Unity Backend

"게임 유저 관리" 기능을 이용한 로그인, 계정 생성, 이메일 설정, 아이디/비밀번호 찾기, 유저 정보 출력, 닉네임 설정

Created in 2023-01-26

Last Updated 2023-02-01

Unity Version 2022.2.2f1

Index

- 뒤끝 함수 호출 방식과 비동기 호출
- 게임 유저 관리

뒤끝 함수 호출 방식과 비동기 호출

- 뒤끝 함수 호출 방식
- 비동기 호출을 위한 처리

뒤끝 함수 호출 방식과 비동기 호출

뒤끝 함수 호출 방식

- 뒤끝에서 제공하는 함수는 동기, 비동기, SendQueue로 호출 가능

- 동기
  - 요청 시점에서 시간이 얼마나 걸리던지 요청한 자리에서 결과가 주어지는 함수
  - 함수를 호출하면 리턴 값이 반환될 때까지 메인 스레드의 동작이 중지
  - 비동기 함수와 다르게 순차적으로 함수가 호출되어 순서가 섞일 가능성이 없으며, 작업 도중 다른 스레드에 의해 오류가 발생할 경우가 없기 때문에 확실한 결과 제공

```
11          // 동기
12          var bro = Backend.BMember.CustomLogin(ID, PW);
13          // 로그인에 성공했을 때 처리
14          if ( bro.IsSuccess() )
15          {
16          }
17          // 로그인에 실패했을 때 처리
18          else
19          {
20              // statusCode의 값에 따라 실패 원인을 알 수 있다.
21              int statusCode = int.Parse(bro.GetStatusCode());
22          }
```

뒤끝 함수 호출 방식과 비동기 호출

- 비동기
  - 동기 함수와 다르게 호출 시점에서 실행 결과를 기다리지 않는 함수
  - 비동기 함수의 경우 비동기 함수를 호출한 쓰레드에서 해당 함수 호출에 대한 결과를 기다리지 않고 바로 다음 작업을 수행할 수 있다.

  - 비동기 함수의 콜백 함수
    - **비동기 IO 스레드 내에서 실행**하는 방법
      - 유니티의 정책에 따라 별도의 스레드에서 **유니티 MonoBehaviour 객체에 접근 불가능**
      - 콜백 함수 내에서 유니티 객체, UI 객체 등에 접근할 수 없기 때문에 비동기 함수의 요청 결과에 따른 처리를 위해 별도의 **Dispatcher 사용 필요**

    - 콜백 함수 풀링을 사용해 **메인 스레드에서 실행**하는 방법
      - 콜백 함수 풀링 기능을 사용하면 콜백 함수 내에서도 **유니티 MonoBehaviour 객체에 접근 가능**
      - 콜백 함수 내에서 유니티 객체, UI 객체 등에 접근 가능
      - **뒤끝 초기화 시 useAsnynPoll=true 설정과 Update() 메소드에서 AsyncPoll() 메소드 호출 필요**

뒤끝 함수 호출 방식과 비동기 호출

- 비동기 - 콜백 풀링 함수

```
24      // 비동기 - 콜백 풀링 함수
25      // 별도의 큐에 저장된 콜백 함수를 실행하려면 AsnycPoll() 메소드 호출이 필요하다.
26      // https://developer.thebackend.io/unity3d/guide/Async/AsyncFuncPoll/
27      Backend.BMember.CustomLogin(ID, PW, callback =>
28      {
29          // 로그인에 성공했을 때
30          if ( callback.IsSuccess() )
31          {
32          }
33          // 로그인에 실패했을 때
34          else
35          {
36              // statusCode의 값에 따라 실패 원인을 알 수 있다.
37              int statusCode = int.Parse(callback.GetStatusCode());
38          }
39      });
```

뒤끝 함수 호출 방식과 비동기 호출

- SendQueue
  - 비동기 함수 호출 시 바로 호출하지 않고 큐에 적재한 후 순차적으로 함수를 호출
  - 동기의 장점인 순차적으로 호출되는 점과 비동기의 장점인
    함수 호출의 결과를 기다리지 않고 바로 다음 작업이 수행되며,
    프로그램이 중지되지 않는 것처럼 보인다는 점이 합쳐져 있다.

```
41      // SendQueue
42      // SendQueue의 Enqueue()에 호출할 메소드를 등록하고 별도의 시작 처리가 필요하다.
43      // https://developer.thebackend.io/unity3d/guide/Async/SendQueueDetail/
44      SendQueue.Enqueue(Backend.BMember.CustomLogin, ID, PW, callback =>
45      {
46          // 로그인에 성공했을 때
47          if ( callback.IsSuccess() )
48          {
49          }
50          // 로그인에 실패했을 때
51          else
52          {
53              // statusCode의 값에 따라 실패 원인을 알 수 있다.
54              int statusCode = int.Parse(callback.GetStatusCode());
55          }
56      });
```

SendQueue는 메인 스레드가 아닌
별도의 쓰레드에서 동작

뒤끝 함수 호출 방식과 비동기 호출

- 동기, 비동기 호출 예시

동기

비동기 - 콜백 풀링 함수

뒤끝 함수 호출 방식과 비동기 호출

비동기 호출을 위한 처리

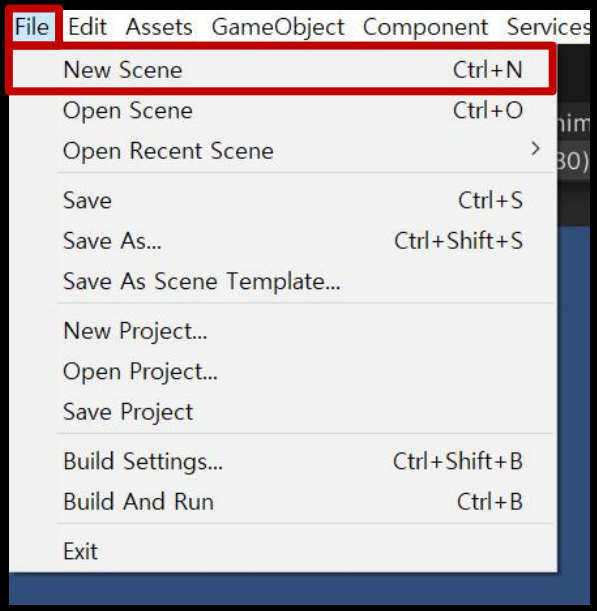
- 비동기 메소드 호출을 위해 AsyncPoll() 호출
  - BackendManager Script 수정

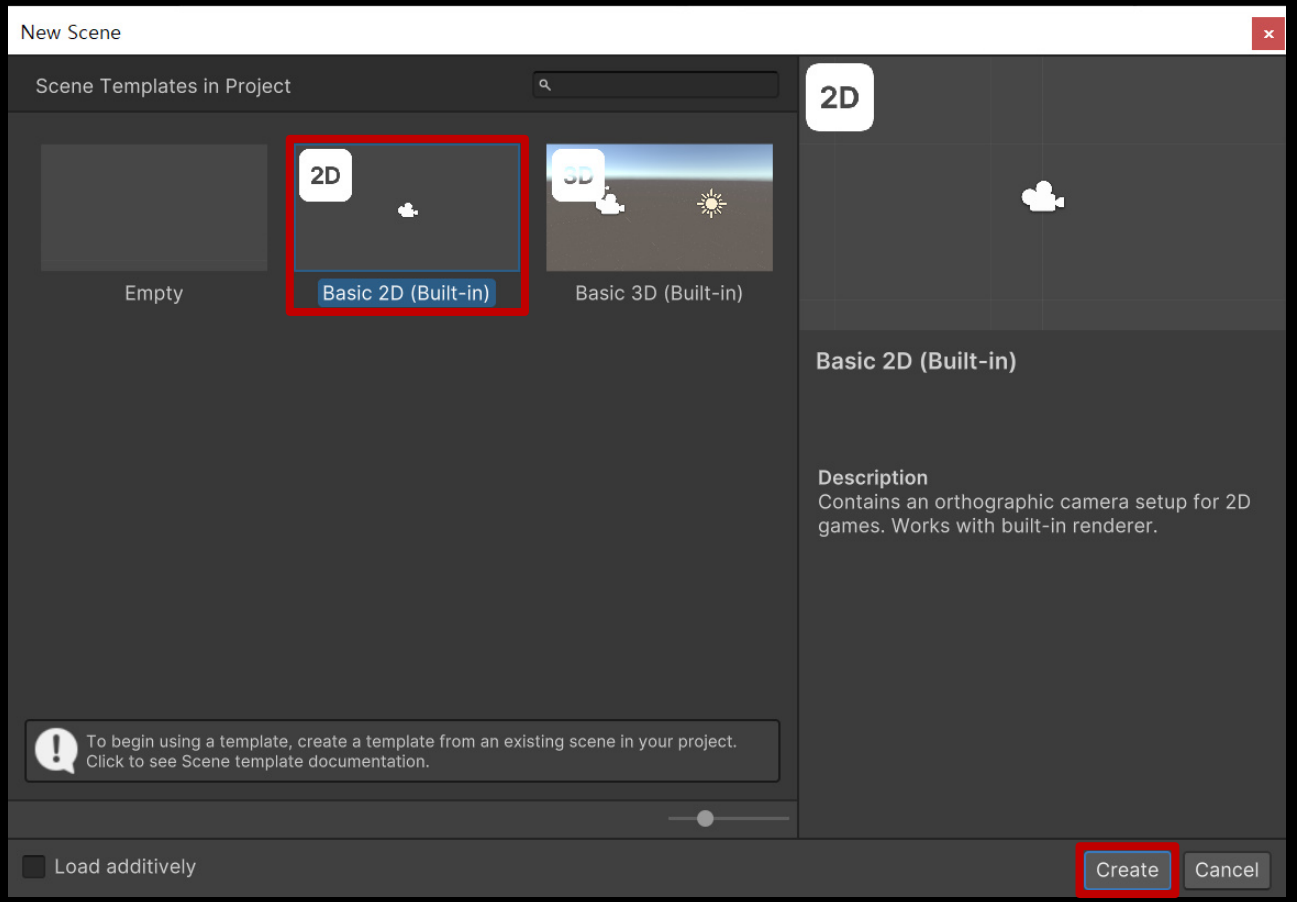
```
1   using UnityEngine;
2   using BackEnd;      // 뒤끝 SDK
3
4   public class BackendManager : MonoBehaviour
5   {
6       private void Awake()
7       {
8           // Update() 메소드의 Backend.AsyncPoll(); 호출을 위해 오브젝트를 파괴하지 않는다
9           DontDestroyOnLoad(gameObject);
10
11          // 뒤끝 서버 초기화
12          BackendSetup();
13      }
14
15      private void Update()
16      {
17          // 서버의 비동기 메소드 호출(콜백 함수 풀링)을 위해 작성
18          // 참고 : https://developer.thebackend.io/unity3d/guide/Async/AsyncFuncPoll/
19          if ( Backend.IsInitialized )
20          {
21              Backend.AsyncPoll();
22          }
23      }
24
25      private void BackendSetup()...
42  }
```

AsyncPoll() 메소드를 호출해야 별도의 큐에 저장된 콜백 함수를 호출할 수 있기 때문에 모든 씬에서 삭제되지 않고 계속 남아있도록 DontDestroyOnLoad()를 호출

뒤끝 함수 호출 방식과 비동기 호출

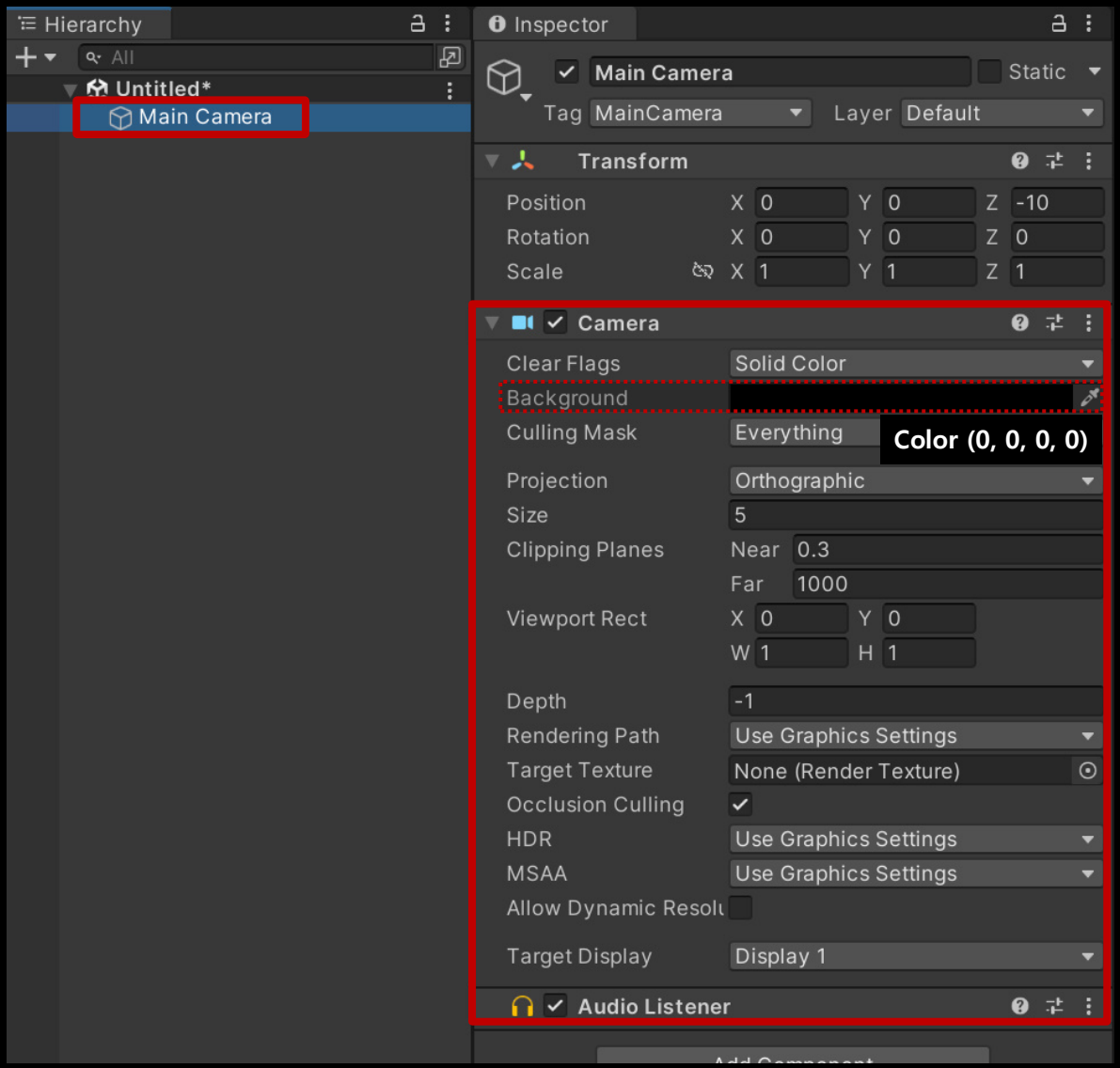
- Login 씬 생성
  - File - New Scene





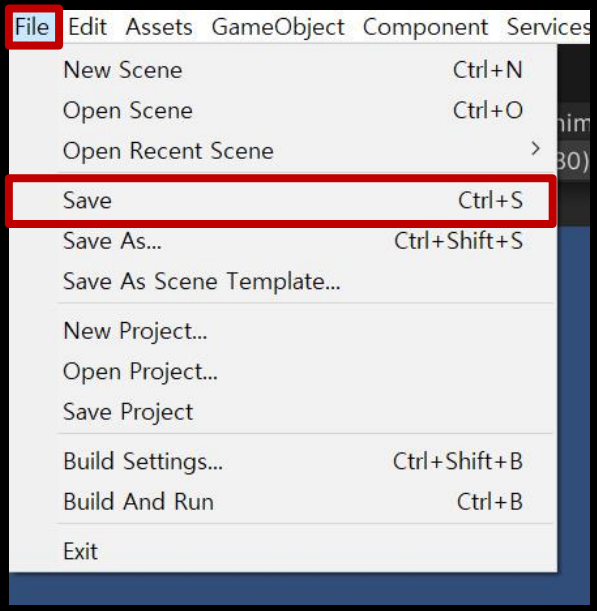
뒤끝 함수 호출 방식과 비동기 호출

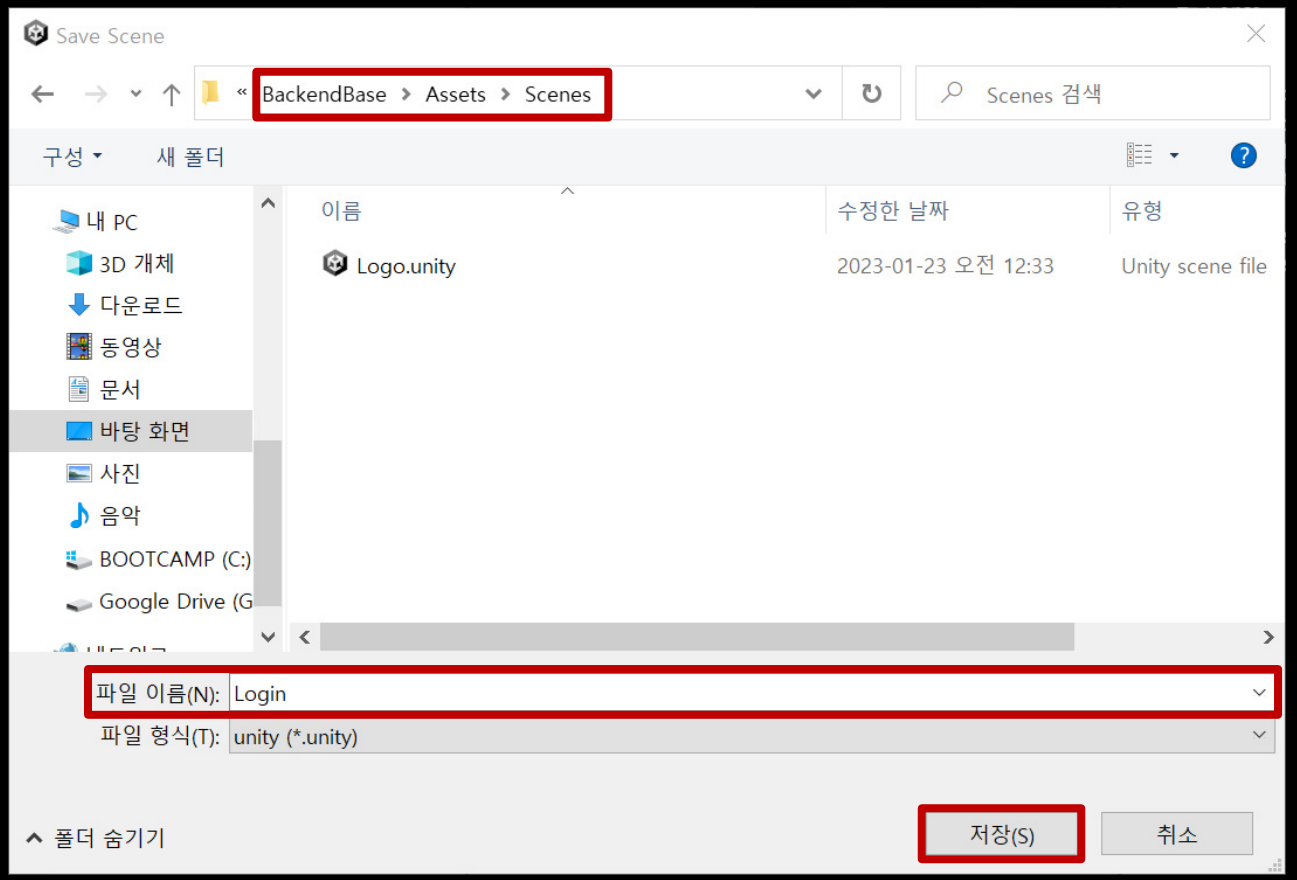
- 카메라 설정



뒤끝 함수 호출 방식과 비동기 호출

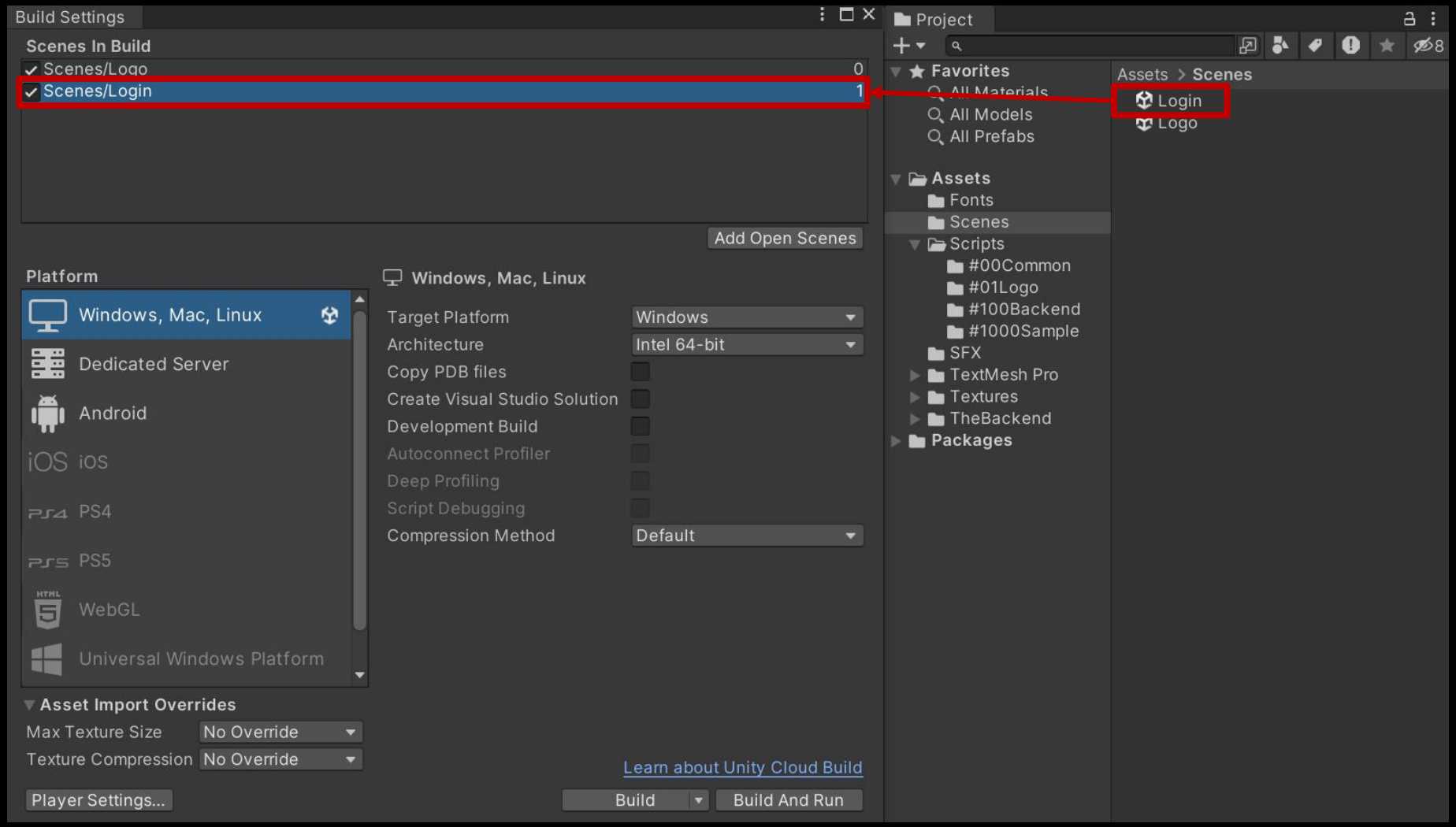
- Login 씬 저장
  - File - Save





뒤끝 함수 호출 방식과 비동기 호출

- Scenes In Build에 씬 등록
  - File - Build Settings



뒤끝 함수 호출 방식과 비동기 호출

- 씬 전환과 같은 유틸 메소드를 정의하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "Utils"로 변경

```
1  using UnityEngine.SceneManagement;
2
3  public enum SceneNames { Logo=0, Login, }
4
5  public static class Utils
6  {
7      public static string GetActiveScene()
8      {
9          return SceneManager.GetActiveScene().name;
10     }
11
12     public static void LoadScene(string sceneName="")
13     {
14         if ( sceneName == "" )
15         {
16             SceneManager.LoadScene(GetActiveScene());
17         }
18         else
19         {
20             SceneManager.LoadScene(sceneName);
21         }
22     }
23
24     public static void LoadScene(SceneNames sceneName)
25     {
26         // SceneNames 열거형으로 매개변수를 받아온 경우 ToString() 처리
27         SceneManager.LoadScene(sceneName.ToString());
28     }
29 }
```

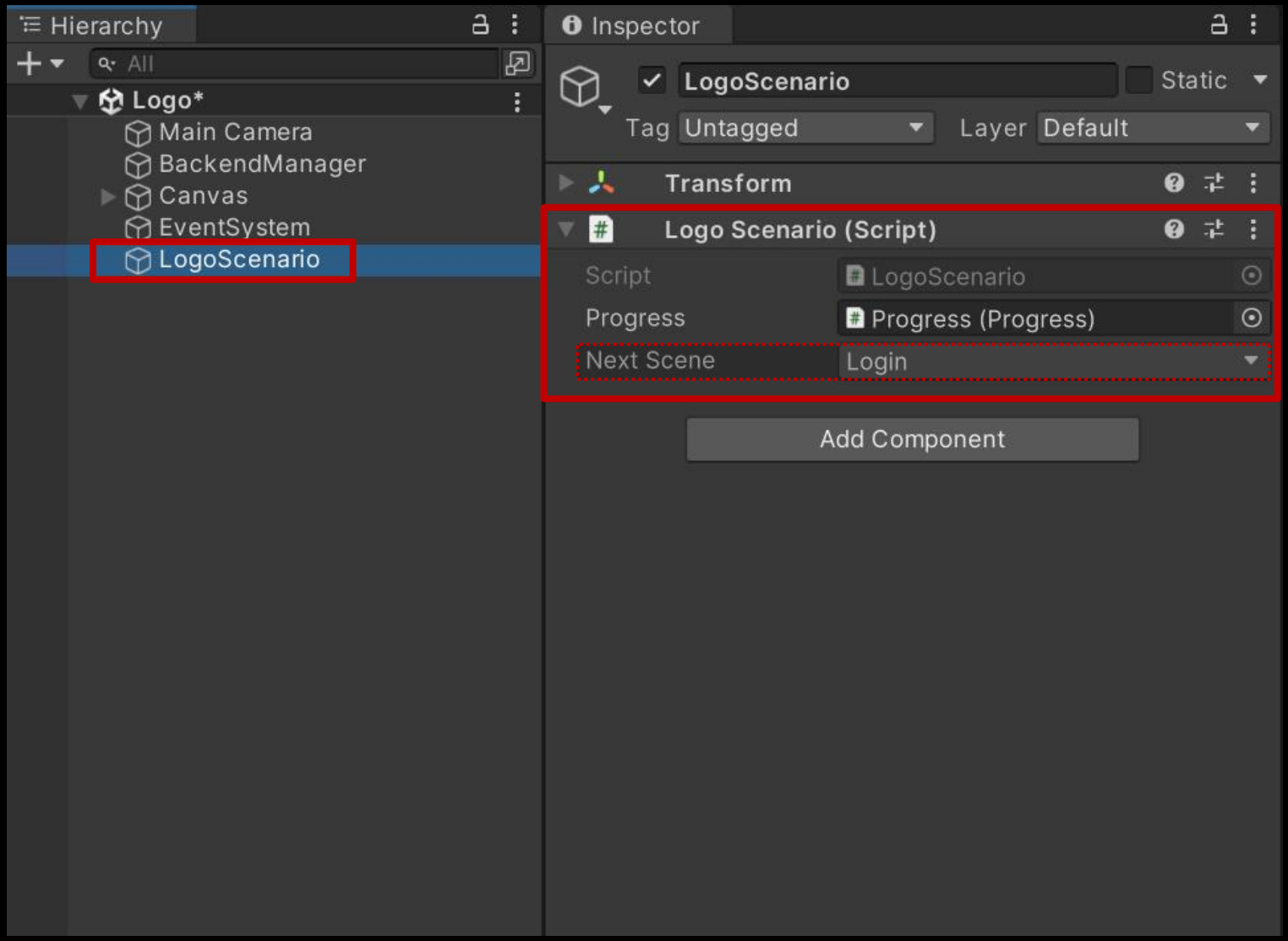
뒤끝 함수 호출 방식과 비동기 호출

- 로딩이 완료되면 "Login" 씬으로 이동
  - LogoScenario Script 수정

```
1   using UnityEngine;
2
3   public class LogoScenario : MonoBehaviour
4   {
5       [SerializeField]
6       private Progress    progress;
7       [SerializeField]
8       private SceneNames  nextScene;
9
10      private void Awake()...
14
15      private void SystemSetup()...
31
32      private void OnAfterProgress()
33      {
34          Utils.LoadScene(nextScene);
35      }
36  }
```

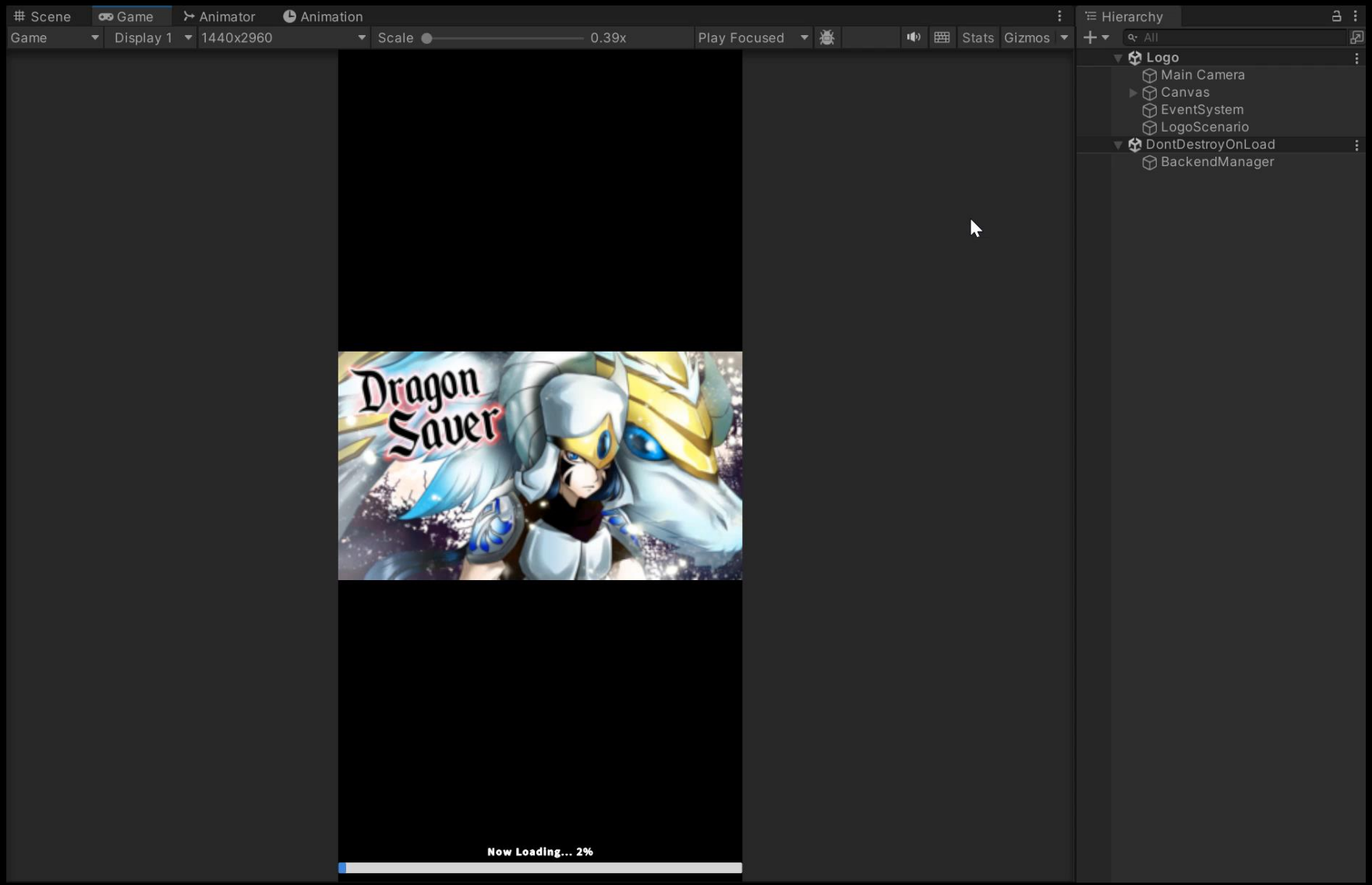
뒤끝 함수 호출 방식과 비동기 호출

- LogoScenario 오브젝트의 "LogoScenario" 컴포넌트 변수 설정



뒤끝 함수 호출 방식과 비동기 호출

- 결과 화면



게임 유저 관리

- 개요
- 로그인
- 계정 생성, 이메일 설정
- 아이디 찾기
- 비밀번호 찾기
- 유저 정보 출력
- 닉네임 설정

게임 유저 관리

개요

- 회원가입, 이메일 설정, 로그인, 아이디/비밀번호 찾기, 닉네임 설정 메소드

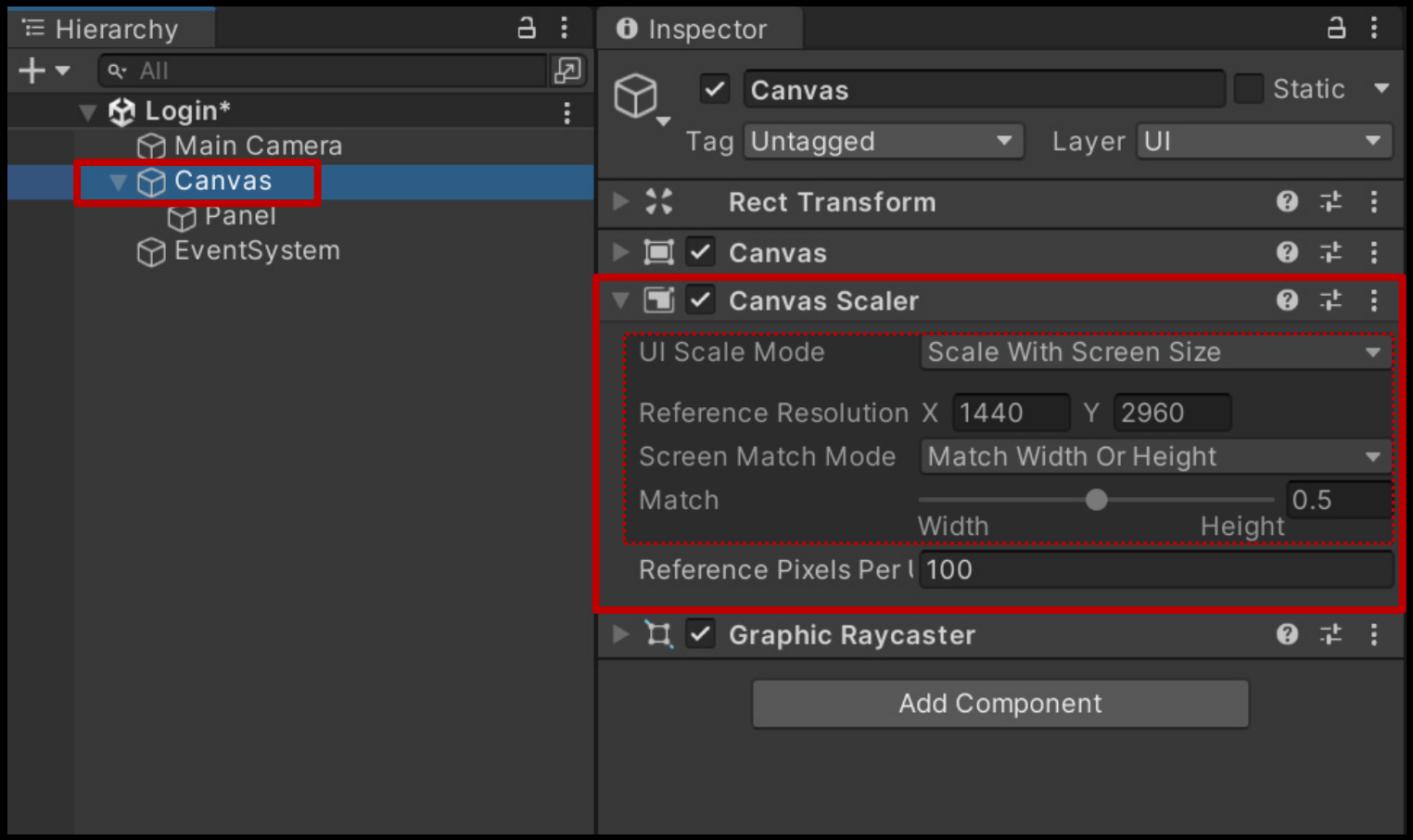
```
using UnityEngine;
using BackEnd;

public class LoginSample : MonoBehaviour
{
    private void Awake()
    {
        string ID       = "user01";
        string PW       = "1234";
        string email    = "user01@gmail.com";
        string nickname = "첫번째유저";

        /// 회원가입
        Backend.BMember.CustomSignUp(ID, PW);

        /// 이메일 설정
        Backend.BMember.UpdateCustomEmail(email);

        /// 로그인
        Backend.BMember.CustomLogin(ID, PW);

        /// 아이디 찾기
        Backend.BMember.FindCustomID(email);

        /// 비밀번호 찾기
        Backend.BMember.ResetPassword(ID, email);

        /// 닉네임 설정
        // 닉네임이 없을 때 최초 닉네임 설정
        Backend.BMember.CreateNickname(nickname);
        // 이미 있는 닉네임을 수정 (만약 닉네임이 없으면 CreateNickname()이 호출된다.
        Backend.BMember.UpdateNickname(nickname);
    }
}
```

회원가입, 로그인 등과 같이 뒤끝 서버의 게임 유저를 관리하기 위해서는 Backend 클래스에 정의되어 있는 BackendMember 타입의 BMember 정적 변수에 접근해 설정할 수 있습니다.

게임 유저 관리

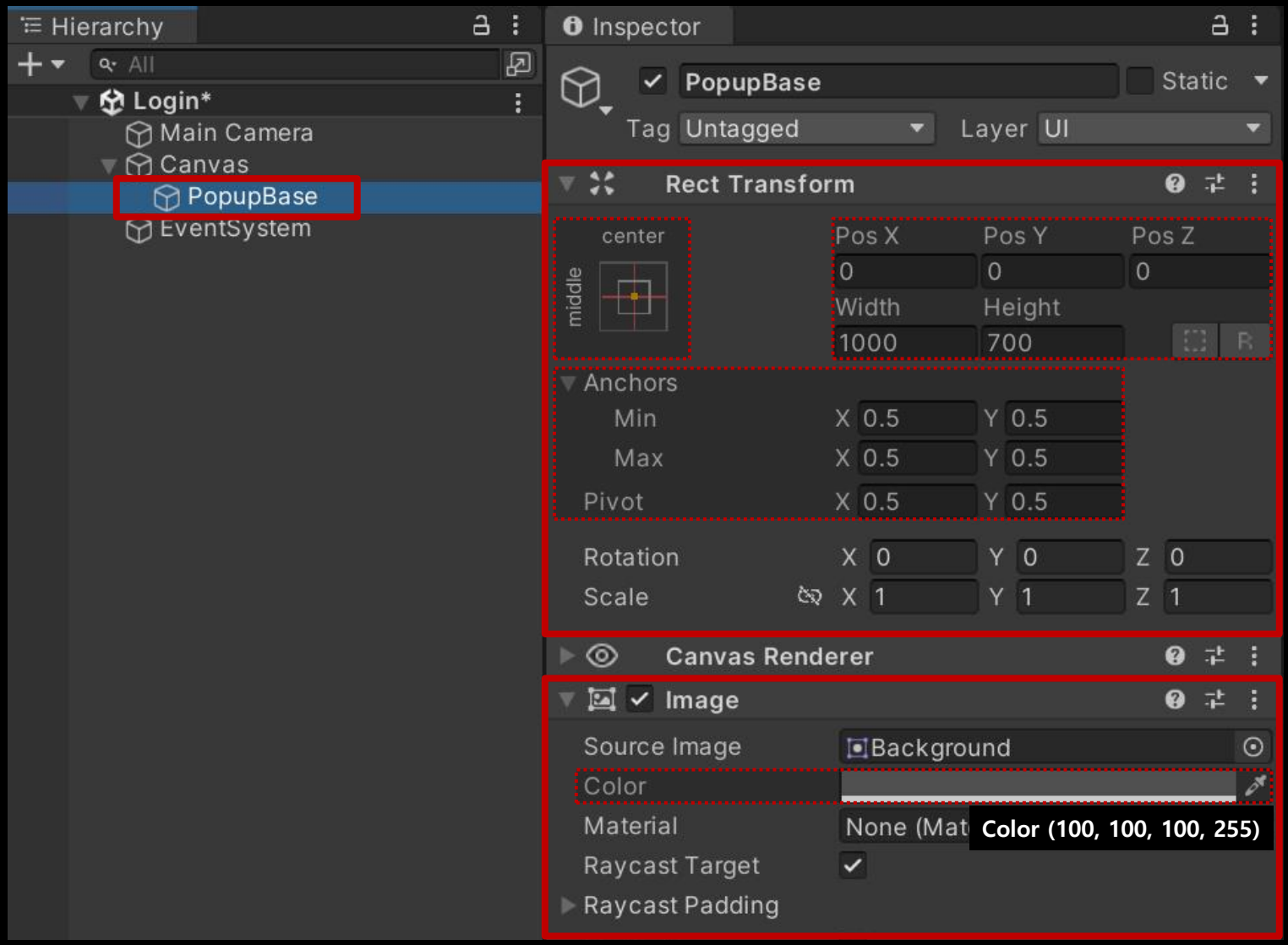
로그인

- 팝업 윈도우에 출력하는 UI들을 관리하는 Panel UI 생성 및 설정
  - GameObject - UI - Panel



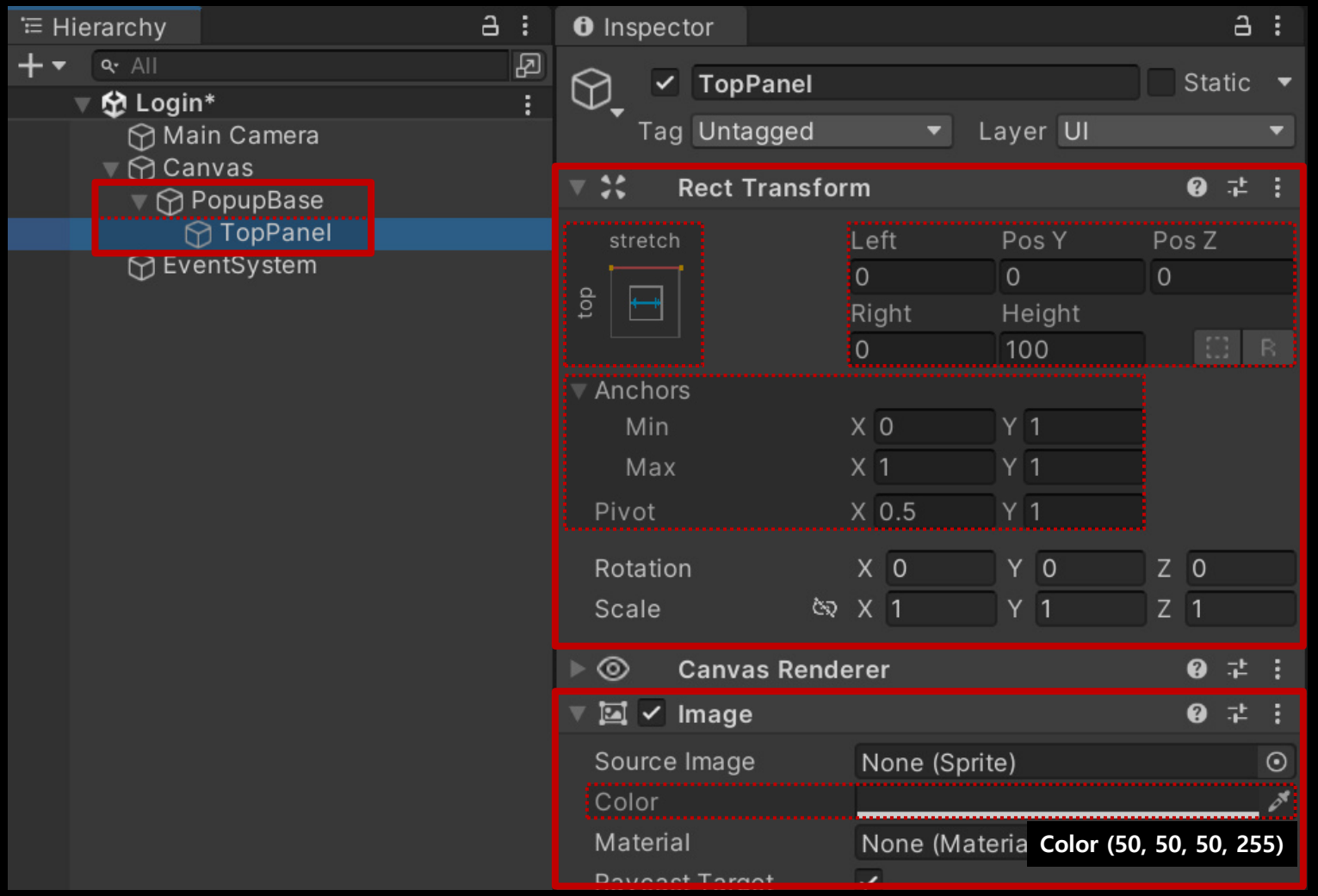
게임 유저 관리

- 팝업 윈도우에 출력하는 UI들을 관리하는 Panel UI 생성 및 설정 (계속)



게임 유저 관리

- 윈도우 상단 배경을 출력하는 Image UI 생성 및 설정
  - GameObject - UI - Image



게임 유저 관리

- 윈도우 상단에 제목을 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"

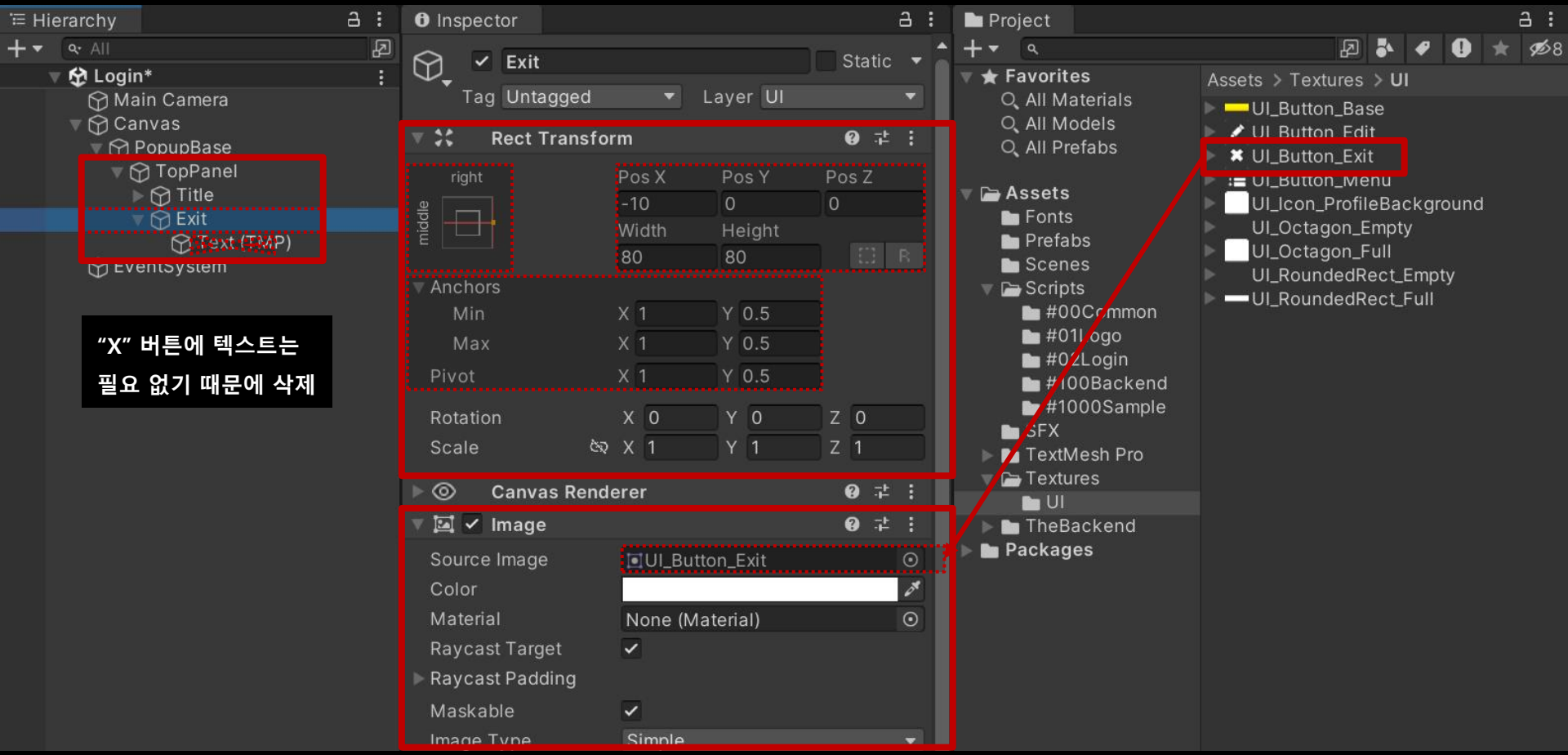
Hierarchy
All
Login*
Main Camera
Canvas
PopupBase
TopPanel
Title
EventSystem

Inspector
Title
Static
Tag Untagged
Layer UI
Rect Transform
stretch
middle
Left 100
Pos Y 0
Pos Z 0
Right 100
Height 100
Anchors
Min X 0 Y 0.5
Max X 1 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
TextMeshPro - Text (UI)
NotoSansKR-Bold SDF Material (Material)
Shader TextMeshPro/Distance Fi
Edit...
Add Component

TextMeshPro - Text (UI)
Text Input
Enable RTL Editor
New Text
Text Style Normal
Main Settings
Font Asset NotoSansKR-Bold SDF (TMP_Fc
Material Preset NotoSansKR-Bold SDF Material
Font Style B I U S ab AB SC
Font Size 40
Auto Size
Vertex Color
Color Gradient
Override Tags
Spacing Options (em) Character 0 Word 0
Line 0 Paragraph 0
Alignment
Wrapping Enabled
Overflow Overflow
Horizontal Mapping Character
Vertical Mapping Character

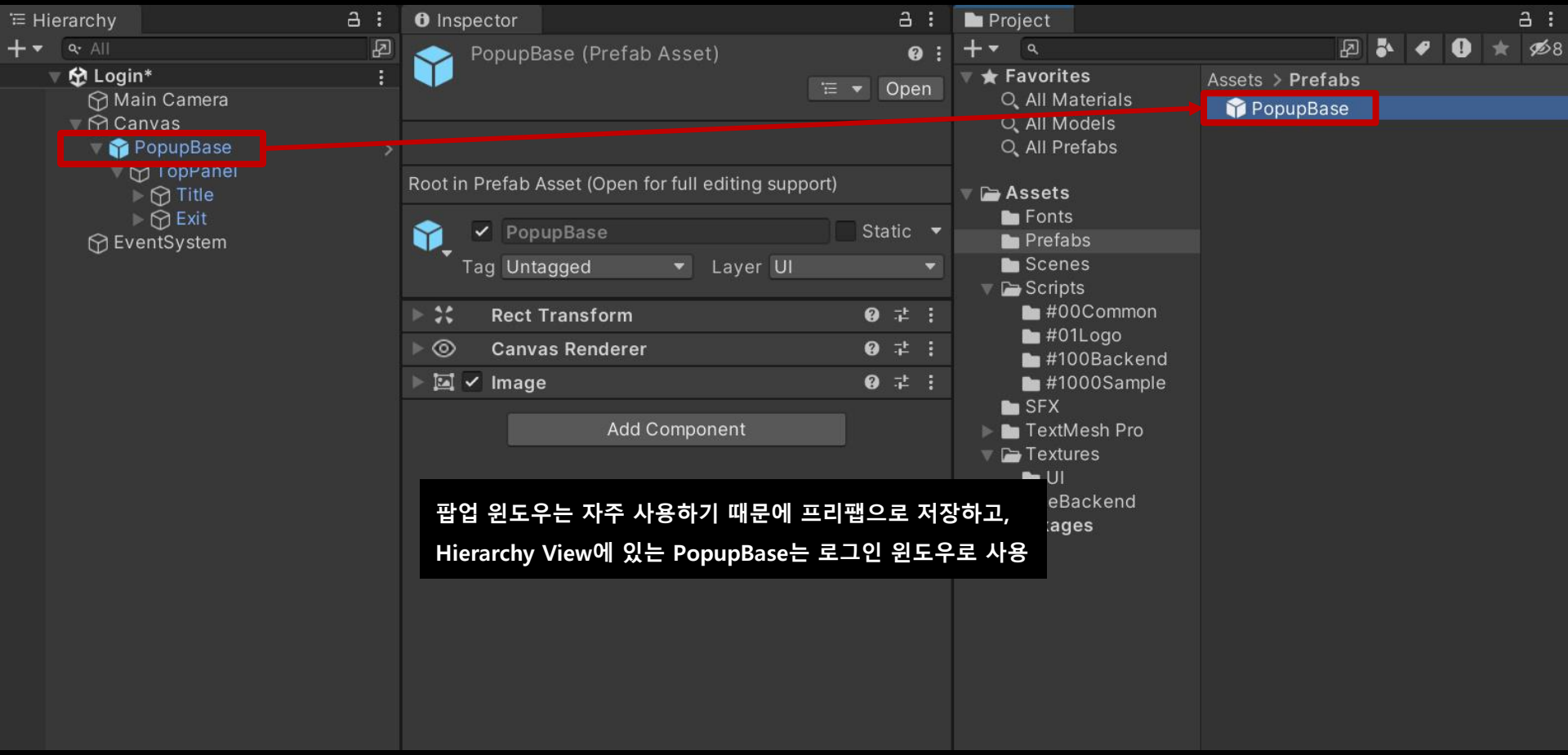
게임 유저 관리

- 윈도우를 종료하는 "Button - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Button - TextMeshPro"



