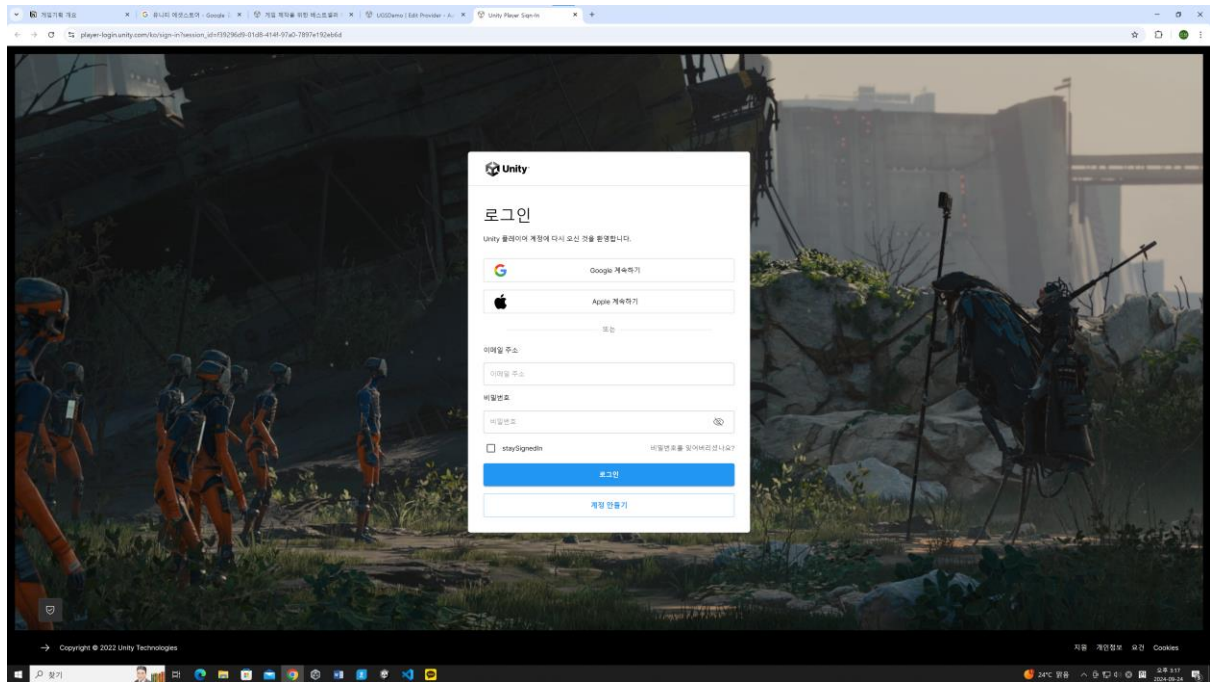
        private async void Awake()
        {
            // 초기화
            await UnityServices.InitializeAsync();

            signInButton.onClick.AddListener(async () =>
            {
                await PlayerAccountService.Instance.StartSignInAsync();
            });
        }
    }
}

```

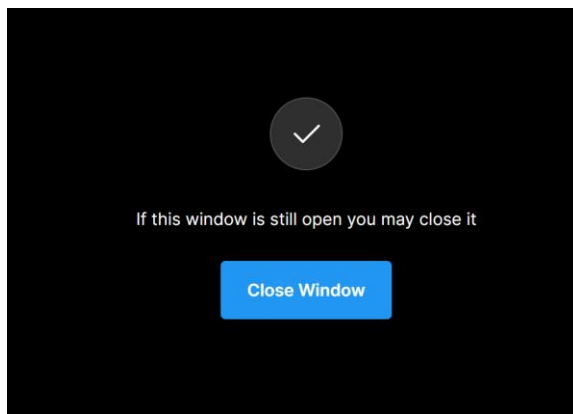
Awake 초기화 및 어카운트 연결

이후 로그인 버튼 눌러보면

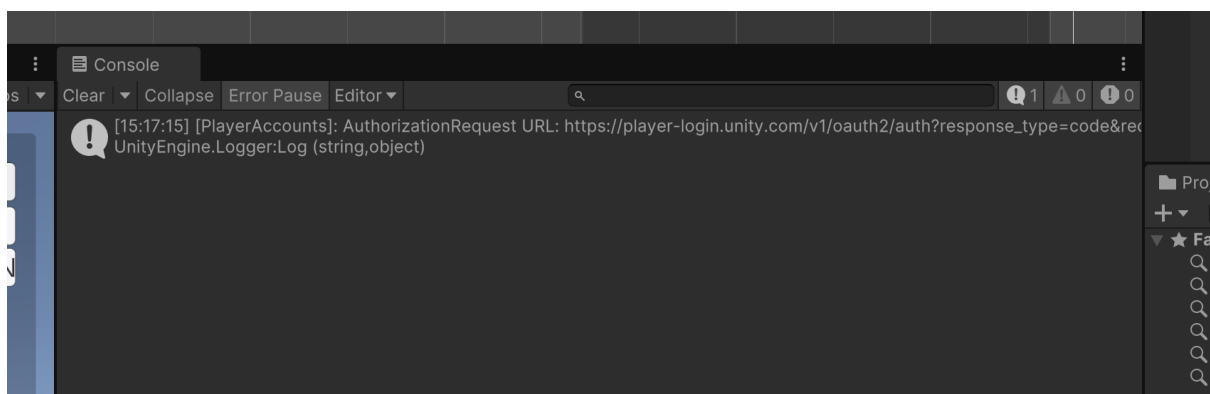


웹브라우저 연결됨

유니티, 구글 등 계정으로 로그인 하거나 계정만들기 하면



로그인 완료



콘솔로그에서 확인 가능