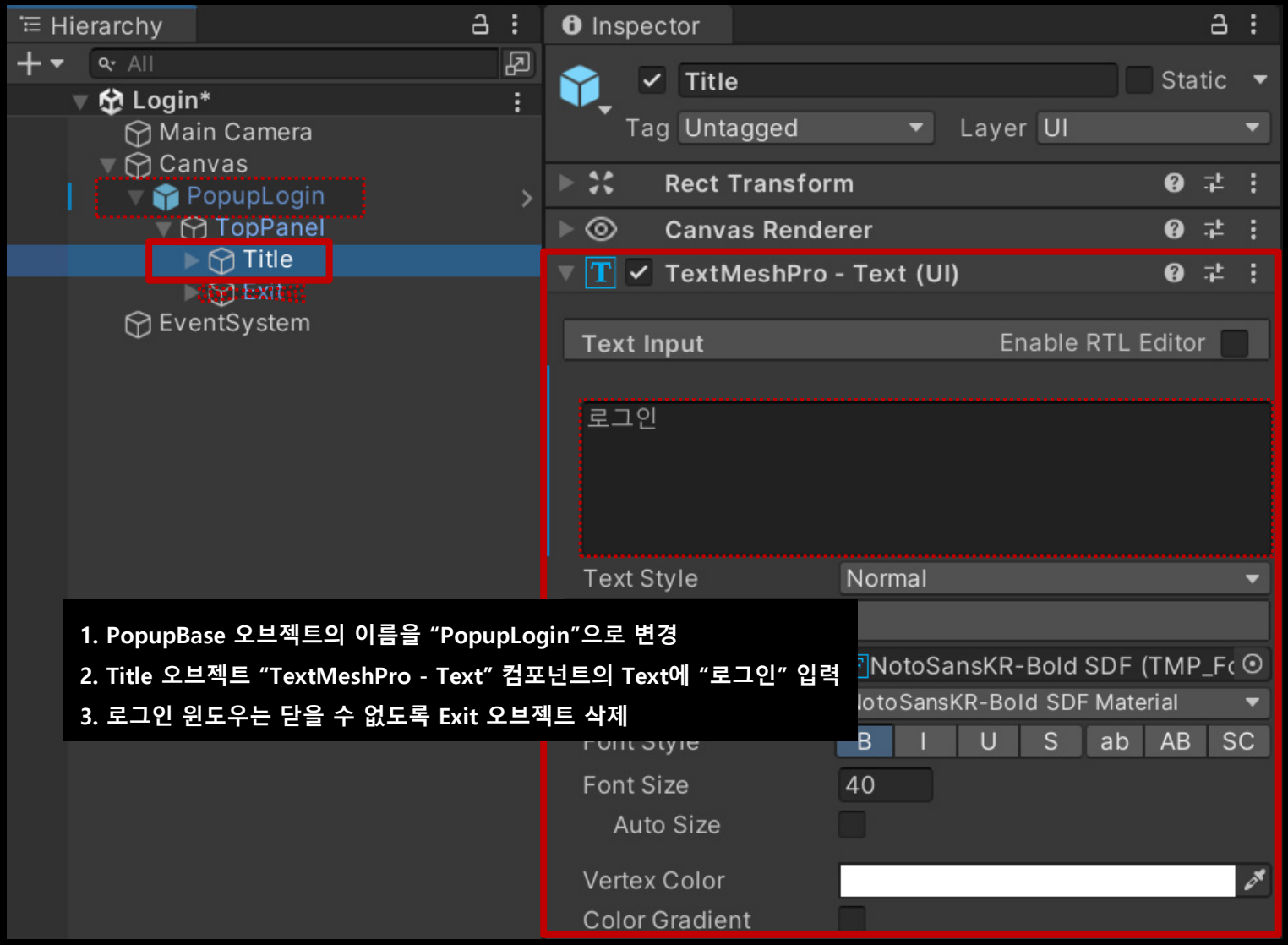
게임 유저 관리

- 팝업 윈도우 오브젝트 Prefab 생성
  - Hierarchy View의 "PopupBase" 오브젝트를 Project View로 Drag & Drop



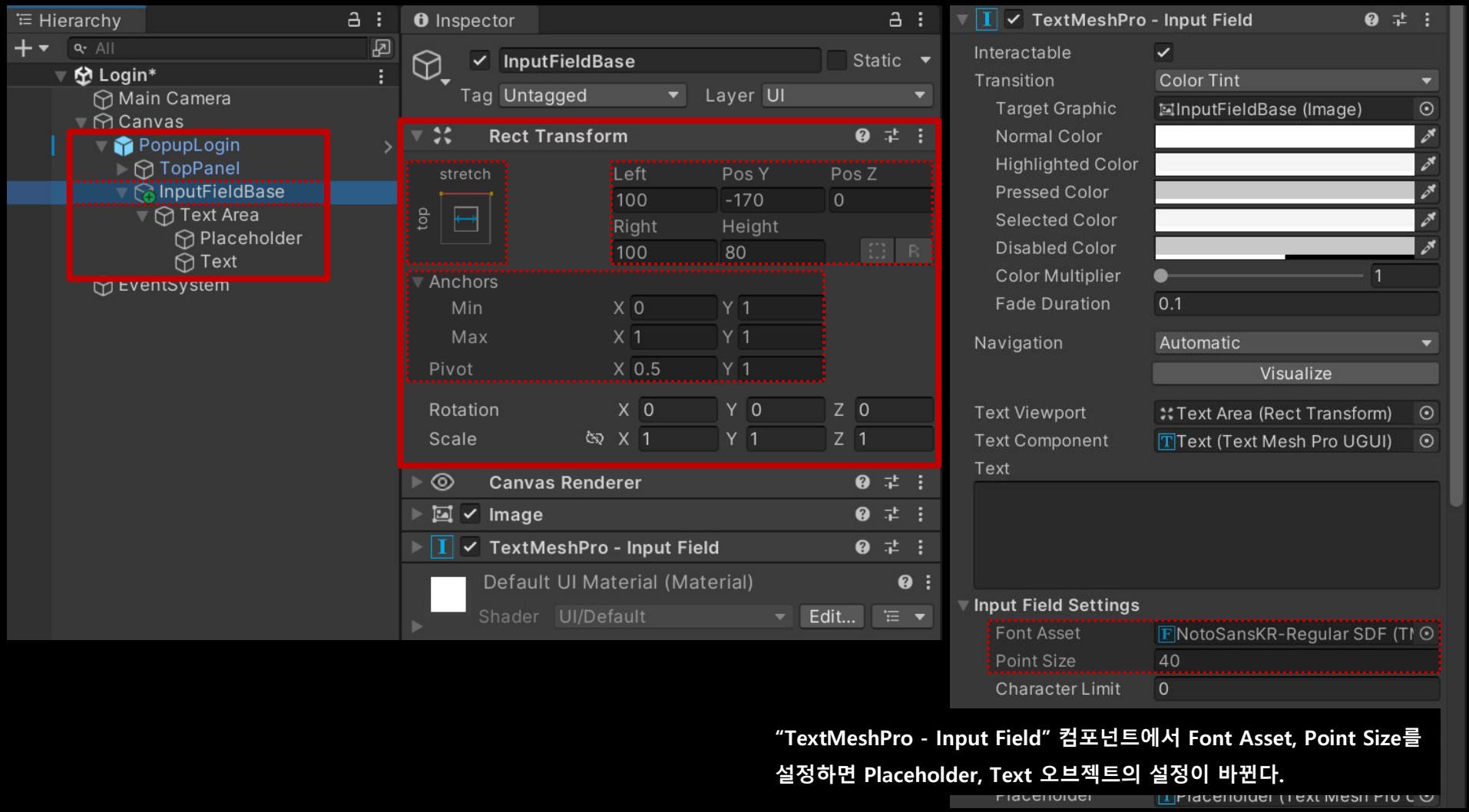
게임 유저 관리

- 로그인 팝업 제작을 위한 오브젝트 설정



게임 유저 관리

- 텍스트 입력을 위한 “Input Field - TextMeshPro” UI 생성 및 설정
  - GameObject - UI - “Input Field - TextMeshPro”



“TextMeshPro - Input Field” 컴포넌트에서 Font Asset, Point Size를 설정하면 Placeholder, Text 오브젝트의 설정이 바뀐다.

게임 유저 관리

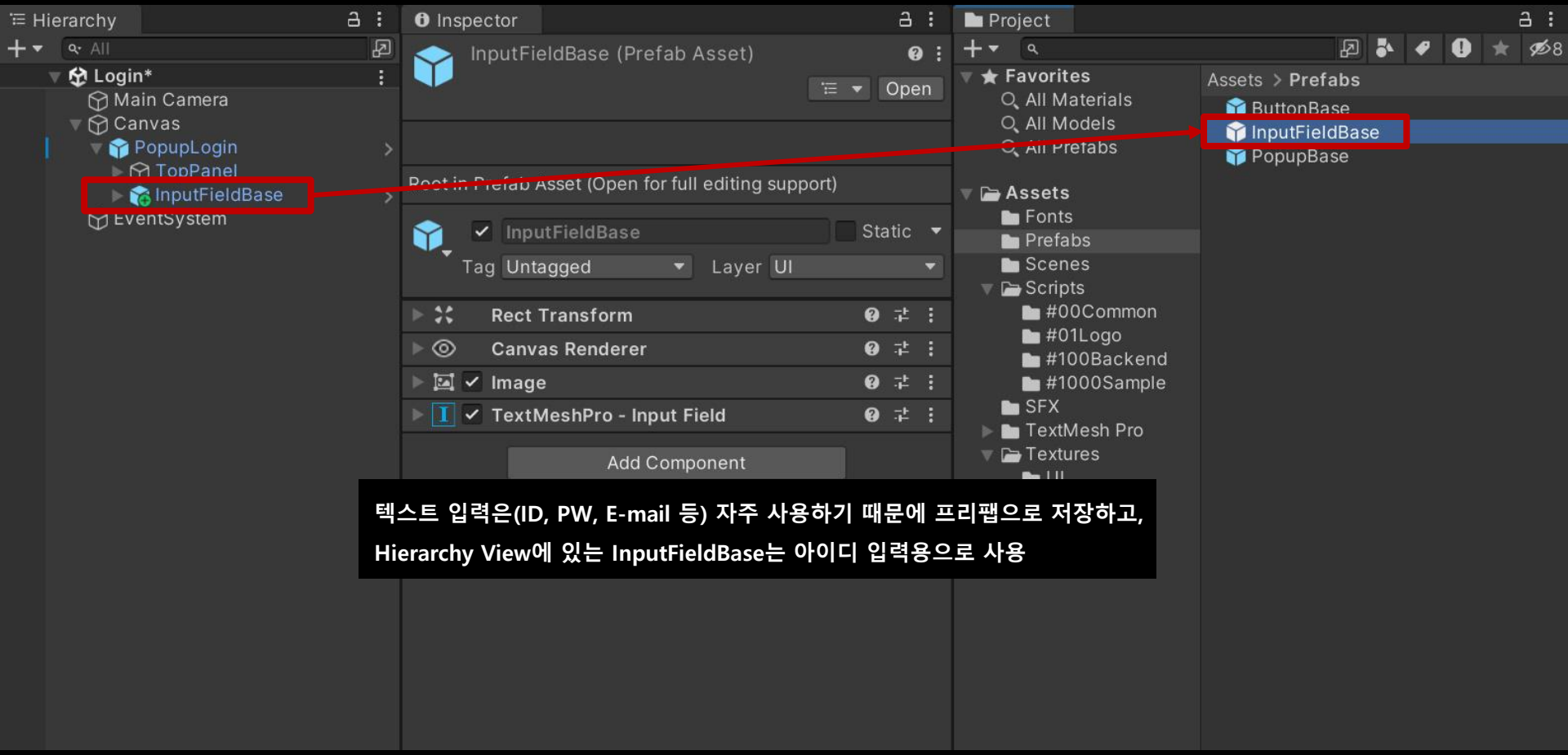
- 텍스트 입력을 위한 “Input Field - TextMeshPro” UI 생성 및 설정 (계속)

Hierarchy
All
Login*
Main Camera
Canvas
PopupLogin
TopPanel
InputFieldBase
Text Area
Placeholder
Text
EventSystem

Inspector
Placeholder
Static
Tag Untagged
Layer UI
Rect Transform
Canvas Renderer
TextMeshPro - Text (UI)
Text Input
Enable RTL Editor
Enter text...
Text Style
Normal
Main Settings
Font Asset
NotoSansKR-Regular SDF (TMF
Material Preset
NotoSansKR-Regular SDF Material
Font Style
B I U S ab AB SC
Font Size
40
Auto Size
Vertex Color
Color Gradient
Color (50, 50, 50, 255)

게임 유저 관리

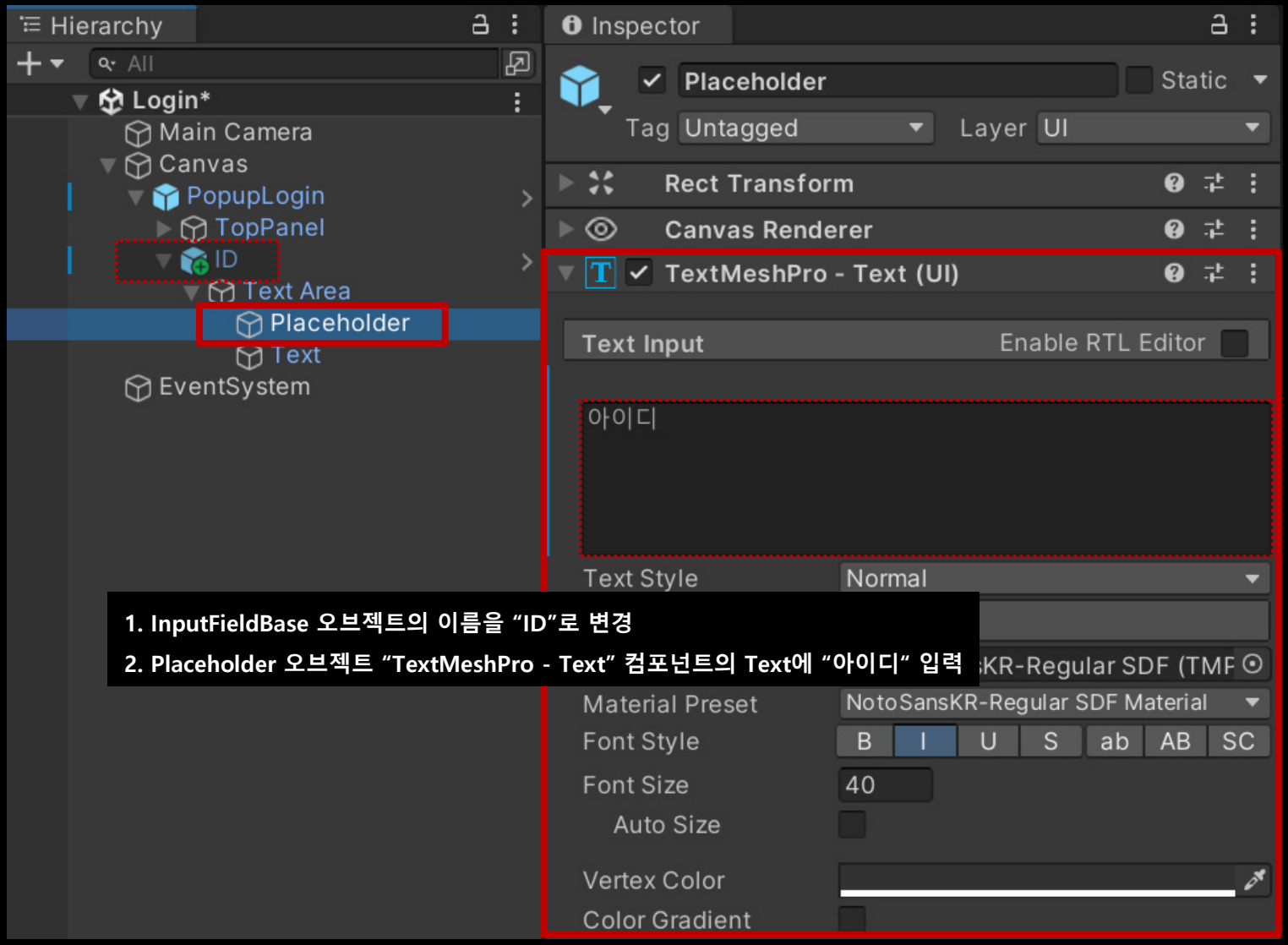
- 텍스트 입력 오브젝트 Prefab 생성
  - Hierarchy View의 "InputFieldBase" 오브젝트를 Project View로 Drag & Drop



텍스트 입력은(ID, PW, E-mail 등) 자주 사용하기 때문에 프리팹으로 저장하고,
Hierarchy View에 있는 InputFieldBase는 아이디 입력용으로 사용

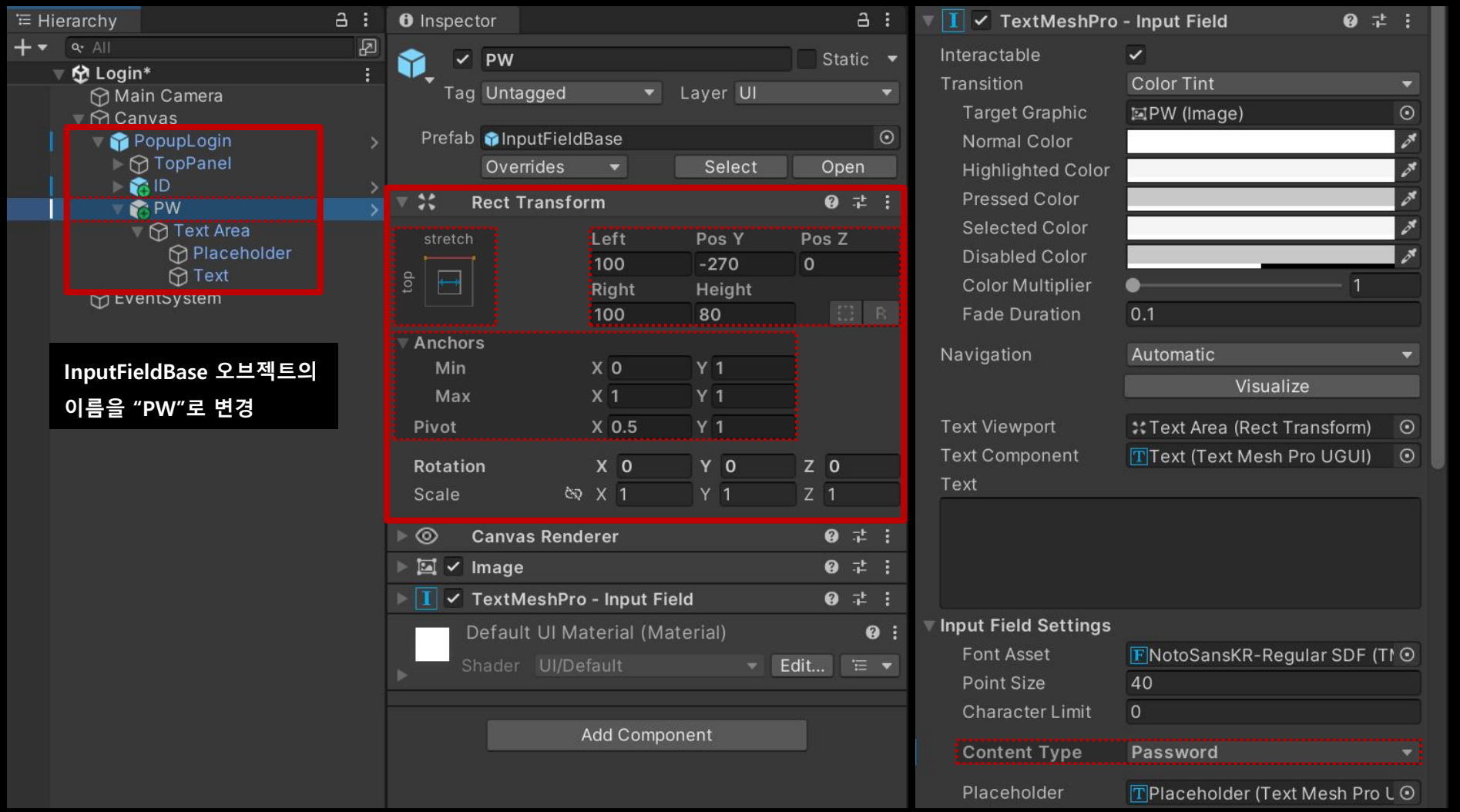
게임 유저 관리

- 아이디 입력 필드 제작을 위한 오브젝트 설정



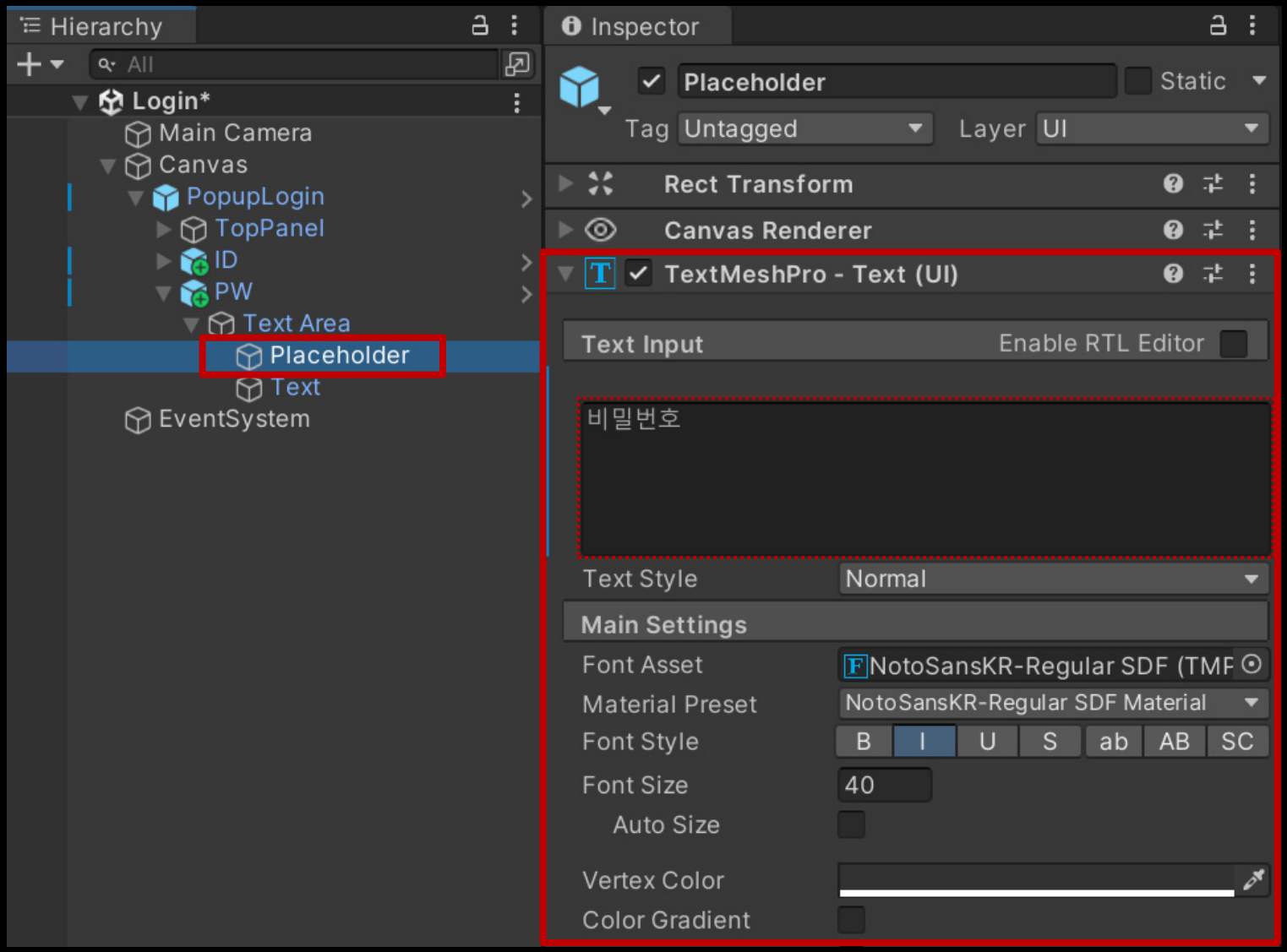
게임 유저 관리

- 비밀번호 입력 필드 생성 및 설정
  - InputFieldBase 프리팹을 Hierarchy View로 Drag & Drop



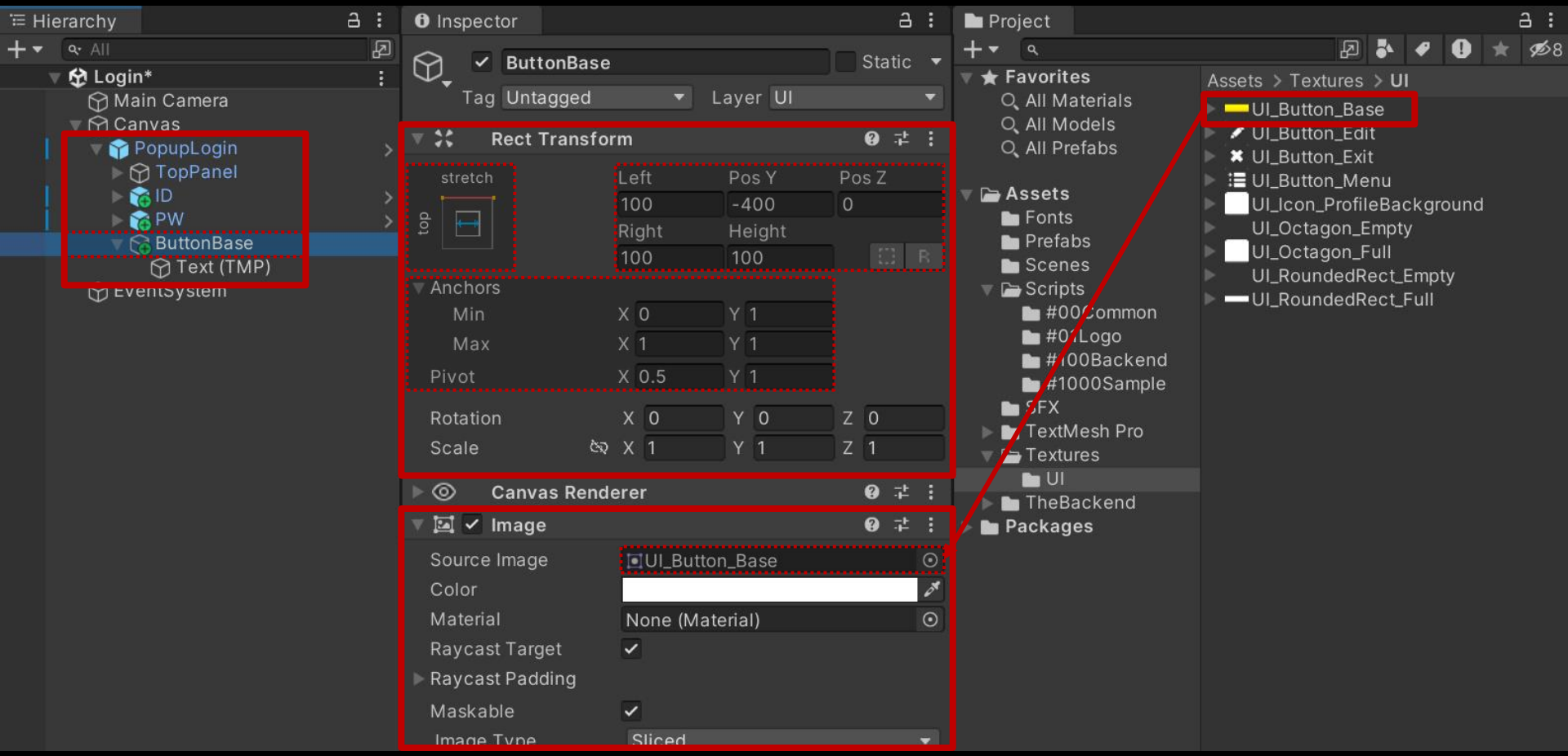
게임 유저 관리

- 비밀번호 입력 필드 생성 및 설정 (계속)



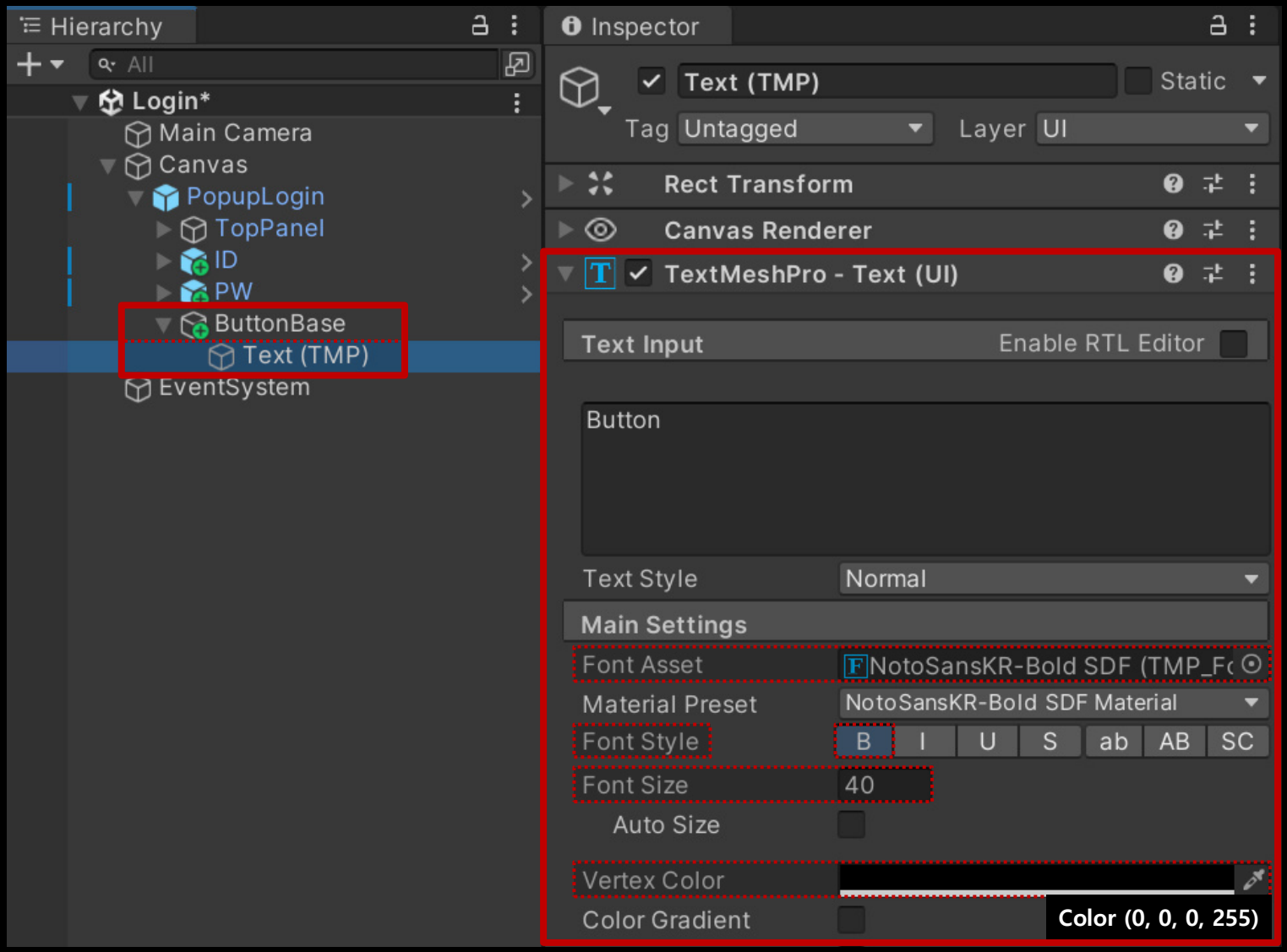
게임 유저 관리

- 클릭 가능한 버튼 "Button - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Button - TextMeshPro"



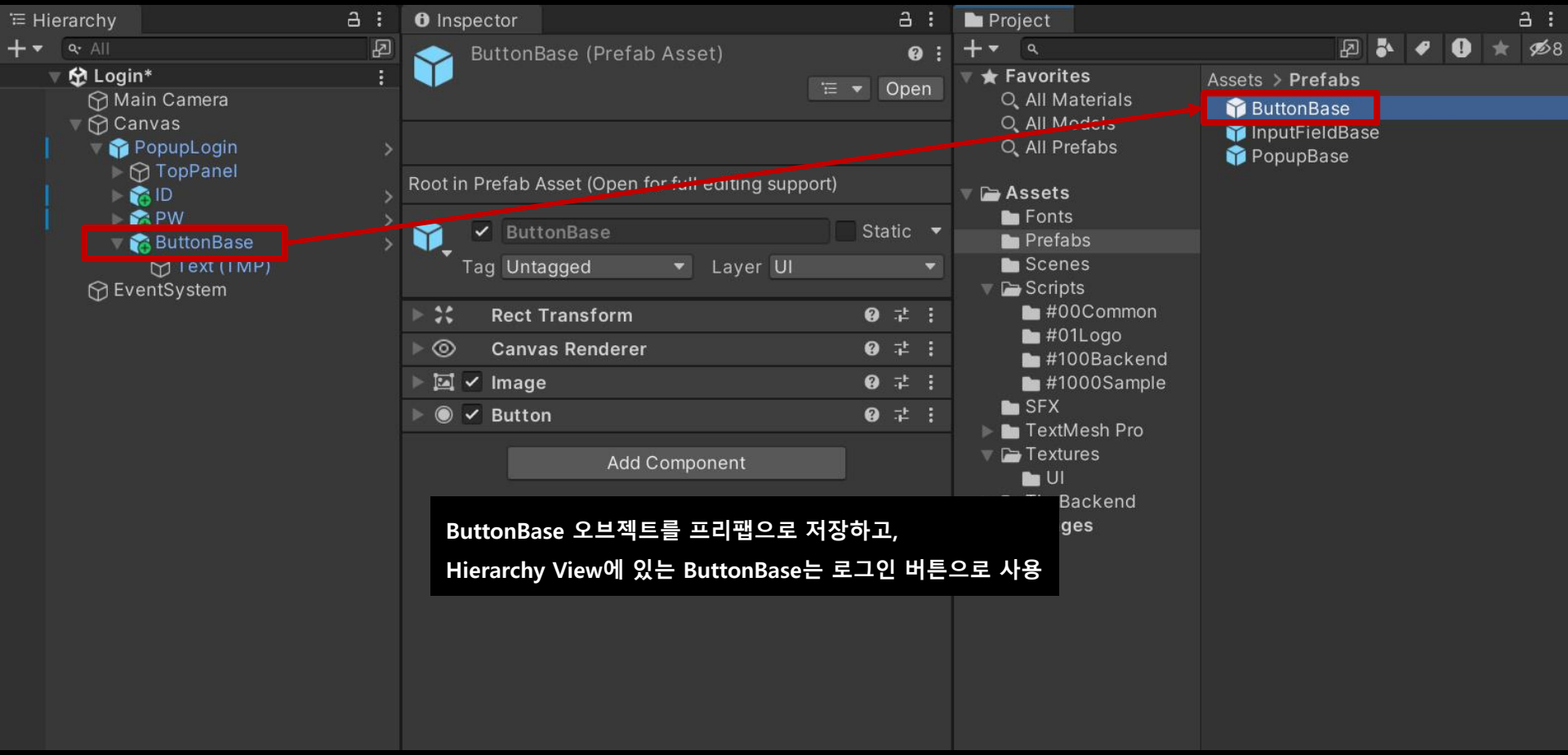
게임 유저 관리

- 클릭 가능한 버튼 "Button - TextMeshPro" UI 생성 및 설정 (계속)



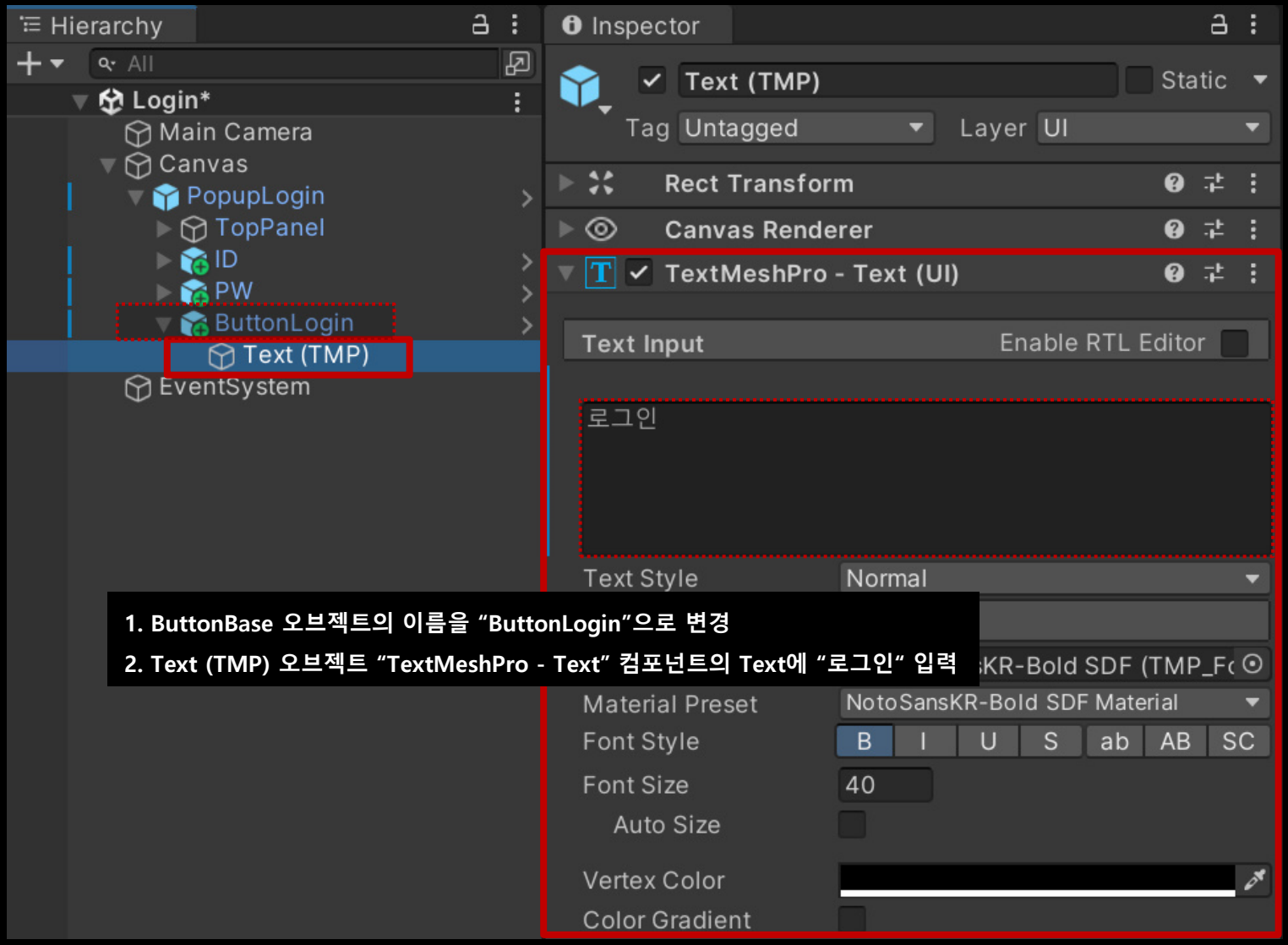
게임 유저 관리

- 버튼 오브젝트 Prefab 생성
  - Hierarchy View의 "ButtonBase" 오브젝트를 Project View로 Drag & Drop



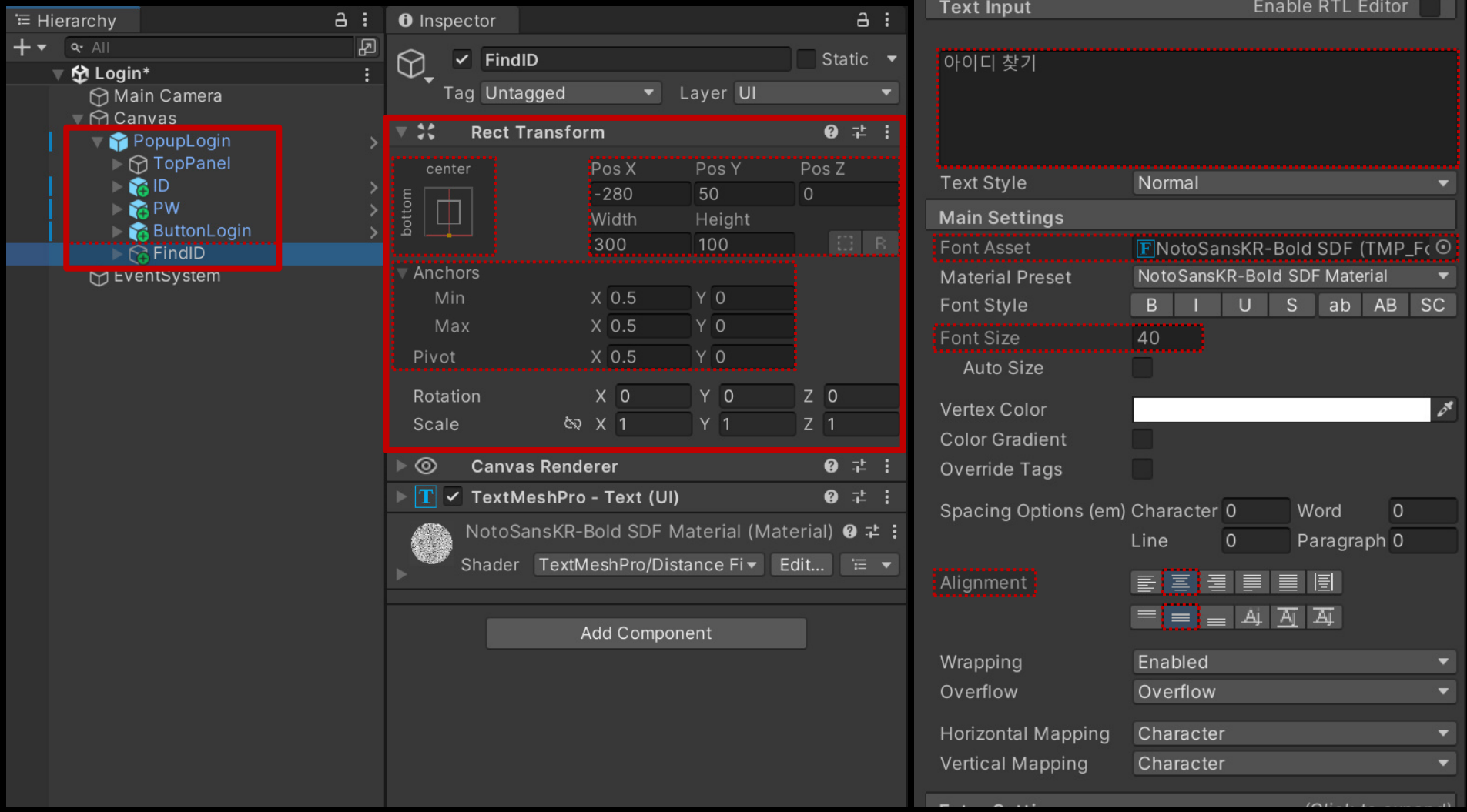
게임 유저 관리

- "로그인" 버튼 제작을 위한 오브젝트 설정



게임 유저 관리

- "아이디 찾기" 텍스트를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"



게임 유저 관리

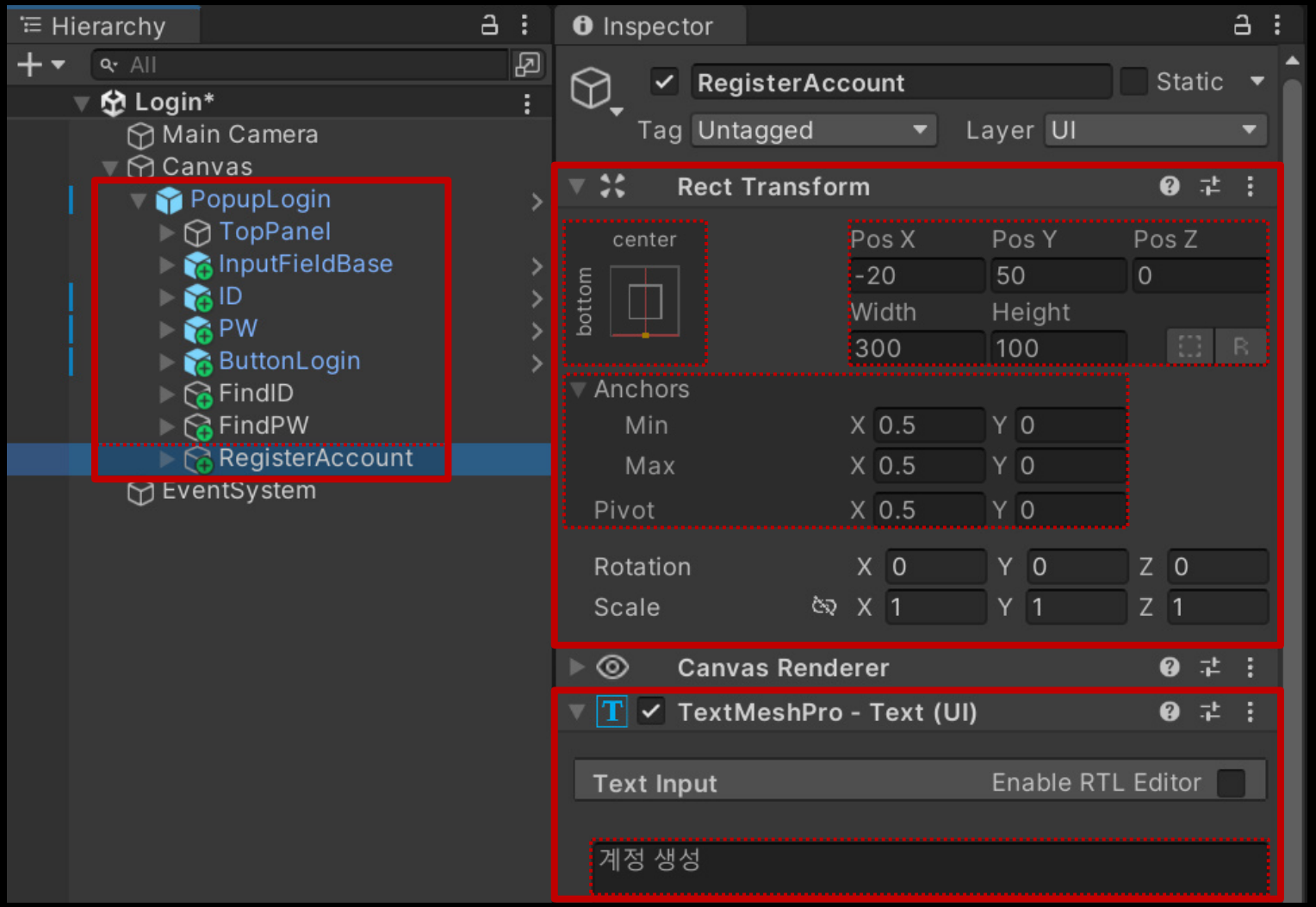
- "비밀번호 찾기" 텍스트를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - FindID 오브젝트를 Ctrl+D로 복제한 후 이름을 FindPW로 변경

Hierarchy
All
Login*
Main Camera
Canvas
PopupLogin
TopPanel
InputFieldBase
ID
PW
ButtonLogin
FindID
FindPW
EventSystem

Inspector
FindPW
Static
Tag Untagged
Layer UI
Rect Transform
center
bottom
Pos X 260
Pos Y 50
Pos Z 0
Width 300
Height 100
Anchors
Min X 0.5 Y 0
Max X 0.5 Y 0
Pivot X 0.5 Y 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
TextMeshPro - Text (UI)
Text Input
Enable RTL Editor
비밀번호 찾기

게임 유저 관리

- "계정 생성" 텍스트를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - FindID 오브젝트를 Ctrl+D로 복제한 후 이름을 RegisterAccount로 변경



게임 유저 관리

- 시스템 메시지를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"

Hierarchy
All
Login*
Main Camera
Canvas
PopupLogin
Message
EventSystem

Inspector
Message
Static
Tag Untagged
Layer UI
Rect Transform
stretch
middle
Left 0
Pos Y -450
Pos Z 0
Right 0
Height 100
Anchors
Min X 0 Y 0.5
Max X 1 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
TextMeshPro - Text (UI)
LiberationSans SDF Material (Material)
Shader TextMeshPro/Mobile/Dist
Edit...
Add Component

TextMeshPro - Text (UI)
Text Input
Enable RTL Editor
Text Style Normal
Main Settings
Font Asset NotoSansKR-Bold SDF (TMP_Fo
Material Preset NotoSansKR-Bold SDF Material
Font Style B I U S ab AB SC
Font Size 40
Auto Size
Vertex Color
Color Gradient
Override Tags
Spacing Options (em) Character 0 Word 0
Line 0 Paragraph 0
Alignment
Wrapping Enabled
Overflow Overflow
Horizontal Mapping Character
Vertical Mapping Character

게임 유저 관리

- 게임 유저 관리에서 사용하는 UI들을 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "LoginBase"로 변경

```
1   using UnityEngine;
2   using UnityEngine.UI;
3   using TMPro;
4
5   public class LoginBase : MonoBehaviour
6   {
7       [SerializeField]
8       private TextMeshProUGUI textMessage;
9
10      /// <summary>
11      /// 메세지 내용, InputField 색상 초기화
12      /// </summary>
13      protected void ResetUI(params Image[] images)
14      {
15          textMessage.text = string.Empty;
16
17          for ( int i = 0; i < images.Length; ++ i )
18          {
19              images[i].color = Color.white;
20          }
21      }
22
```

게임 유저 관리

- 게임 유저 관리에서 사용하는 UI들을 제어하는 스크립트 생성 및 작성 (계속)

```
23      /// <summary>
24      /// 매개변수에 있는 내용을 출력
25      /// </summary>
26      protected void SetMessage(string msg)
27      {
28          textMessage.text = msg;
29      }
30
31      /// <summary>
32      /// 입력 오류가 있는 InputField 색상 변경
33      /// 오류에 대한 메세지 출력
34      /// </summary>
35      protected void GuideForIncorrectlyEnteredData(Image image, string msg)
36      {
37          textMessage.text = msg;
38          image.color      = Color.red;
39      }
40
```

게임 유저 관리

- 게임 유저 관리에서 사용하는 UI들을 제어하는 스크립트 생성 및 작성 (계속)

```
41      /// <summary>
42      /// 필드 값이 비어있는지 확인 (image:필드 색상, field:내용, result:출력될 내용)
43      /// </summary>
44      protected bool IsFieldDataEmpty(Image image, string field, string result)
45      {
46          if ( field.Trim().Equals("") )
47          {
48              GuideForIncorrectlyEnteredData(image, $"\"{result}\" 필드를 채워주세요.");
49
50              return true;
51          }
52
53          return false;
54      }
55  }
```

게임 유저 관리

- 뒤끝 서버와 연동해 로그인을 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "Login"으로 변경

```
1  using System.Collections;
2  using UnityEngine;
3  using UnityEngine.UI;
4  using TMPro;
5  using BackEnd;
6
7  public class Login : LoginBase
8  {
9      [SerializeField]
10     private Image              imageID;          // ID 필드 색상 변경
11     [SerializeField]
12     private TMP_InputField     inputFieldID;     // ID 필드 텍스트 정보 추출
13     [SerializeField]
14     private Image              imagePW;          // PW 필드 색상 변경
15     [SerializeField]
16     private TMP_InputField     inputFieldPW;     // PW 필드 텍스트 정보 추출
17
18     [SerializeField]
19     private Button             btnLogin;         // 로그인 버튼 (상호작용 가능/불가능)
20
```

게임 유저 관리

- 뒤끝 서버와 연동해 로그인을 제어하는 스크립트 생성 및 작성 (계속)

```
21      /// <summary>
22      /// "로그인" 버튼을 눌렀을 때 호출
23      /// </summary>
24      public void OnClickLogin()
25      {
26          // 매개변수로 입력한 InputField UI의 색상과 Message 내용 초기화
27          ResetUI(imageID, imagePW);
28
29          // 필드 값이 비어있는지 체크
30          if ( IsFieldDataEmpty(imageID, inputFieldID.text, "아이디") )    return;
31          if ( IsFieldDataEmpty(imagePW, inputFieldPW.text, "비밀번호") )  return;
32
33          // 로그인 버튼을 연타하지 못하도록 상호작용 비활성화
34          btnLogin.interactable = false;
35
36          // 서버에 로그인을 요청하는 동안 화면에 출력하는 내용 업데이트
37          // ex) 로그인 관련 텍스트 출력, 톱니바퀴 아이콘 회전 등
38          StartCoroutine(nameof(LoginProcess));
39
40          // 뒤끝 서버 로그인 시도
41          ResponseToLogin(inputFieldID.text, inputFieldPW.text);
42      }
43
```

게임 유저 관리

- 뒤끝 서버와 연동해 로그인을 제어하는 스크립트 생성 및 작성 (계속)

```
44      /// <summary>
45      /// 로그인 시도 후 서버로부터 전달받은 message를 기반으로 로직 처리
46      /// </summary>
47      private void ResponseToLogin(string ID, string PW)
48      {
49          // 서버에 로그인 요청
50          Backend.BMember.CustomLogin(ID, PW, callback =>
51          {
52              StopCoroutine(nameof(LoginProcess));
53
54              // 로그인 성공
55              if ( callback.IsSuccess() )
56              {
57                  SetMessage($"{inputFieldID.text}님 환영합니다.");
58              }
59              // 로그인 실패
60              else...
93          });
94      }
95
```

뒷장

게임 유저 관리

- 뒤끝 서버와 연동해 로그인을 제어하는 스크립트 생성 및 작성 (계속)

```
59              // 로그인 실패
60              else
61              {
62                  // 로그인에 실패했을 때는 다시 로그인을 해야하기 때문에 "로그인" 버튼 상호작용 활성화
63                  btnLogin.interactable = true;
64
65                  string message = string.Empty;
66
67                  switch ( int.Parse(callback.GetStatusCode()) )
68                  {
69                      case 401:   // 존재하지 않는 아이디, 잘못된 비밀번호
70                          message = callback.GetMessage().Contains("customId") ? "존재하지 않는 아이디입니다." : "잘못된 비밀번호 입니다.";
71                          break;
72                      case 403:   // 유저 or 디바이스 차단
73                          message = callback.GetMessage().Contains("user") ? "차단당한 유저입니다." : "차단당한 디바이스입니다.";
74                          break;
75                      case 410:   // 탈퇴 진행중
76                          message = "탈퇴가 진행중인 유저입니다.";
77                          break;
78                      default:
79                          message = callback.GetMessage();
80                          break;
81                  }
82
83                  // StatusCode 401에서 "잘못된 비밀번호 입니다." 일 때
84                  if ( message.Contains("비밀번호") )
85                  {
86                      GuideForIncorrectlyEnteredData(imagePW, message);
87                  }
88                  else
89                  {
90                      GuideForIncorrectlyEnteredData(imageID, message);
91                  }
92              }
```

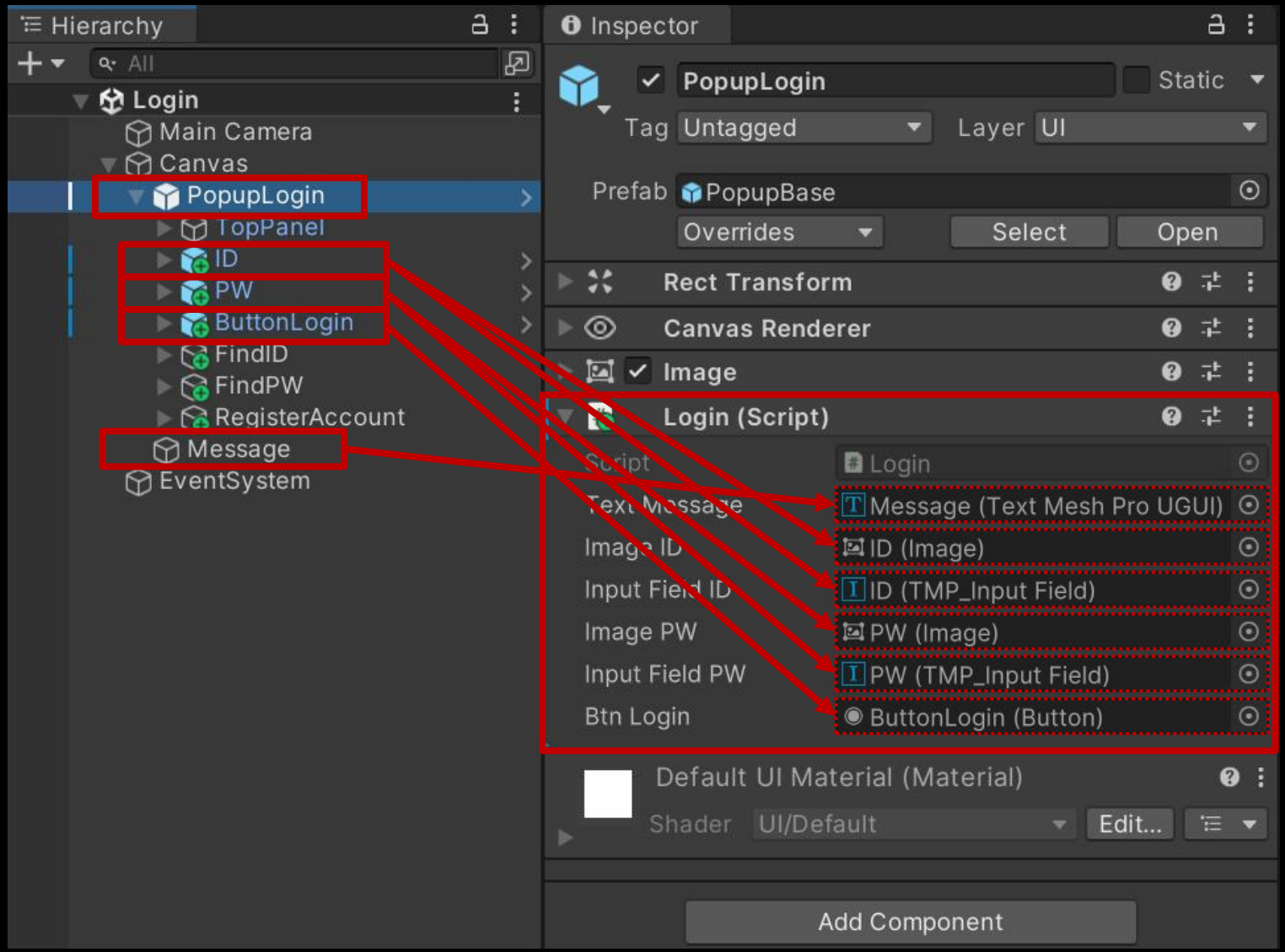
게임 유저 관리

- 뒤끝 서버와 연동해 로그인을 제어하는 스크립트 생성 및 작성 (계속)

```
96      private IEnumerator LoginProcess()
97      {
98          float time = 0;
99
100         while ( true )
101         {
102             time += Time.deltaTime;
103
104             SetMessage($"로그인 중입니다... {time:F1}");
105
106             yield return null;
107         }
108     }
109 }
```

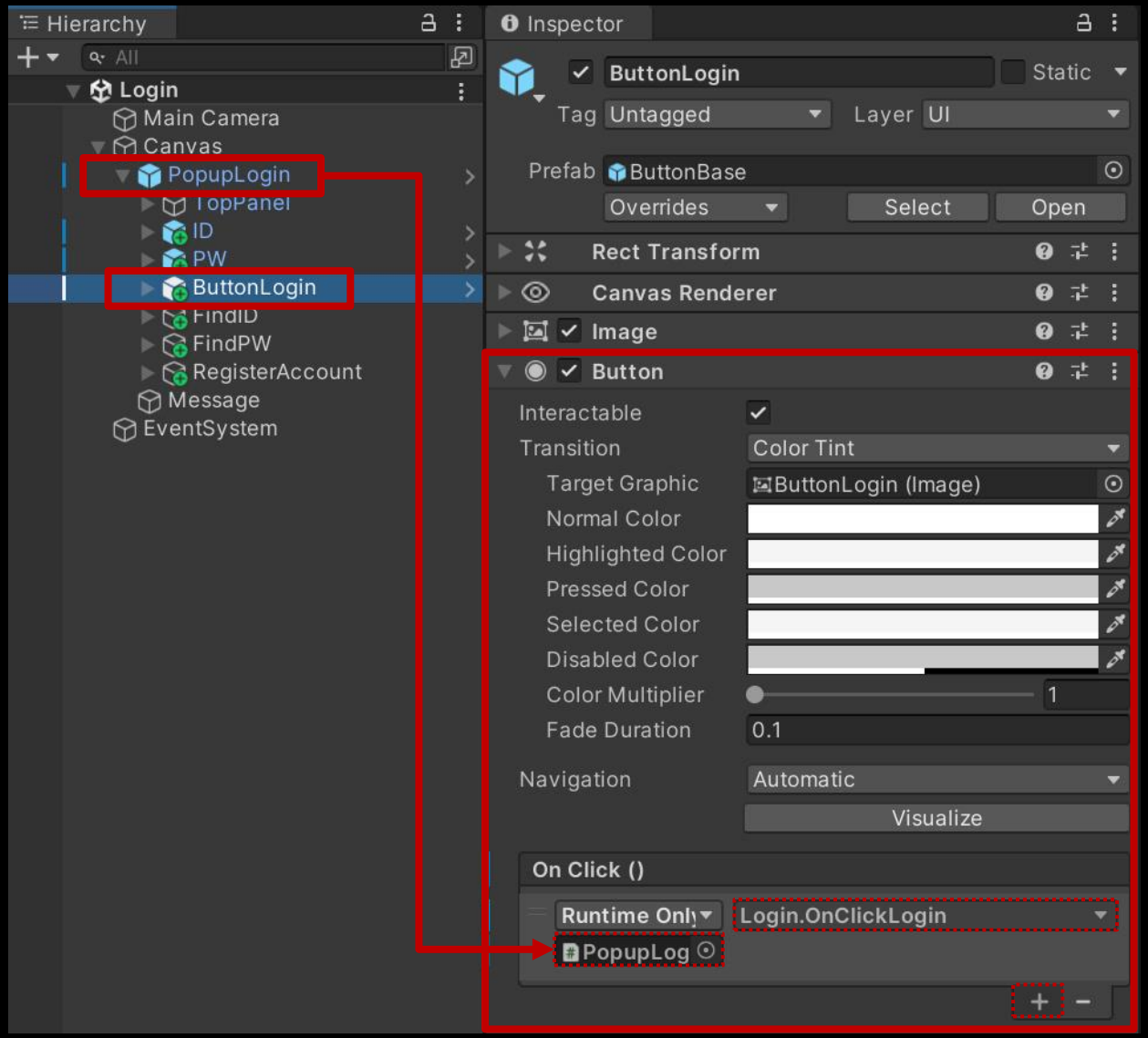
게임 유저 관리

- PopupLogin 오브젝트에 "Login" 컴포넌트 추가 및 설정



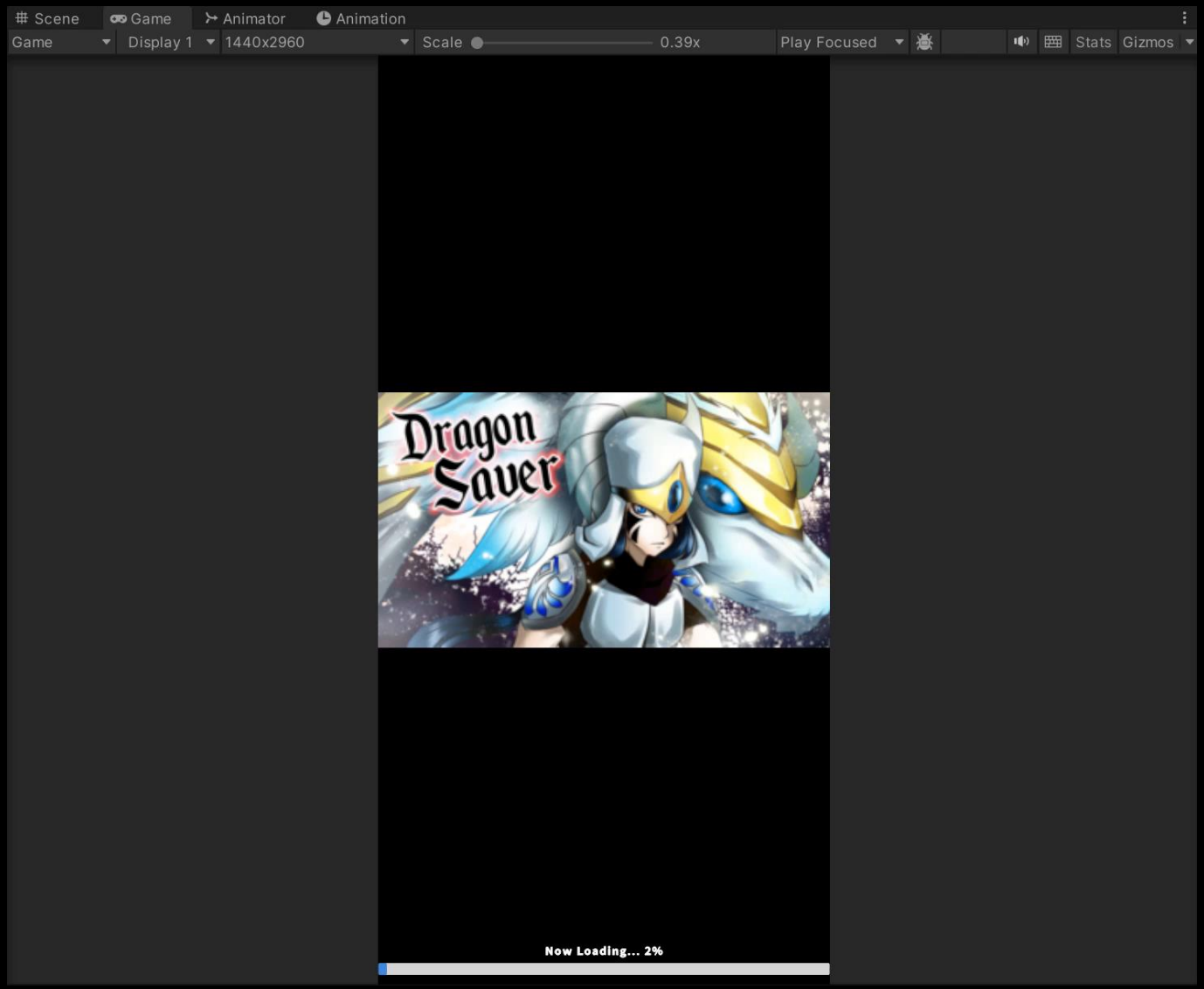
게임 유저 관리

- ButtonLogin 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



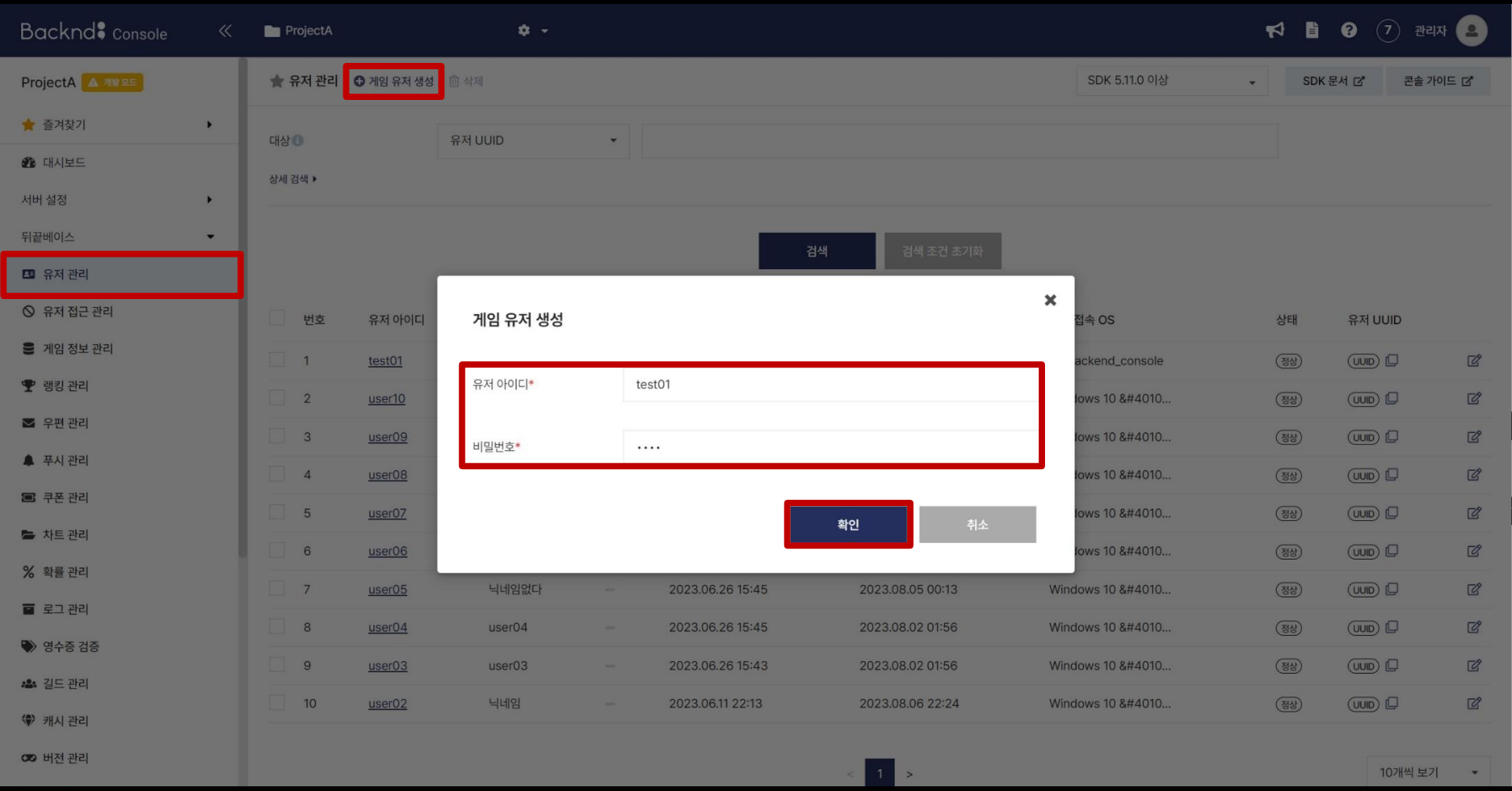
게임 유저 관리

- 결과 화면


Scene
Game
Animator
Animation
Game
Display 1
1440x2960
Scale
0.39x
Play Focused
Stats
Gizmos
Dragon Saver
Now Loading... 2%

게임 유저 관리

- Backend Console에서 계정 생성
  - "유저 관리" 탭 - "게임 유저 생성" 버튼



게임 유저 관리

- Backend Console에서 계정 생성 (계속)

Backnd Console

ProjectA

관리자

ProjectA 개발 모드

- 즐겨찾기
- 대시보드
- 서버 설정
- 뒤끝베이스
- 유저 관리
- 유저 접근 관리
- 게임 정보 관리
- 랭킹 관리
- 우편 관리
- 푸시 관리
- 쿠폰 관리
- 차트 관리
- 확률 관리
- 로그 관리
- 영수증 검증
- 길드 관리
- 캐시 관리
- 버전 관리
- 공지사항

유저 관리 | 게임 유저 생성 | 삭제

SDK 5.11.0 이상 | SDK 문서 | 콘솔 가이드

대상 | 유저 UUID

상세 검색

검색 | 검색 조건 초기화

번호	유저 아이디	닉네임	국가	가입일	최근 접속일	최근 접속 OS	상태	유저 UUID
1	test01	—	—	2023.08.13 21:28	2023.08.13 21:28	thebackend_console	정상	UUID
2	user10	user10	—	2023.08.03 14:18	2023.08.06 22:45	Windows 10 &#4010...	정상	UUID
3	user09	user09	—	2023.08.03 14:18	2023.08.06 22:27	Windows 10 &#4010...	정상	UUID
4	user08	user08	—	2023.08.03 14:17	2023.08.06 22:27	Windows 10 &#4010...	정상	UUID
5	user07	user07	—	2023.08.03 14:17	2023.08.03 14:17	Windows 10 &#4010...	정상	UUID
6	user06	user06	—	2023.06.26 20:36	2023.08.03 14:16	Windows 10 &#4010...	정상	UUID
7	user05	닉네임없다	—	2023.06.26 15:45	2023.08.05 00:13	Windows 10 &#4010...	정상	UUID
8	user04	user04	—	2023.06.26 15:45	2023.08.02 01:56	Windows 10 &#4010...	정상	UUID
9	user03	user03	—	2023.06.26 15:43	2023.08.02 01:56	Windows 10 &#4010...	정상	UUID
10	user02	닉네임	—	2023.06.11 22:13	2023.08.06 22:24	Windows 10 &#4010...	정상	UUID

< 1 >

10개씩 보기

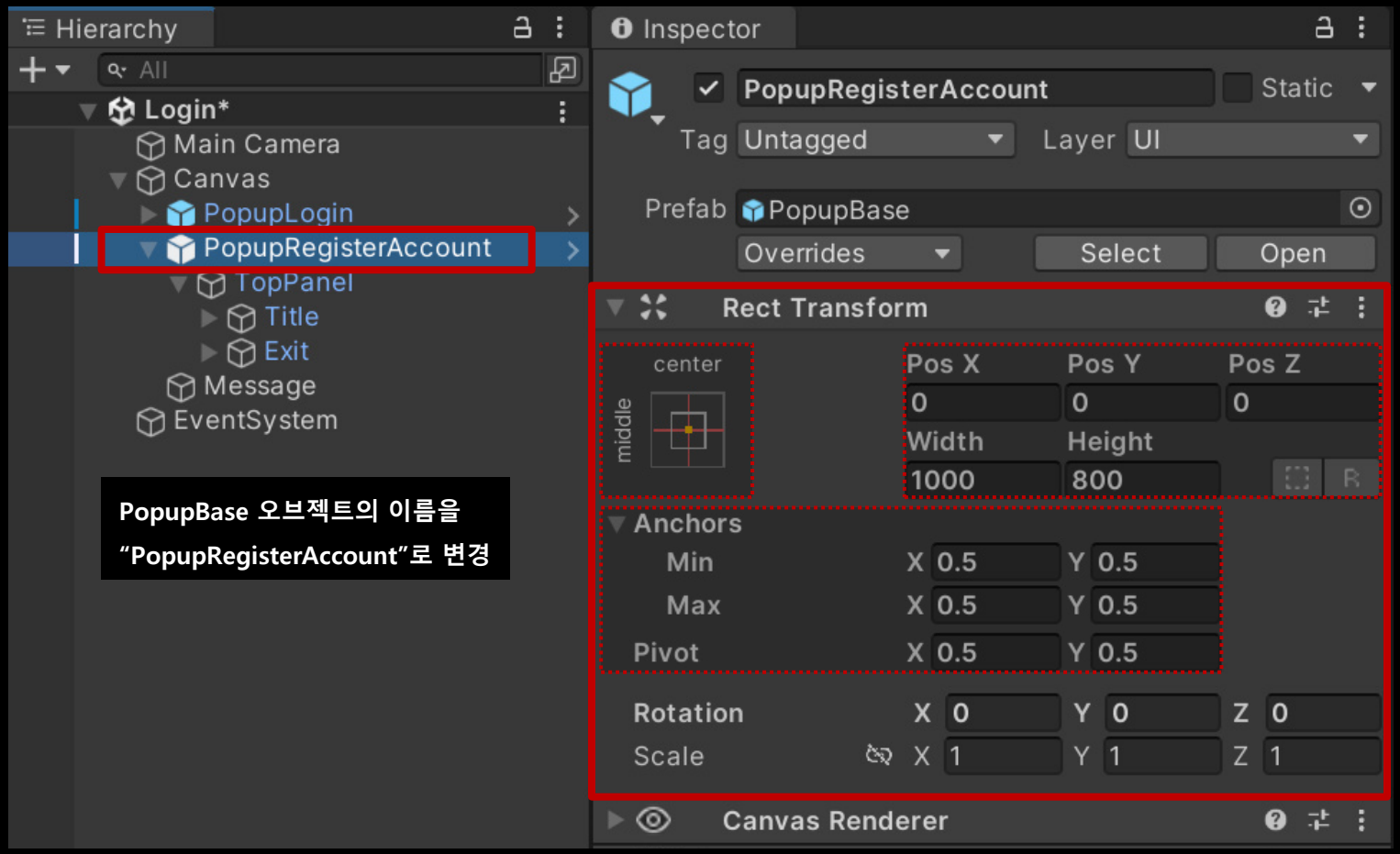
회사소개 이용약관 서비스수준협약 개인정보처리방침

게임 유저 관리

계정 생성, 이메일 설정

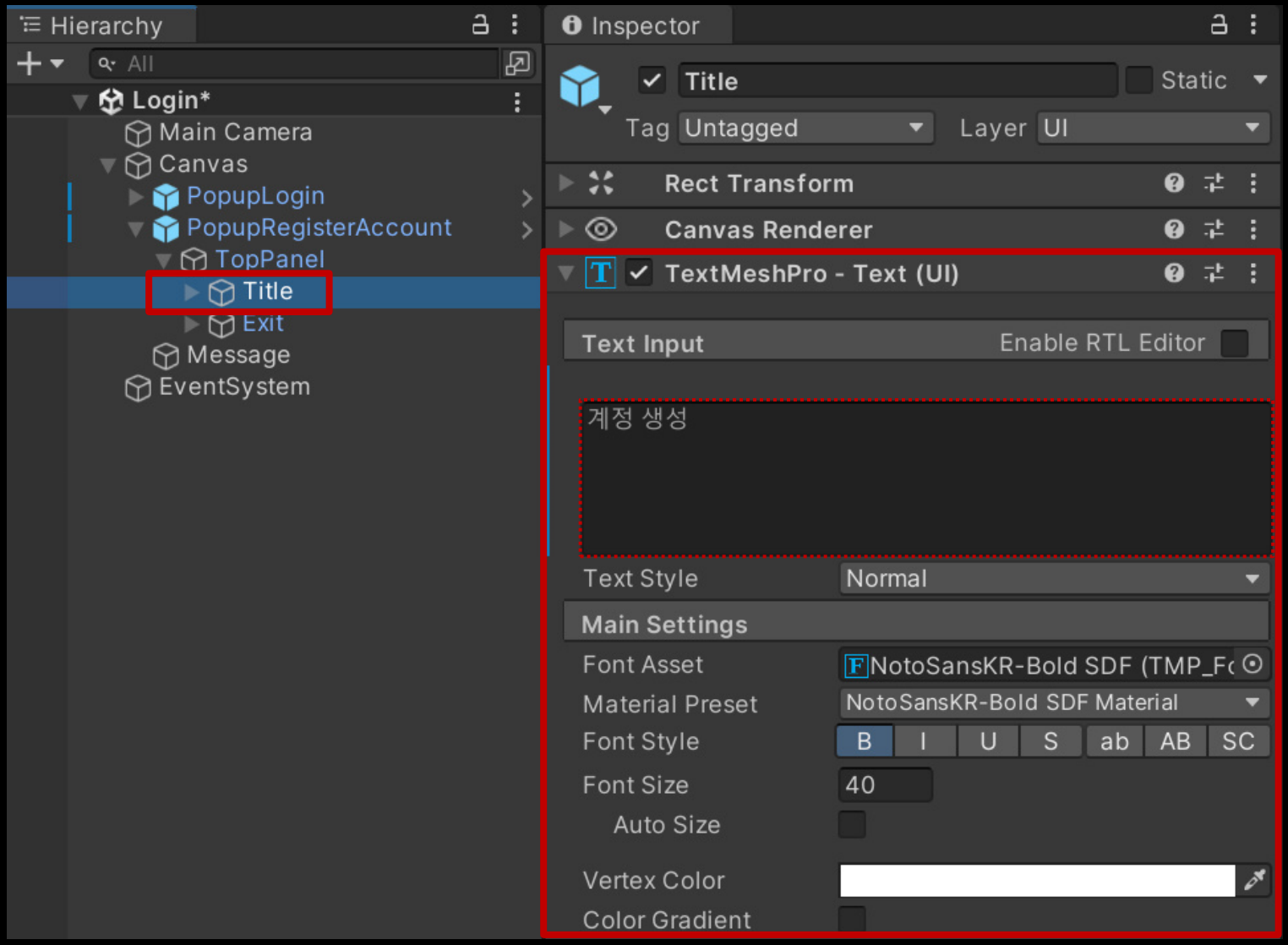
계정 생성 팝업 윈도우 생성 및 설정

- PopupBase 프리팹을 Hierarchy View로 Drag & Drop



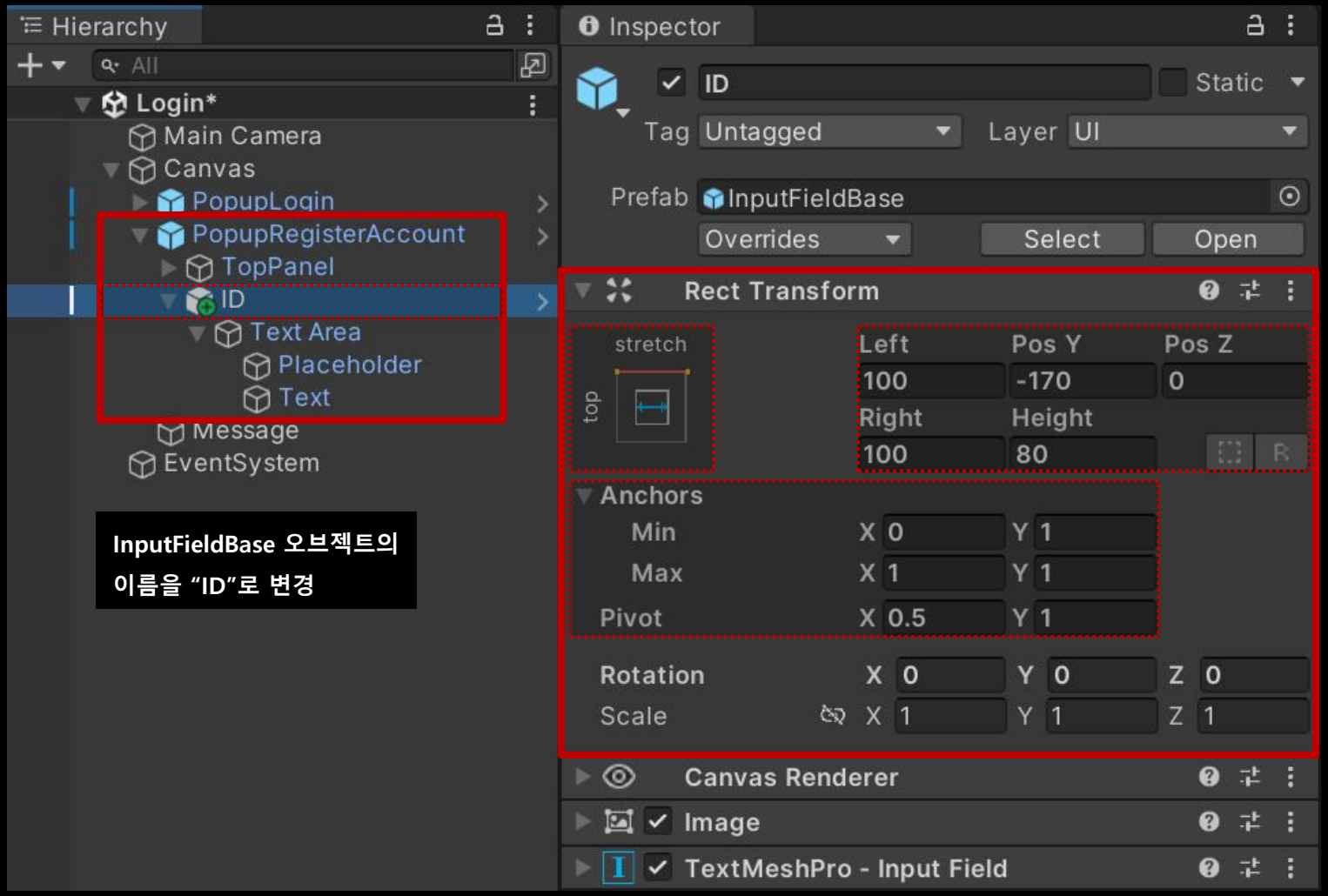
게임 유저 관리

- 계정 생성 팝업 윈도우 상단 이름 설정



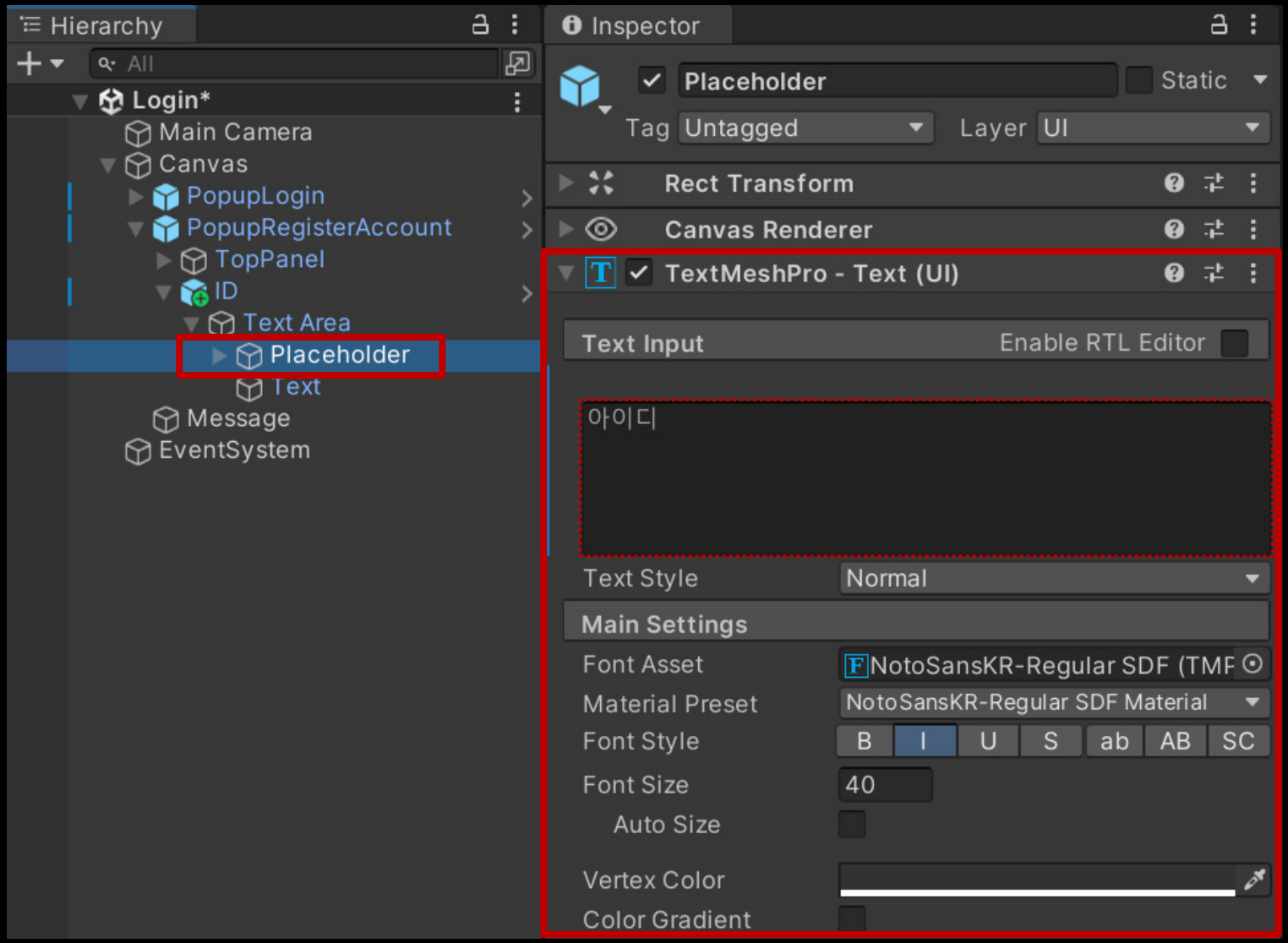
게임 유저 관리

- 아이디 입력 필드 생성 및 설정
  - InputFieldBase 프리팹을 Hierarchy View로 Drag & Drop



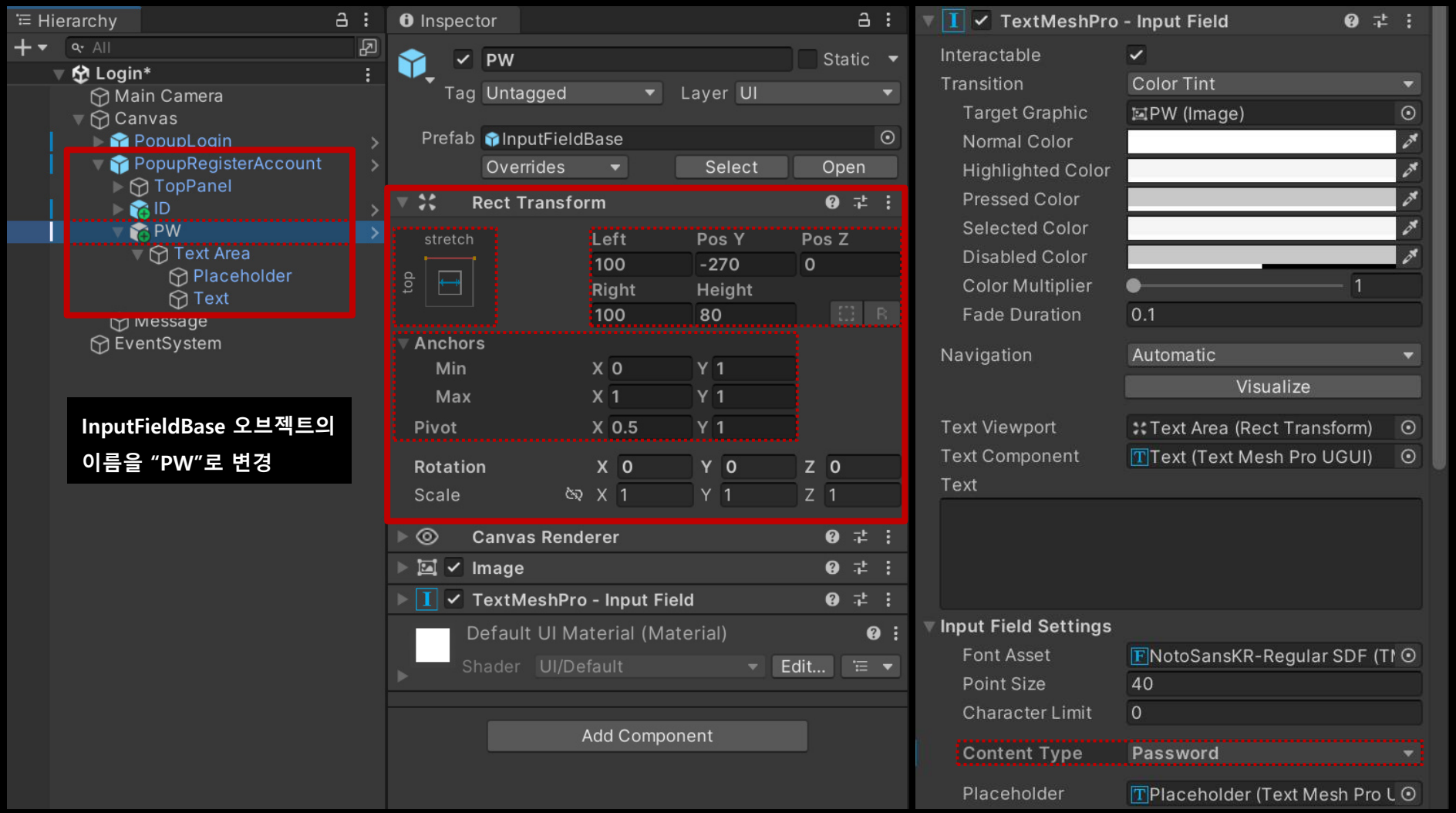
게임 유저 관리

- 아이디 입력 필드 생성 및 설정 (계속)



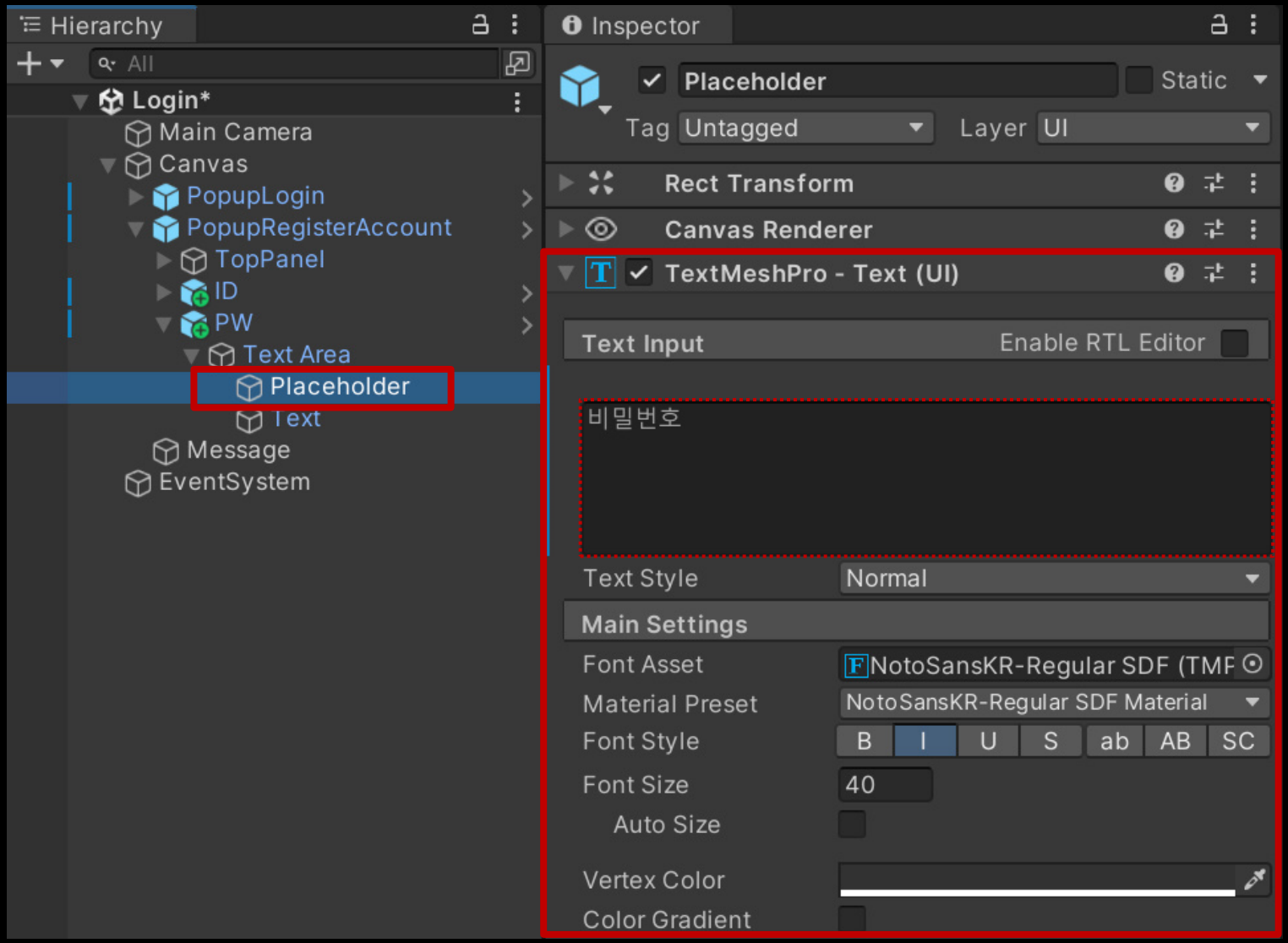
게임 유저 관리

- 비밀번호 입력 필드 생성 및 설정
  - InputFieldBase 프리팹을 Hierarchy View로 Drag & Drop



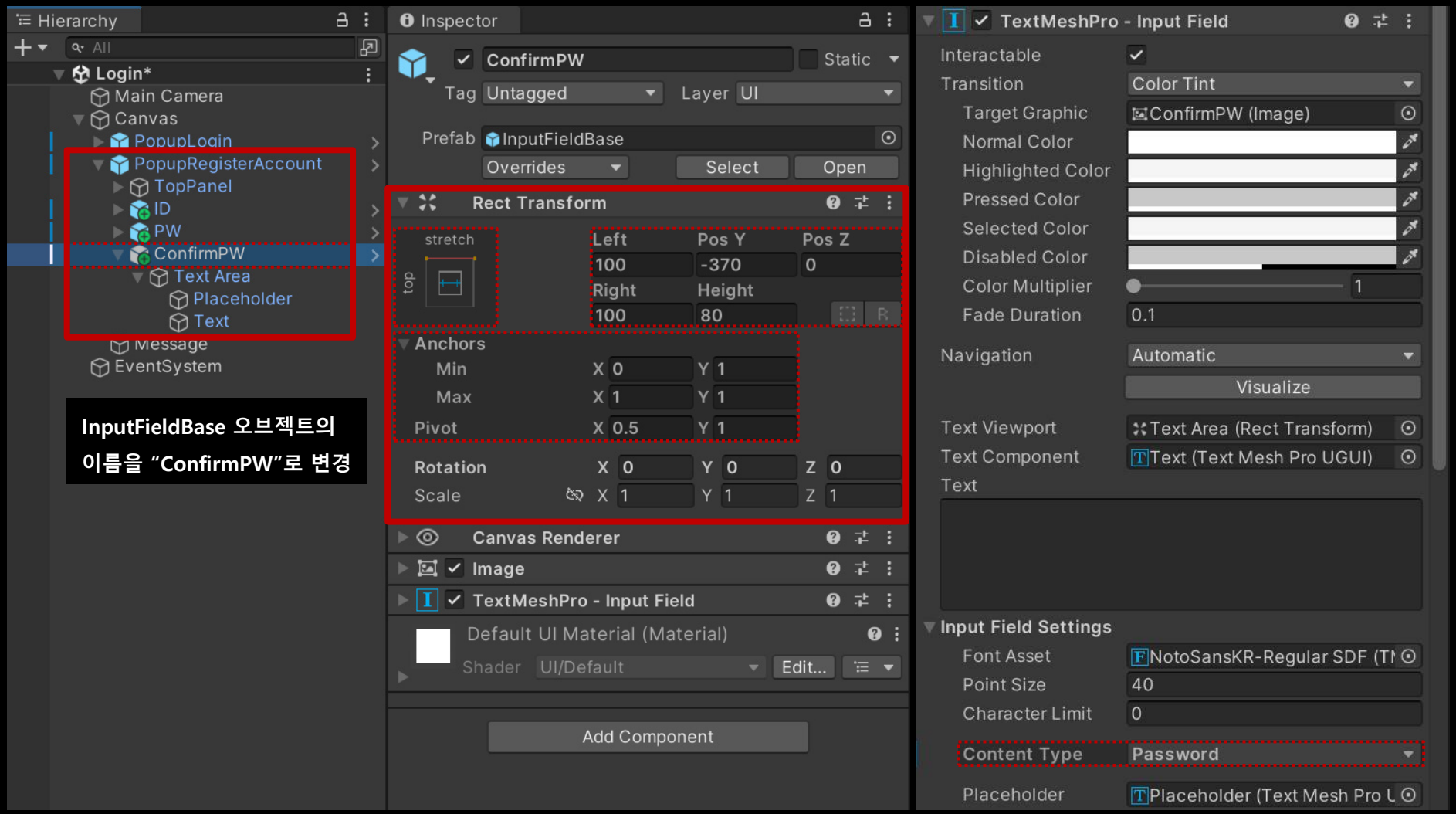
게임 유저 관리

- 비밀번호 입력 필드 생성 및 설정 (계속)



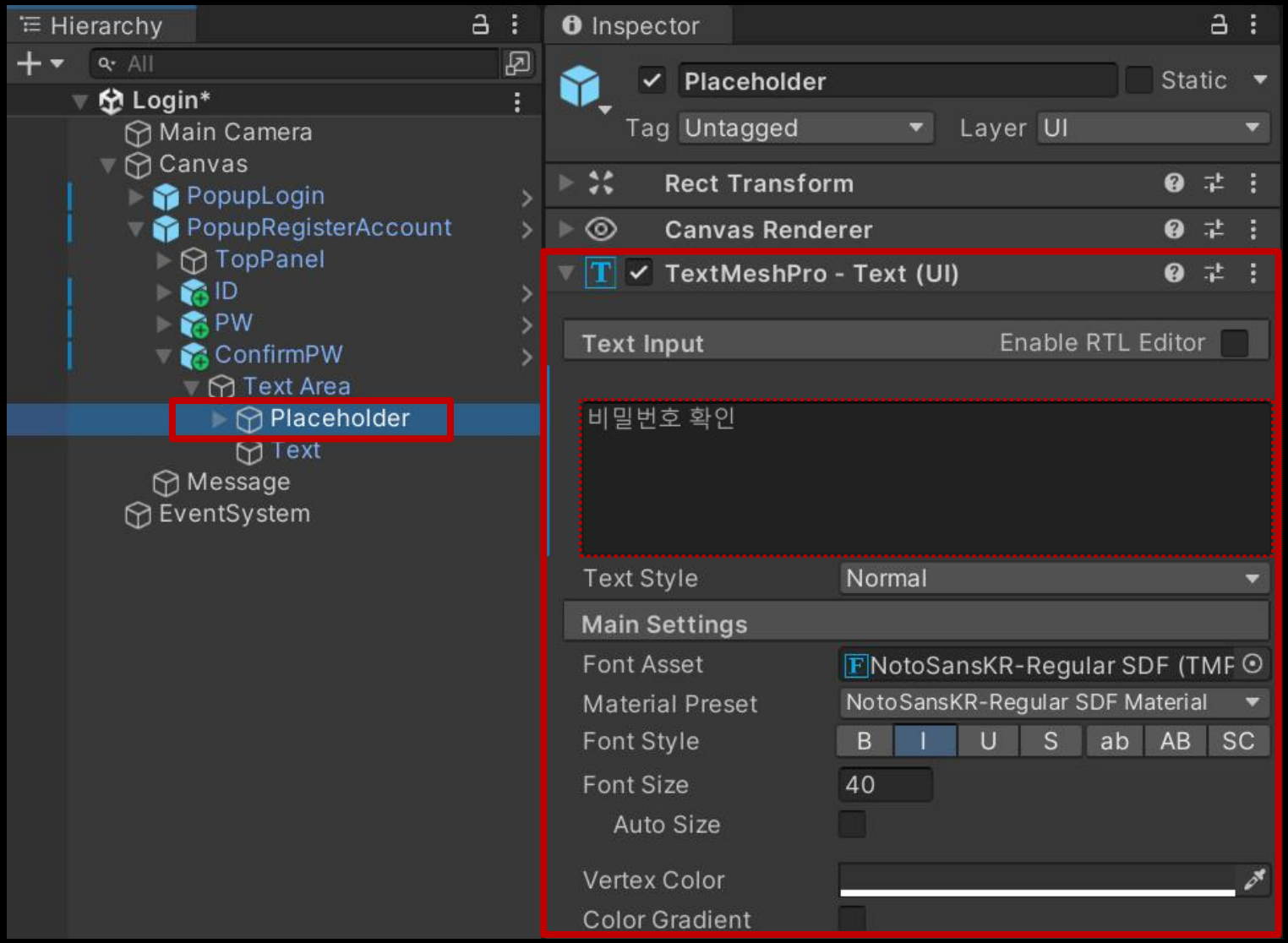
게임 유저 관리

- 비밀번호 확인 입력 필드 생성 및 설정
  - InputFieldBase 프리팹을 Hierarchy View로 Drag & Drop



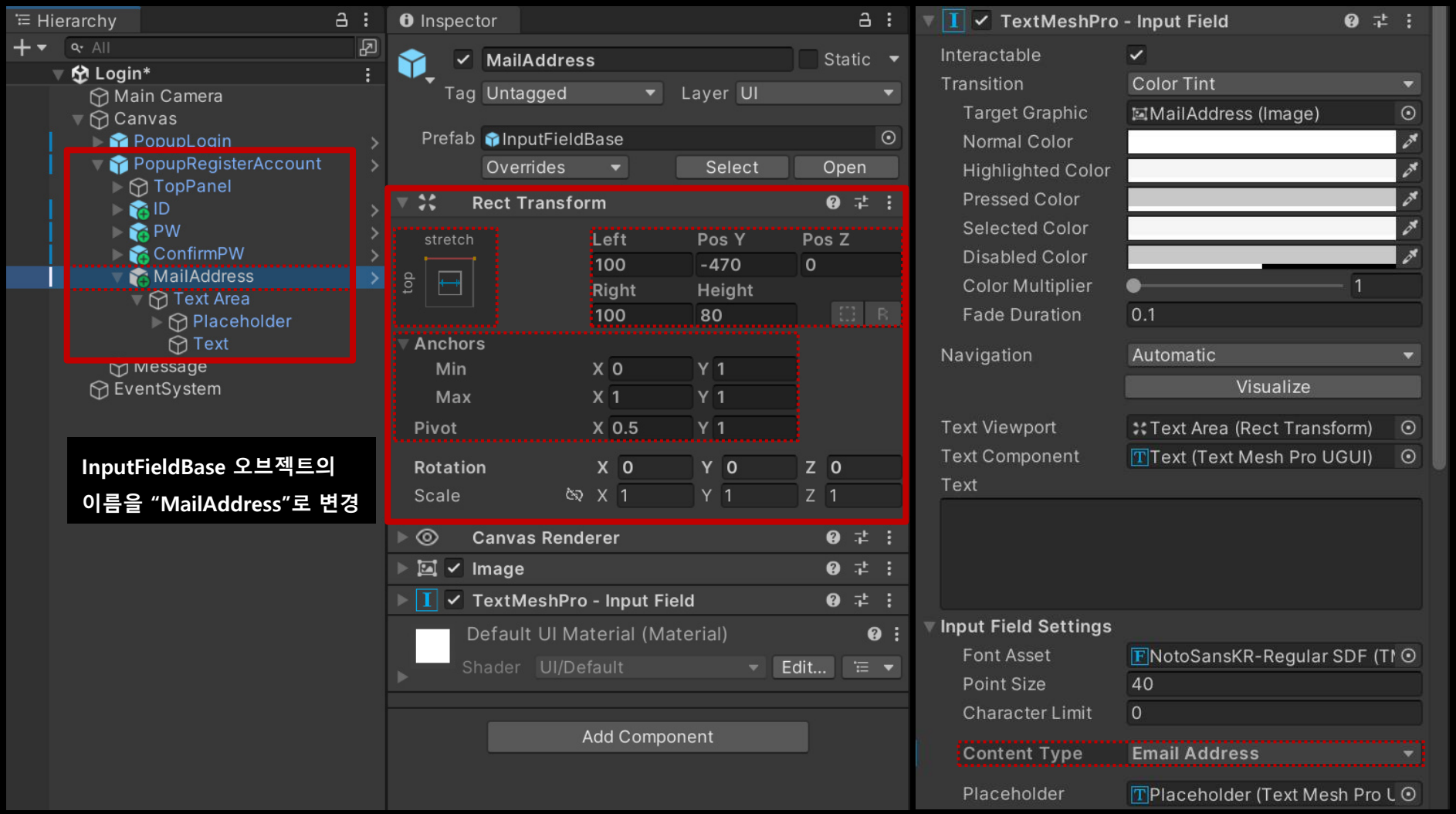
게임 유저 관리

- 비밀번호 확인 입력 필드 생성 및 설정 (계속)