```

namespace AuthUnityPlayer
{
    public class AuthManager : MonoBehaviour
    {
        [SerializeField] private Button signInButton, signOutButton, playerNameSaveButton;
        [SerializeField] private TMP_InputField playerNameIf;

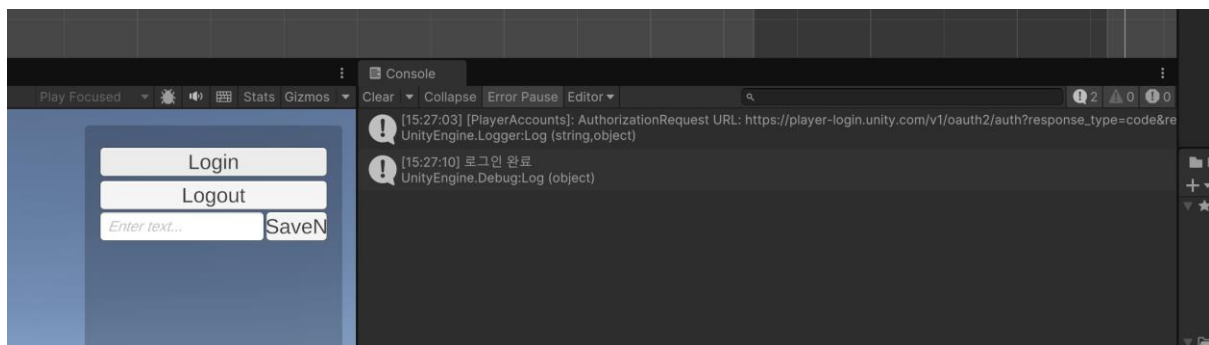
        private async void Awake()
        {
            // 초기화
            await UnityServices.InitializeAsync();
            // 로그인 후 콜백 연결
            PlayerAccountService.Instance.SignedIn += OnSignedIn;

            signInButton.onClick.AddListener(async () =>
            {
                await PlayerAccountService.Instance.StartSignInAsync();
            });

            // 로그인 프로세스
            private async void OnSignedIn()
            {
                // 액세스토큰 접속할 수 있는 토큰을 보내줌. 그걸 받아옴
                string accessToken = PlayerAccountService.Instance.AccessToken;
                // 액세스 토큰을 가져와서 로그인을 해 달라~
                await AuthenticationService.Instance.SignInWithUnityAsync(accessToken);
                Debug.Log("로그인 완료");
            }
        }
    }
}

```

로그인 프로세스 함수 만들어서 콜백 연결



로그인 누르면 아까와같이 웹 떴다가, 로그인 하면 콘솔로그.

```

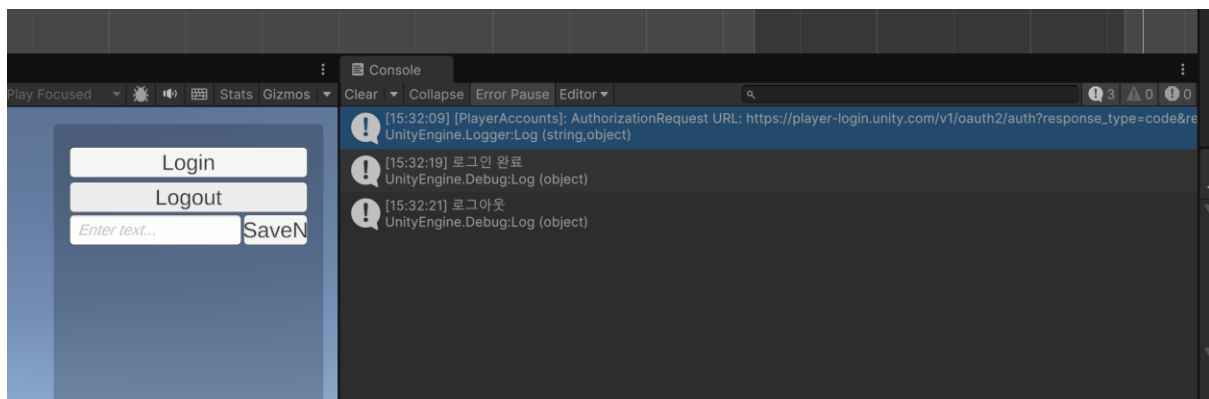
private async void Awake()
{
    // 초기화
    await UnityServices.InitializeAsync();
    // 로그인 후 콜백 연결
    PlayerAccountService.Instance.SignedIn += OnSignedIn;
    PlayerAccountService.Instance.SignedOut += () =>
    {
        Debug.Log("로그아웃");
    };

    signInButton.onClick.AddListener(async () =>
    {
        await PlayerAccountService.Instance.StartSignInAsync();
    });

    signOutButton.onClick.AddListener(() =>
    {
        PlayerAccountService.Instance.SignOut();
    });
}

```

로그아웃 버튼 및 콜백연결



로그아웃 잘 됨.

*****버그 픽스*****

같은 세션에서 재로그인 할 때 이미 로그인된 유저라고 뜨는 버그를 고침

```

37
38 // 로그인 프로세스
39 private async void OnSignedIn()
40 {
41     // 액세스토큰 접속할 수 있는 토큰을 보내줌. 그걸 받아옴
42     string accessToken = PlayerAccountService.Instance.AccessToken;
43
44     if (AuthenticationService.Instance.IsSignedIn) return;
45
46     // 액세스 토큰을 가져와서 로그인을 해 달라~
47     await AuthenticationService.Instance.SignInWithUnityAsync(accessToken);
48     Debug.Log("로그인 완료");
49 }
50
51 }

```

If (~) 추가함