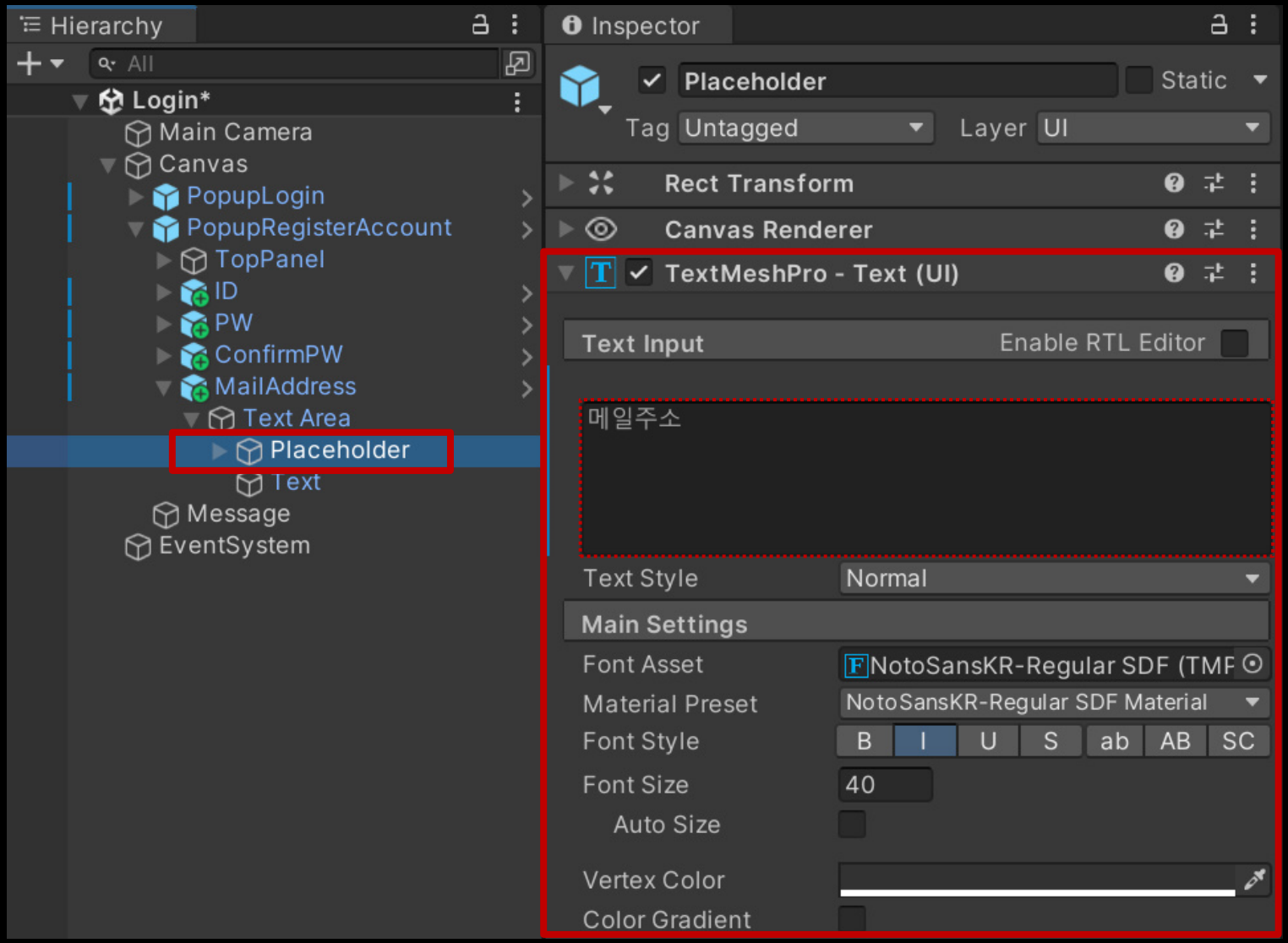
게임 유저 관리

- 메일 주소 입력 필드 생성 및 설정
  - InputFieldBase 프리팹을 Hierarchy View로 Drag & Drop



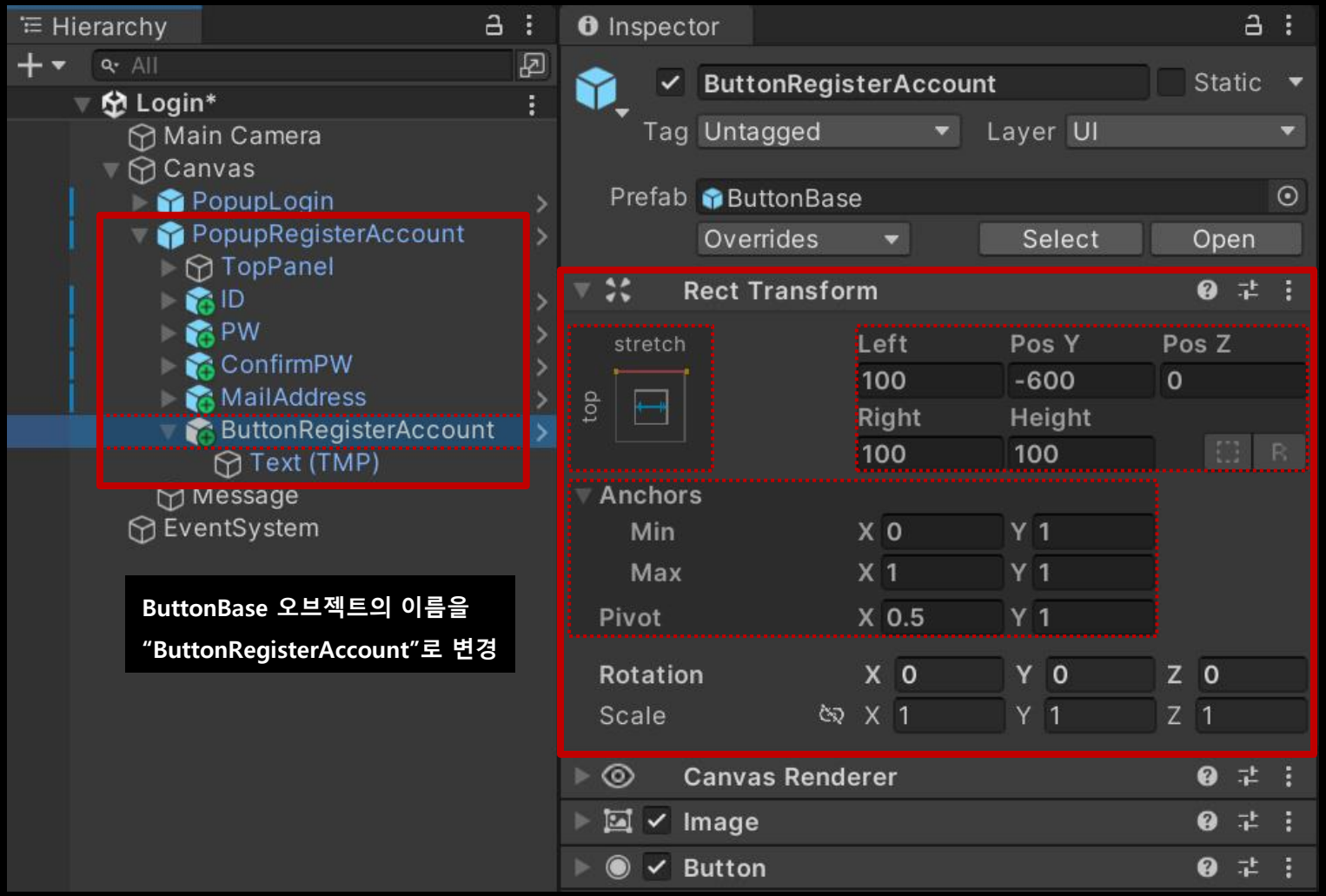
게임 유저 관리

- 메일 주소 입력 필드 생성 및 설정 (계속)



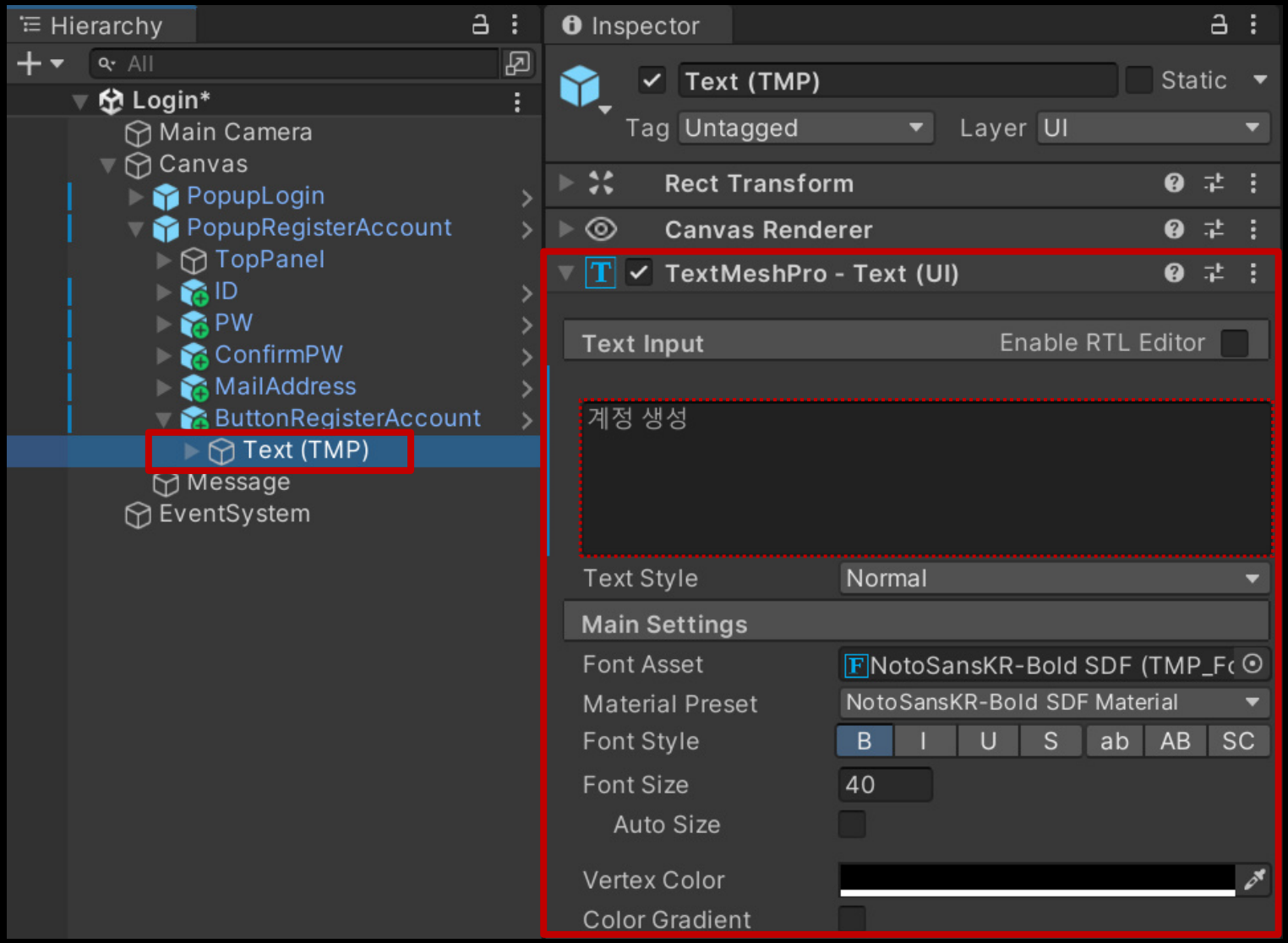
게임 유저 관리

- "계정 생성" 버튼 생성 및 설정
  - ButtonBase 프리팹을 Hierarchy View로 Drag & Drop



게임 유저 관리

- "계정 생성" 버튼 생성 및 설정 (계속)



게임 유저 관리

- 뒤끝 서버와 연동해 계정 생성을 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "RegisterAccount"로 변경

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using BackEnd;

public class RegisterAccount : LoginBase
{
    [SerializeField]
    private Image              imageID;              // ID 필드 색상 변경
    [SerializeField]
    private TMP_InputField     inputFieldID;         // ID 필드 텍스트 정보 추출
    [SerializeField]
    private Image              imagePW;              // PW 필드 색상 변경
    [SerializeField]
    private TMP_InputField     inputFieldPW;         // PW 필드 텍스트 정보 추출
    [SerializeField]
    private Image              imageConfirmPW;       // Confirm PW 필드 색상 변경
    [SerializeField]
    private TMP_InputField     inputFieldConfirmPW;  // Confirm PW 필드 텍스트 정보 추출
    [SerializeField]
    private Image              imageEmail;           // E-mail 필드 색상 변경
    [SerializeField]
    private TMP_InputField     inputFieldEmail;      // E-mail 필드 텍스트 정보 추출

    [SerializeField]
    private Button             btnRegisterAccount;   // "계정 생성" 버튼 (상호작용 가능/불가능)

```

게임 유저 관리

- 뒤끝 서버와 연동해 계정 생성을 제어하는 스크립트 생성 및 작성 (계속)

```
28      /// <summary>
29      /// "계정 생성" 버튼을 눌렀을 때 호출
30      /// </summary>
31      public void OnClickRegisterAccount()
32      {
33          // 매개변수로 입력한 InputField UI의 색상과 Message 내용 초기화
34          ResetUI(imageID, imagePW, imageConfirmPW, imageEmail);
35
36          // 필드 값이 비어있는지 체크
37          if ( IsFieldDataEmpty(imageID, inputFieldID.text, "아이디") )                    return;
38          if ( IsFieldDataEmpty(imagePW, inputFieldPW.text, "비밀번호") )                  return;
39          if ( IsFieldDataEmpty(imageConfirmPW, inputFieldConfirmPW.text, "비밀번호 확인") )  return;
40          if ( IsFieldDataEmpty(imageEmail, inputFieldEmail.text, "메일 주소") )           return;
41
42          // 비밀번호와 비밀번호 확인의 내용이 다를 때
43          if ( !inputFieldPW.text.Equals(inputFieldConfirmPW.text) )
44          {
45              GuideForIncorrectlyEnteredData(imageConfirmPW, "비밀번호가 일치하지 않습니다.");
46              return;
47          }
48
49          // 메일 형식 검사
50          if ( !inputFieldEmail.text.Contains("@") )
51          {
52              GuideForIncorrectlyEnteredData(imageEmail, "메일 형식이 잘못되었습니다.(ex. address@xx.xx)");
53              return;
54          }
55
56          // 계정 생성 버튼의 상호작용 비활성화
57          btnRegisterAccount.interactable = false;
58          SetMessage("계정 생성중입니다.");
59
60          // 뒤끝 서버 계정 생성 시도
61          CustomSignUp();
62      }
63
```

게임 유저 관리

- 뒤끝 서버와 연동해 계정 생성을 제어하는 스크립트 생성 및 작성 (계속)

```
64      /// <summary>
65      /// 계정 생성 시도 후 서버로부터 전달받은 message를 기반으로 로직 처리
66      /// </summary>
67      private void CustomSignUp()
68      {
69          Backend.BMember.CustomSignUp(inputFieldID.text, inputFieldPW.text, callback =>
70          {
71              // "계정 생성" 버튼 상호작용 활성화
72              btnRegisterAccount.interactable = true;
73
74              // 계정 생성 성공
75              if ( callback.IsSuccess() )
76              {
77                  // E-mail 정보 업데이트
78                  Backend.BMember.UpdateCustomEmail(inputFieldEmail.text, callback =>
79                  {
80                      if ( callback.IsSuccess() )
81                      {
82                          SetMessage($"계정 생성 성공. {inputFieldID.text}님 환영합니다.");
83                      }
84                  });
85              }
86              // 계정 생성 실패
87              else...
115         });
116     }
117 }
```

뒷장

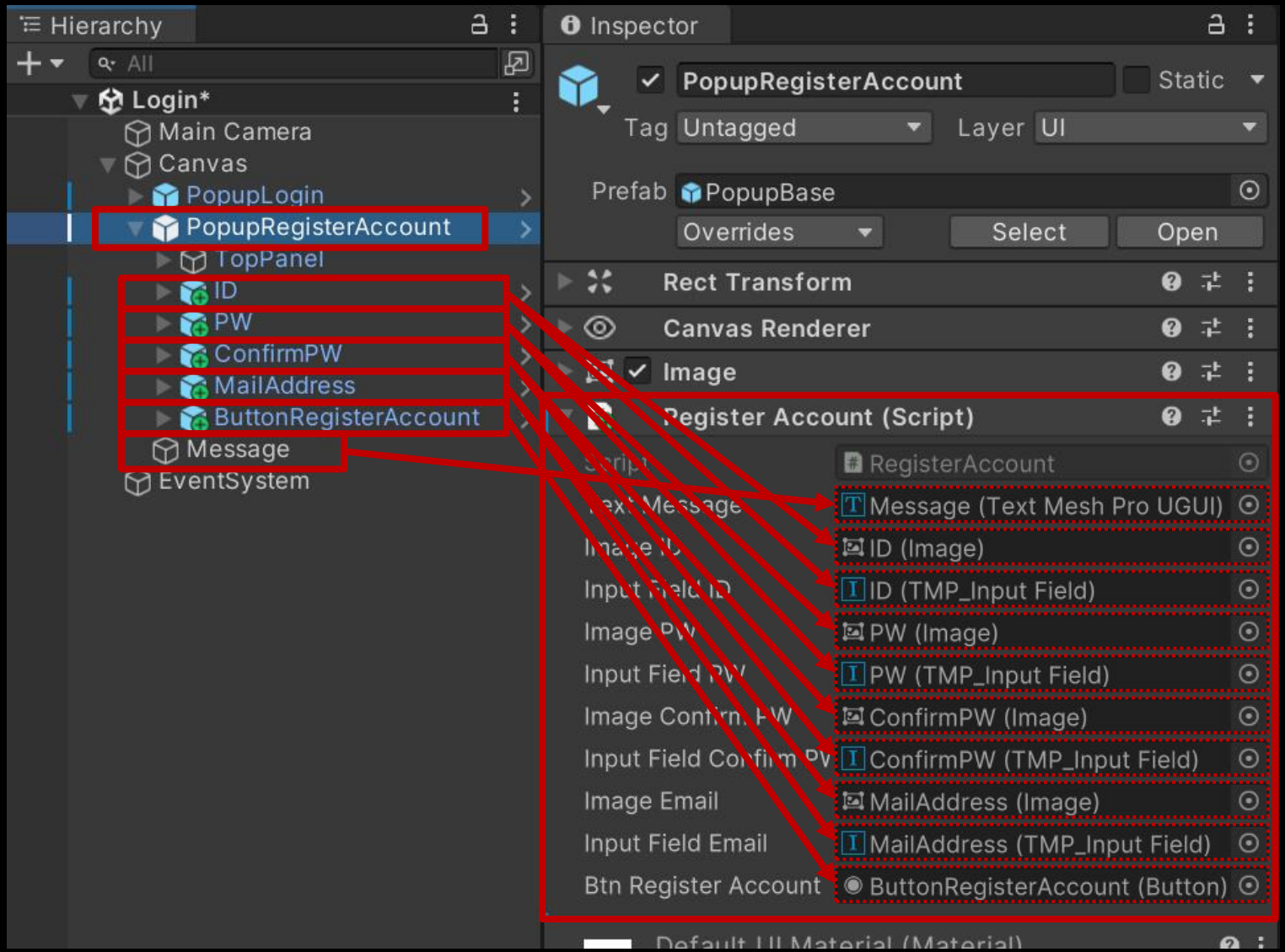
게임 유저 관리

- 뒤끝 서버와 연동해 계정 생성을 제어하는 스크립트 생성 및 작성 (계속)

```
86              // 계정 생성 실패
87              else
88              {
89                  string message = string.Empty;
90
91                  switch ( int.Parse(callback.GetStatusCode()) )
92                  {
93                      case 409:   // 중복된 customId 가 존재하는 경우
94                          message = "이미 존재하는 아이디입니다.";
95                          break;
96                      case 403:   // 차단당한 디바이스일 경우
97                          message = callback.GetMessage();
98                          break;
99                      case 401:   // 프로젝트 상태가 '점검'일 경우
100                     case 400:   // 디바이스 정보가 null일 경우
101                     default:
102                         message = callback.GetMessage();
103                         break;
104                 }
105
106                 if ( message.Contains("아이디") )
107                 {
108                     GuideForIncorrectlyEnteredData(imageID, message);
109                 }
110                 else
111                 {
112                     SetMessage(message);
113                 }
114             }
```

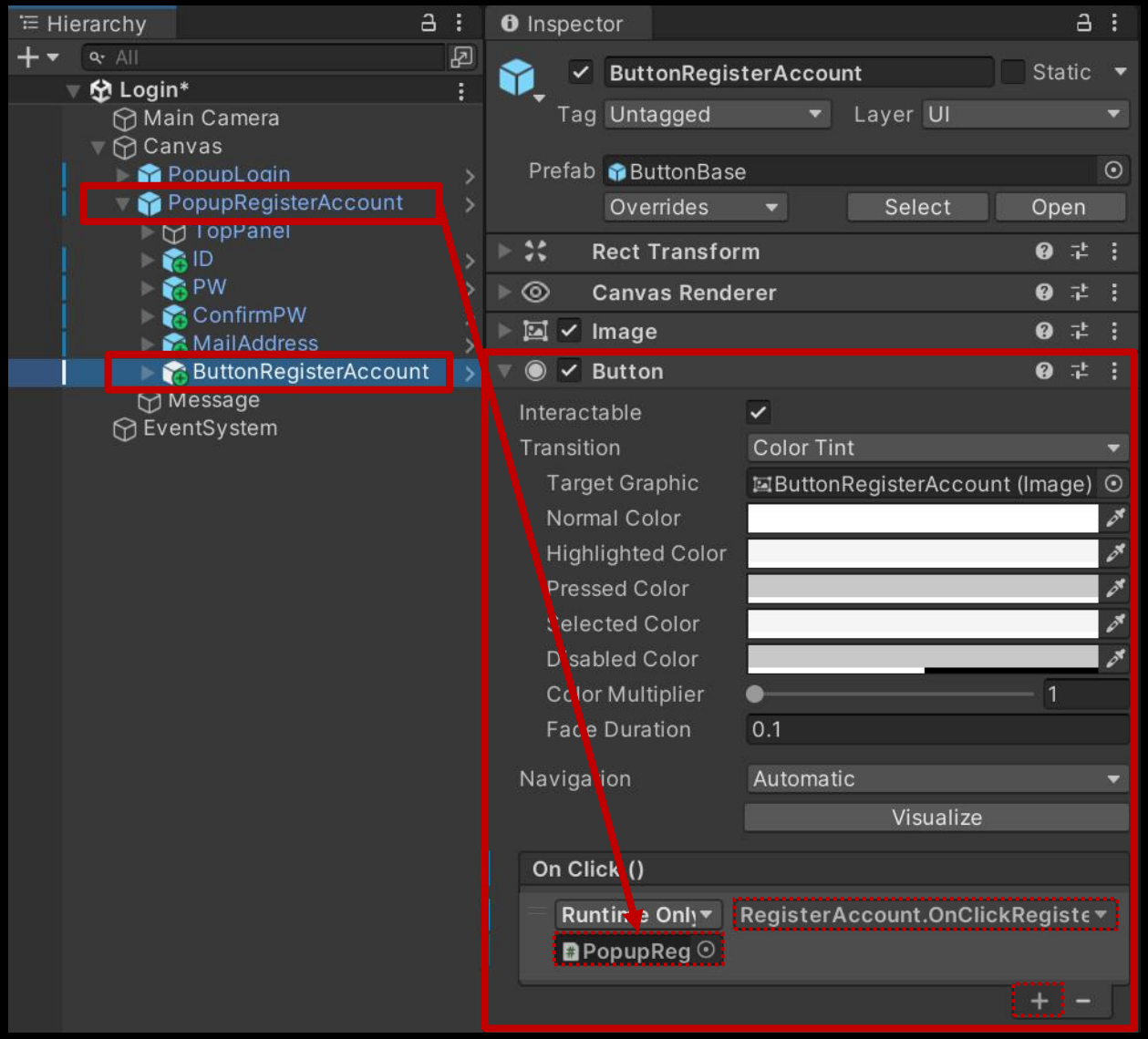
게임 유저 관리

- PopupRegisterAccount 오브젝트에 “RegisterAccount” 컴포넌트 추가 및 설정



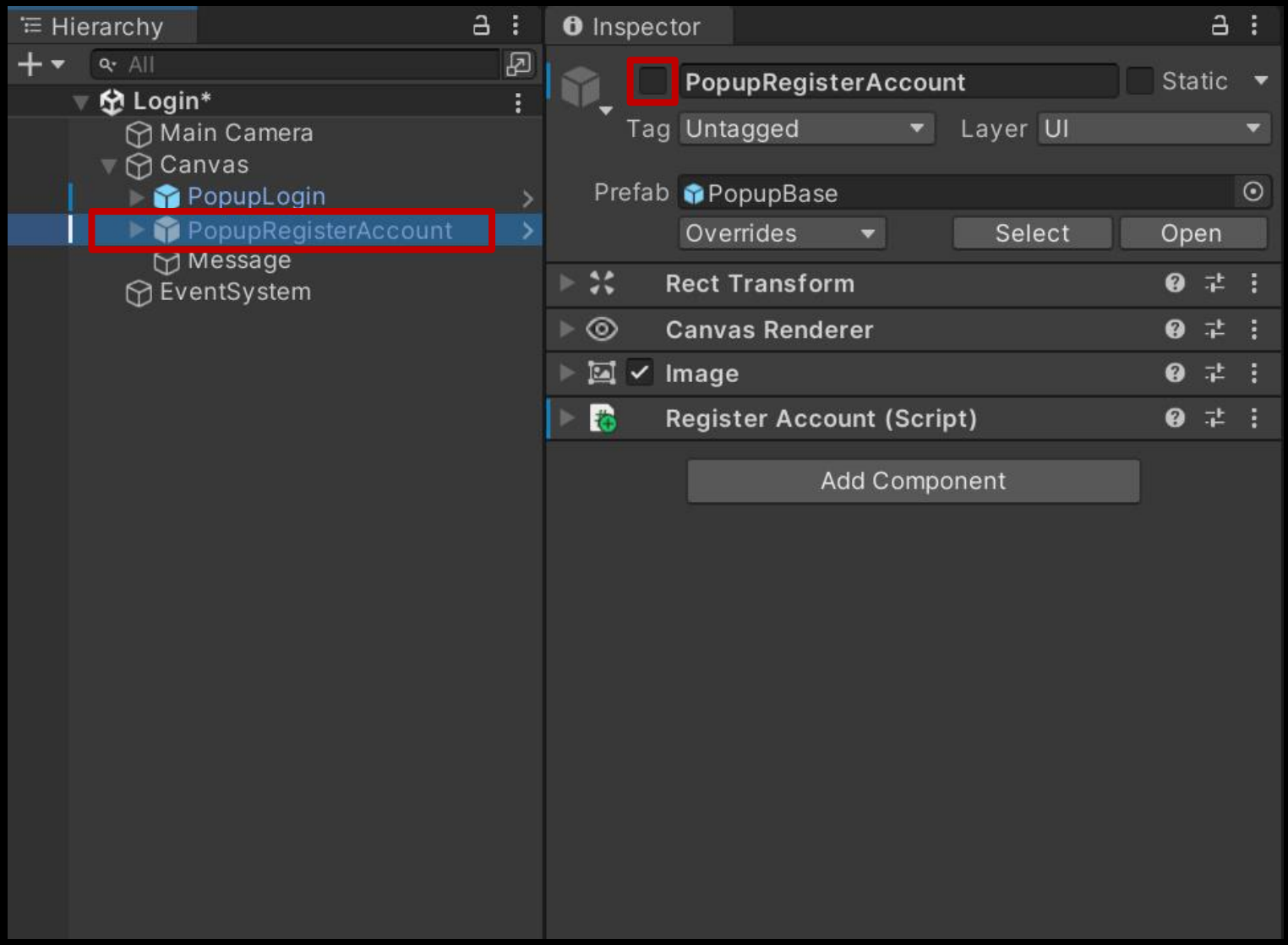
게임 유저 관리

- ButtonRegisterAccount 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



게임 유저 관리

- PopupRegisterAccount 오브젝트 비활성화



게임 유저 관리

- Text UI 상호작용을 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "UITextInteraction"으로 변경

```
1  using UnityEngine;
2  using UnityEngine.EventSystems;
3  using UnityEngine.Events;
4  using TMPro;
5
6  public class UITextInteraction : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler, IPointerClickHandler
7  {
8      [System.Serializable]
9      private class OnClickEvent : UnityEvent { }
10
11     // Text UI를 클릭했을 때 호출하고 싶은 메소드 등록
12     [SerializeField]
13     private OnClickEvent    onClickEvent;
14
15     // 색상이 바뀌고, 터치가 되는 TextMeshProGUI
16     private TextMeshProUGUI text;
17
```

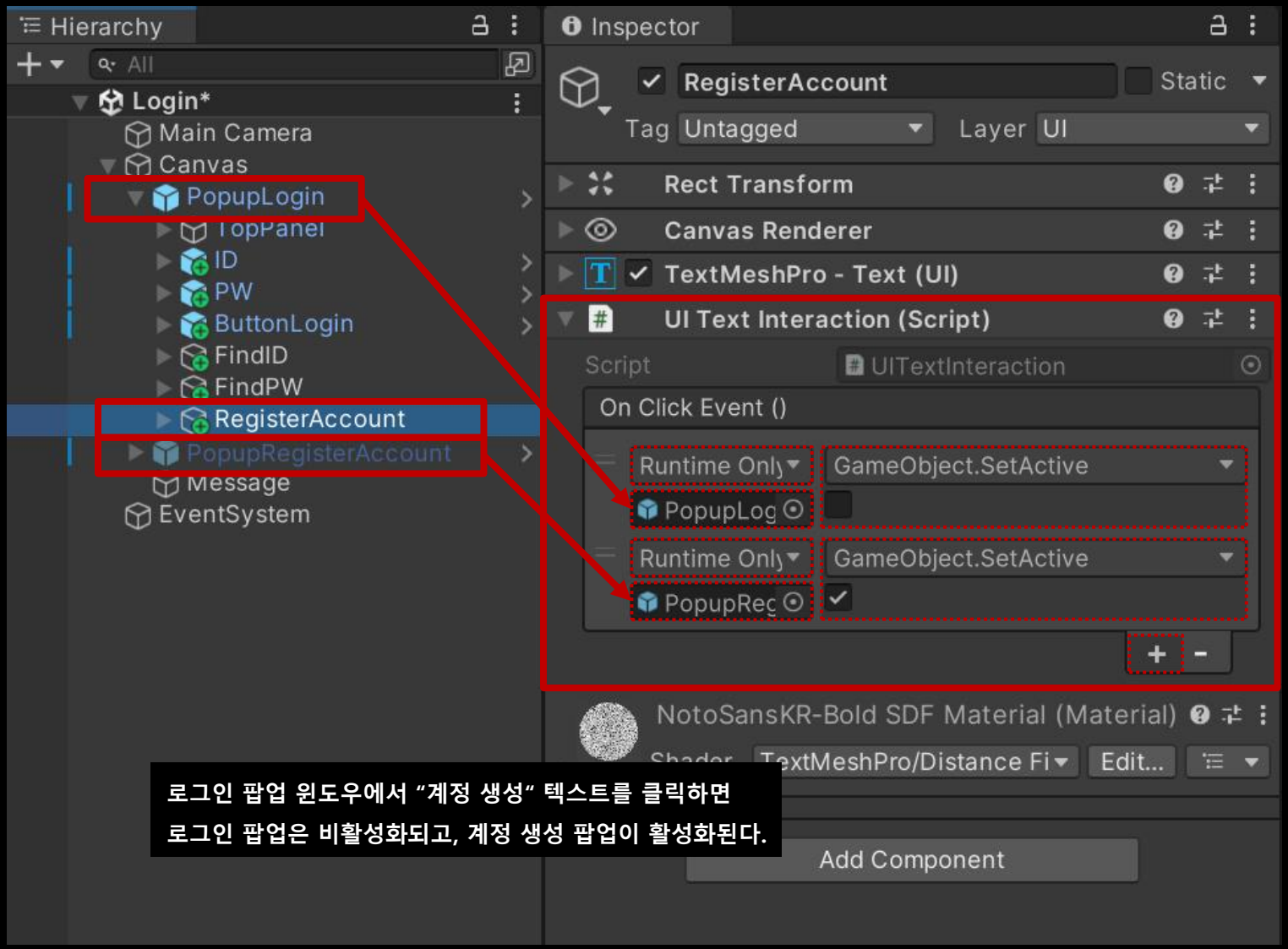
게임 유저 관리

- Text UI 상호작용을 제어하는 스크립트 생성 및 작성 (계속)

```
18      private void Awake()
19      {
20          text = GetComponent<TextMeshProUGUI>();
21      }
22
23      public void OnPointerEnter(PointerEventData eventData)
24      {
25          text.fontStyle = FontStyles.Bold;
26      }
27
28      public void OnPointerExit(PointerEventData eventData)
29      {
30          text.fontStyle = FontStyles.Normal;
31      }
32
33      public void OnPointerClick(PointerEventData eventData)
34      {
35          onClickEvent?.Invoke();
36      }
37  }
```

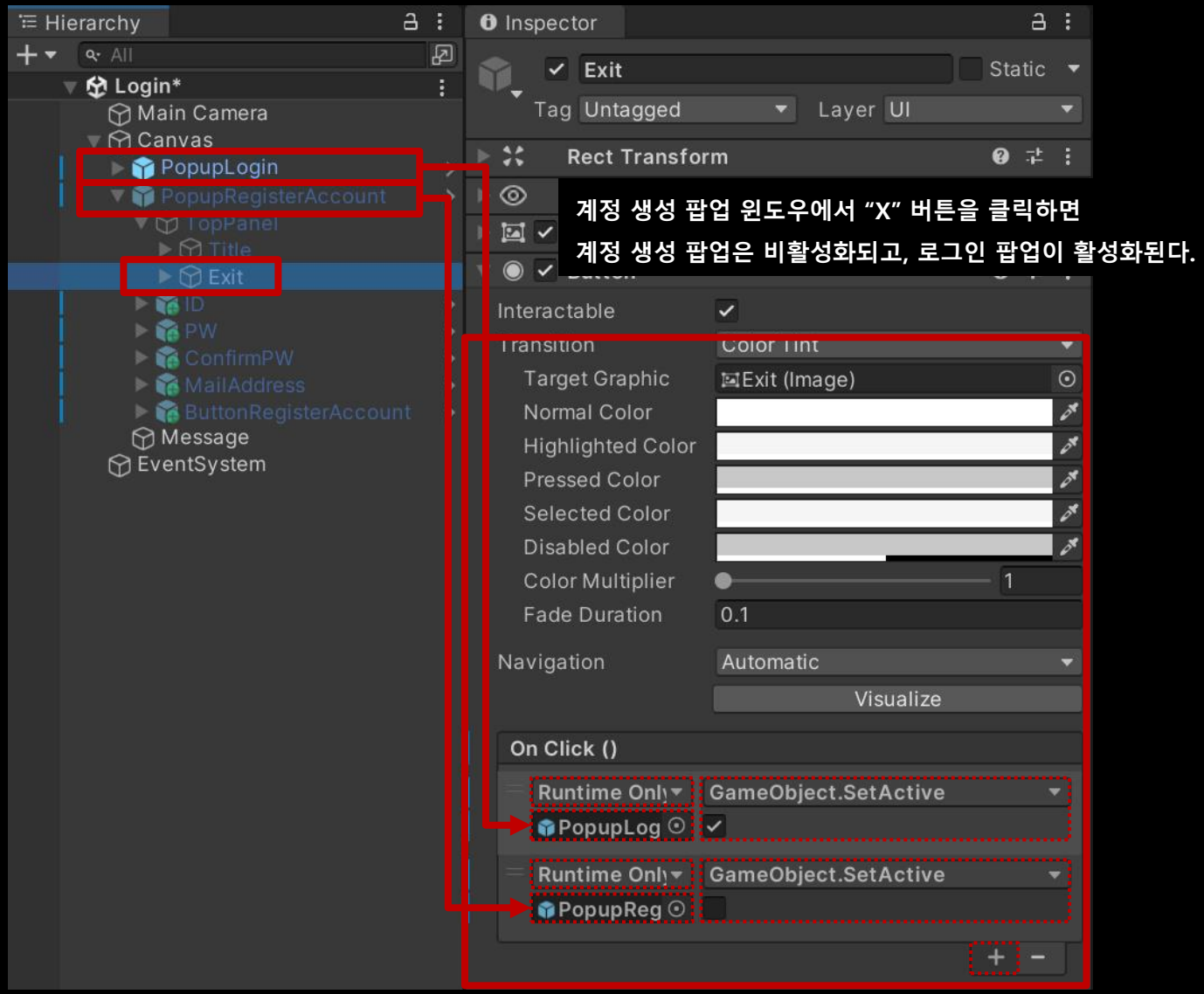
게임 유저 관리

- RegisterAccount 오브젝트에 "UITextInteraction" 컴포넌트 추가 및 설정



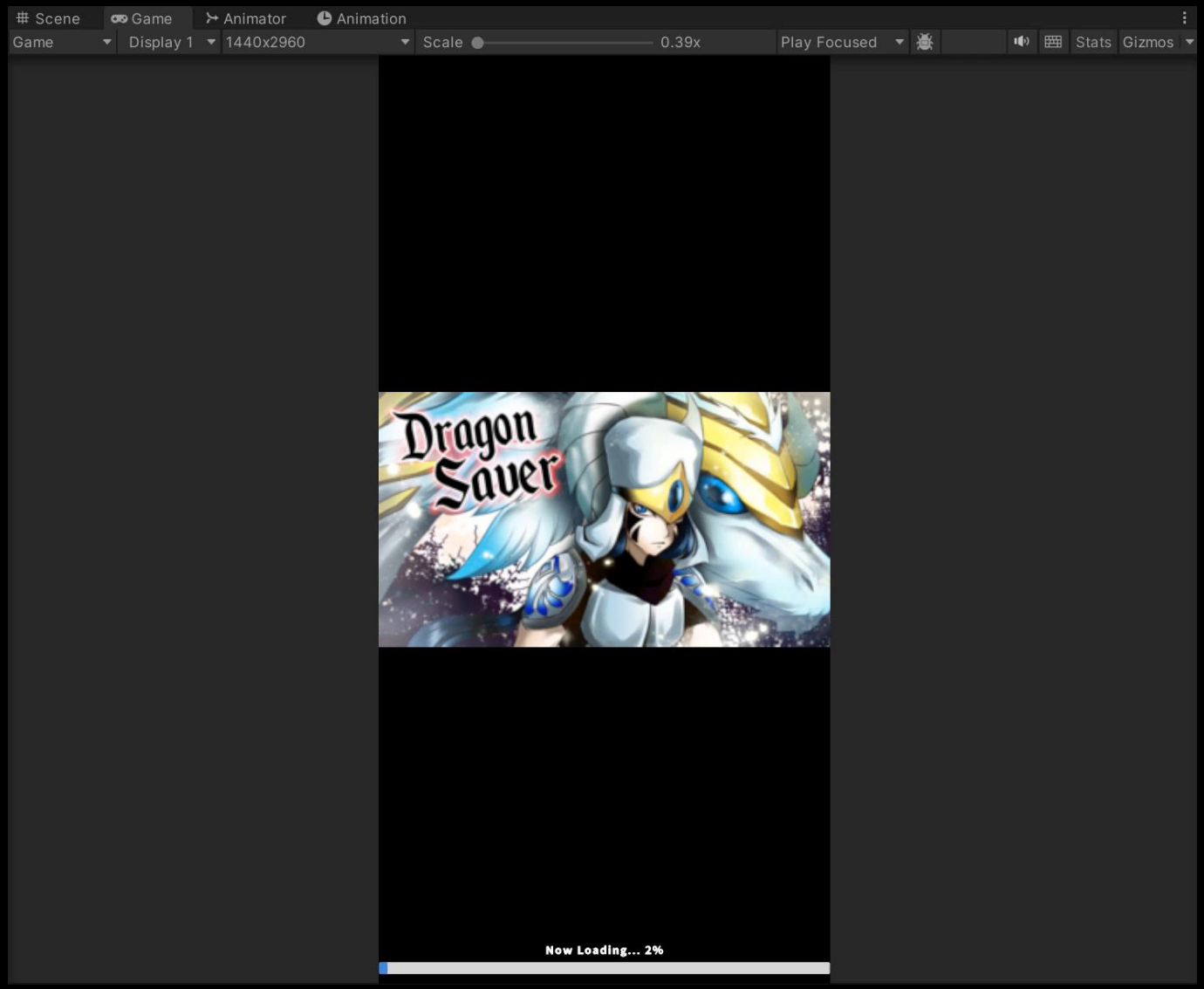
게임 유저 관리

- Exit 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



게임 유저 관리

- 결과 화면

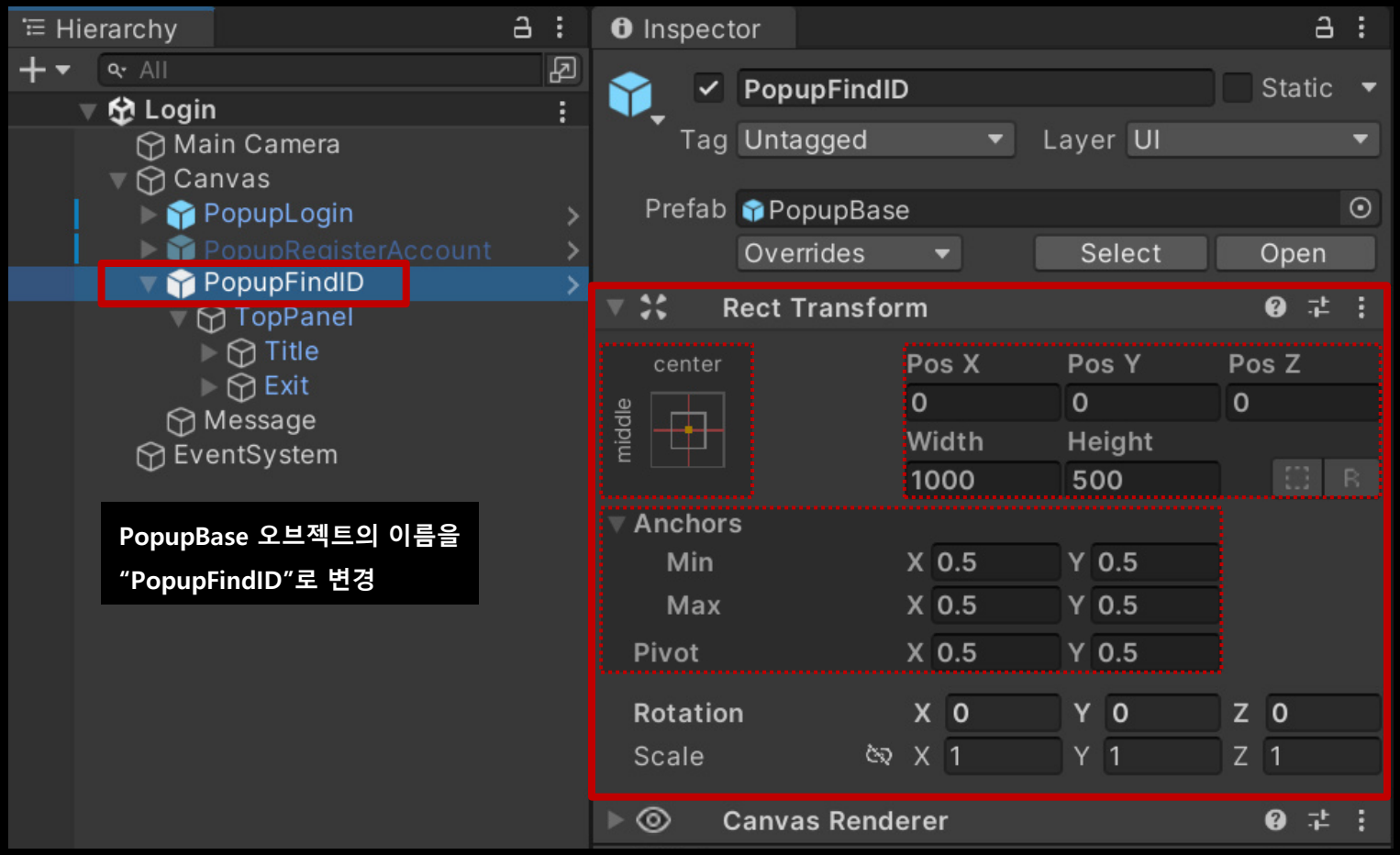

Scene
Game
Animator
Animation
Game
Display 1
1440x2960
Scale
0.39x
Play Focused
Stats
Gizmos
Dragon Saver
Now Loading... 2%

게임 유저 관리

아이디 찾기

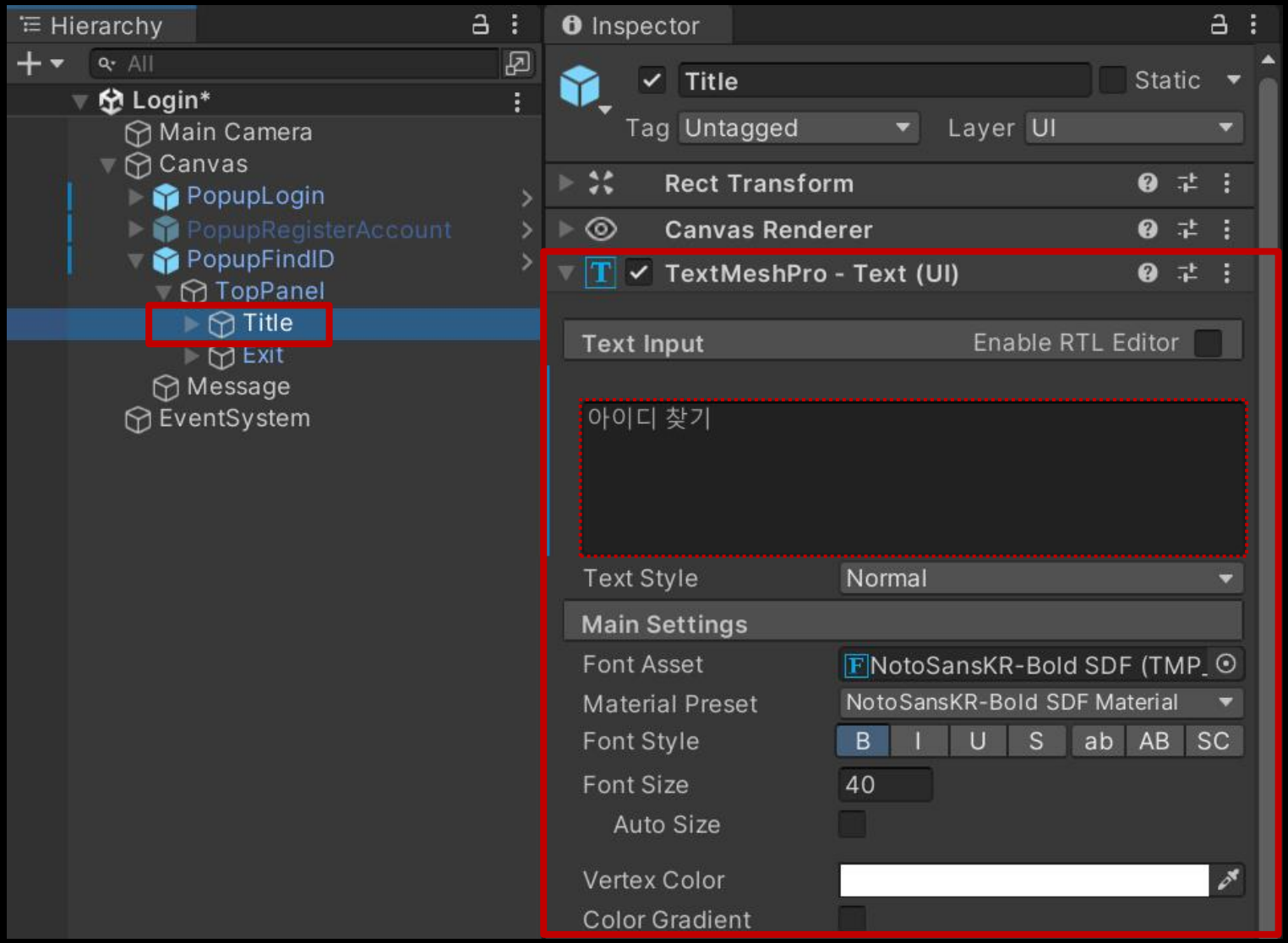
아이디 찾기 팝업 윈도우 생성 및 설정

- PopupBase 프리팹을 Hierarchy View로 Drag & Drop



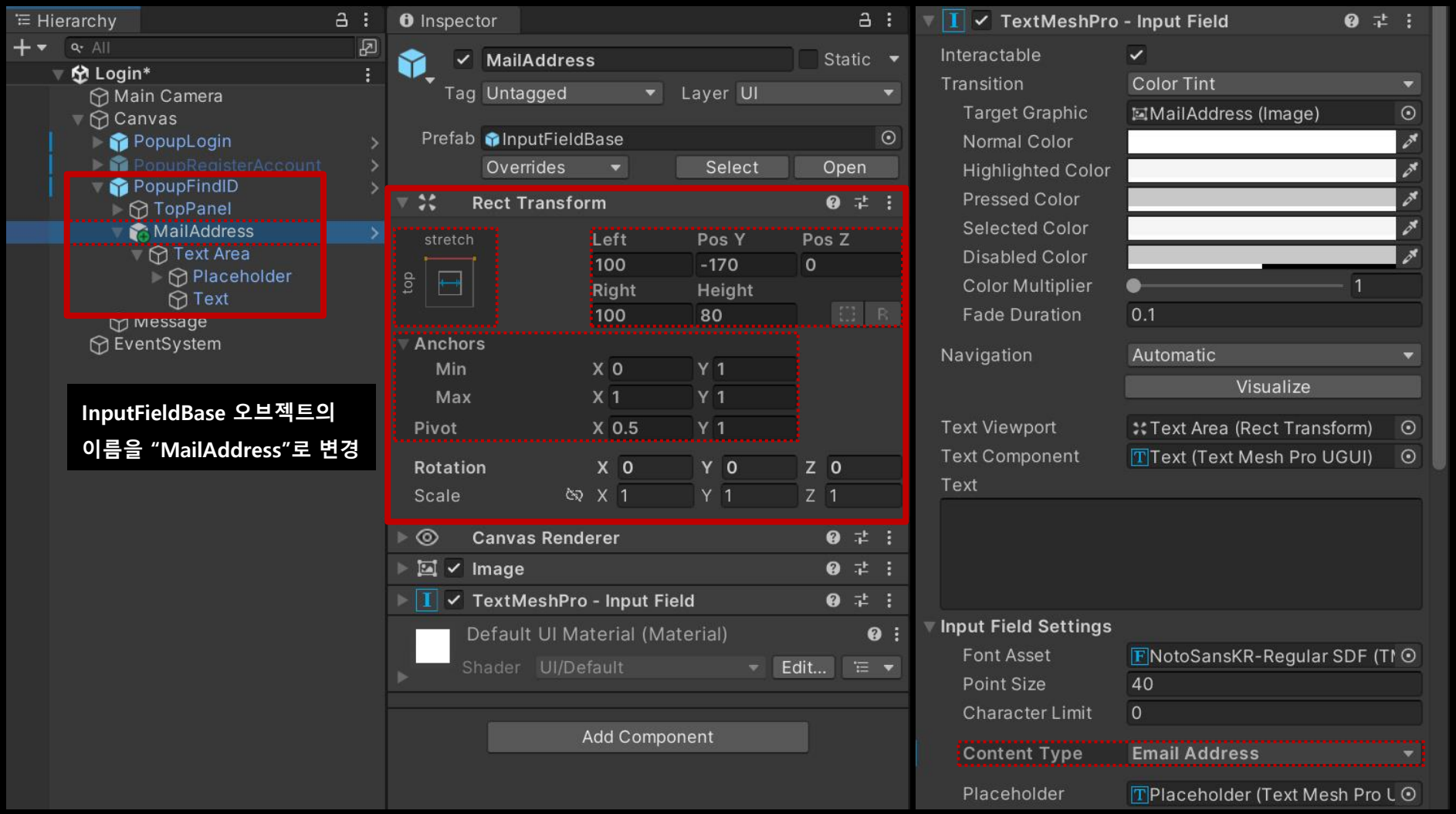
게임 유저 관리

- 아이디 찾기 팝업 윈도우 상단 이름 설정



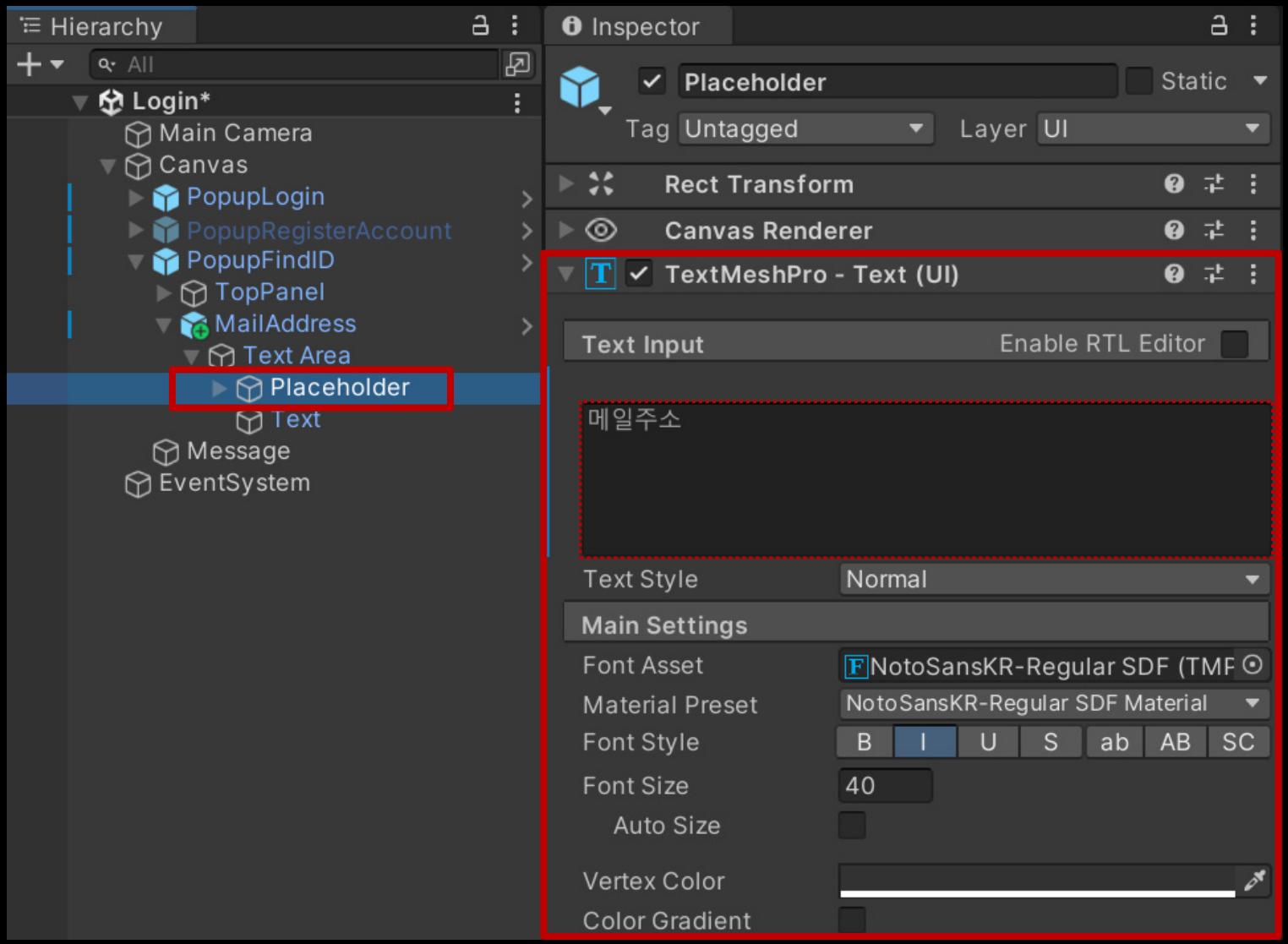
게임 유저 관리

- 메일 주소 입력 필드 생성 및 설정
  - InputFieldBase 프리팹을 Hierarchy View로 Drag & Drop



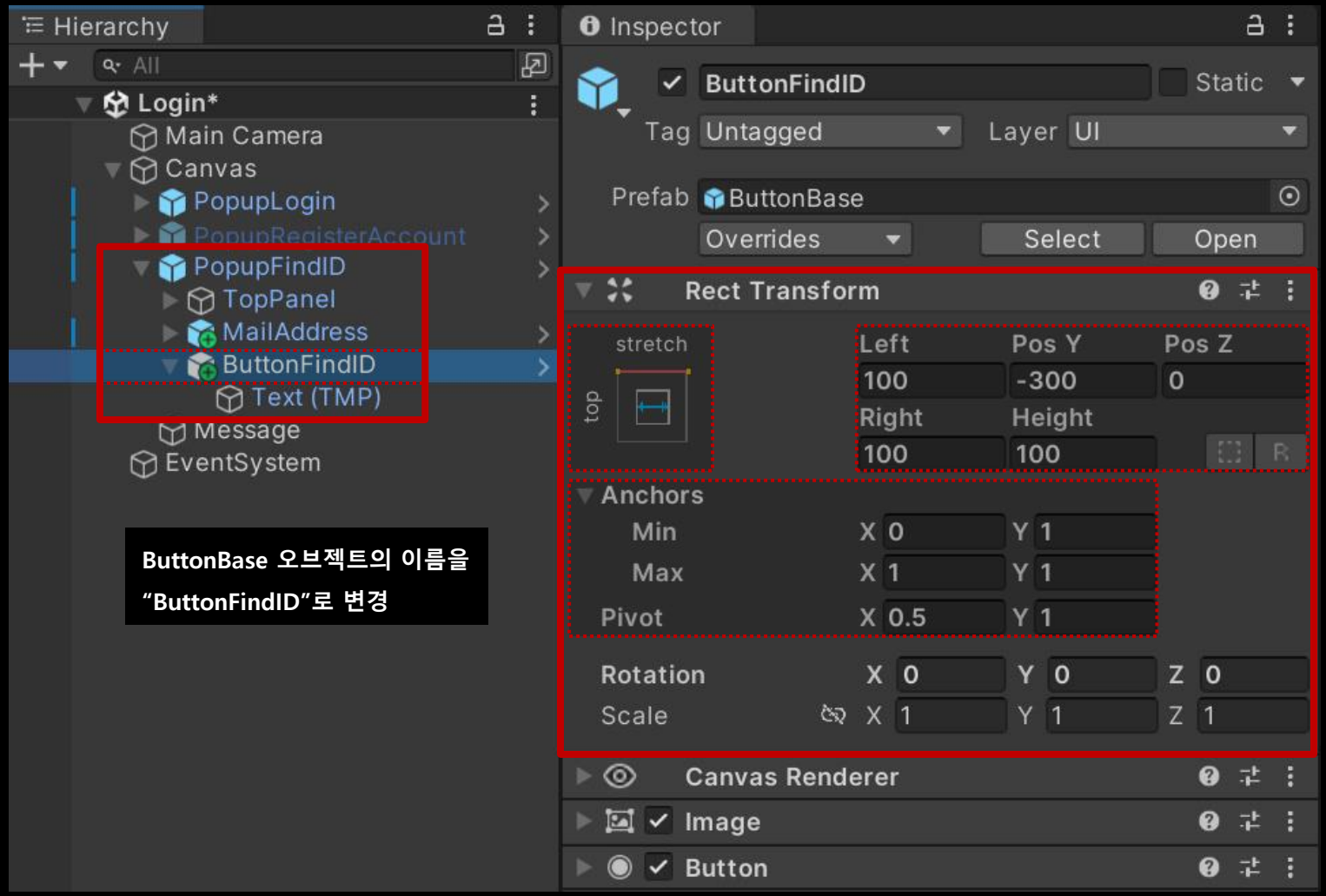
게임 유저 관리

- 메일 주소 입력 필드 생성 및 설정 (계속)



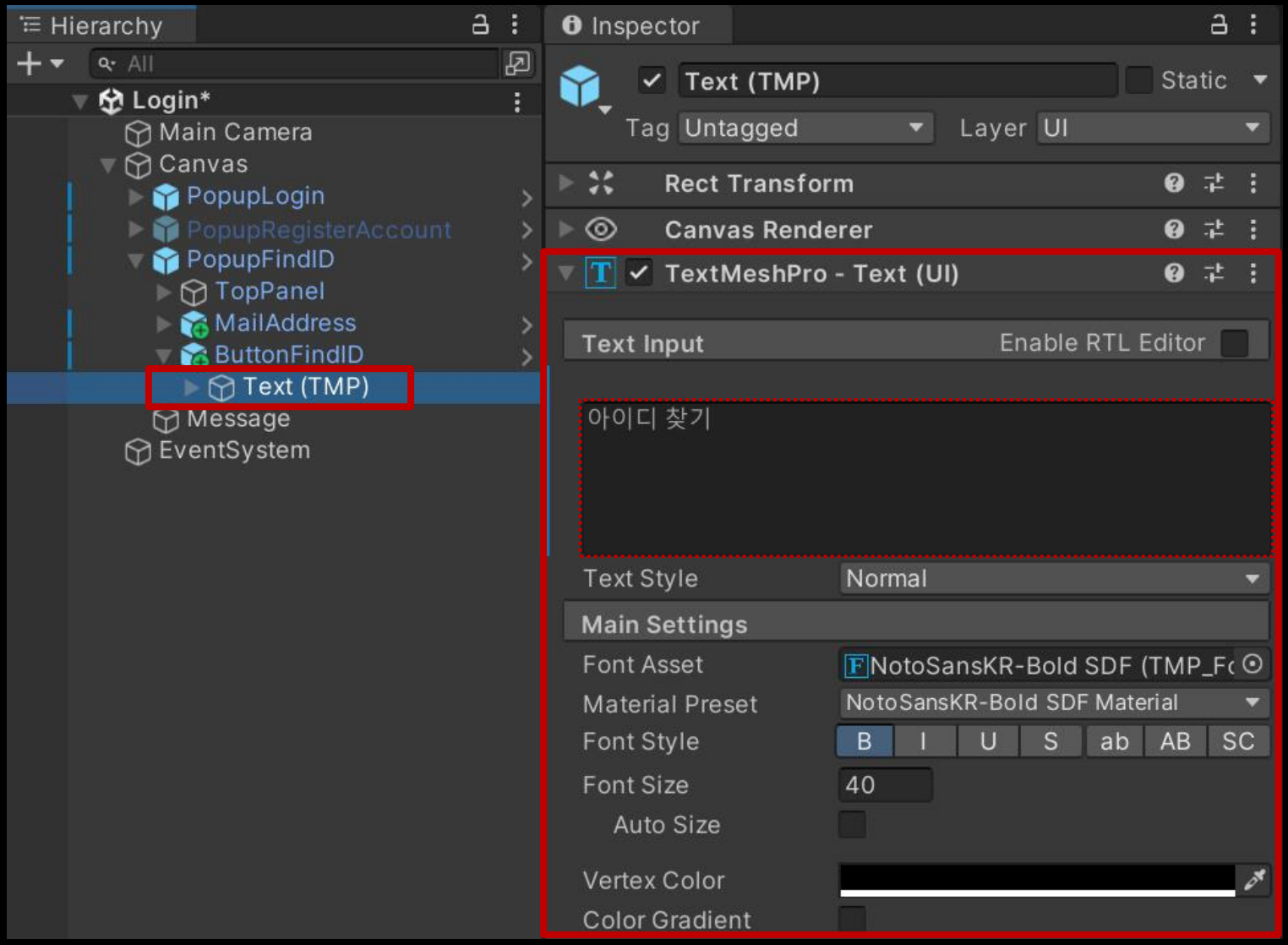
게임 유저 관리

- "아이디 찾기" 버튼 생성 및 설정
  - ButtonBase 프리팹을 Hierarchy View로 Drag & Drop



게임 유저 관리

- "아이디 찾기" 버튼 생성 및 설정 (계속)



게임 유저 관리

- 뒤끝 서버와 연동해 아이디 찾기를 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "FindID"로 변경

```
1   using UnityEngine;
2   using UnityEngine.UI;
3   using TMPro;
4   using BackEnd;
5
6   public class FindID : LoginBase
7   {
8       [SerializeField]
9       private Image              imageEmail;         // E-mail 필드 색상 변경
10      [SerializeField]
11      private TMP_InputField     inputFieldEmail;    // E-mail 필드 텍스트 정보 추출
12
13      [SerializeField]
14      private Button             btnFindID;          // "아이디 찾기" 버튼 (상호작용 가능/불가능)
15
```

게임 유저 관리

- 뒤끝 서버와 연동해 아이디 찾기를 제어하는 스크립트 생성 및 작성 (계속)

```
16  public void OnClickFindID()
17  {
18      // 매개변수로 입력한 InputField UI의 색상과 Message 내용 초기화
19      ResetUI(imageEmail);
20
21      // 필드 값이 비어있는지 체크
22      if ( IsFieldDataEmpty(imageEmail, inputFieldEmail.text, "메일 주소") )   return;
23
24      // 메일 형식 검사
25      if ( !inputFieldEmail.text.Contains("@") )
26      {
27          GuideForIncorrectlyEnteredData(imageEmail, "메일 형식이 잘못되었습니다.(ex. address@xx.xx)");
28          return;
29      }
30
31      // "아이디 찾기" 버튼의 상호작용 비활성화
32      btnFindID.interactable = false;
33      SetMessage("메일 발송중입니다.");
34
35      // 뒤끝 서버 아이디 찾기 시도
36      FindCustomID();
37  }
38
```

게임 유저 관리

- 뒤끝 서버와 연동해 아이디 찾기를 제어하는 스크립트 생성 및 작성 (계속)

```
39      /// <summary>
40      /// 아이디 찾기를 위해 이메일 발송 시도 후 서버로부터 전달받은 message를 기반으로 로직 처리
41      /// </summary>
42      private void FindCustomID()
43      {
44          // 아이디 정보를 이메일로 발송
45          Backend.BMember.FindCustomID(inputFieldEmail.text, callback =>
46          {
47              // "아이디 찾기" 버튼 상호작용 활성화
48              btnFindID.interactable = true;
49
50              // 메일 발송 성공
51              if ( callback.IsSuccess() )
52              {
53                  SetMessage($"{inputFieldEmail.text} 주소로 메일을 발송하였습니다.");
54              }
55              // 메일 발송 실패
56              else...
83          });
84      }
85  }
```

뒷장

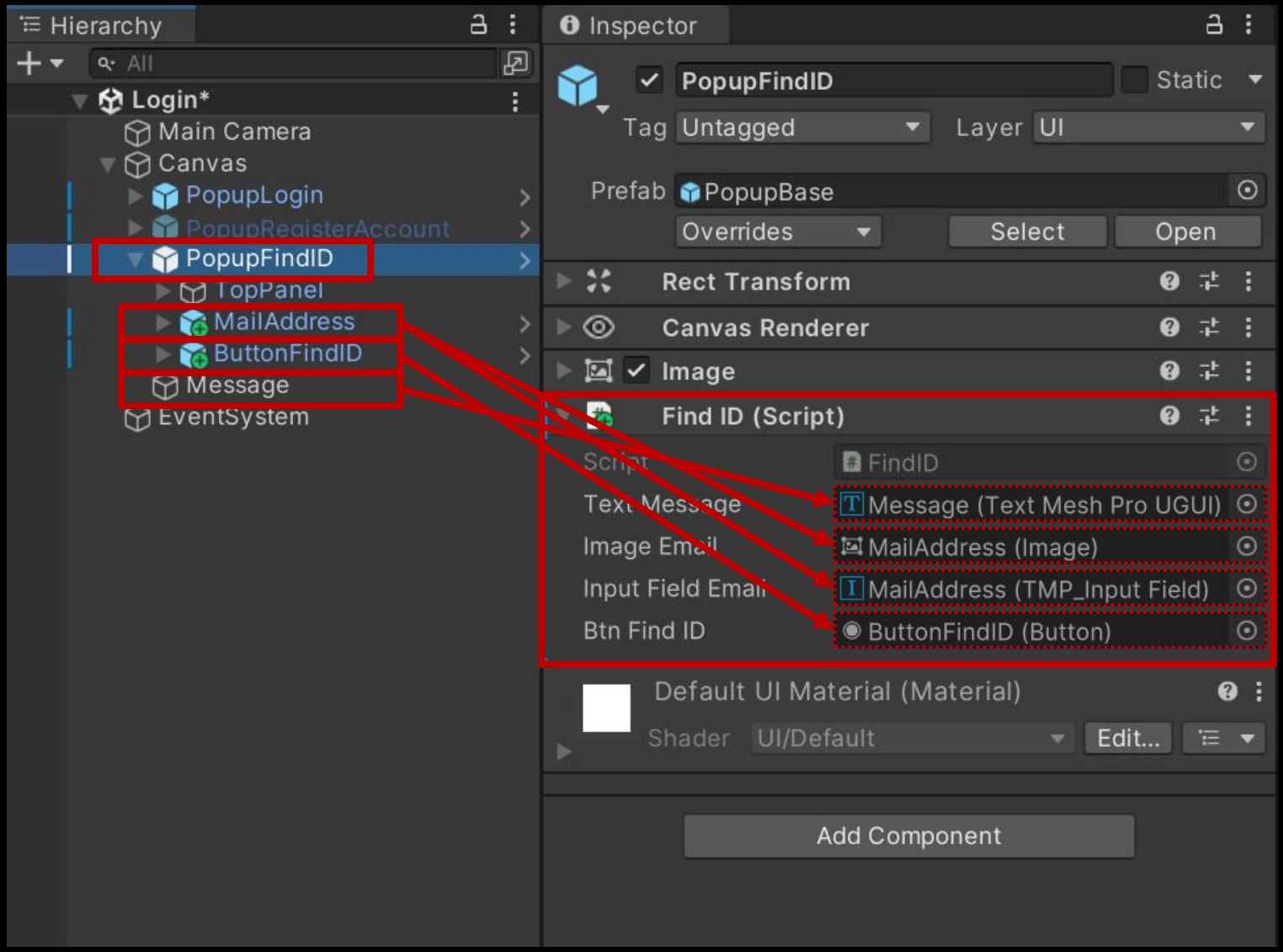
게임 유저 관리

- 뒤끝 서버와 연동해 아이디 찾기를 제어하는 스크립트 생성 및 작성 (계속)

```
55          // 메일 발송 실패
56          else
57          {
58              string message = string.Empty;
59
60              switch ( int.Parse(callback.GetStatusCode()) )
61              {
62                  case 404:   // 해당 이메일의 게이머가 없는 경우
63                      message = "해당 이메일을 사용하는 사용자가 없습니다.";
64                      break;
65                  case 429:   // 24시간 이내에 5회 이상 같은 이메일 정보로 아이디/비밀번호 찾기를 시도한 경우
66                      message = "24시간 이내에 5회 이상 아이디/비밀번호 찾기를 시도했습니다.";
67                      break;
68                  default:
69                      // statusCode : 400 => 프로젝트 명에 특수문자가 추가된 경우 (안내 메일 미발송 및 에러 발생)
70                      message = callback.GetMessage();
71                      break;
72              }
73
74              if ( message.Contains("이메일") )
75              {
76                  GuideForIncorrectlyEnteredData(imageEmail, message);
77              }
78              else
79              {
80                  SetMessage(message);
81              }
82          }
```

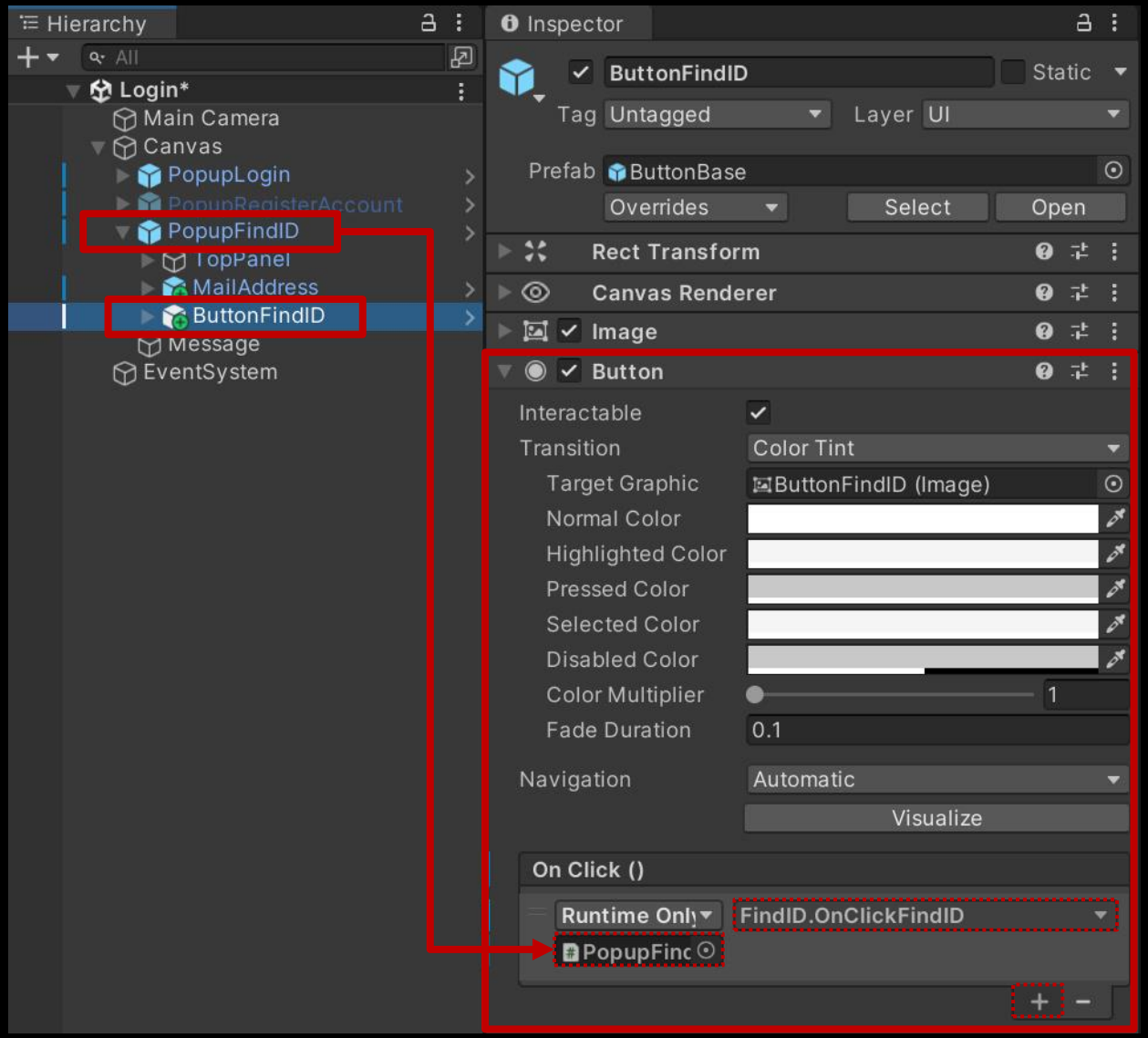
게임 유저 관리

- PopupFindID 오브젝트에 "FindID" 컴포넌트 추가 및 설정



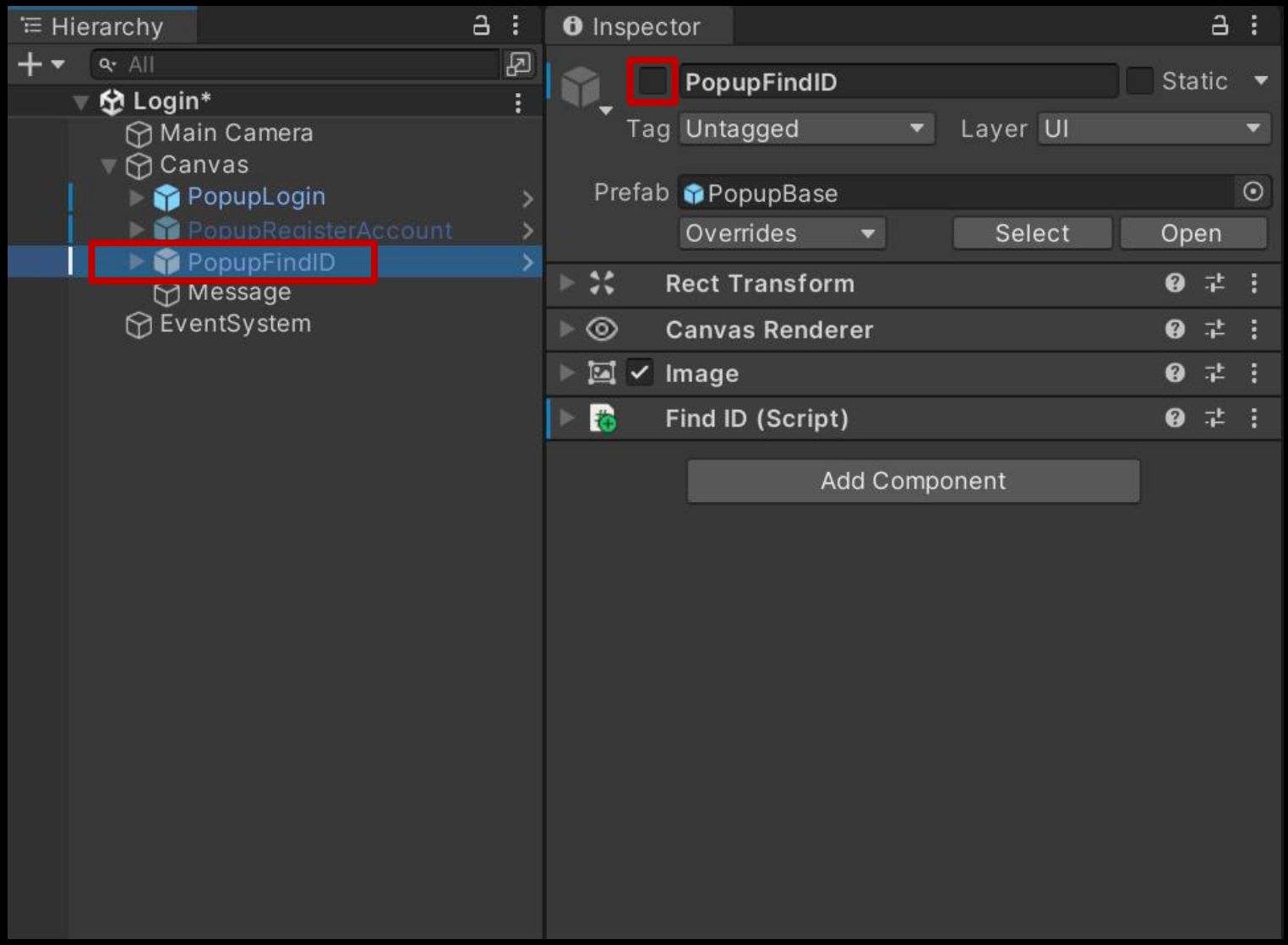
게임 유저 관리

- ButtonFindID 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



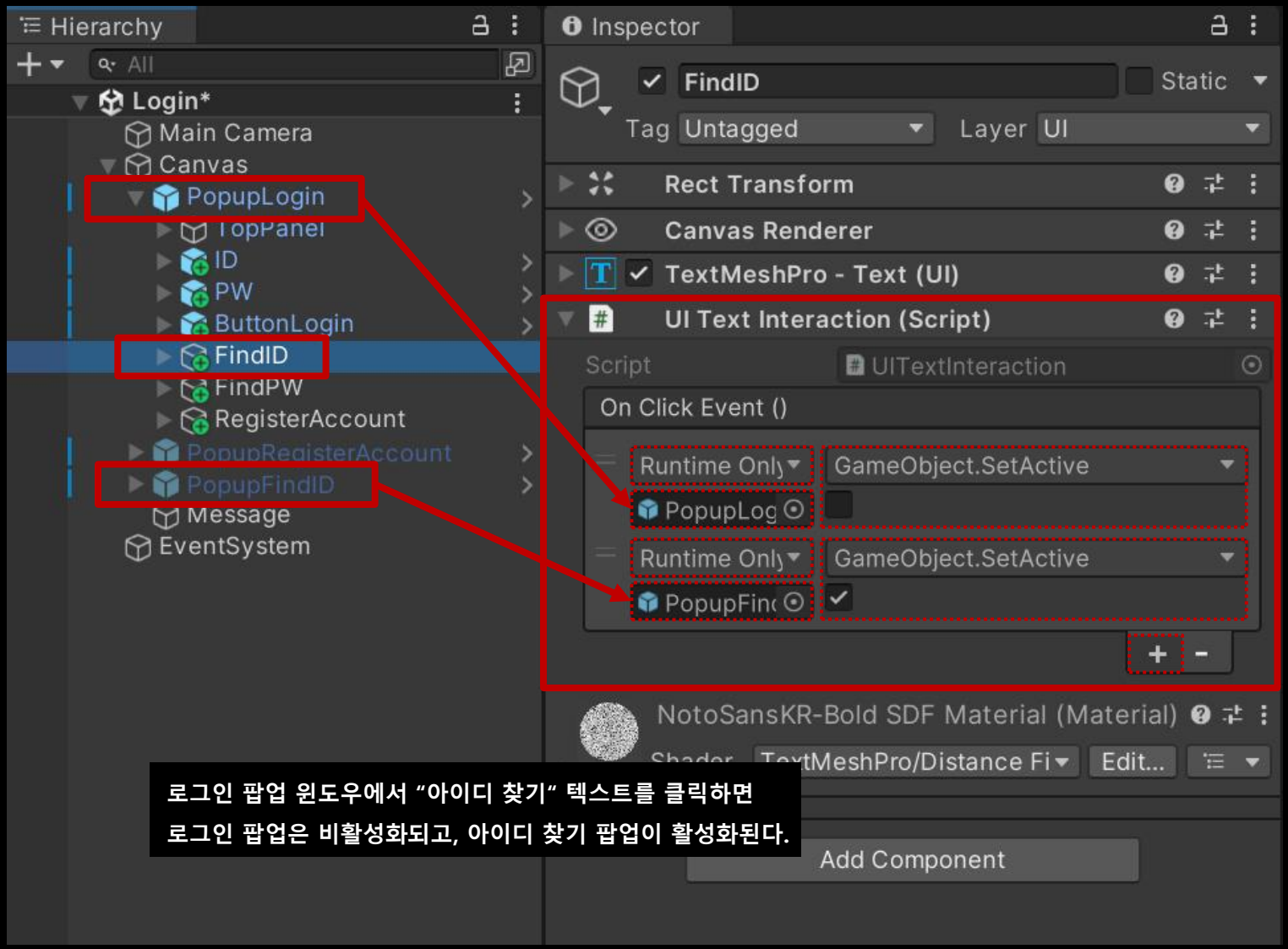
게임 유저 관리

- PopupFindID 오브젝트 비활성화



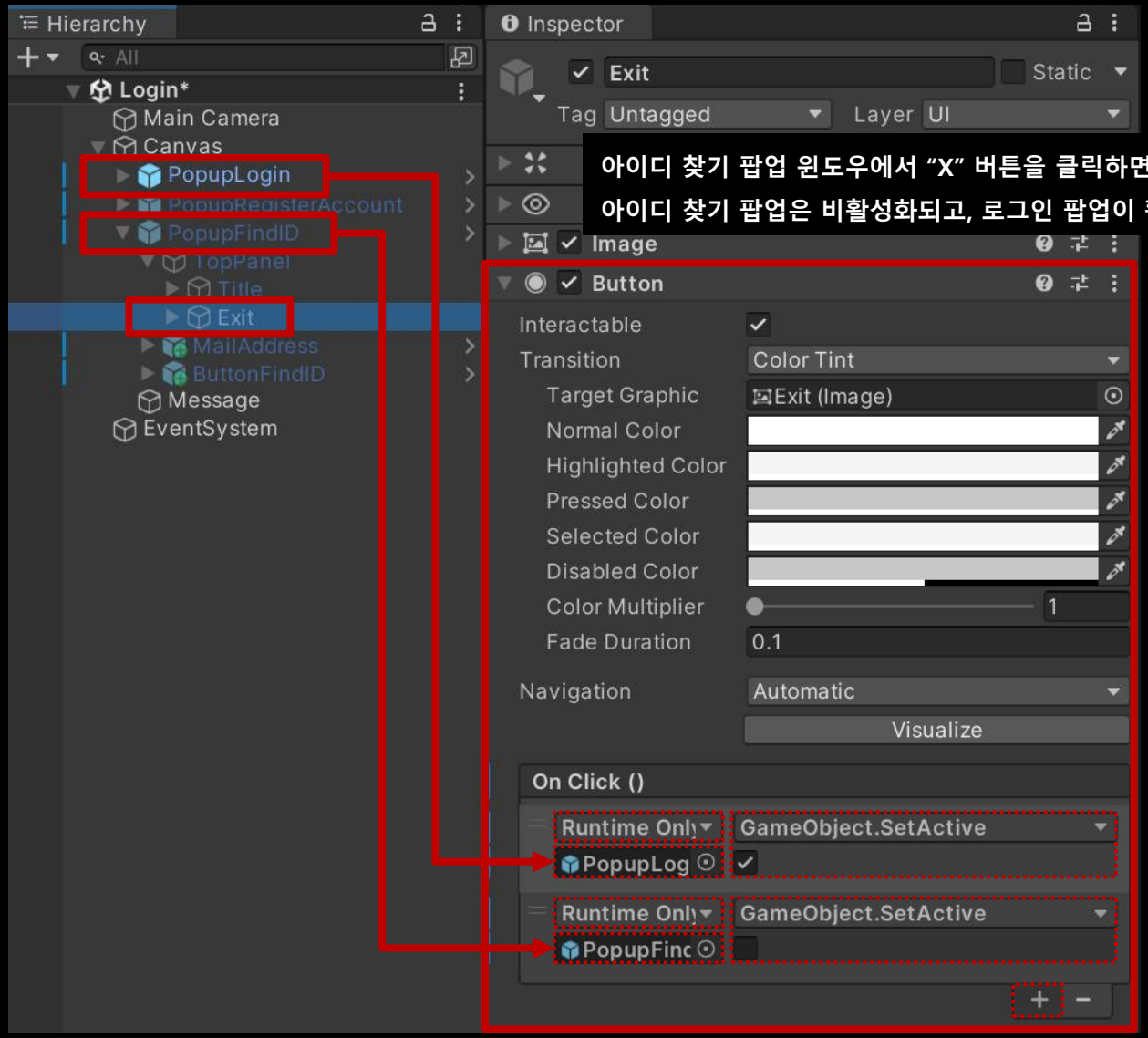
게임 유저 관리

- FindID 오브젝트에 "UITextInteraction" 컴포넌트 추가 및 설정



게임 유저 관리

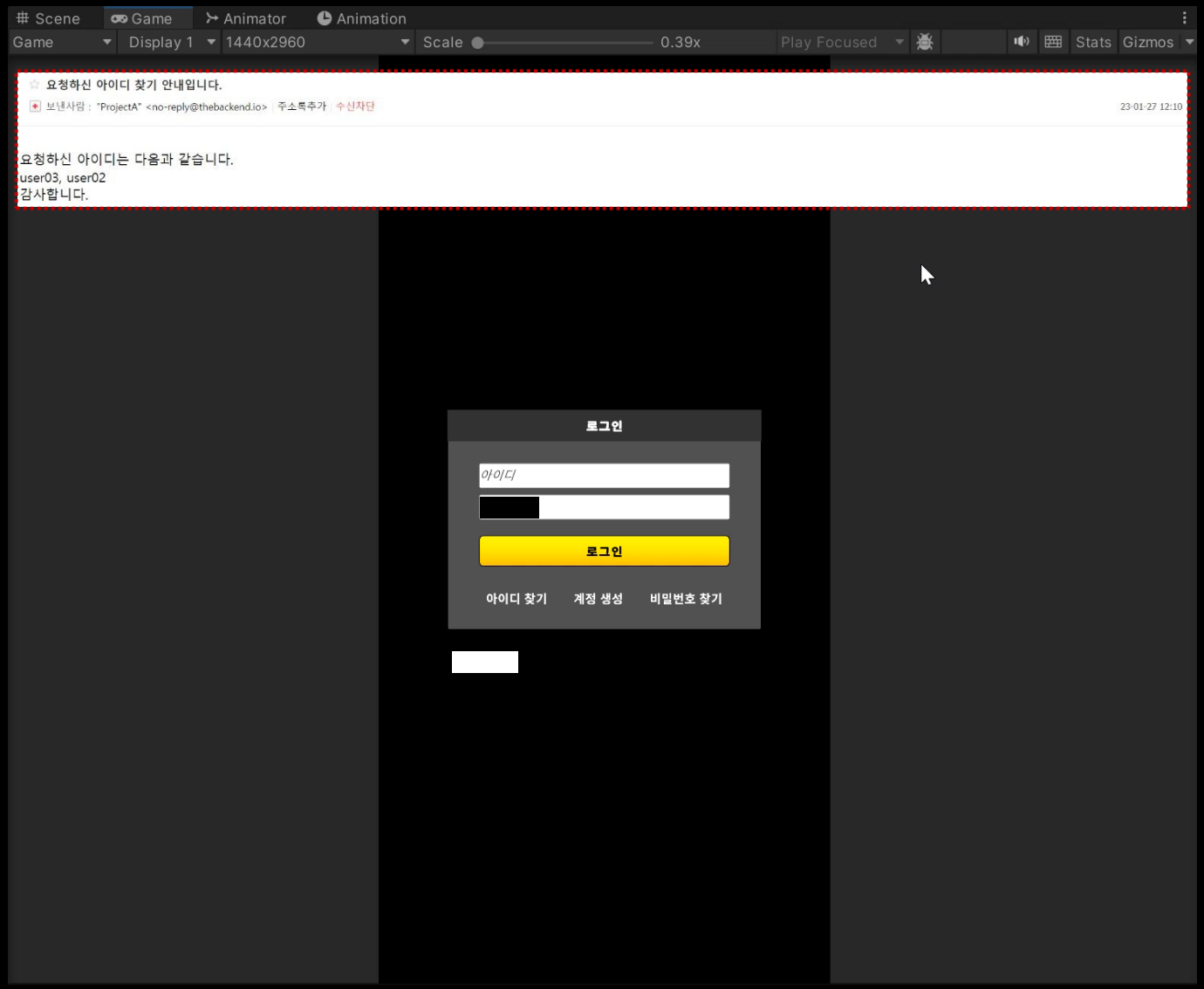
- Exit 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



아이디 찾기 팝업 윈도우에서 "X" 버튼을 클릭하면
아이디 찾기 팝업은 비활성화되고, 로그인 팝업이 활성화된다.

게임 유저 관리

- 결과 화면

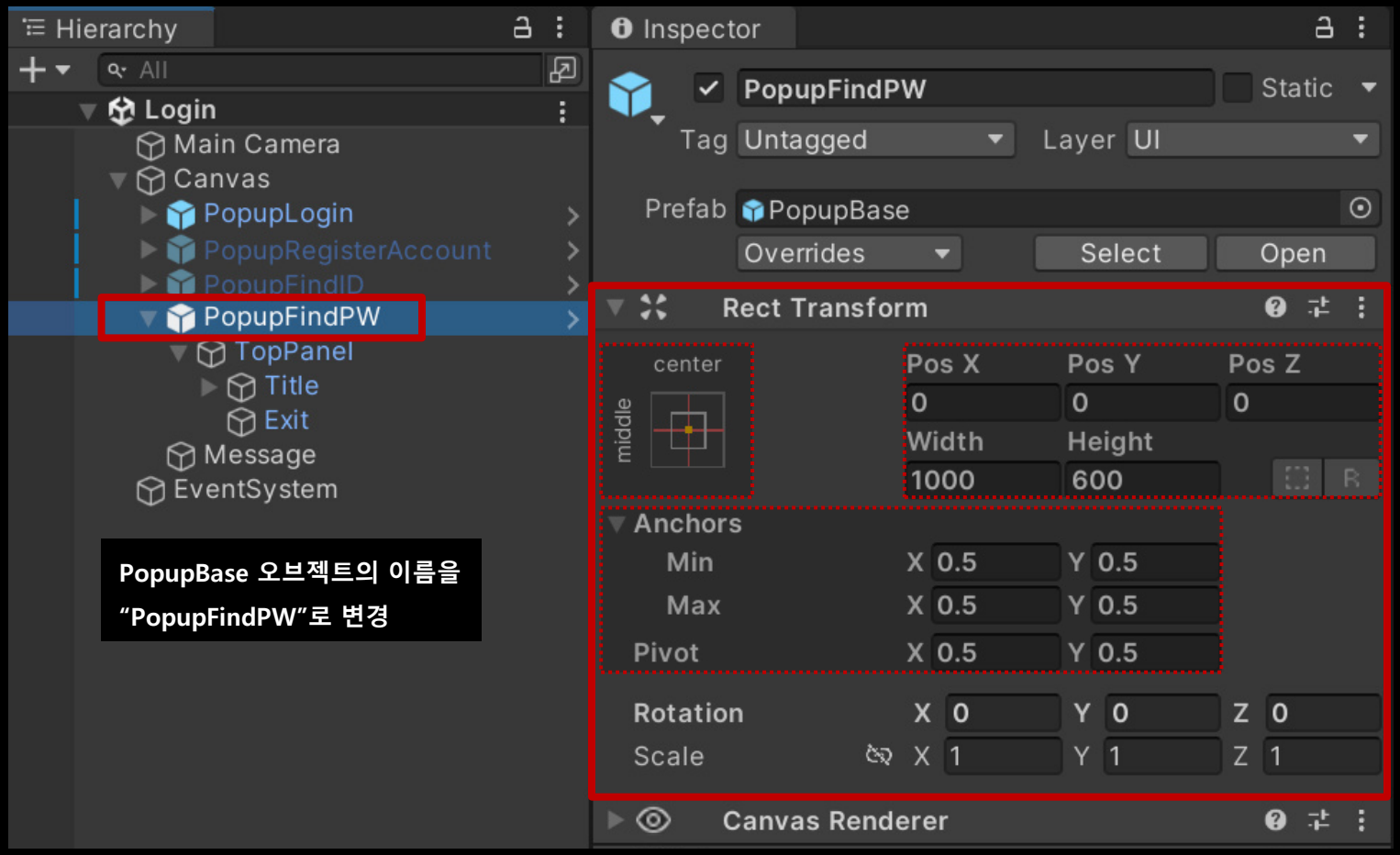

Scene
Game
Animator
Animation
Game
Display 1
1440x2960
Scale
0.39x
Play Focused
Stats
Gizmos
요청하신 아이디 찾기 안내입니다.
보낸사람 : "ProjectA" <no-reply@thebackend.io>
주소록추가
수신차단
23-01-27 12:10
요청하신 아이디는 다음과 같습니다.
user03, user02
감사합니다.
로그인
아이디
로그인
아이디 찾기
계정 생성
비밀번호 찾기

게임 유저 관리

비밀번호 찾기

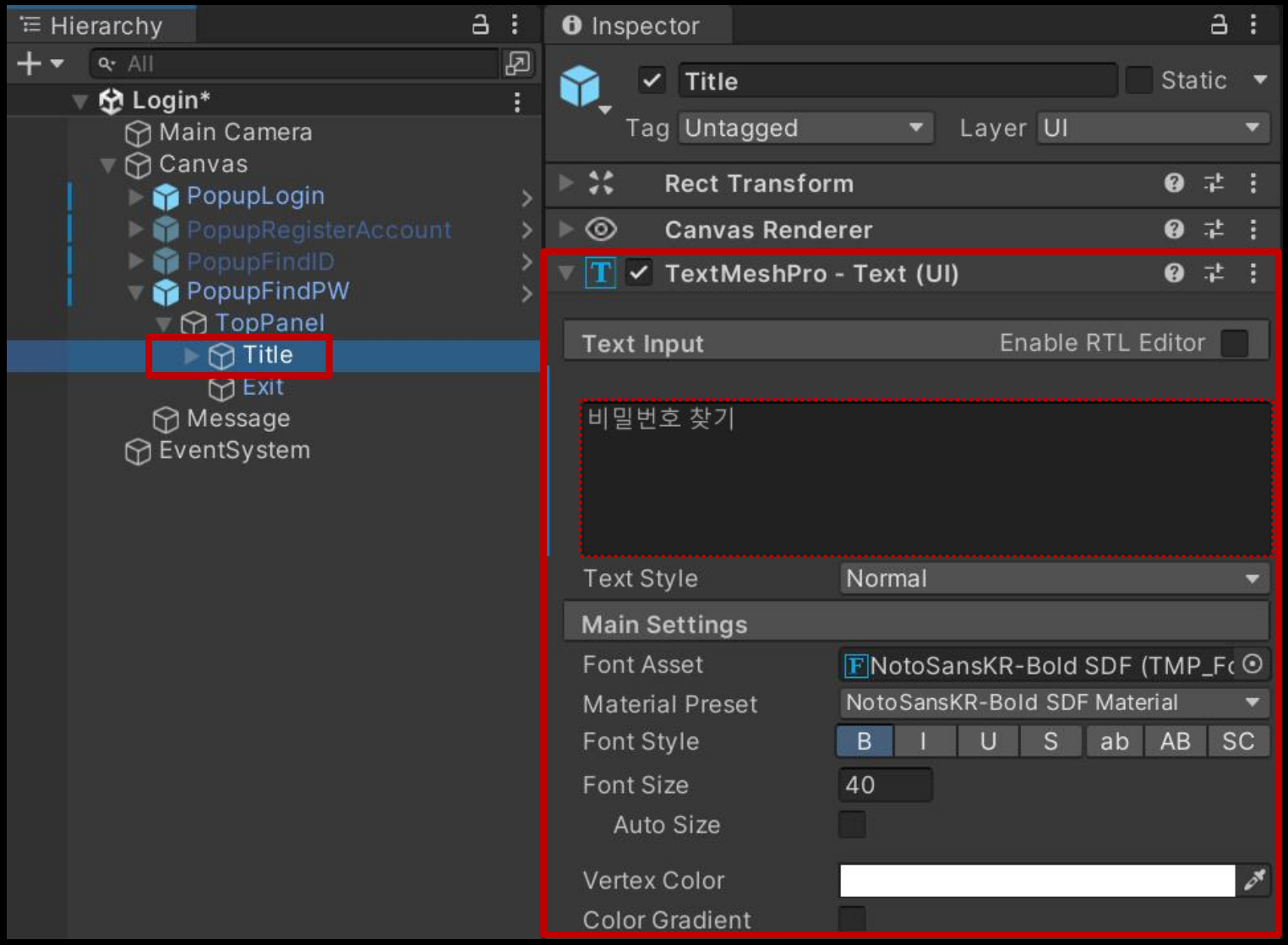
비밀번호 찾기 팝업 윈도우 생성 및 설정

- PopupBase 프리팹을 Hierarchy View로 Drag & Drop



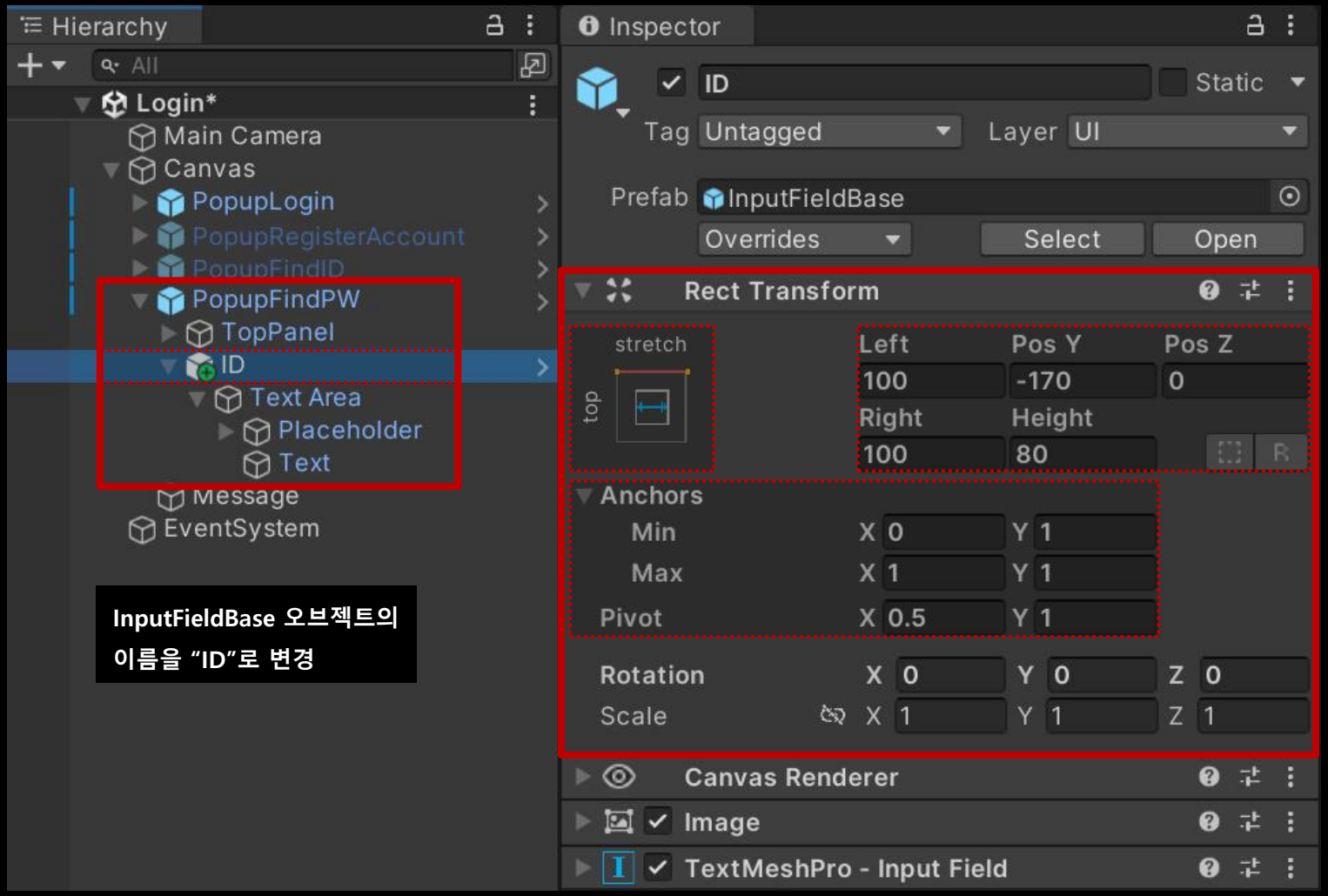
게임 유저 관리

- 비밀번호 찾기 팝업 윈도우 상단 이름 설정



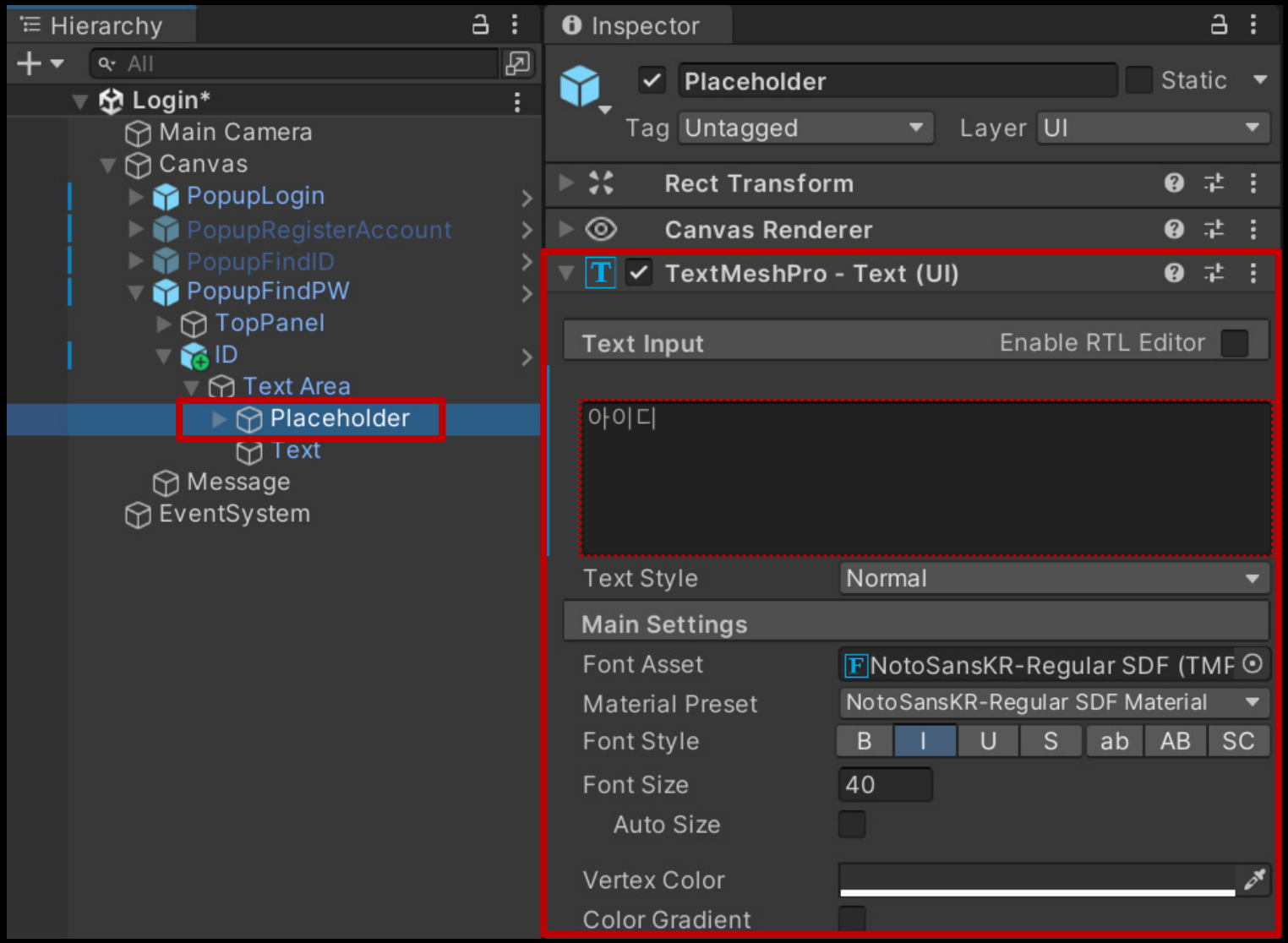
게임 유저 관리

- 아이디 입력 필드 생성 및 설정
  - InputFieldBase 프리팹을 Hierarchy View로 Drag & Drop



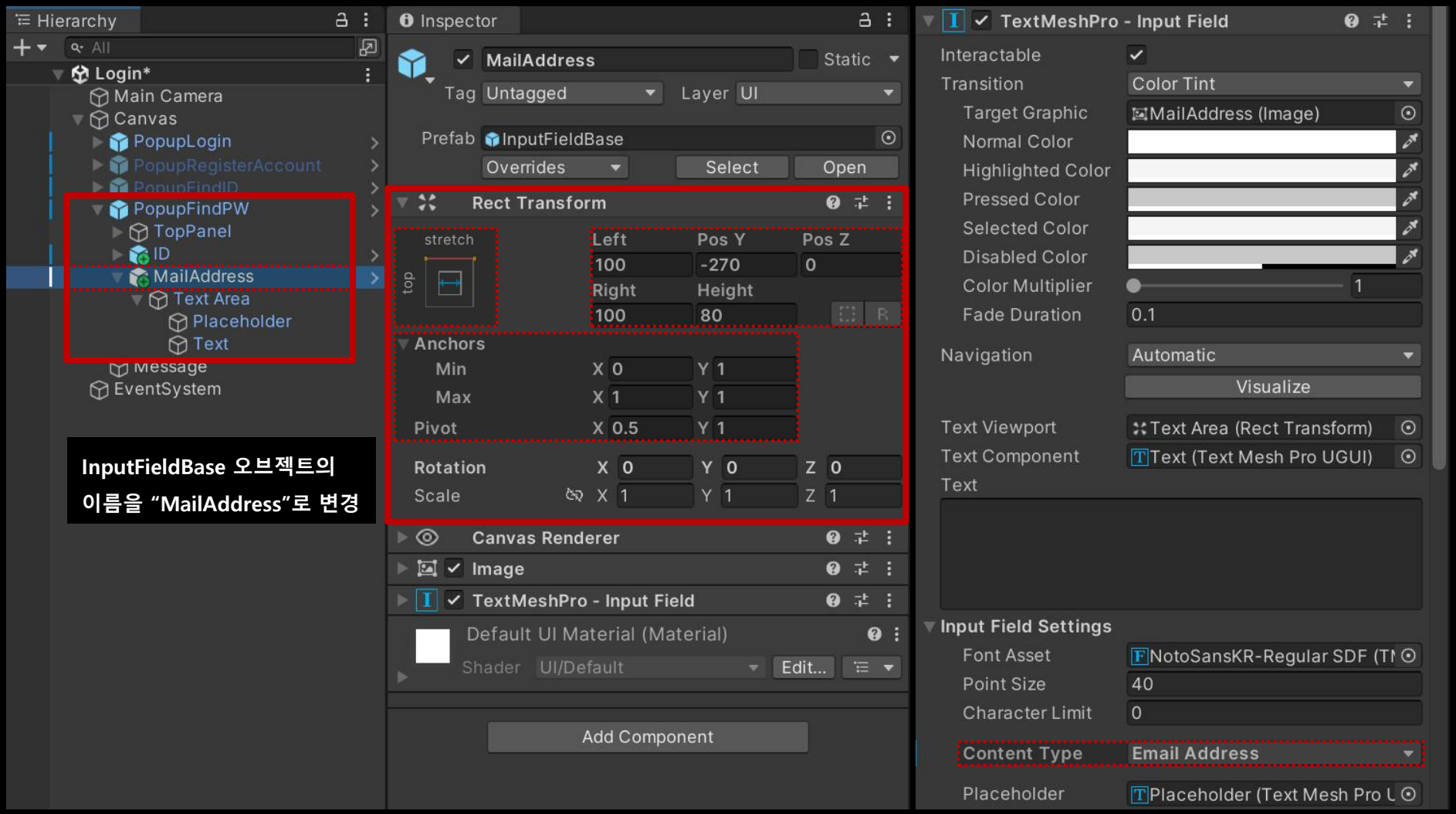
게임 유저 관리

- 아이디 입력 필드 생성 및 설정 (계속)



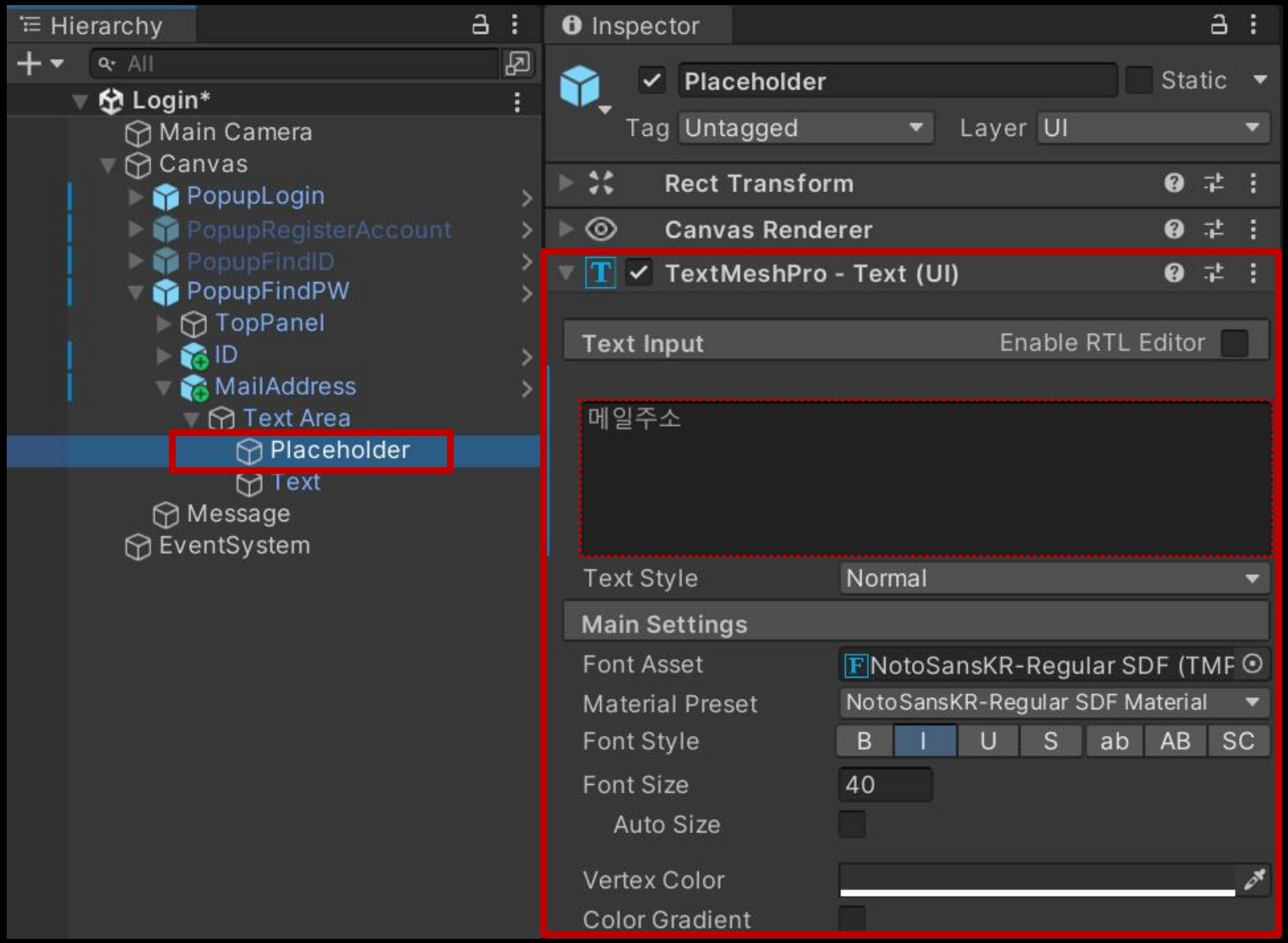
게임 유저 관리

- 메일 주소 입력 필드 생성 및 설정
  - InputFieldBase 프리팹을 Hierarchy View로 Drag & Drop



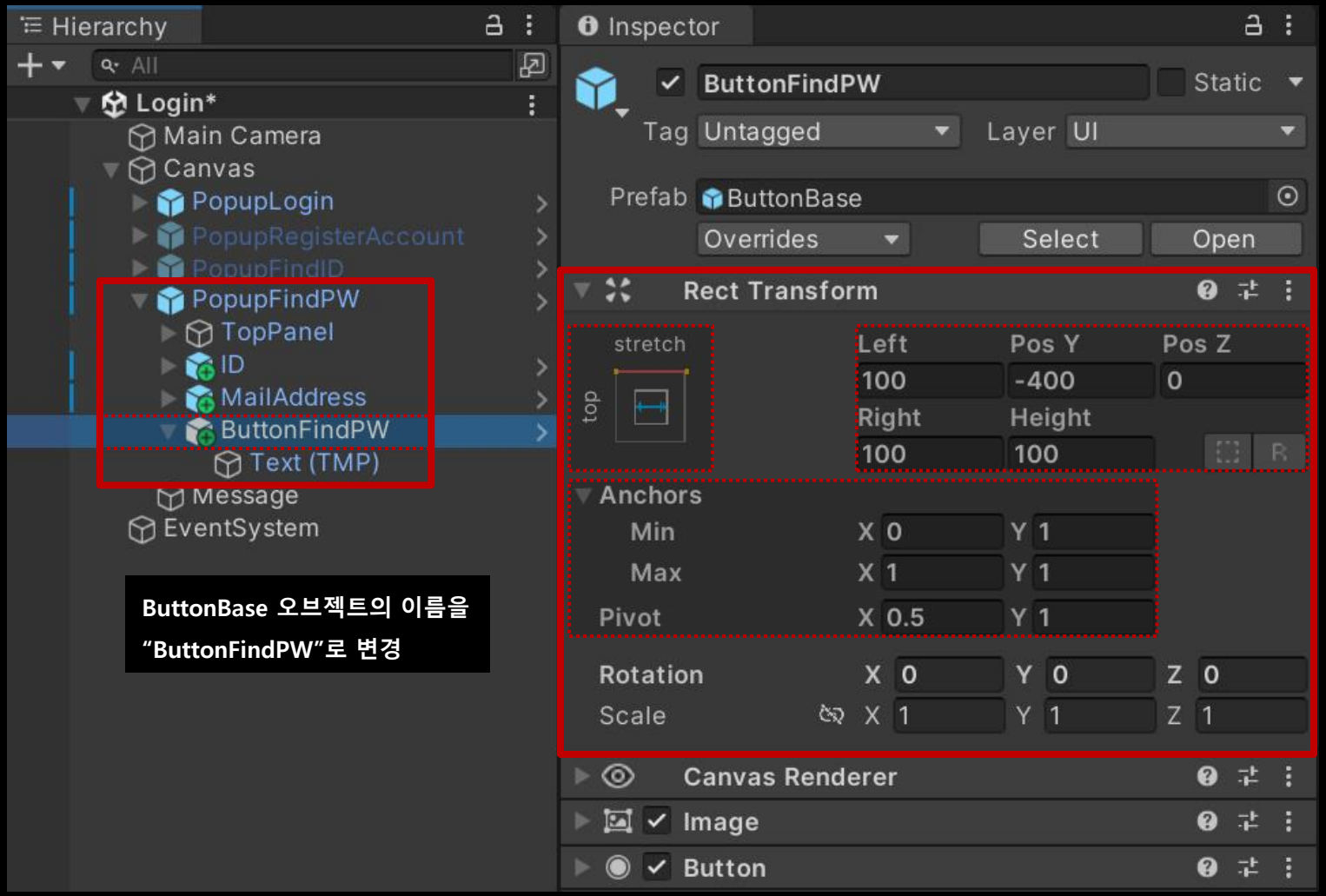
게임 유저 관리

- 메일 주소 입력 필드 생성 및 설정 (계속)



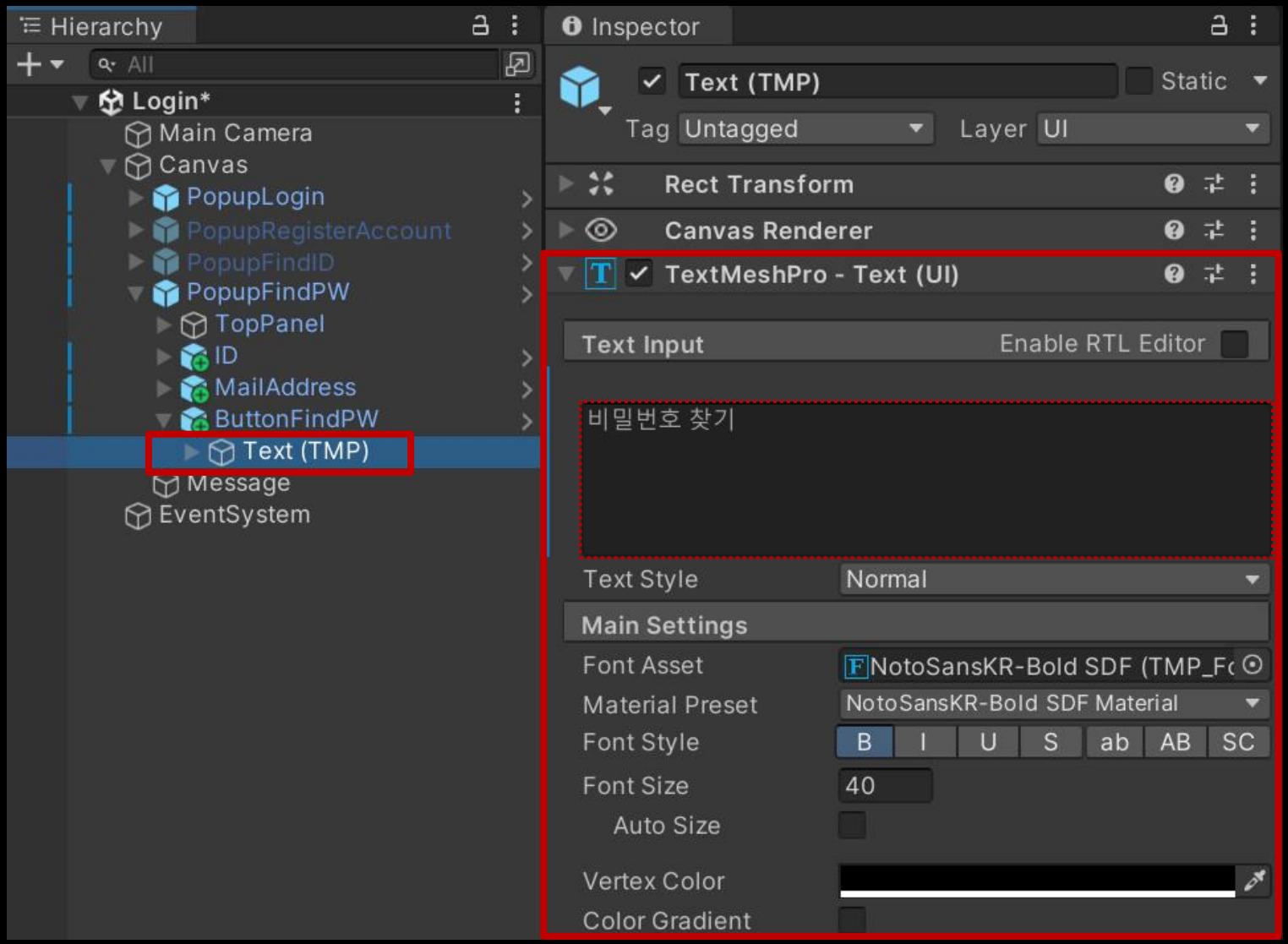
게임 유저 관리

- "비밀번호 찾기" 버튼 생성 및 설정
  - ButtonBase 프리팹을 Hierarchy View로 Drag & Drop



게임 유저 관리

- "비밀번호 찾기" 버튼 생성 및 설정 (계속)



게임 유저 관리

- 뒤끝 서버와 연동해 비밀번호 찾기를 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "FindPW"로 변경

```
1  using UnityEngine;
2  using UnityEngine.UI;
3  using TMPro;
4  using BackEnd;
5
6  public class FindPW : LoginBase
7  {
8      [SerializeField]
9      private Image              imageID;          // ID 필드 색상 변경
10     [SerializeField]
11     private TMP_InputField     inputFieldID;     // ID 필드 텍스트 정보 추출
12     [SerializeField]
13     private Image              imageEmail;       // E-mail 필드 색상 변경
14     [SerializeField]
15     private TMP_InputField     inputFieldEmail;  // E-mail 필드 텍스트 정보 추출
16
17     [SerializeField]
18     private Button             btnFindPW;        // "비밀번호 찾기" 버튼 (상호작용 가능/불가능)
19
```

게임 유저 관리

- 뒤끝 서버와 연동해 비밀번호 찾기를 제어하는 스크립트 생성 및 작성 (계속)

```
20  public void OnClickFindPW()
21  {
22      // 매개변수로 입력한 InputField UI의 색상과 Message 내용 초기화
23      ResetUI(imageID, imageEmail);
24
25      // 필드 값이 비어있는지 체크
26      if ( IsFieldDataEmpty(imageID, inputFieldID.text, "아이디") )           return;
27      if ( IsFieldDataEmpty(imageEmail, inputFieldEmail.text, "메일 주소") )   return;
28
29      // 메일 형식 검사
30      if ( !inputFieldEmail.text.Contains("@") )
31      {
32          GuideForIncorrectlyEnteredData(imageEmail, "메일 형식이 잘못되었습니다.(ex. address@xx.xx)");
33          return;
34      }
35
36      // "비밀번호 찾기" 버튼의 상호작용 비활성화
37      btnFindPW.interactable = false;
38      SetMessage("메일 발송중입니다.");
39
40      // 뒤끝 서버 비밀번호 찾기 시도
41      FindCustomPW();
42  }
43
```

게임 유저 관리

- 뒤끝 서버와 연동해 비밀번호 찾기를 제어하는 스크립트 생성 및 작성 (계속)

```
44      /// <summary>
45      /// 비밀번호를 리셋하기 위해 이메일 발송 시도 후 서버로부터 전달받은 message를 기반으로 로직 처리
46      /// </summary>
47      private void FindCustomPW()
48      {
49          // 리셋된 비밀번호 정보를 이메일로 발송
50          Backend.BMember.ResetPassword(inputFieldID.text, inputFieldEmail.text, callback =>
51          {
52              // "비밀번호 찾기" 버튼 상호작용 활성화
53              btnFindPW.interactable = true;
54
55              // 메일 발송 성공
56              if ( callback.IsSuccess() )
57              {
58                  SetMessage($"{inputFieldEmail.text} 주소로 메일을 발송하였습니다.");
59              }
60              // 메일 발송 실패
61              else...
88          });
89      }
90  }
```

뒷장

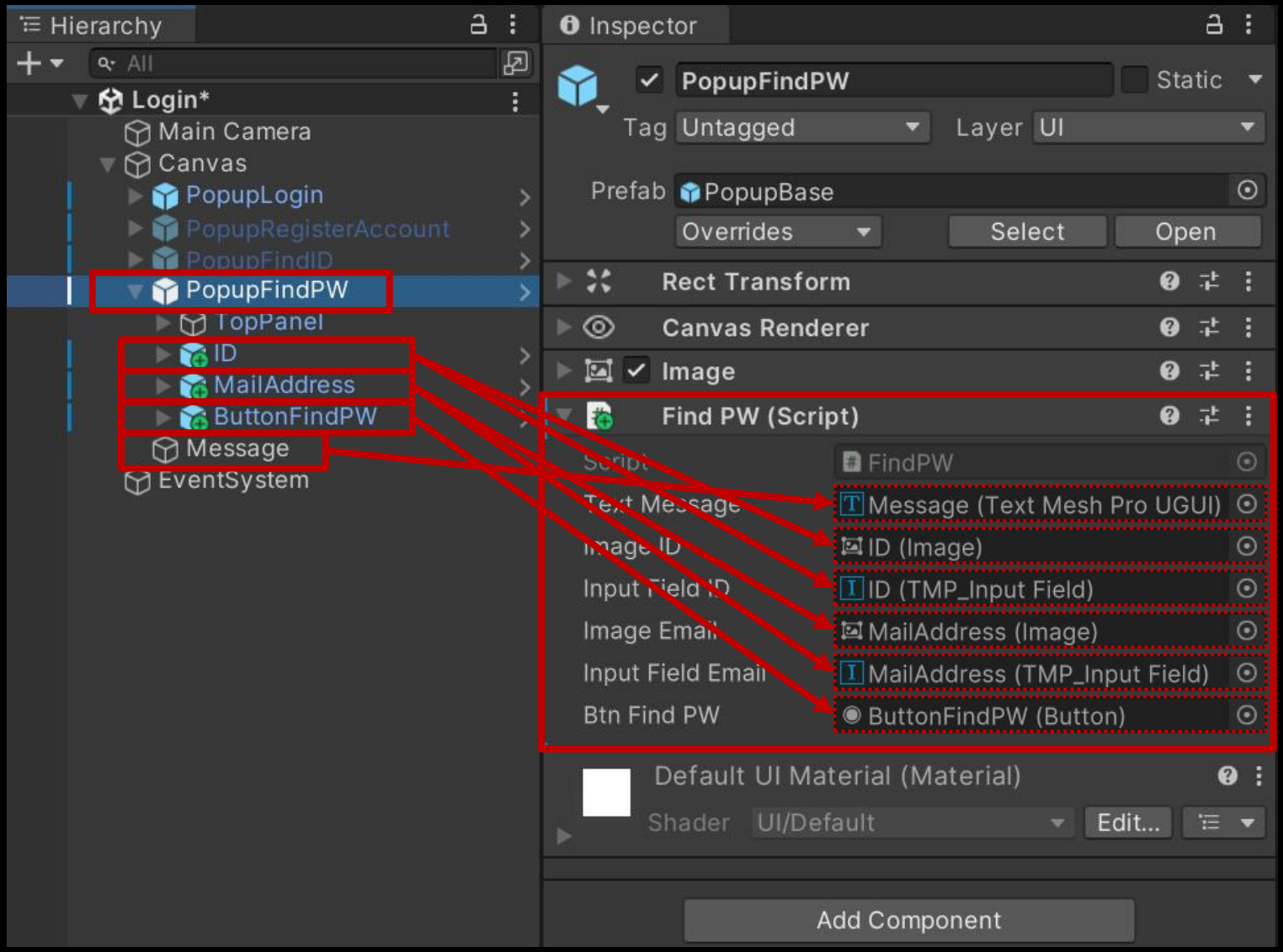
게임 유저 관리

- 뒤끝 서버와 연동해 비밀번호 찾기를 제어하는 스크립트 생성 및 작성 (계속)

```
60          // 메일 발송 실패
61          else
62          {
63              string message = string.Empty;
64
65              switch ( int.Parse(callback.GetStatusCode()) )
66              {
67                  case 404:   // 해당 이메일의 게이머가 없는 경우
68                      message = "해당 이메일을 사용하는 사용자가 없습니다.";
69                      break;
70                  case 429:   // 24시간 이내에 5회 이상 같은 이메일 정보로 아이디/비밀번호 찾기를 시도한 경우
71                      message = "24시간 이내에 5회 이상 아이디/비밀번호 찾기를 시도했습니다.";
72                      break;
73                  default:
74                      // statusCode : 400 => 프로젝트 명에 특수문자가 추가된 경우 (안내 메일 미발송 및 에러 발생)
75                      message = callback.GetMessage();
76                      break;
77              }
78
79              if ( message.Contains("이메일") )
80              {
81                  GuideForIncorrectlyEnteredData(imageEmail, message);
82              }
83              else
84              {
85                  SetMessage(message);
86              }
87          }
```

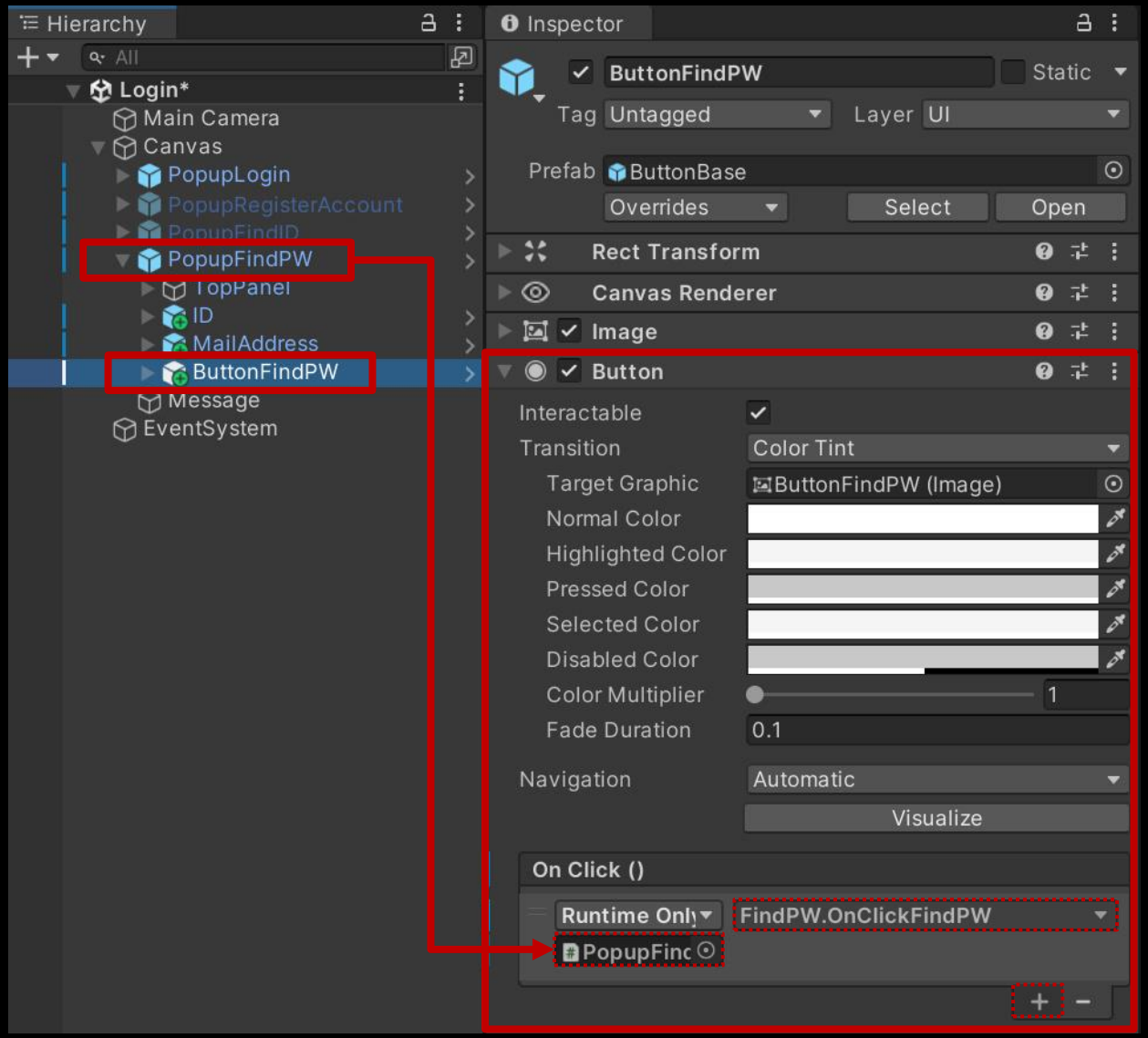
게임 유저 관리

- PopupFindPW 오브젝트에 "FindPW" 컴포넌트 추가 및 설정



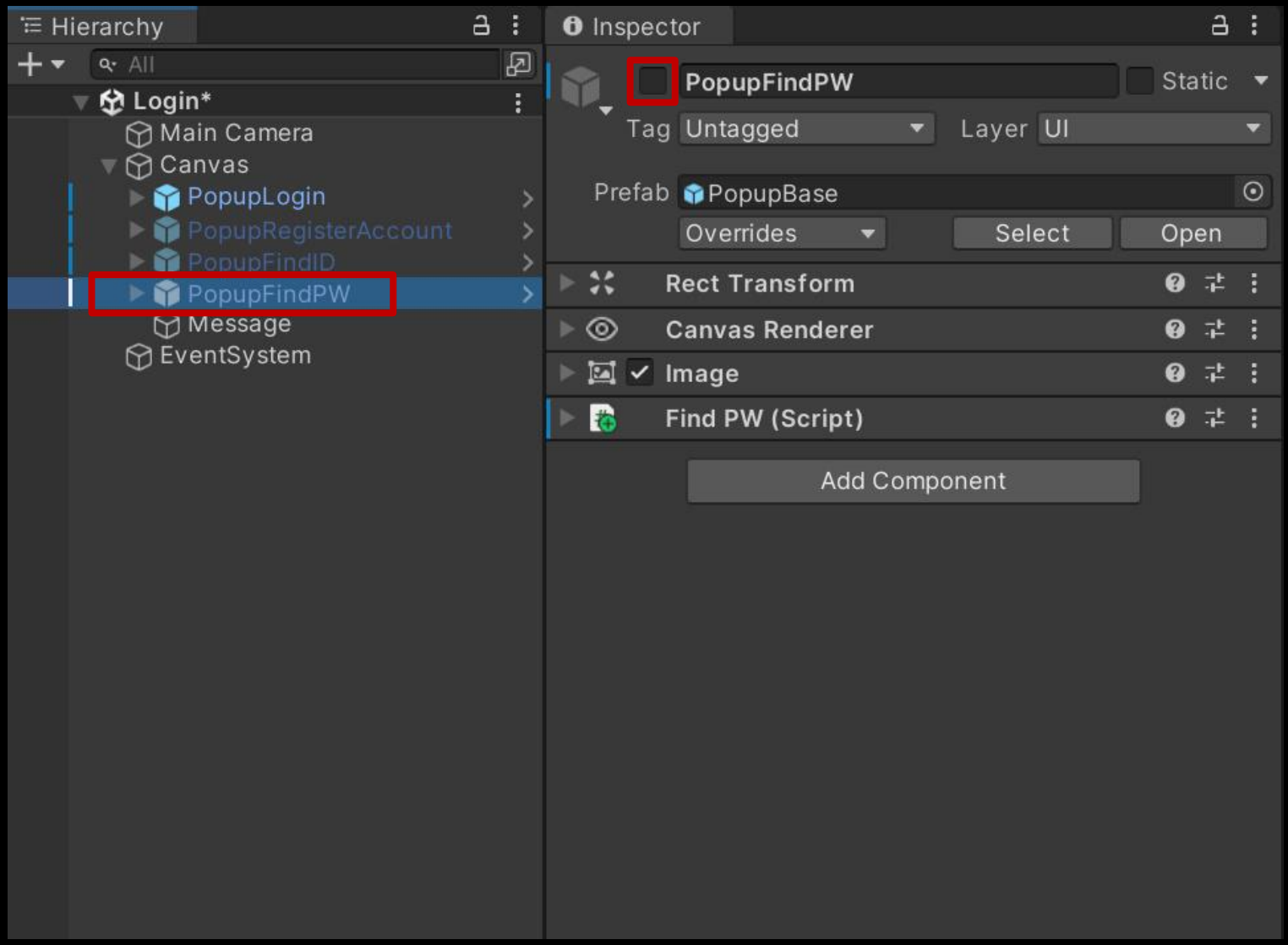
게임 유저 관리

- ButtonFindPW 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



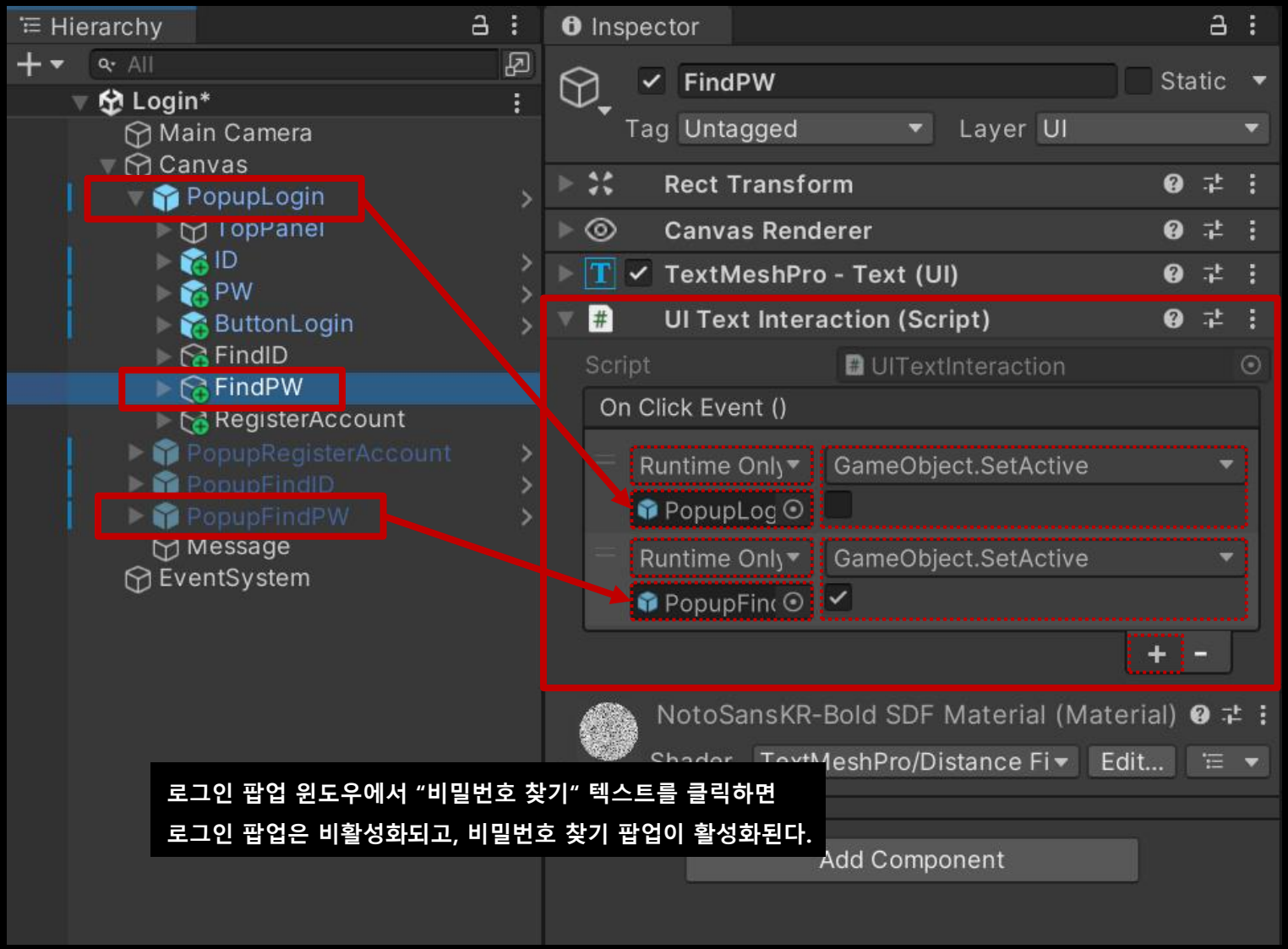
게임 유저 관리

- PopupFindPW 오브젝트 비활성화



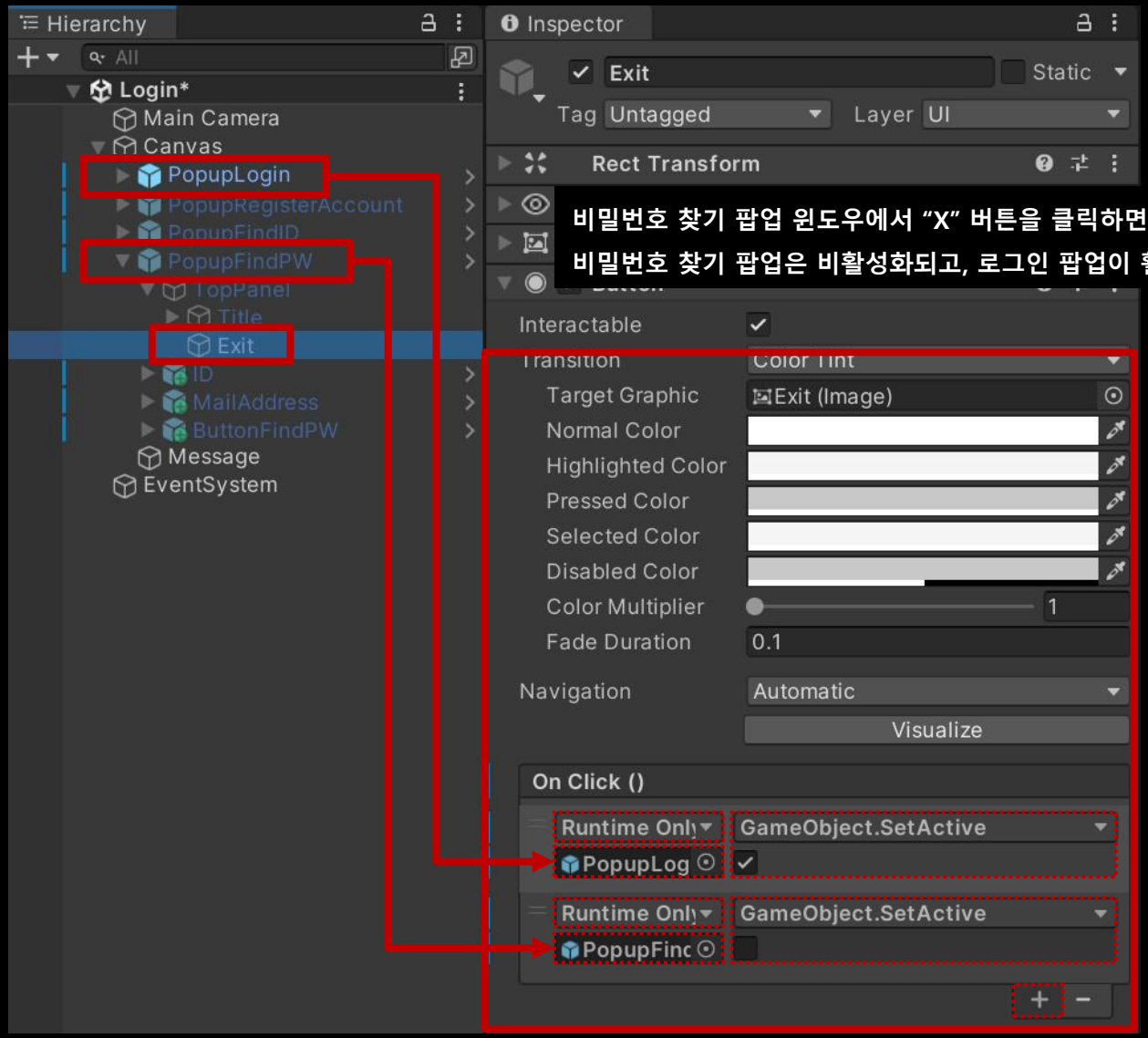
게임 유저 관리

- FindPW 오브젝트에 "UITextInteraction" 컴포넌트 추가 및 설정



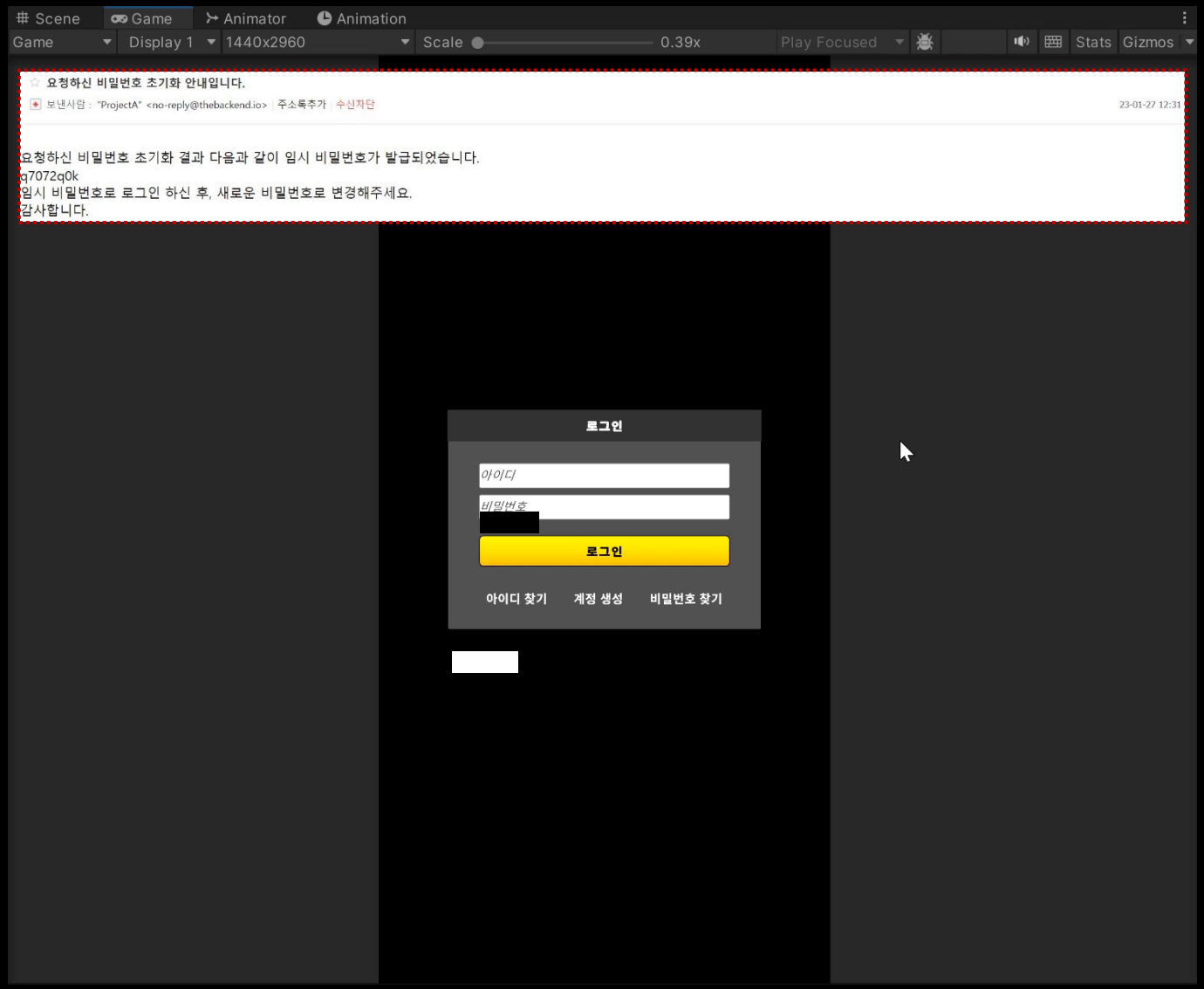
게임 유저 관리

- Exit 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



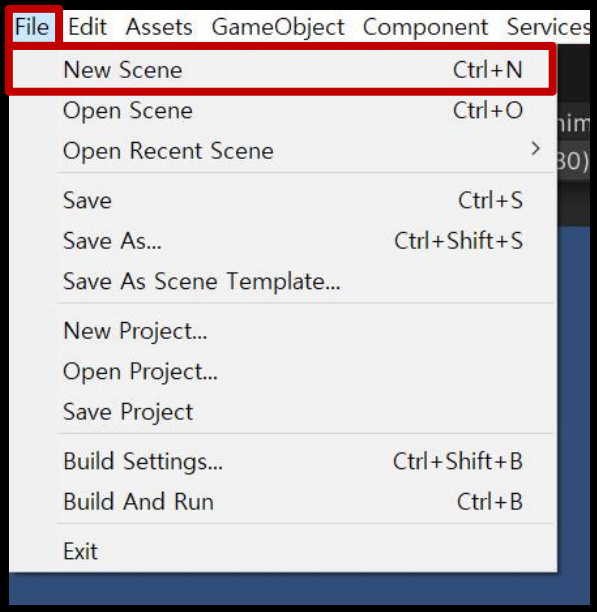
게임 유저 관리

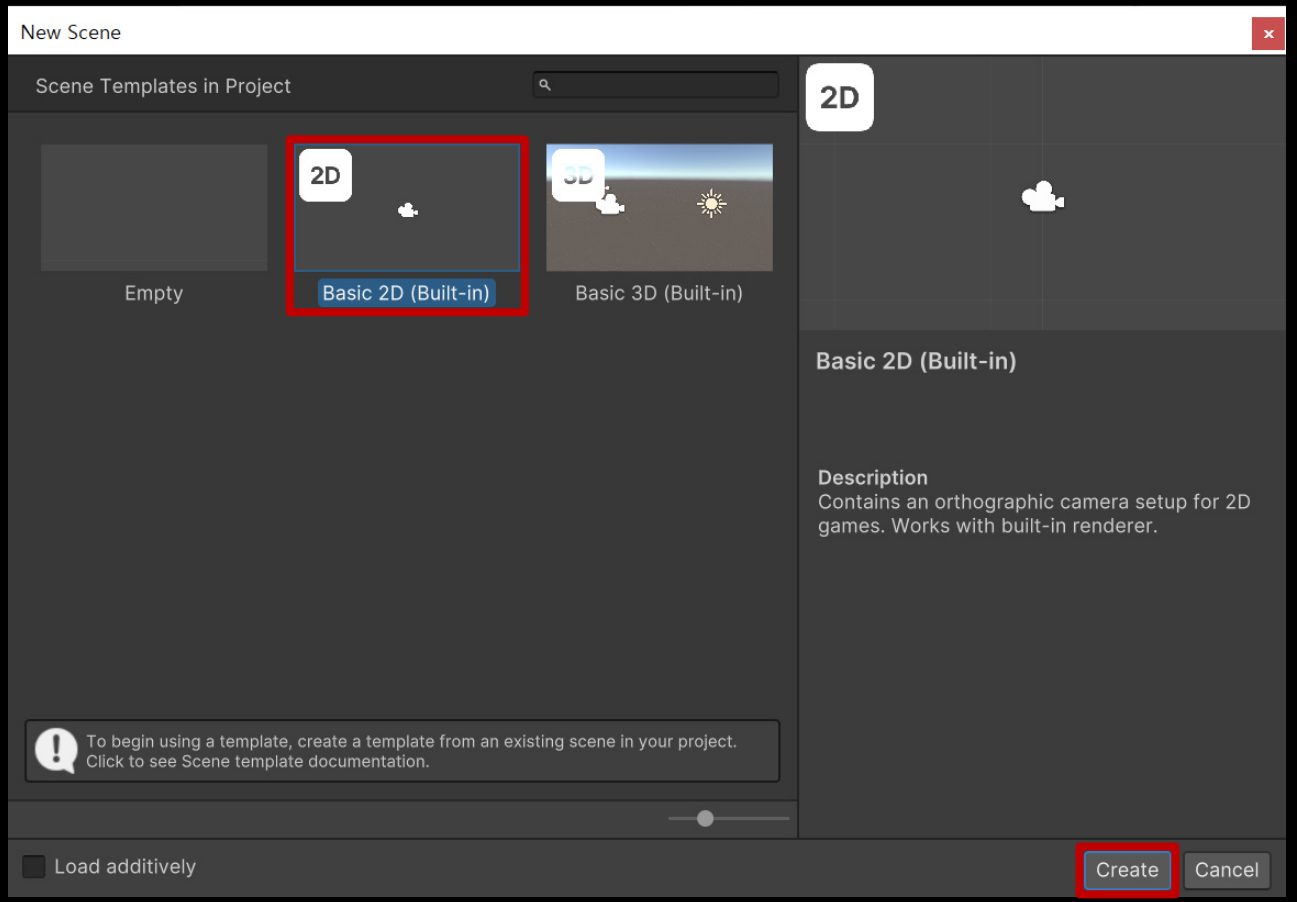
- 결과 화면


Scene
Game
Animator
Animation
Game
Display 1
1440x2960
Scale
0.39x
Play Focused
Stats
Gizmos
요청하신 비밀번호 초기화 안내입니다.
보낸사람 : "ProjectA" <no-reply@thebackend.io> | 주소록추가 | 수신차단
23-01-27 12:31
요청하신 비밀번호 초기화 결과 다음과 같이 임시 비밀번호가 발급되었습니다.
q7072q0k
임시 비밀번호로 로그인 하신 후, 새로운 비밀번호로 변경해주세요.
감사합니다.
로그인
아이디
비밀번호
로그인
아이디 찾기
계정 생성
비밀번호 찾기

게임 유저 관리

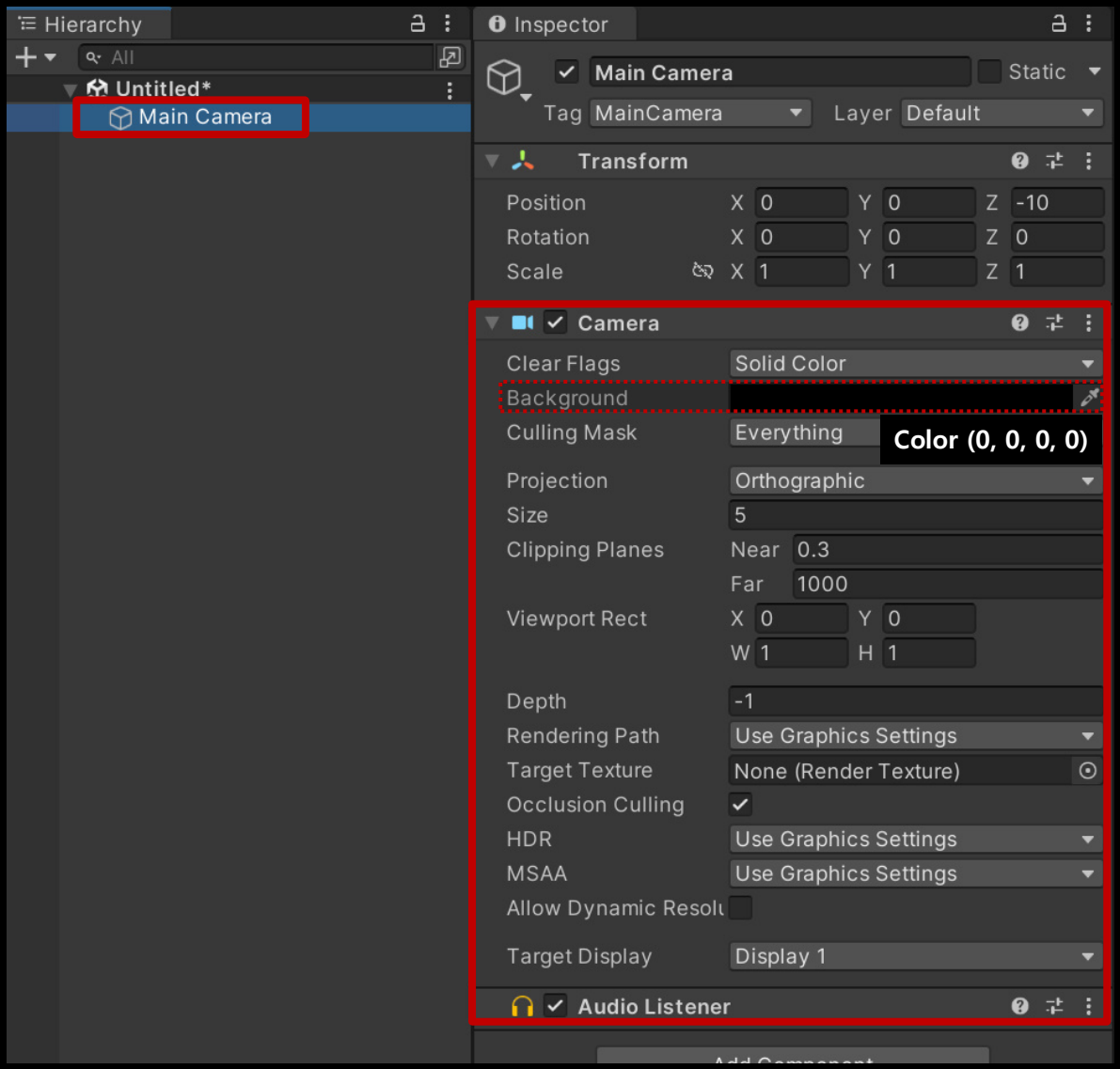
- 유저 정보 출력
  - Lobby 씬 생성
    - File - New Scene





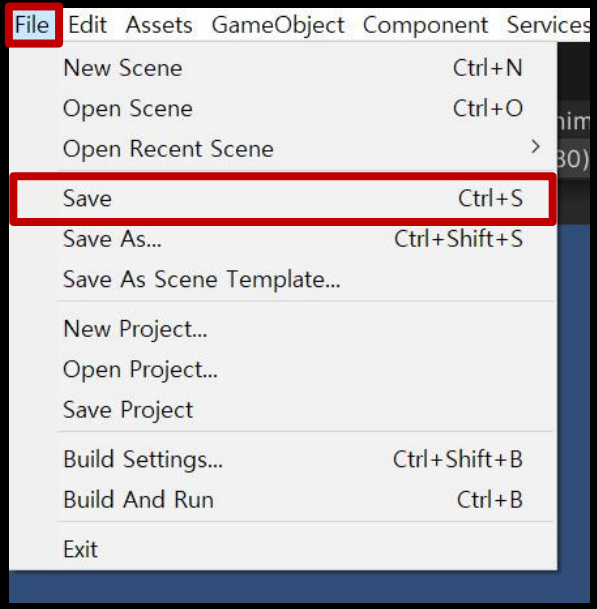
게임 유저 관리

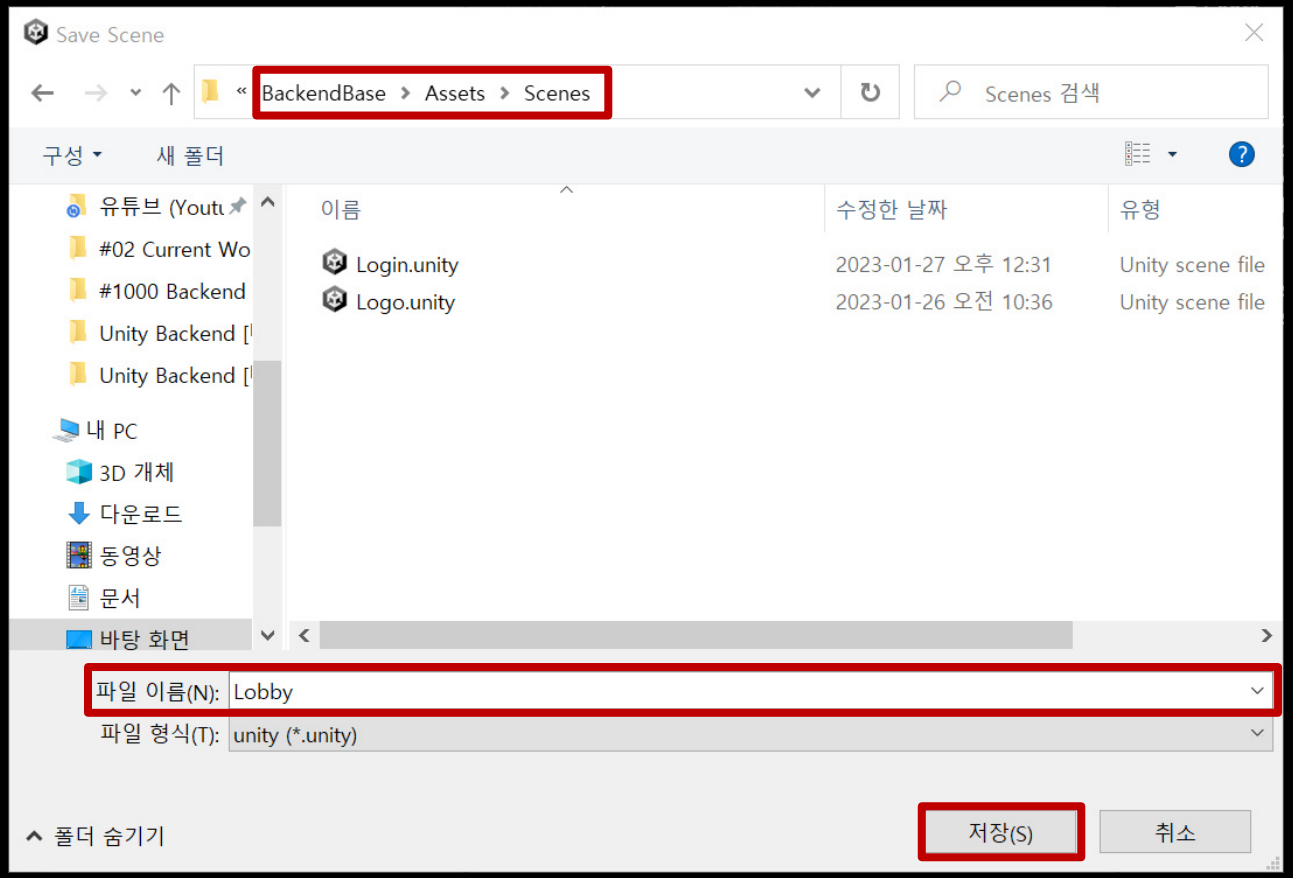
- 카메라 설정



게임 유저 관리

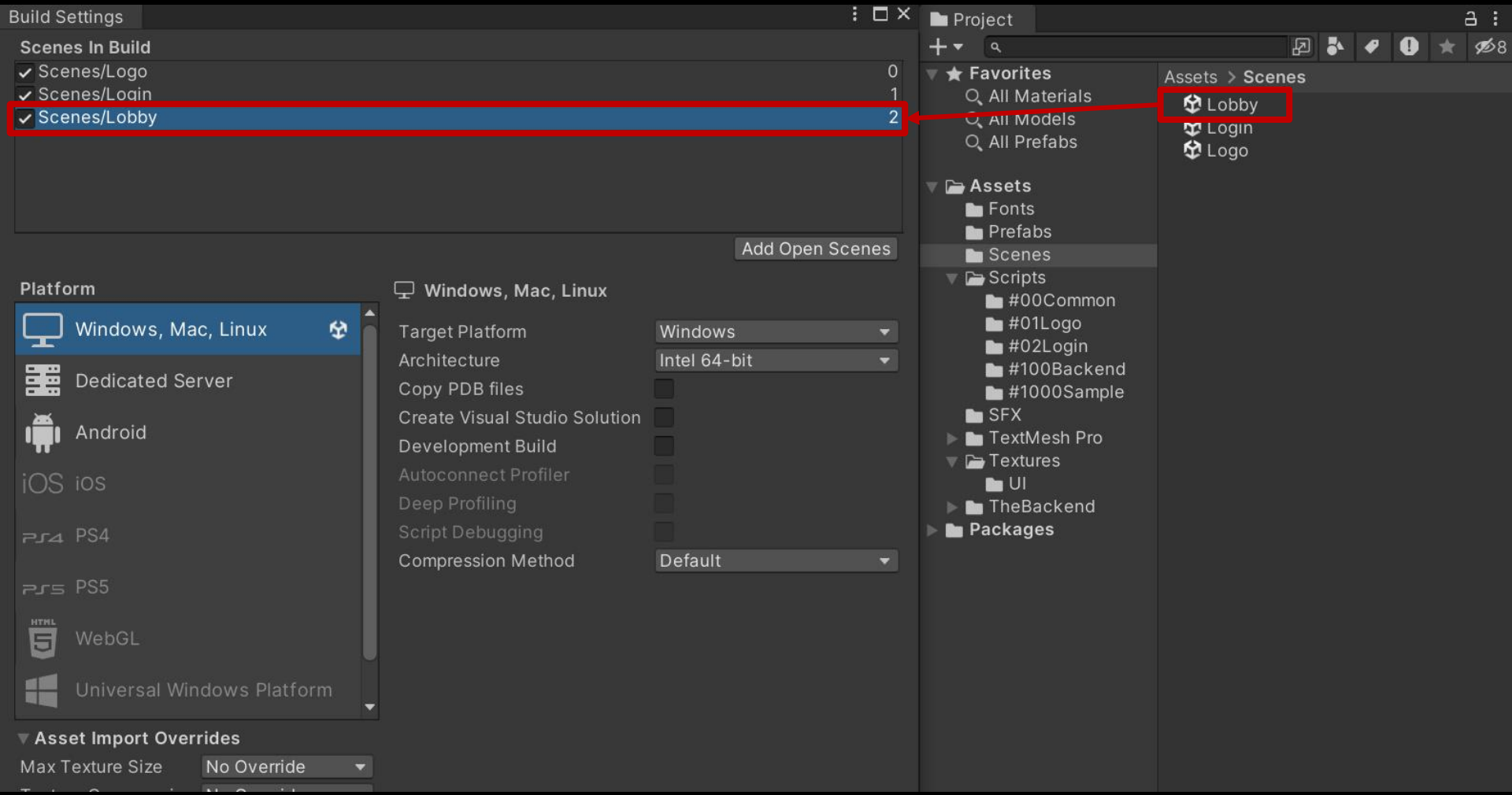
- Lobby 씬 저장
  - File - Save





게임 유저 관리

- Scenes In Build에 씬 등록
  - File - Build Settings



게임 유저 관리

- SceneNames 열거형에 Lobby 추가
  - Utils Script 수정

```
1    using UnityEngine.SceneManagement;
2
3    public enum SceneNames { Logo=0, Login, Lobby, }
4
5    public static class Utils
6    {
7        public static string GetActiveScene()...
11
12       public static void LoadScene(string sceneName="")...
23
24       public static void LoadScene(SceneNames sceneName)...
29   }
```

게임 유저 관리

- 로그인에 성공했을 때 Lobby 씬으로 이동
  - Login Script 수정

```
21      /// <summary> "로그인" 버튼을 눌렀을 때 호출
24      public void OnClickLogin()...
43
44      /// <summary> 로그인 시도 후 서버로부터 전달받은 message를 기반으로 로직 처리
47      private void ResponseToLogin(string ID, string PW)
48      {
49          // 서버에 로그인 요청
50          Backend.BMember.CustomLogin(ID, PW, callback =>
51          {
52              StopCoroutine(nameof(LoginProcess));
53
54              // 로그인 성공
55              if ( callback.IsSuccess() )
56              {
57                  SetMessage($"{inputFieldID.text}님 환영합니다.");
58
59                  // Lobby 씬으로 이동
60                  Utils.LoadScene(SceneNames.Lobby);
61              }
62              // 로그인 실패
63              else...
96          });
97      }
98
99      private IEnumerator LoginProcess()...
112 }
```

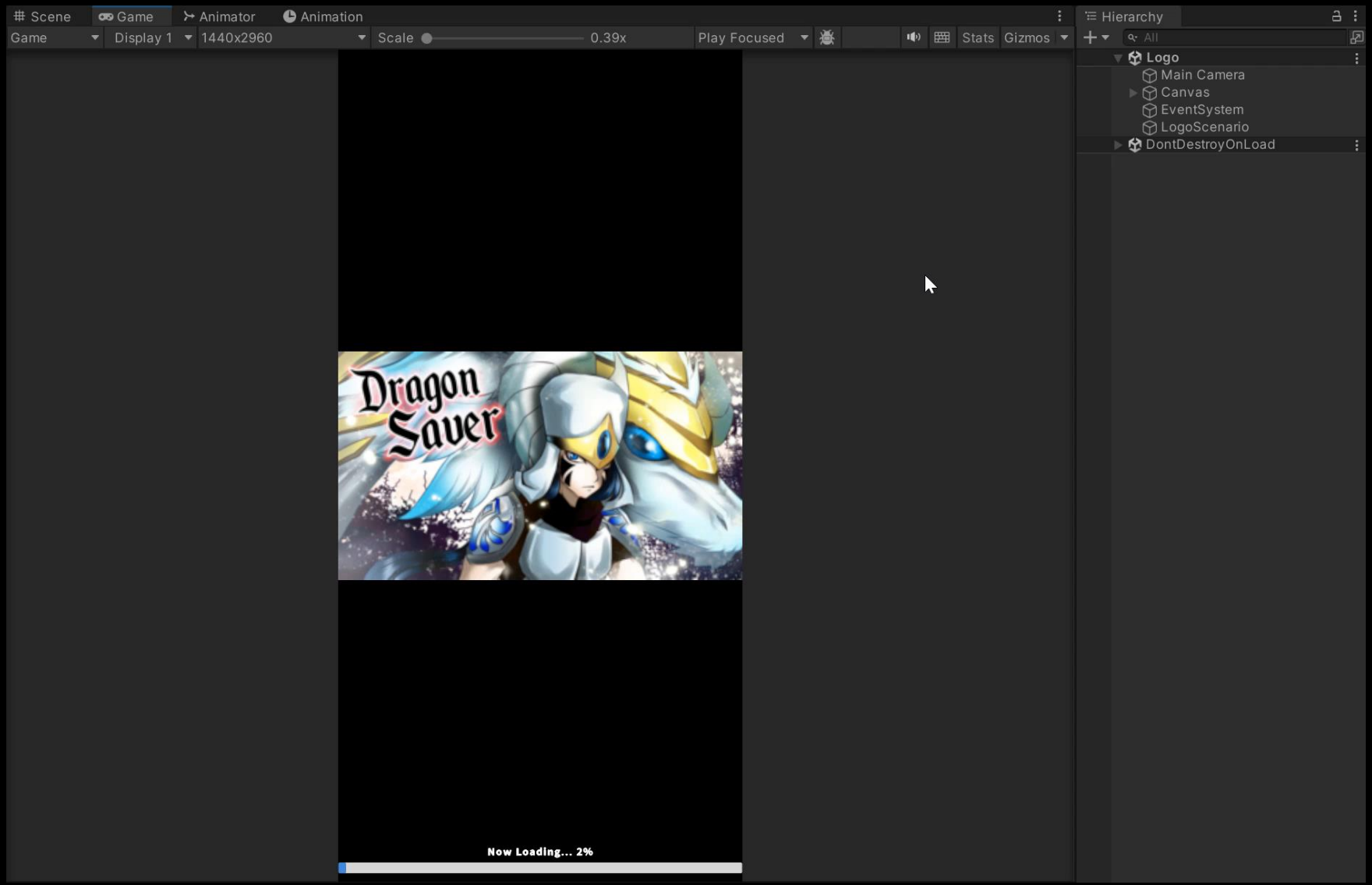
게임 유저 관리

- 계정 생성에 성공했을 때 Lobby 씬으로 이동
  - RegisterAccount Script 수정

```
28      /// <summary> "계정 생성" 버튼을 눌렀을 때 호출
31      public void OnClickRegisterAccount()...
63
64      /// <summary>
65      /// 계정 생성 시도 후 서버로부터 전달받은 message를 기반으로 로직 처리
66      /// </summary>
67      private void CustomSignUp()
68      {
69          Backend.BMember.CustomSignUp(inputFieldID.text, inputFieldPW.text, callback =>
70          {
71              // "계정 생성" 버튼 상호작용 활성화
72              btnRegisterAccount.interactable = true;
73
74              // 계정 생성 성공
75              if ( callback.IsSuccess() )
76              {
77                  // E-mail 정보 업데이트
78                  Backend.BMember.UpdateCustomEmail(inputFieldEmail.text, callback =>
79                  {
80                      if ( callback.IsSuccess() )
81                      {
82                          SetMessage($"계정 생성 성공. {inputFieldID.text}님 환영합니다.");
83
84                          // Lobby 씬으로 이동
85                          Utils.LoadScene(SceneNames.Lobby);
86                      }
87                  });
88              }
89              // 계정 생성 실패
90              else...
118         });
119     }
120 }
```

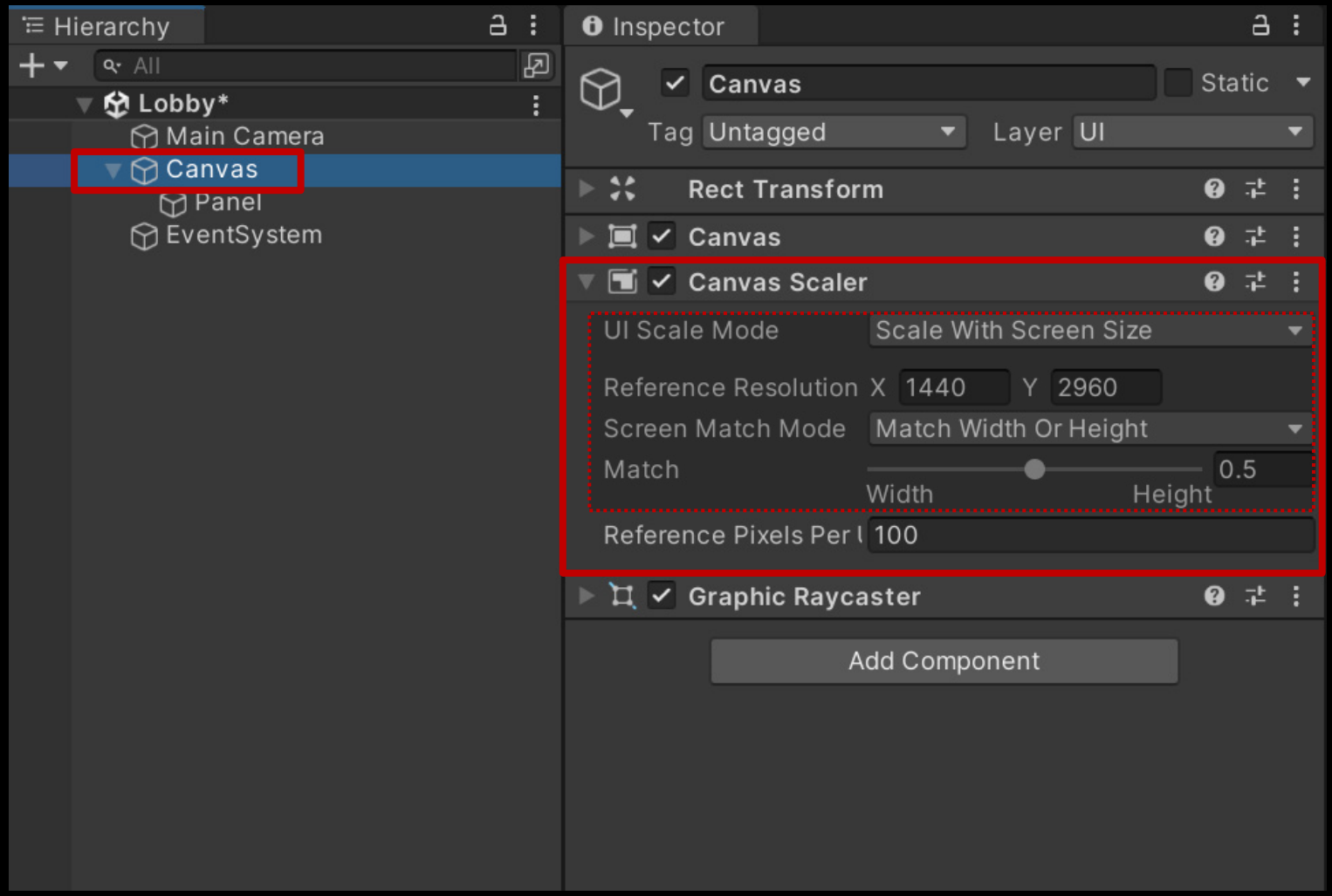
게임 유저 관리

- 결과 화면


Scene
Game
Animator
Animation
Game
Display 1
1440x2960
Scale
0.39x
Play Focused
Stats
Gizmos
Hierarchy
All
Logo
Main Camera
Canvas
EventSystem
LogoScenario
DontDestroyOnLoad
Dragon Saver
Now Loading... 2%

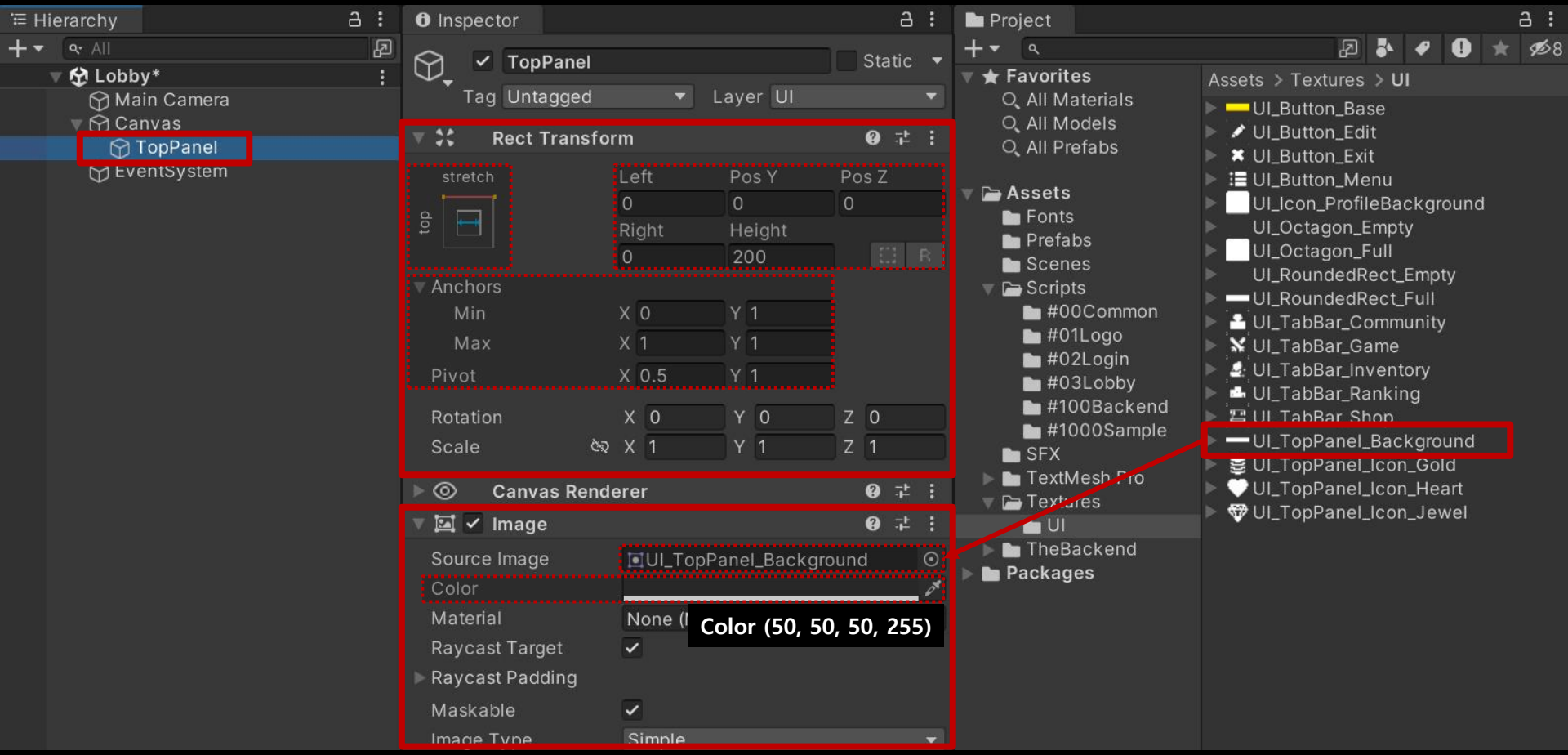
게임 유저 관리

- Lobby 씬 상단의 프로필, 재화 UI들을 관리하는 Panel UI 생성 및 설정
  - GameObject - UI - Panel



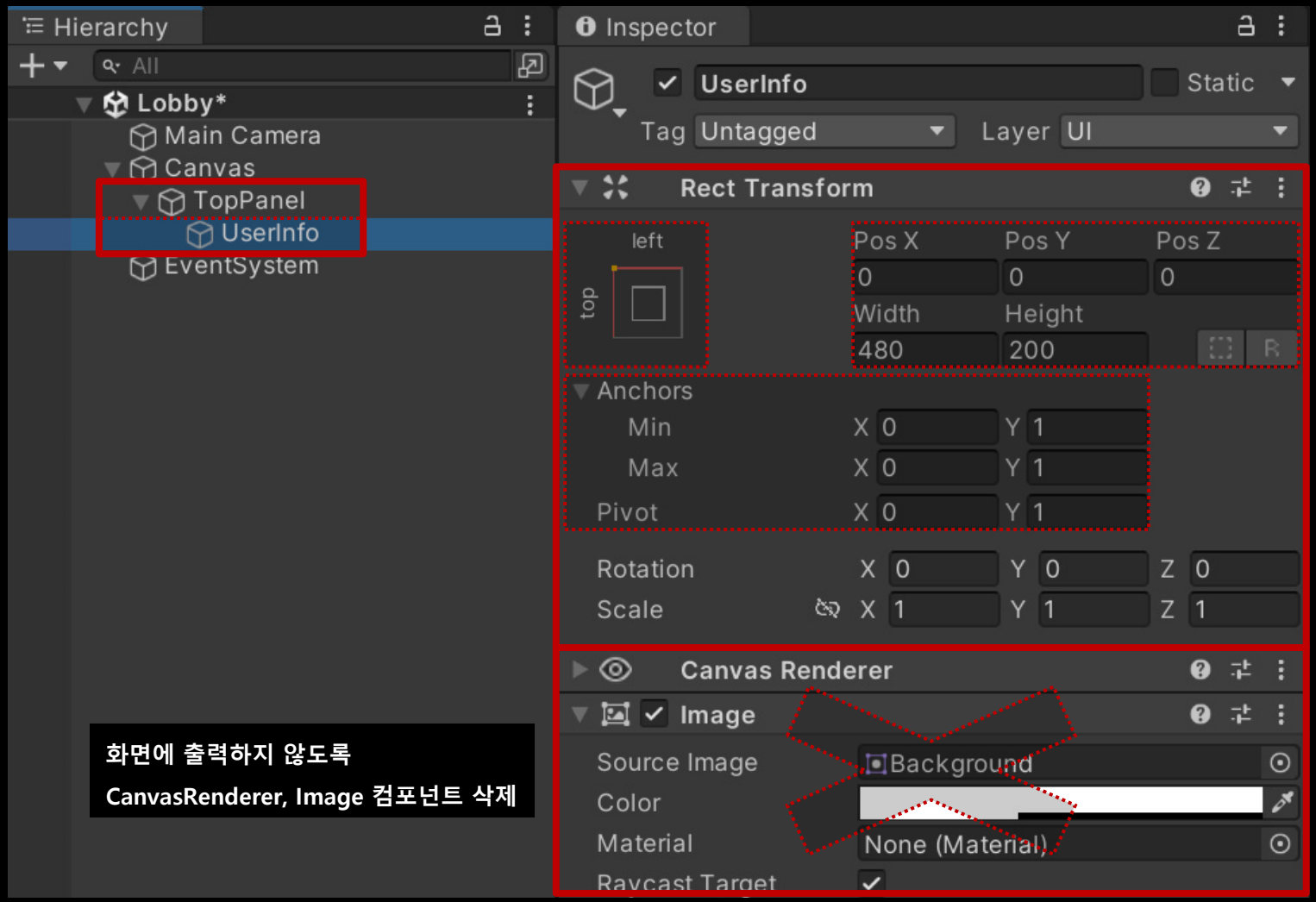
게임 유저 관리

- Lobby 씬 상단의 프로필, 재화 UI들을 관리하는 Panel UI 생성 및 설정 (계속)



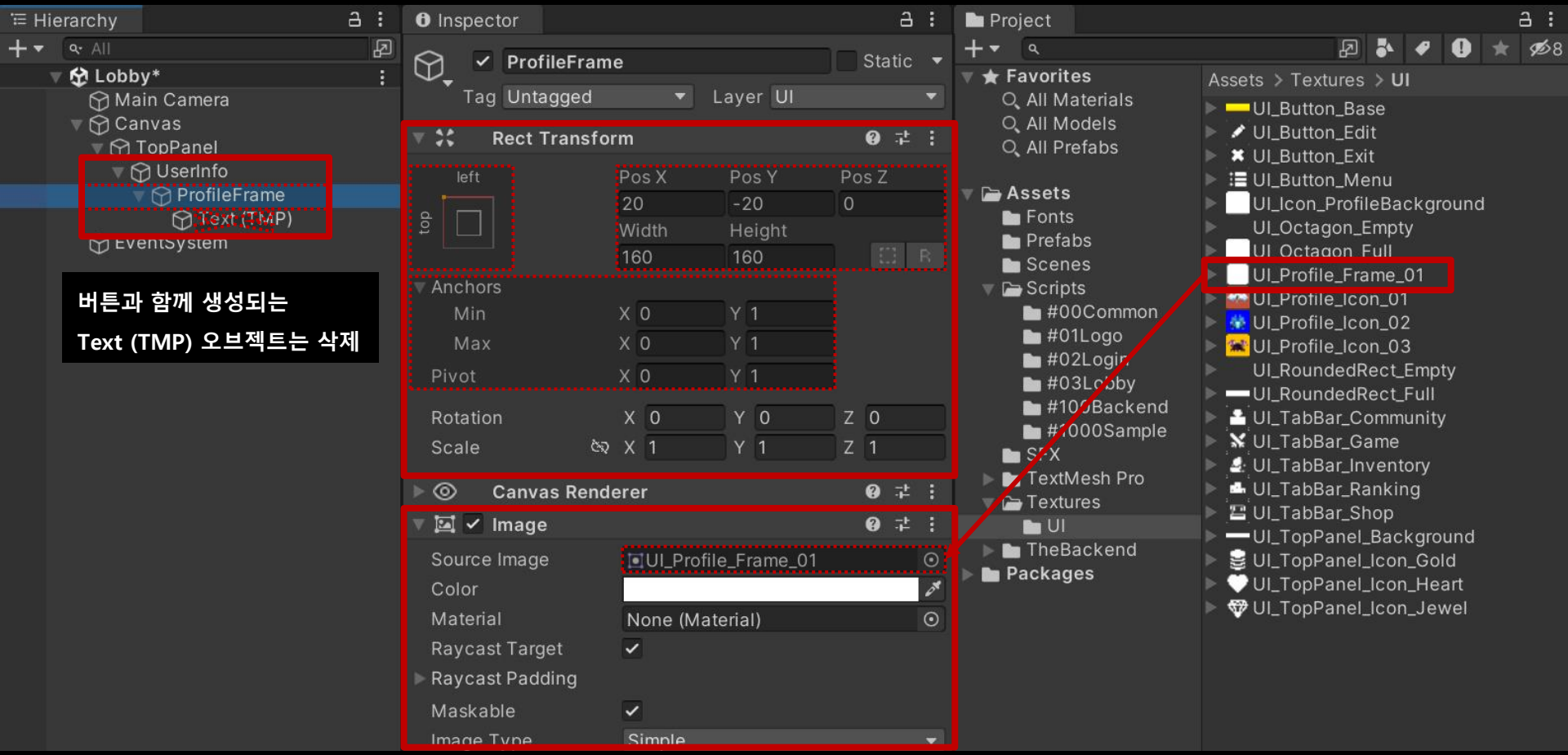
게임 유저 관리

- 유저 프로필 정보 UI들을 관리하는 Panel UI 생성 및 설정
  - GameObject - UI - Panel



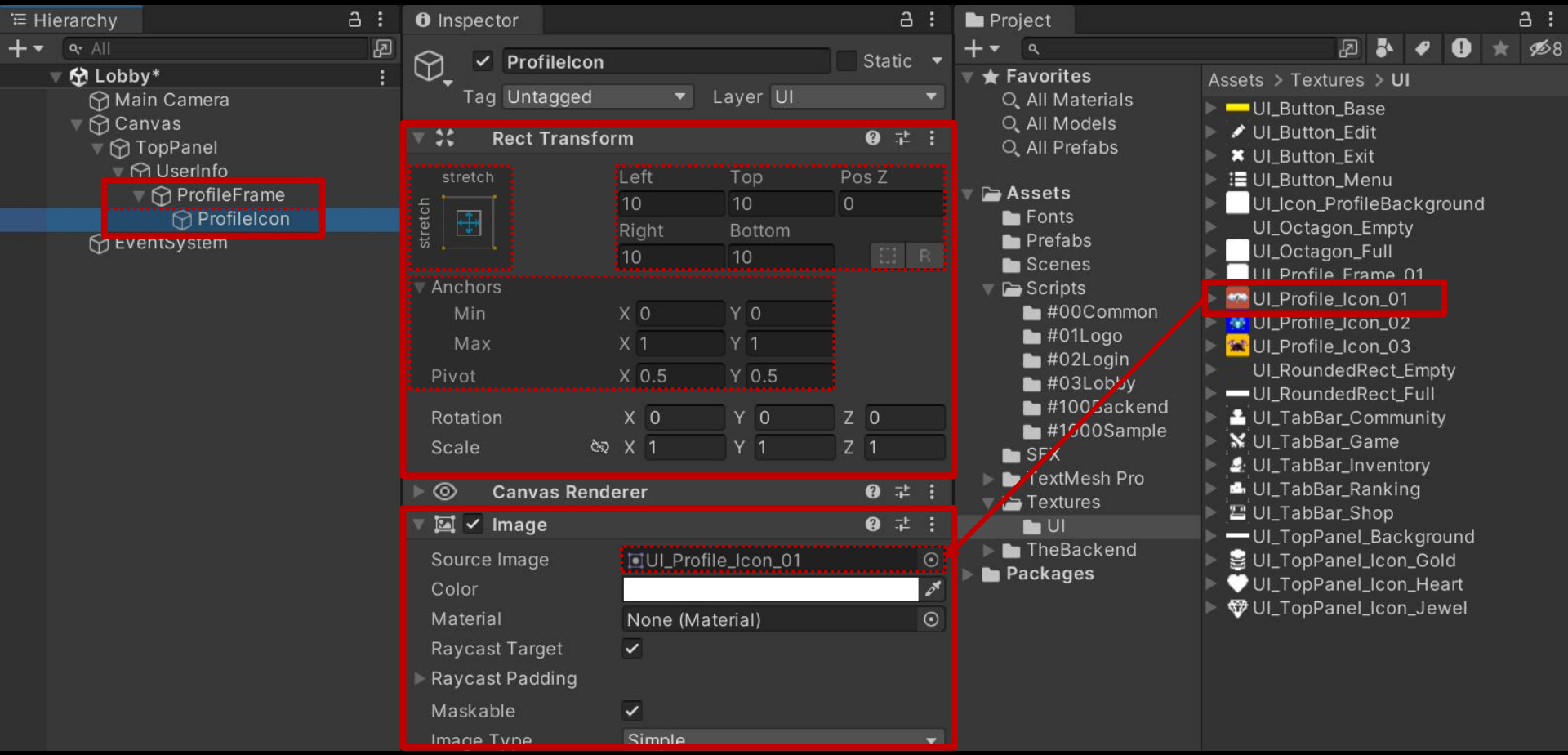
게임 유저 관리

- 프로필 "Button - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Button - TextMeshPro"



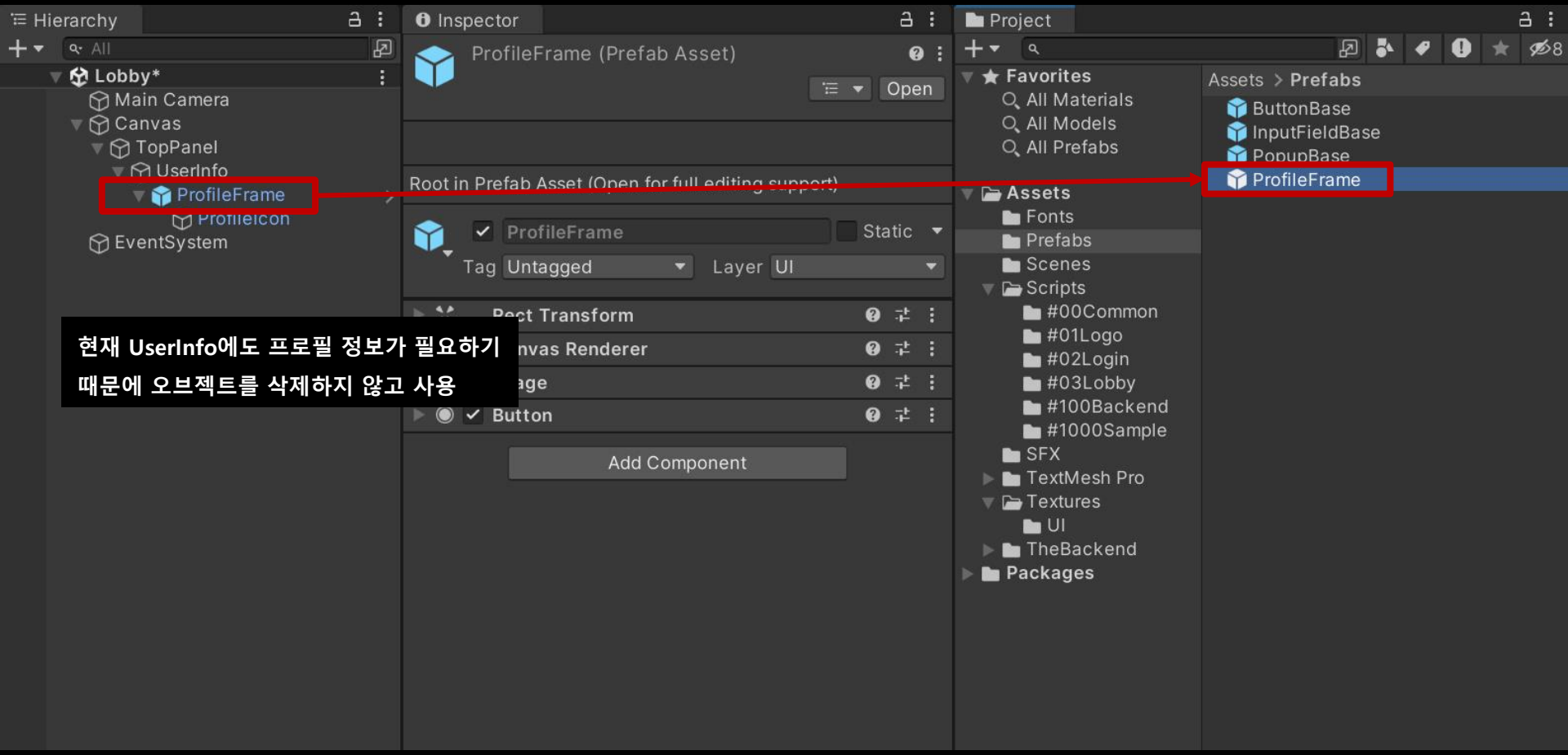
게임 유저 관리

- 프로필 아이콘 Image UI 생성 및 설정
  - GameObject - UI - Image



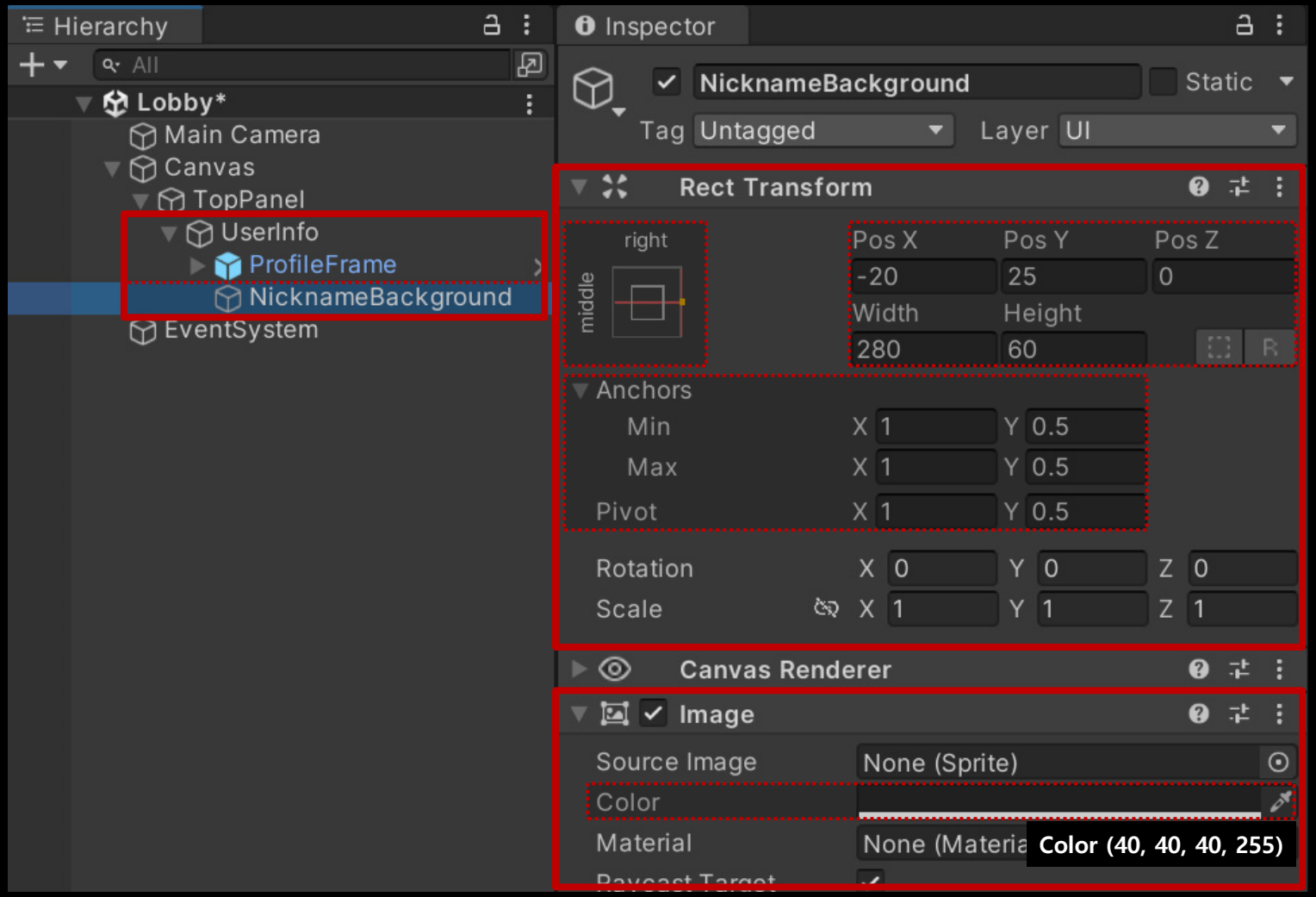
게임 유저 관리

- 프로필 오브젝트 Prefab 생성
  - Hierarchy View의 “ProfileFrame” 오브젝트를 Project View로 Drag & Drop



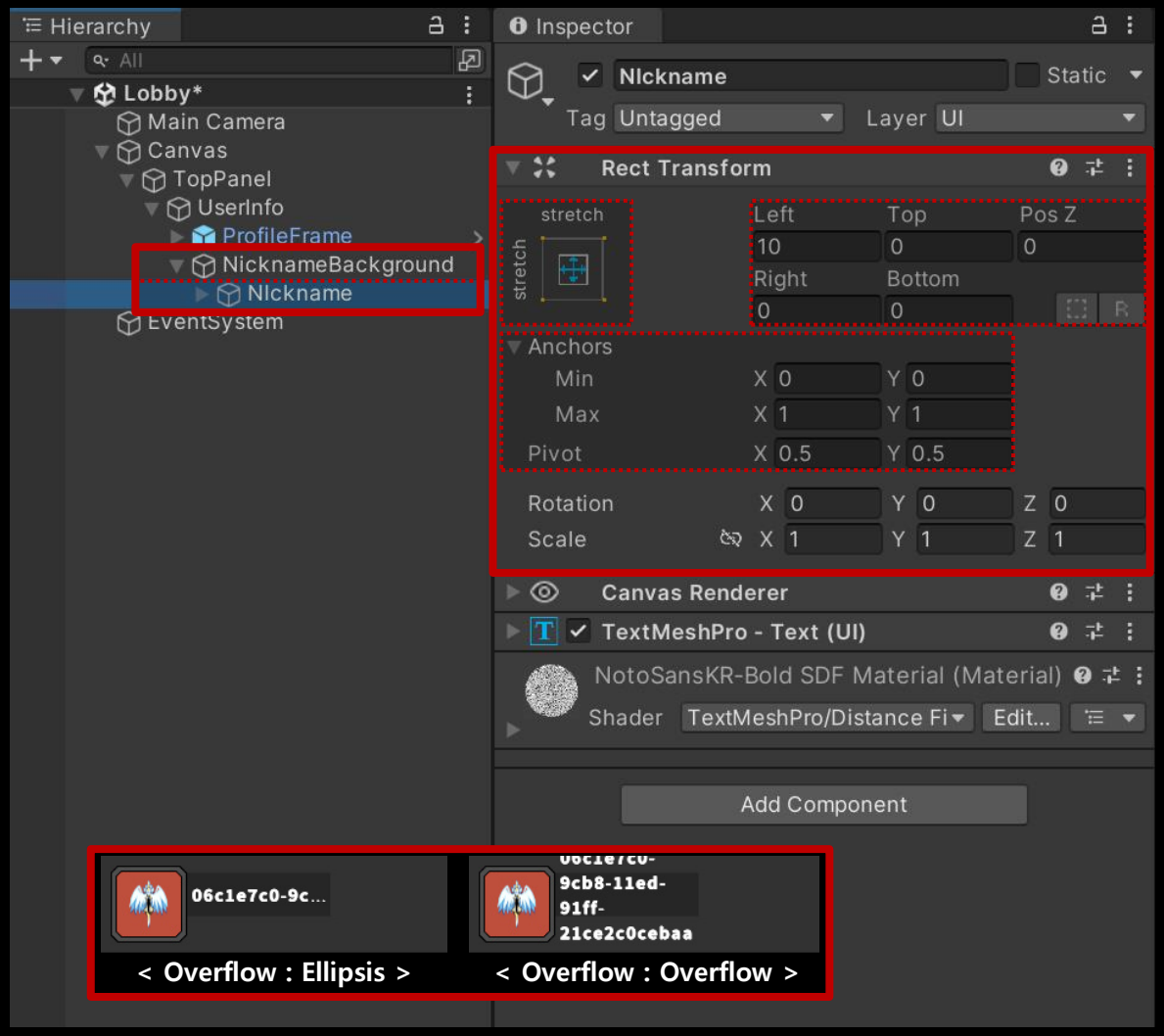
게임 유저 관리

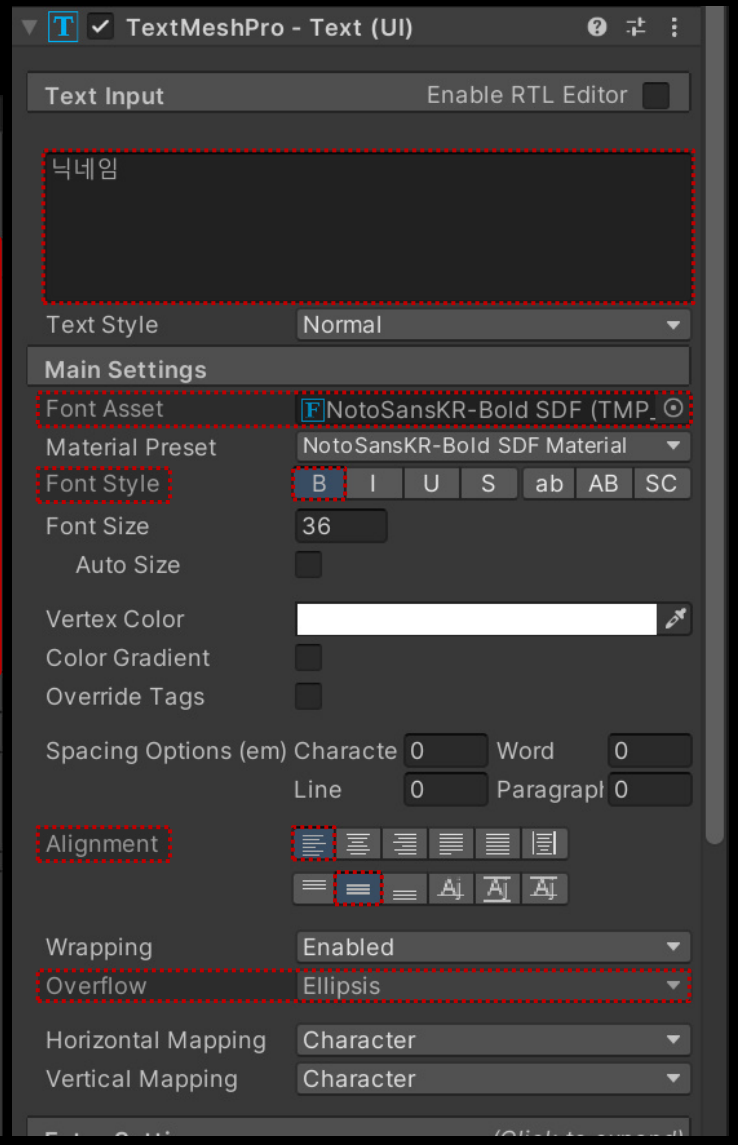
- 닉네임 배경 Image UI 생성 및 설정
  - GameObject - UI - Image



게임 유저 관리

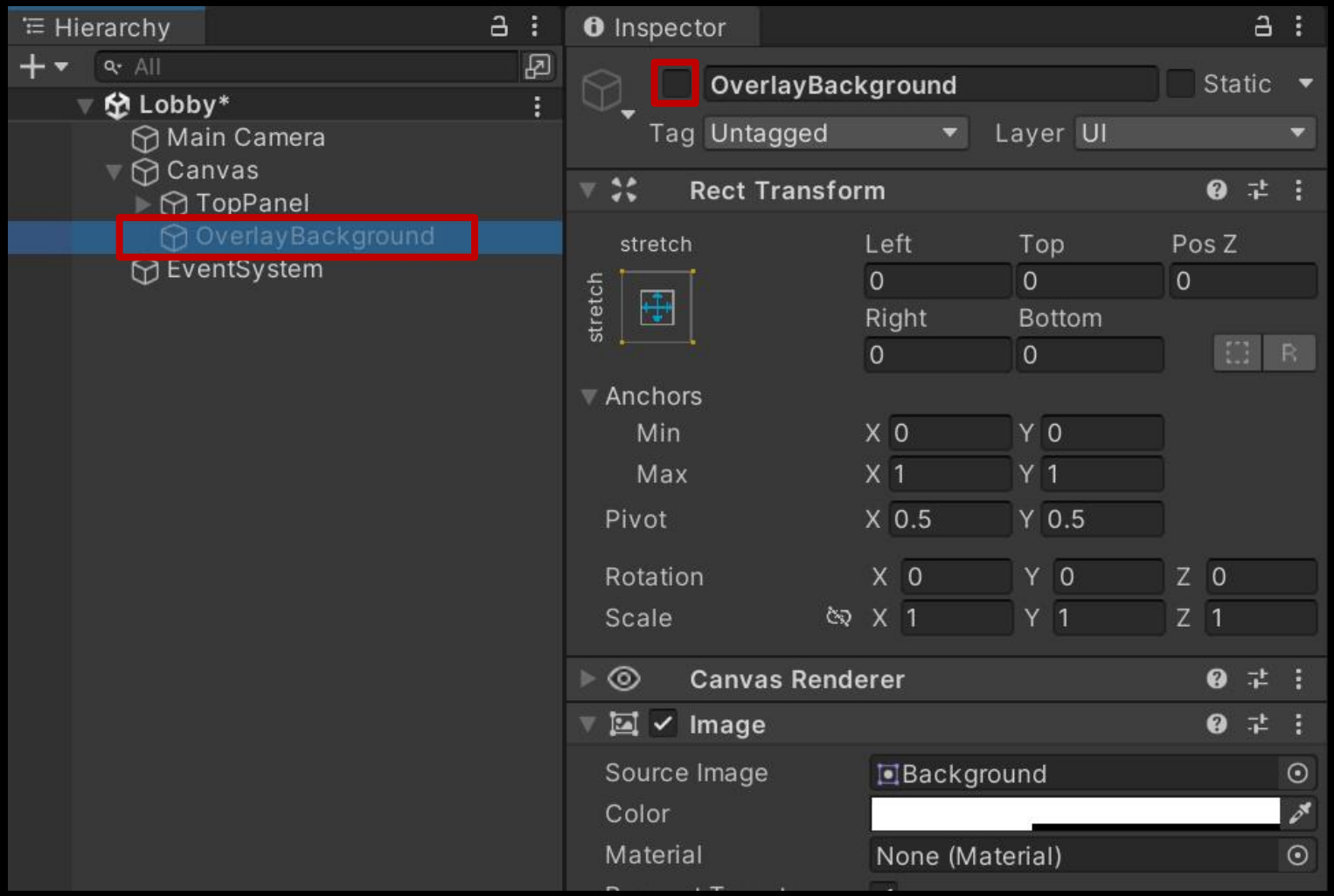
- 닉네임을 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"





게임 유저 관리

- 제일 앞에 활성화되는 팝업 UI 뒤에 배치하는 배경 Panel UI 생성 및 설정
  - GameObject - UI - Panel



게임 유저 관리

- 현재 로그인한 유저의 정보를 불러오는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "UserInfo"로 변경

```
1   using UnityEngine;
2   using BackEnd;
3   using LitJson;
4
5   public class UserInfo : MonoBehaviour
6   {
7       [System.Serializable]
8       public class UserInfoEvent : UnityEngine.Events.UnityEvent { }
9       public UserInfoEvent onUserInfoEvent = new UserInfoEvent();
10
11      private static  UserInfoData data = new UserInfoData();
12      public  static  UserInfoData Data => data;
13
```

게임 유저 관리

- 현재 로그인한 유저의 정보를 불러오는 스크립트 생성 및 작성 (계속)

```
14      public void GetUserInfoFromBackend()
15      {
16          // 현재 로그인한 사용자 정보 불러오기
17          // https://developer.thebackend.io/unity3d/guide/bmember/userInfo/
18          Backend.BMember.GetUserInfo(callback =>
19          {
20              // 정보 불러오기 성공
21              if ( callback.IsSuccess() )...
45              // 정보 불러오기 실패
46              else
47              {
48                  // 유저 정보를 기본 상태로 설정
49                  // Tip. 일반적으로 오프라인 상태를 대비해 기본적인 정보를 저장해두고 오프라인일 때 불러와서 사용
50                  data.Reset();
51                  Debug.LogError(callback.GetMessage());
52              }
53
54              // 유저 정보 불러오기에 성공했을 때 onUserInfoEvent에 등록되어 있는 이벤트 메소드 호출
55              onUserInfoEvent?.Invoke();
56          });
57      }
58  }
59
```

게임 유저 관리

- 현재 로그인한 유저의 정보를 불러오는 스크립트 생성 및 작성 (계속)

```
20              // 정보 불러오기 성공
21              if ( callback.IsSuccess() )
22              {
23                  // JSON 데이터 파싱 성공
24                  try
25                  {
26                      JsonData json = callback.GetReturnValuetoJSON()["row"];
27
28                      data.gamerId                = json["gamerId"].ToString();
29                      data.countryCode            = json["countryCode"]?.ToString();
30                      data.nickname               = json["nickname"]?.ToString();
31                      data.inDate                 = json["inDate"].ToString();
32                      data.emailForFindPassword   = json["emailForFindPassword"]?.ToString();
33                      data.subscriptionType       = json["subscriptionType"].ToString();
34                      data.federationId           = json["federationId"]?.ToString();
35                  }
36                  // JSON 데이터 파싱 실패
37                  catch ( System.Exception e )
38                  {
39                      // 유저 정보를 기본 상태로 설정
40                      data.Reset();
41                      // try-catch 에러 출력
42                      Debug.LogError(e);
43                  }
44              }
```

게임 유저 관리

- 현재 로그인한 유저의 정보를 불러오는 스크립트 생성 및 작성 (계속)

```
60  public class UserInfoData
61  {
62      public  string  gamerId;                // 유저의 gamerID
63      public  string  countryCode;            // 국가코드. 설정 안했으면 null
64      public  string  nickname;               // 닉네임. 설정 안했으면 null
65      public  string  inDate;                 // 유저의 inDate
66      public  string  emailForFindPassword;   // 이메일주소. 설정 안했으면 null
67      public  string  subscriptionType;       // 커스텀, 페더레이션 타입
68      public  string  federationId;           // 구글, 애플, 페이스북 페더레이션 ID. 커스텀 계정은 null
69
70      public void Reset()
71      {
72          gamerId                 = "Offline";
73          countryCode             = "Unknown";
74          nickname                = "Noname";
75          inDate                  = string.Empty;
76          emailForFindPassword    = string.Empty;
77          subscriptionType        = string.Empty;
78          federationId            = string.Empty;
79      }
80  }
```

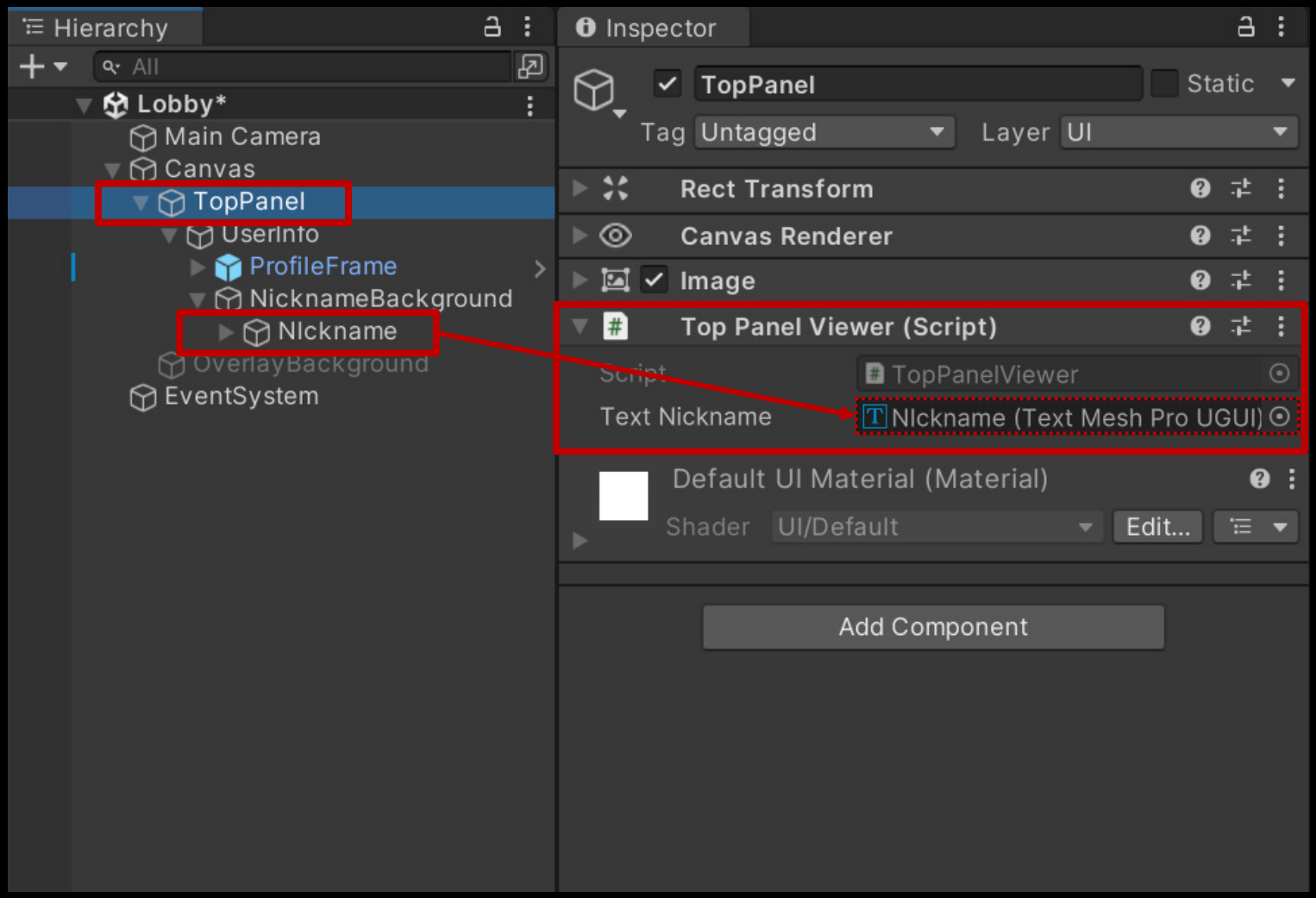
게임 유저 관리

- Top Panel에 출력하는 UI 정보를 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "TopPanelViewer"로 변경

```
1  using UnityEngine;
2  using TMPro;
3
4  public class TopPanelViewer : MonoBehaviour
5  {
6      [SerializeField]
7      private TextMeshProUGUI textNickname;
8
9      public void UpdateNickname()
10     {
11         // 닉네임이 없으면 gamer_id를 출력하고, 닉네임이 있으면 닉네임 출력
12         textNickname.text = UserInfo.Data.nickname == null ?
13                             UserInfo.Data.gamerId : UserInfo.Data.nickname;
14     }
15 }
```

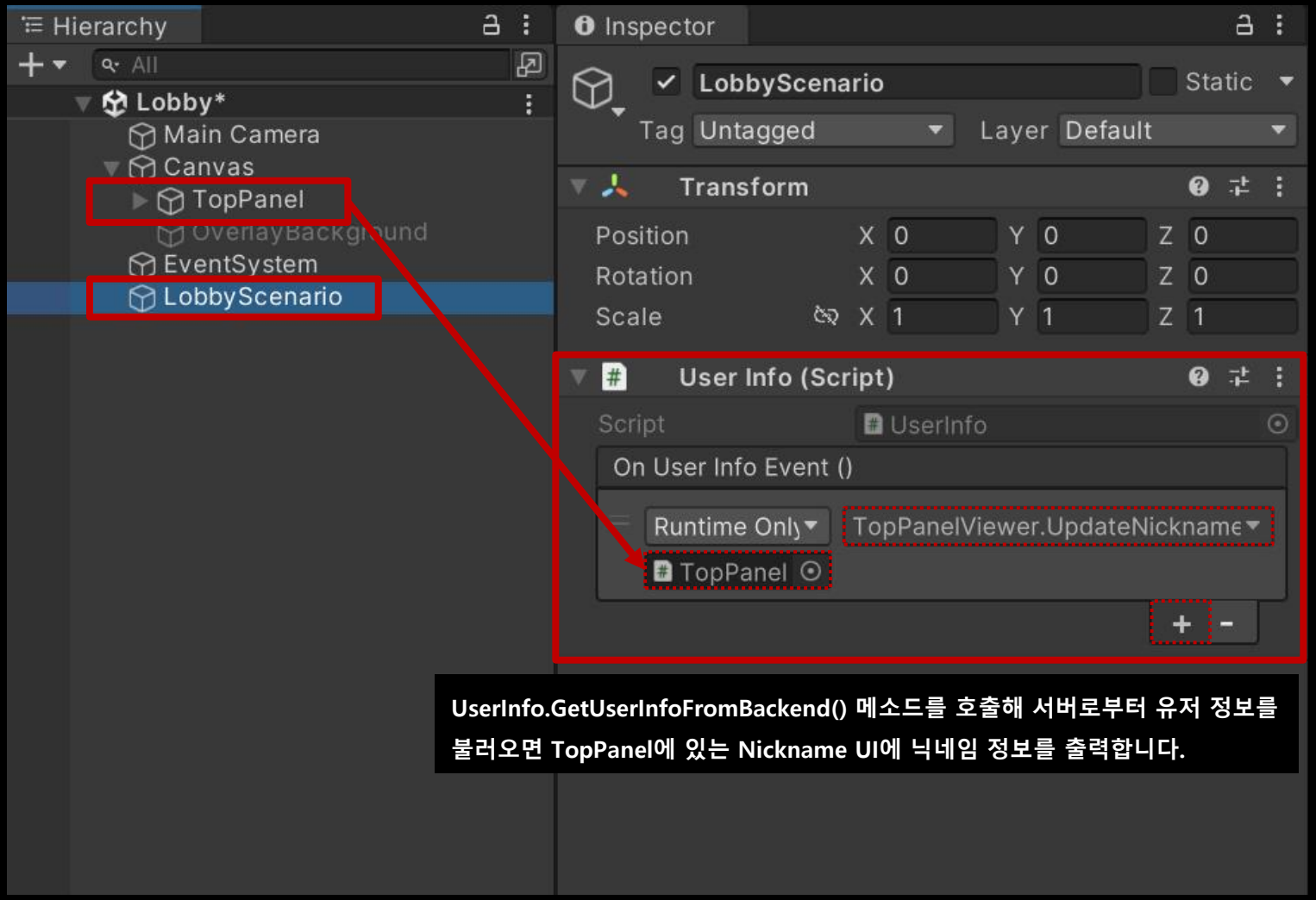
게임 유저 관리

- TopPanel 오브젝트에 "TopPanelViewer" 컴포넌트 추가 및 설정



게임 유저 관리

- LobbyScenario 오브젝트에 "UserInfo" 컴포넌트 추가 및 설정
  - GameObject - Create Empty



UserInfo.GetUserInfoFromBackend() 메소드를 호출해 서버로부터 유저 정보를 불러오면 TopPanel에 있는 Nickname UI에 닉네임 정보를 출력합니다.

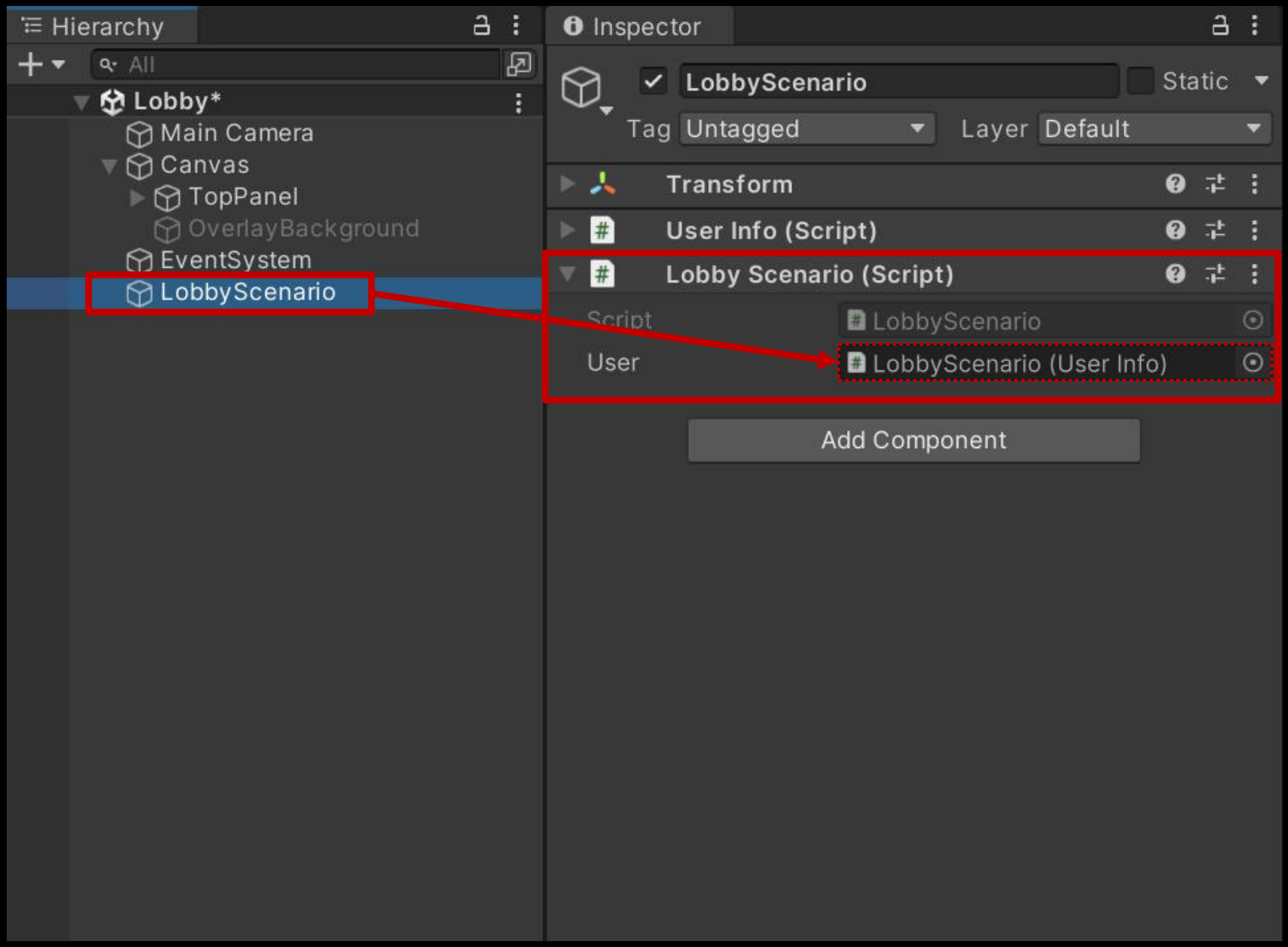
게임 유저 관리

- Lobby 씬에서 정보 갱신 등을 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "LobbyScenario"로 변경

```
1   using UnityEngine;
2
3   public class LobbyScenario : MonoBehaviour
4   {
5       [SerializeField]
6       private UserInfo user;
7
8       private void Awake()
9       {
10          user.GetUserInfoFromBackend();
11      }
12  }
```

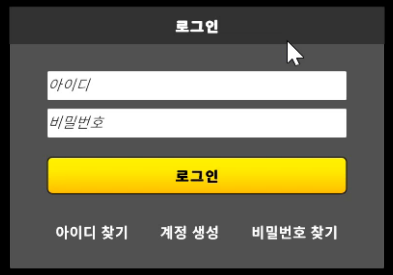
게임 유저 관리

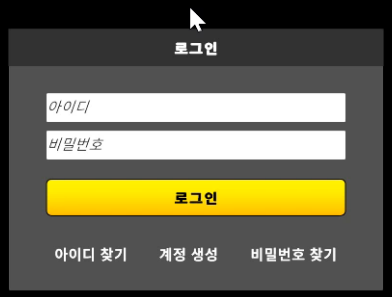
- LobbyScenario 오브젝트에 "LobbyScenario" 컴포넌트 추가 및 설정



게임 유저 관리

- 결과 화면

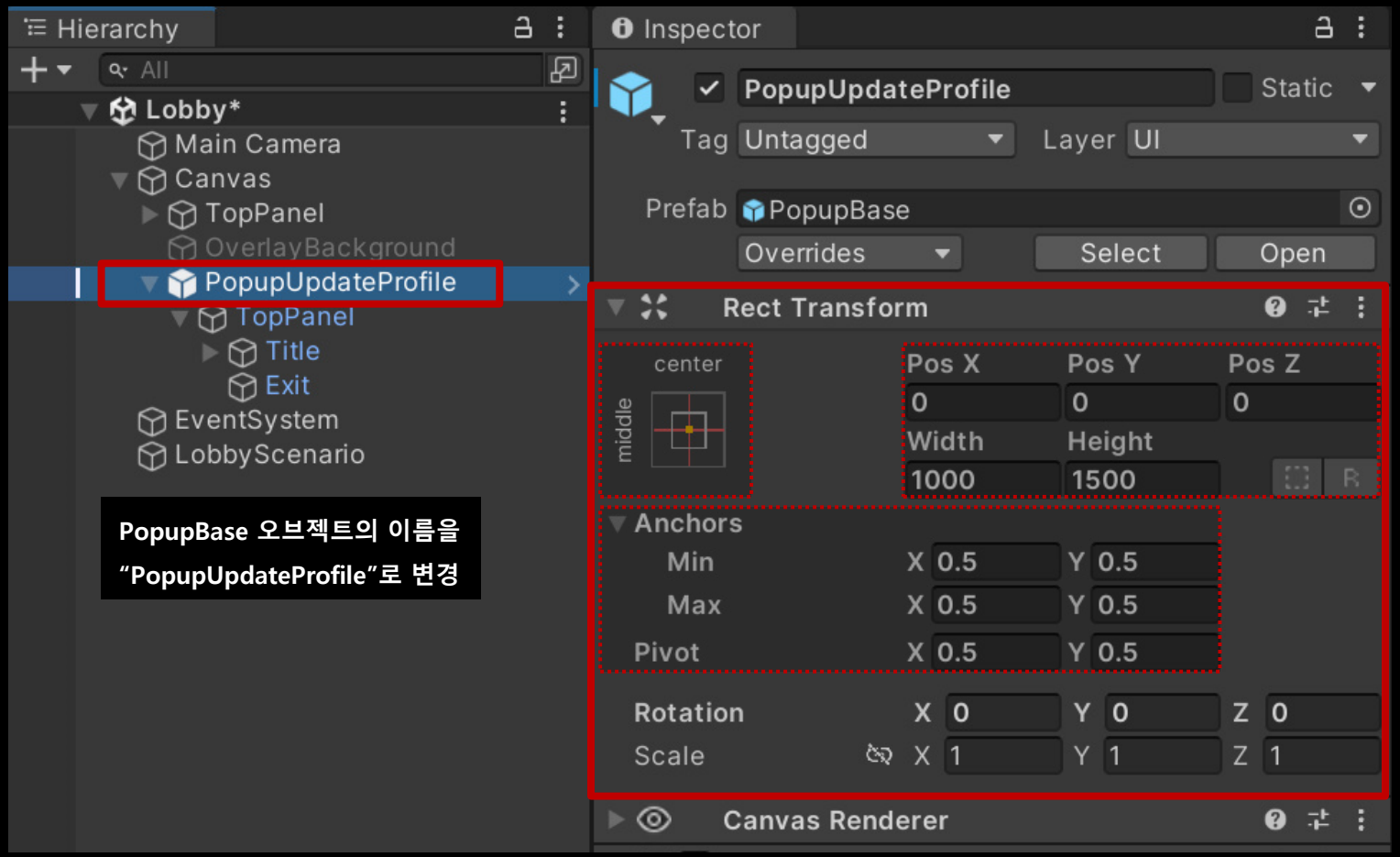

로그인
아이디
비밀번호
로그인
아이디 찾기
계정 생성
비밀번호 찾기


로그인
아이디
비밀번호
로그인
아이디 찾기
계정 생성
비밀번호 찾기

게임 유저 관리

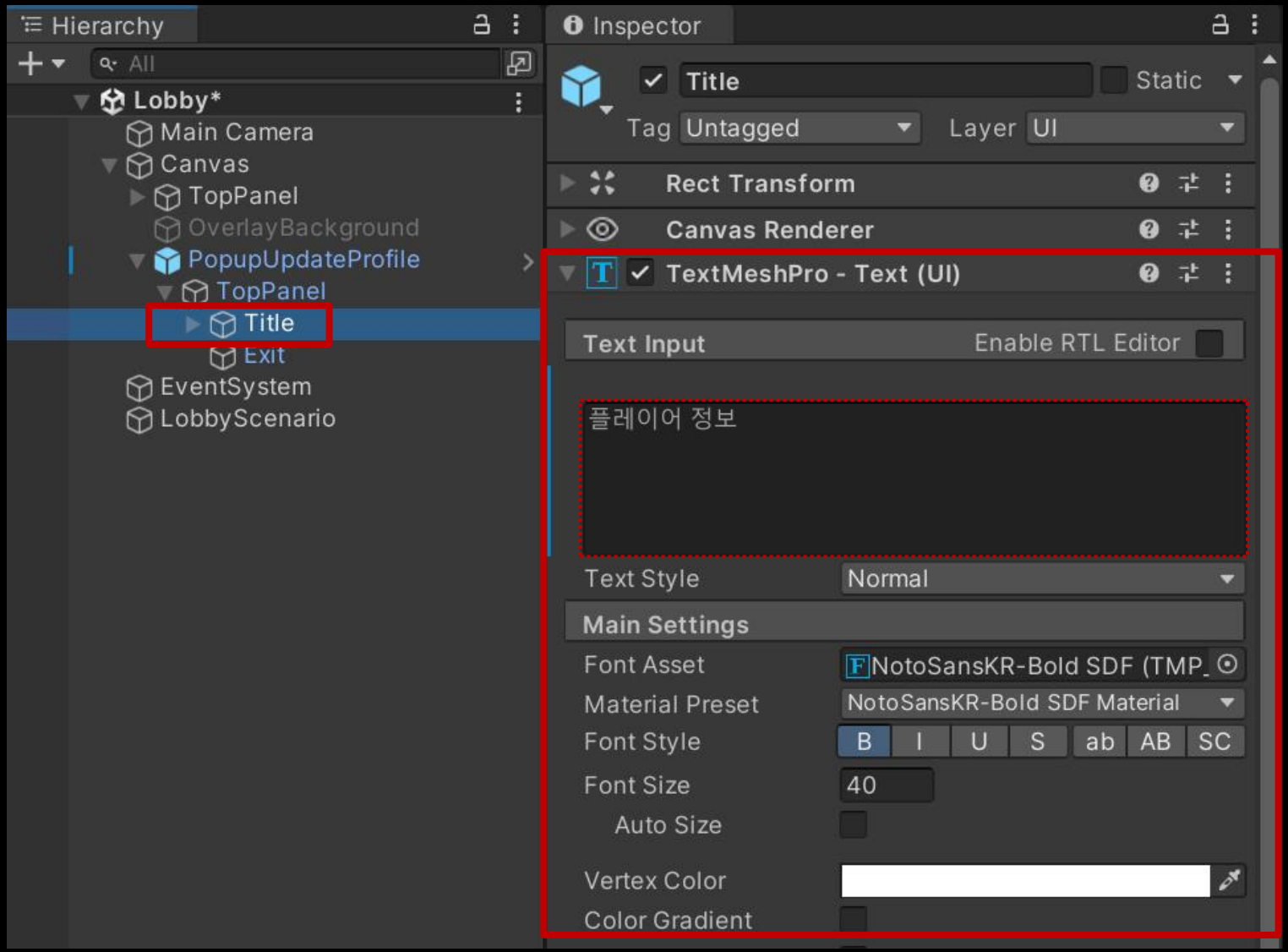
닉네임 설정

- 플레이어 정보 팝업 윈도우 생성 및 설정
  - PopupBase 프리팹을 Hierarchy View로 Drag & Drop



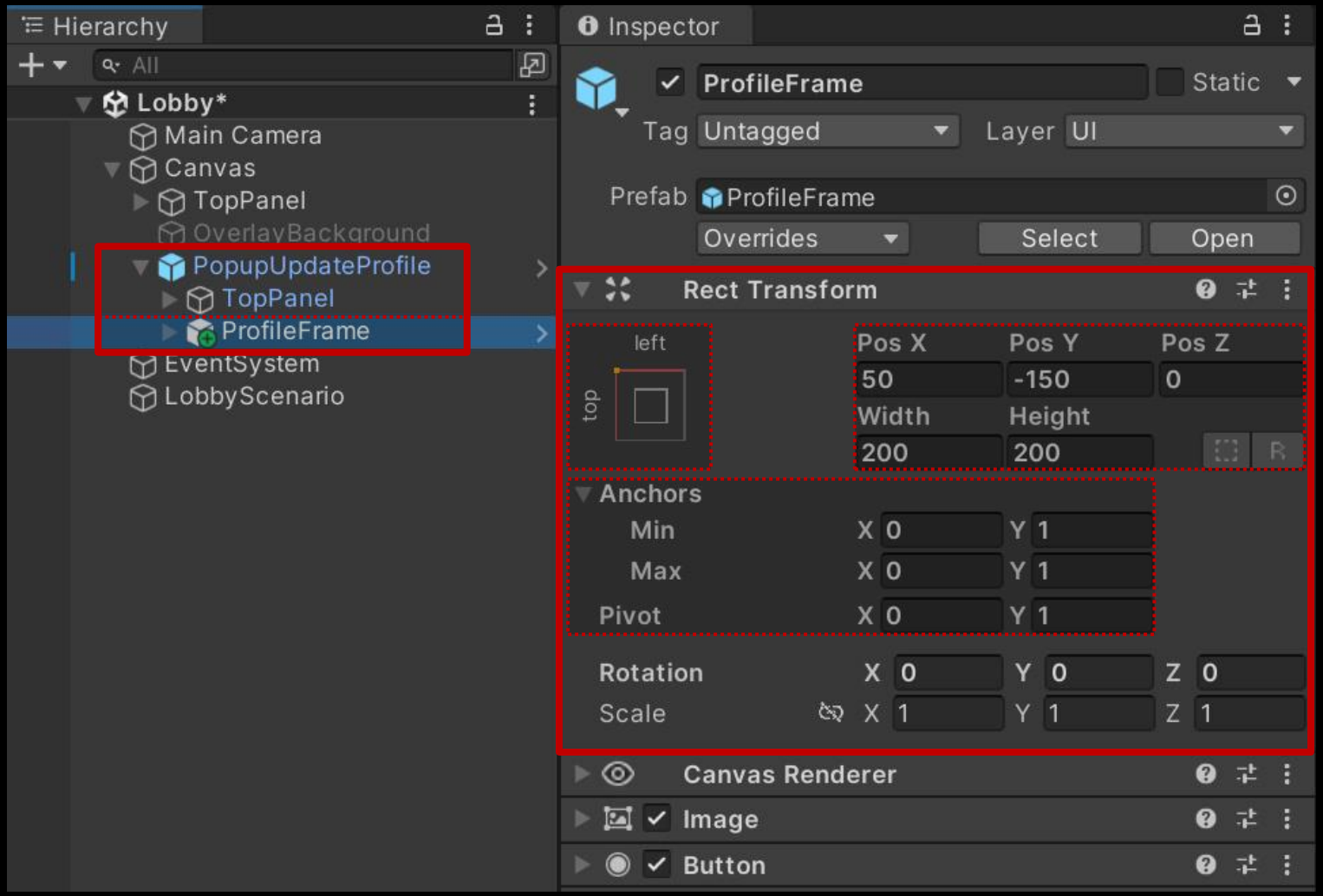
게임 유저 관리

- 플레이어 정보 팝업 윈도우 상단 이름 설정



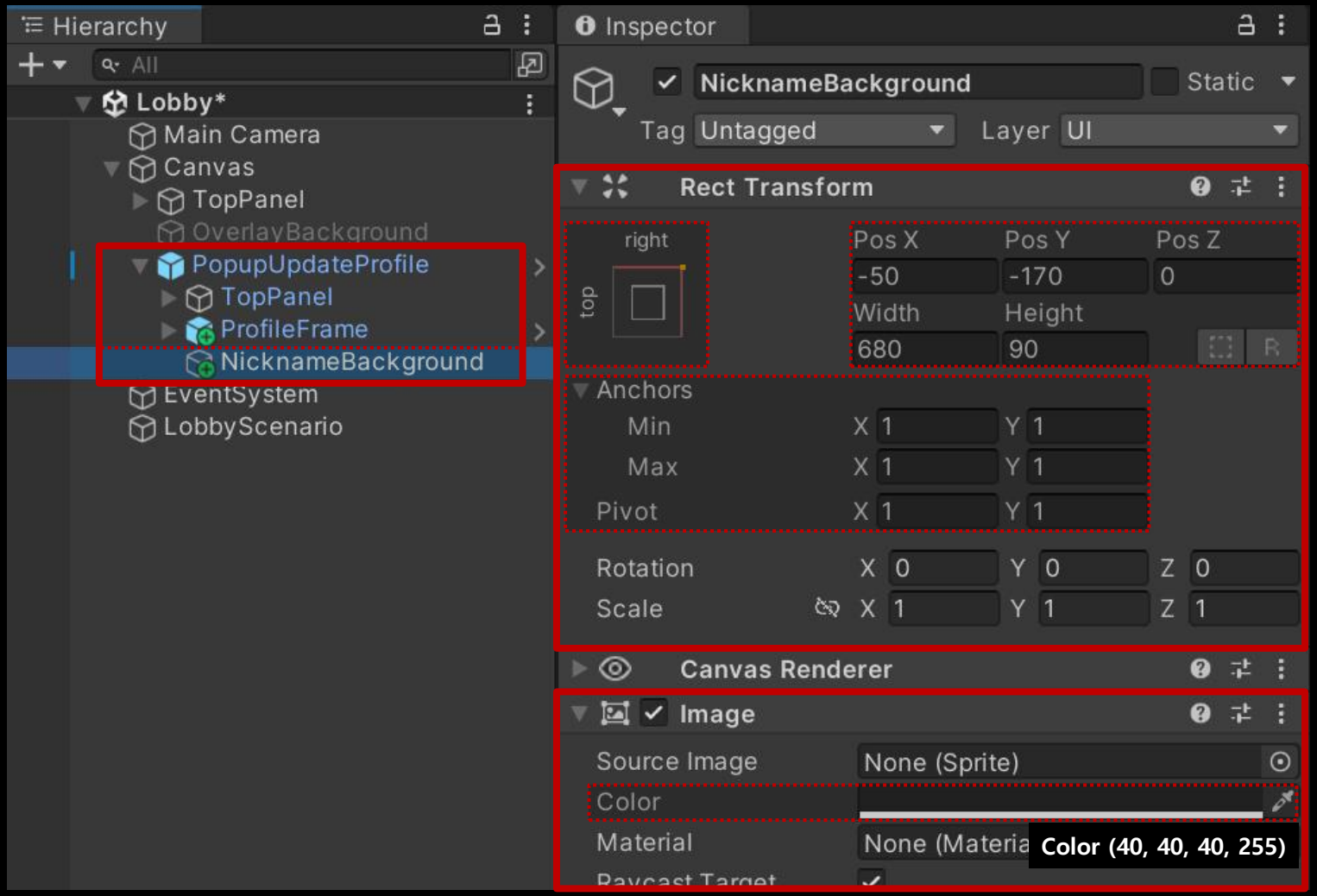
게임 유저 관리

- 프로필 아이콘 생성 및 설정
  - ProfileFrame 프리팹을 Hierarchy View로 Drag & Drop



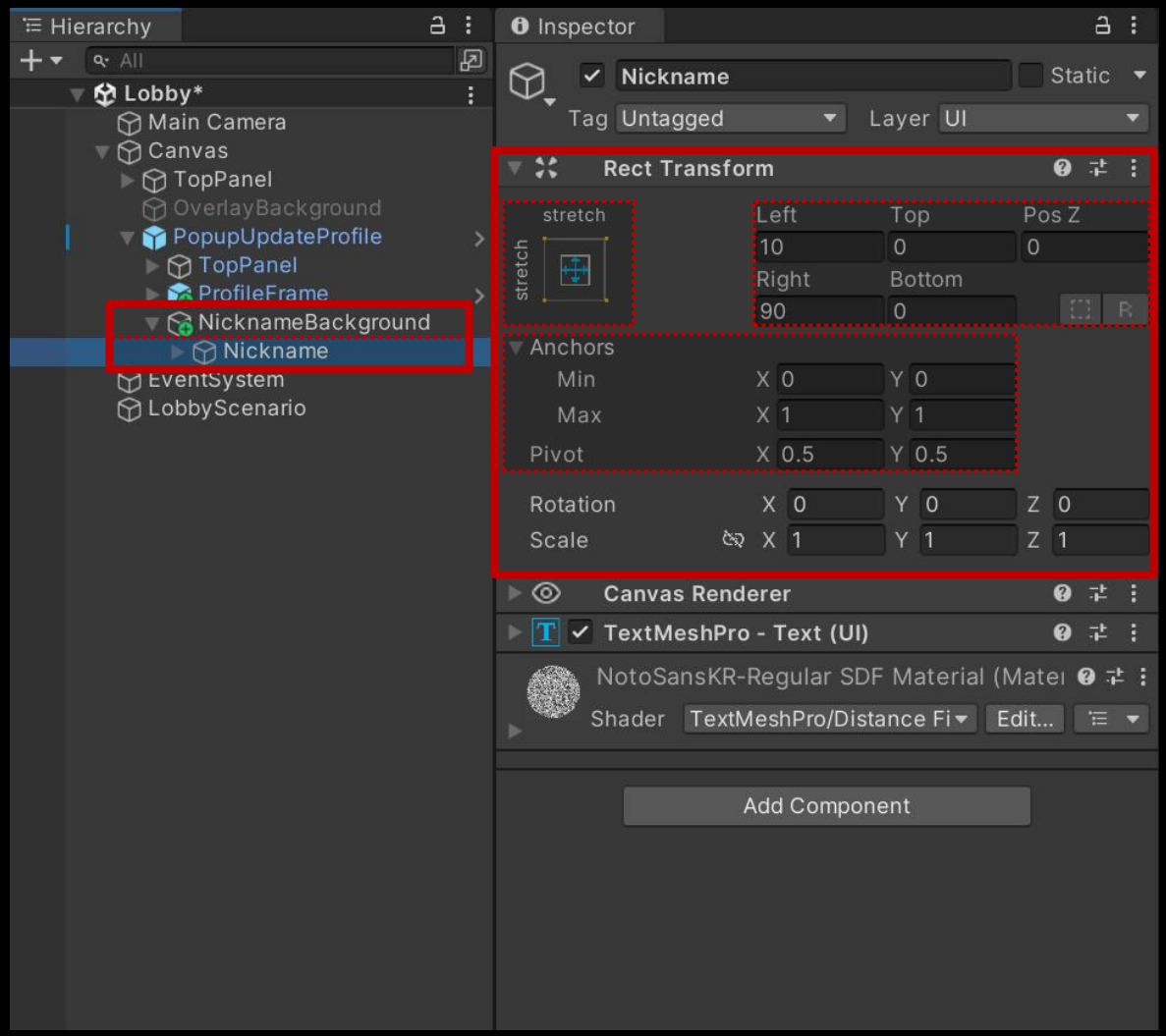
게임 유저 관리

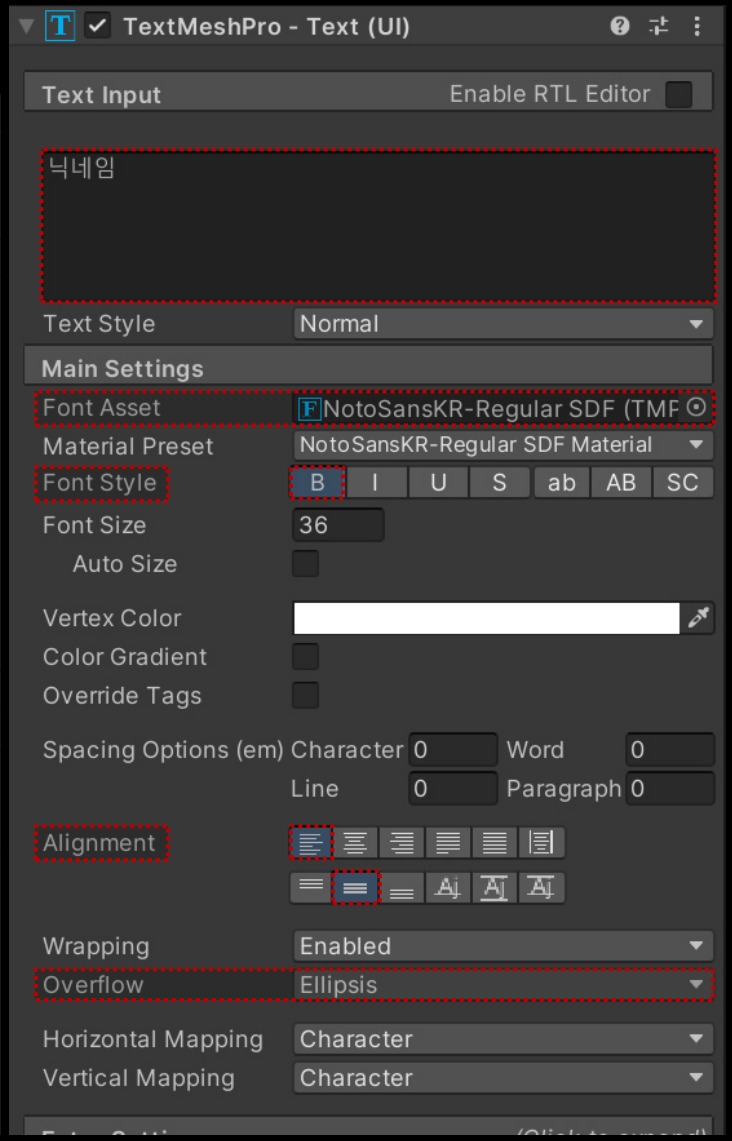
- 닉네임 배경 Image UI 생성 및 설정
  - GameObject - UI - Image



게임 유저 관리

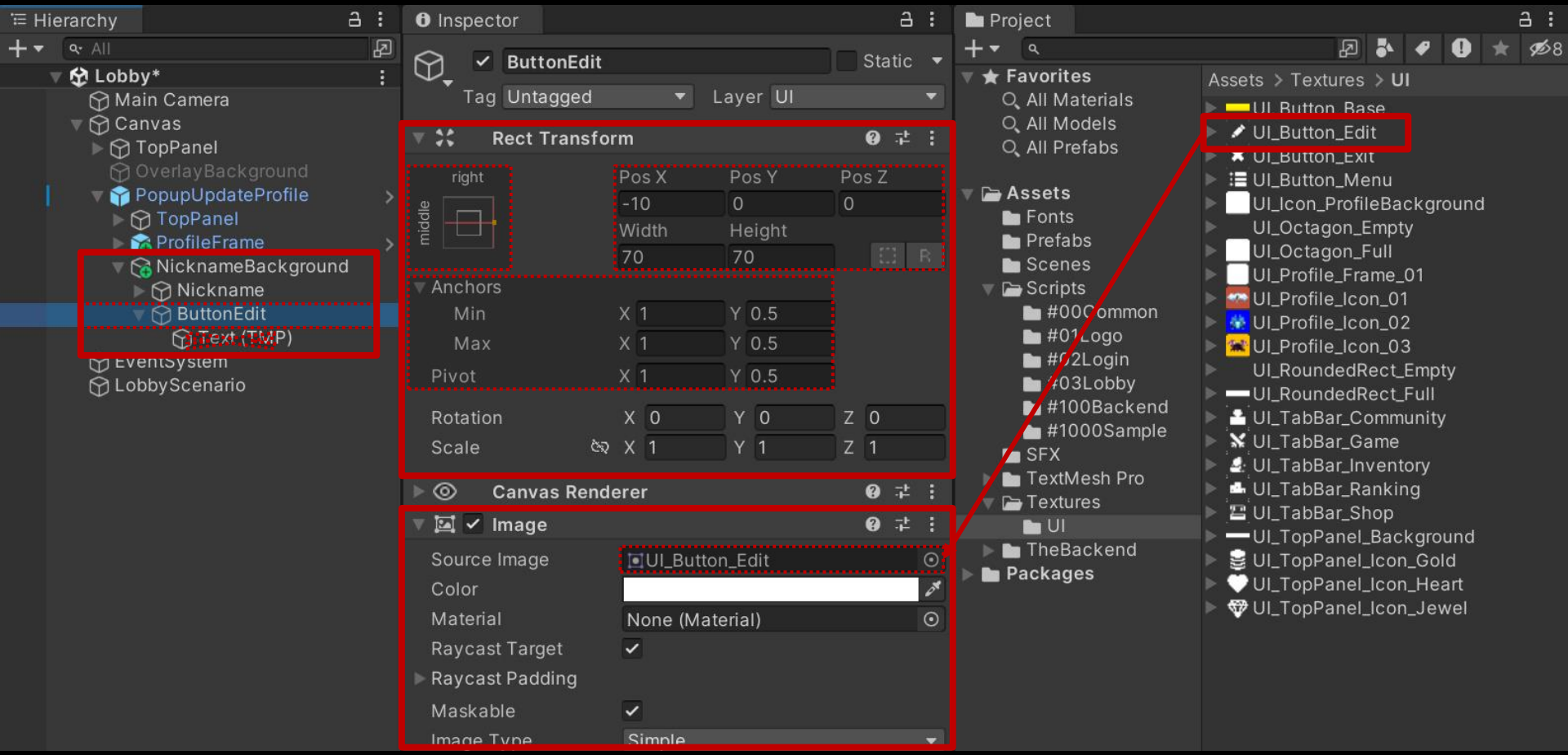
- 닉네임을 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"





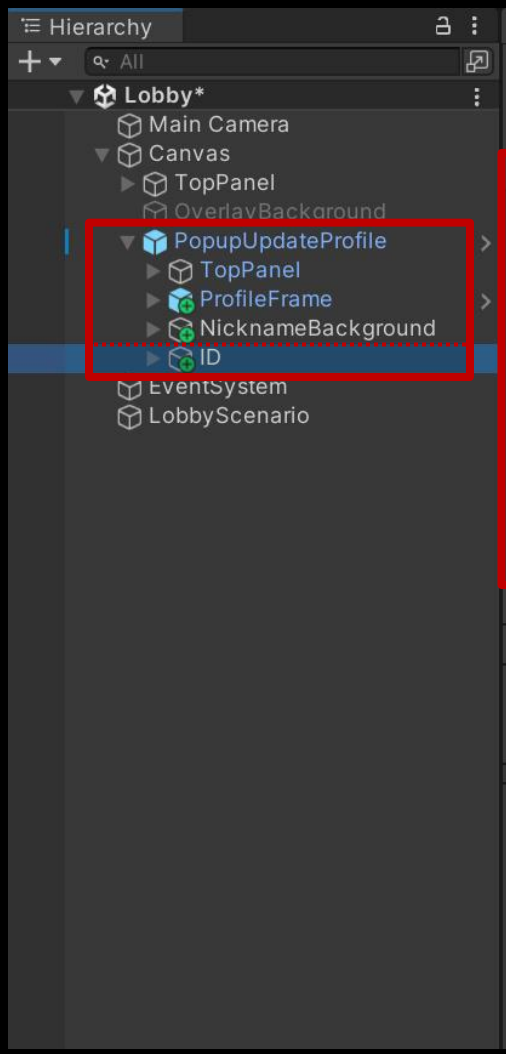
게임 유저 관리

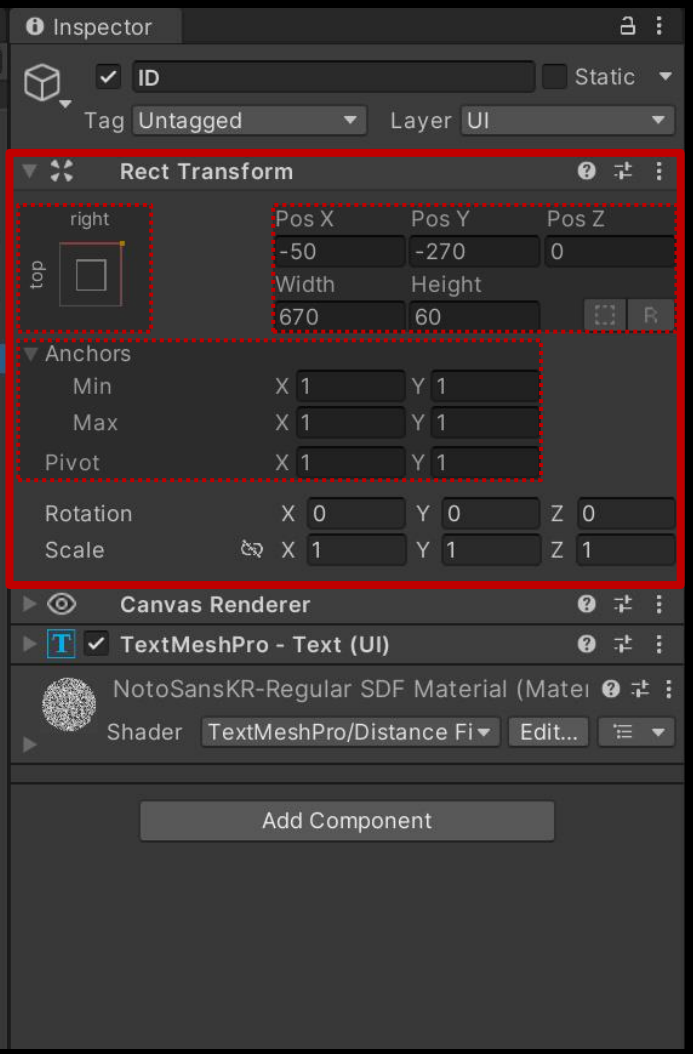
- 닉네임 설정 "Button - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Button - TextMeshPro"

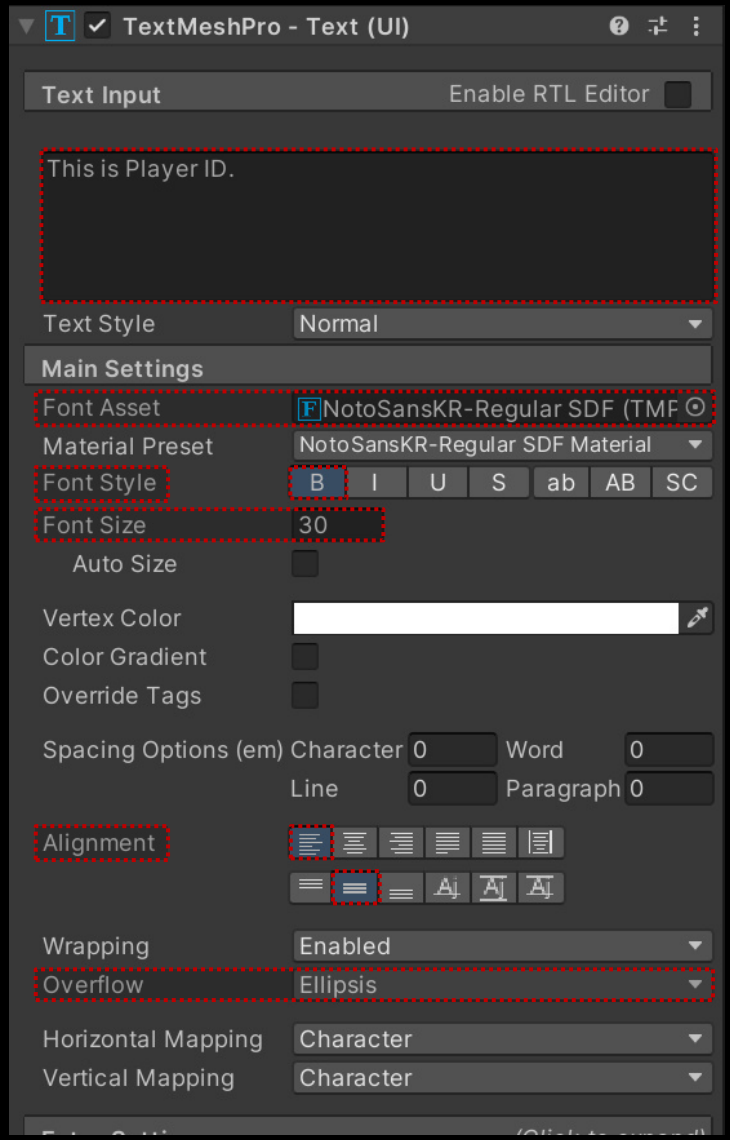


게임 유저 관리

- 아이디(gamer_id)를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"

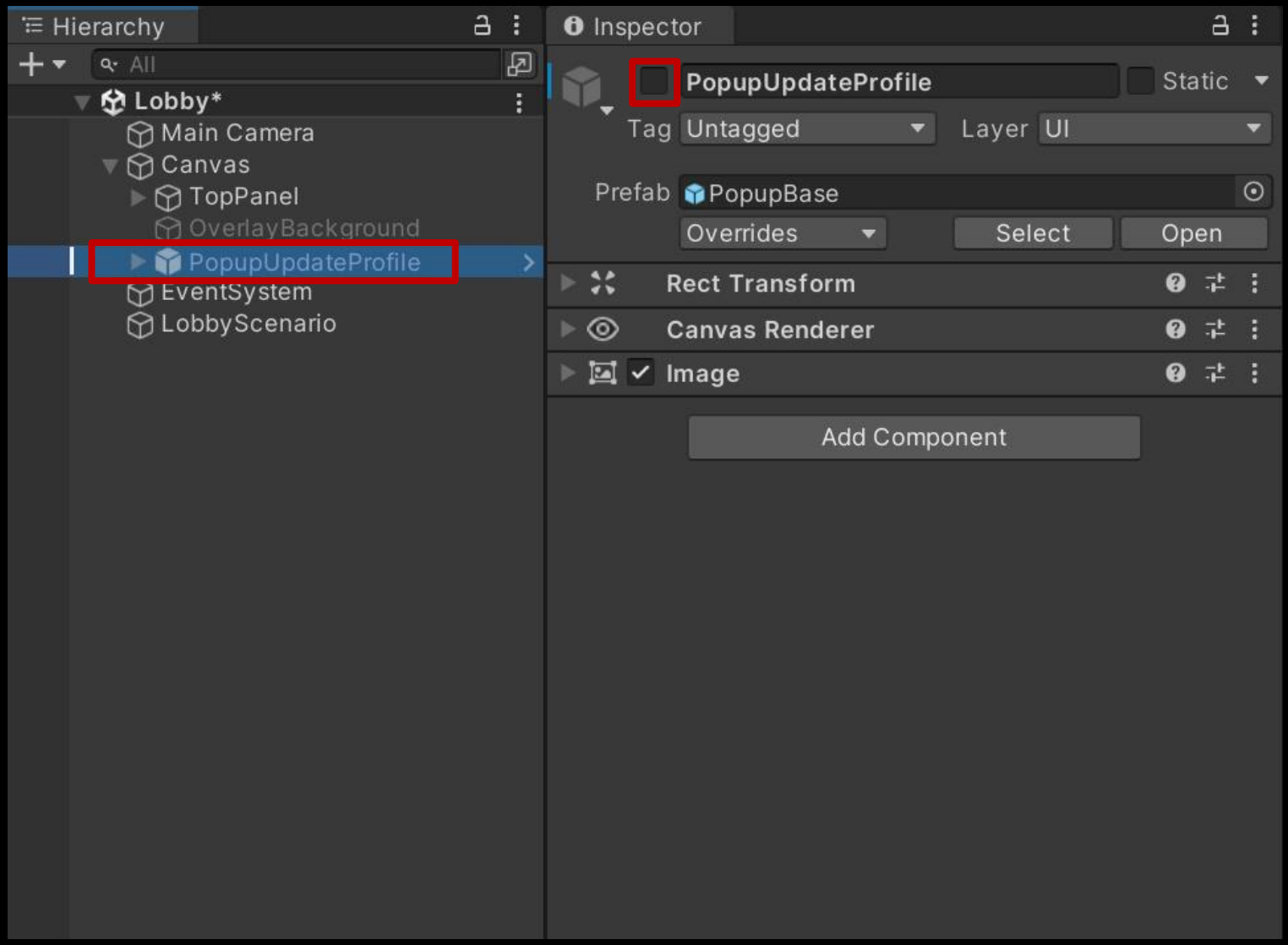






게임 유저 관리

- PopupUpdateProfile 오브젝트 비활성화



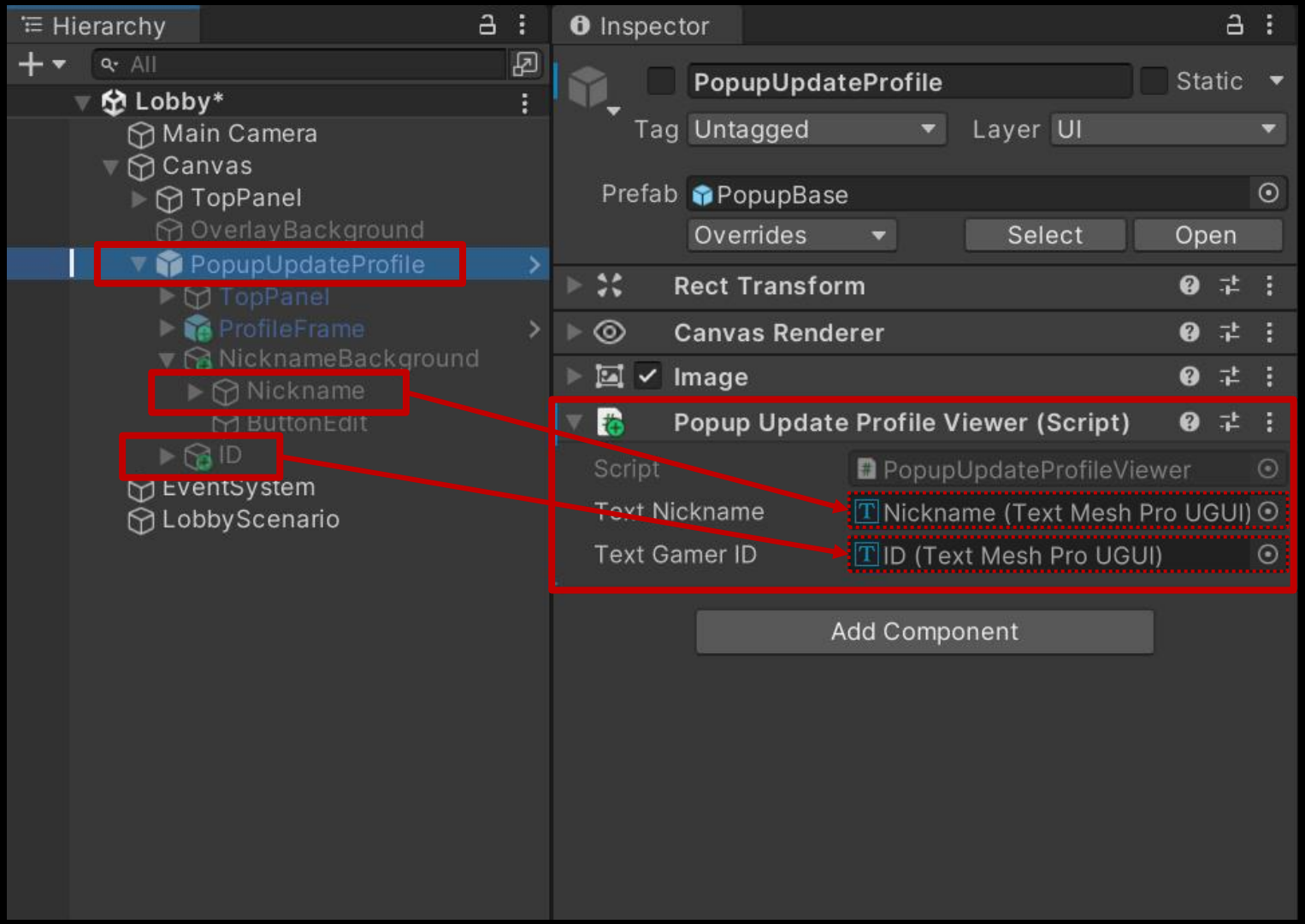
게임 유저 관리

- PopupUpdateProfile에 출력하는 UI 정보를 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "PopupUpdateProfileViewer"로 변경

```
1  using UnityEngine;
2  using TMPro;
3
4  public class PopupUpdateProfileViewer : MonoBehaviour
5  {
6      [SerializeField]
7      private TextMeshProUGUI textNickname;
8      [SerializeField]
9      private TextMeshProUGUI textGamerID;
10
11     public void UpdateNickname()
12     {
13         // 닉네임이 없으면 gamer_id를 출력하고, 닉네임이 있으면 닉네임 출력
14         textNickname.text = UserInfo.Data.nickname == null ?
15                             UserInfo.Data.gamerId : UserInfo.Data.nickname;
16
17         // gamer_id 출력
18         textGamerID.text = UserInfo.Data.gamerId;
19     }
20 }
```

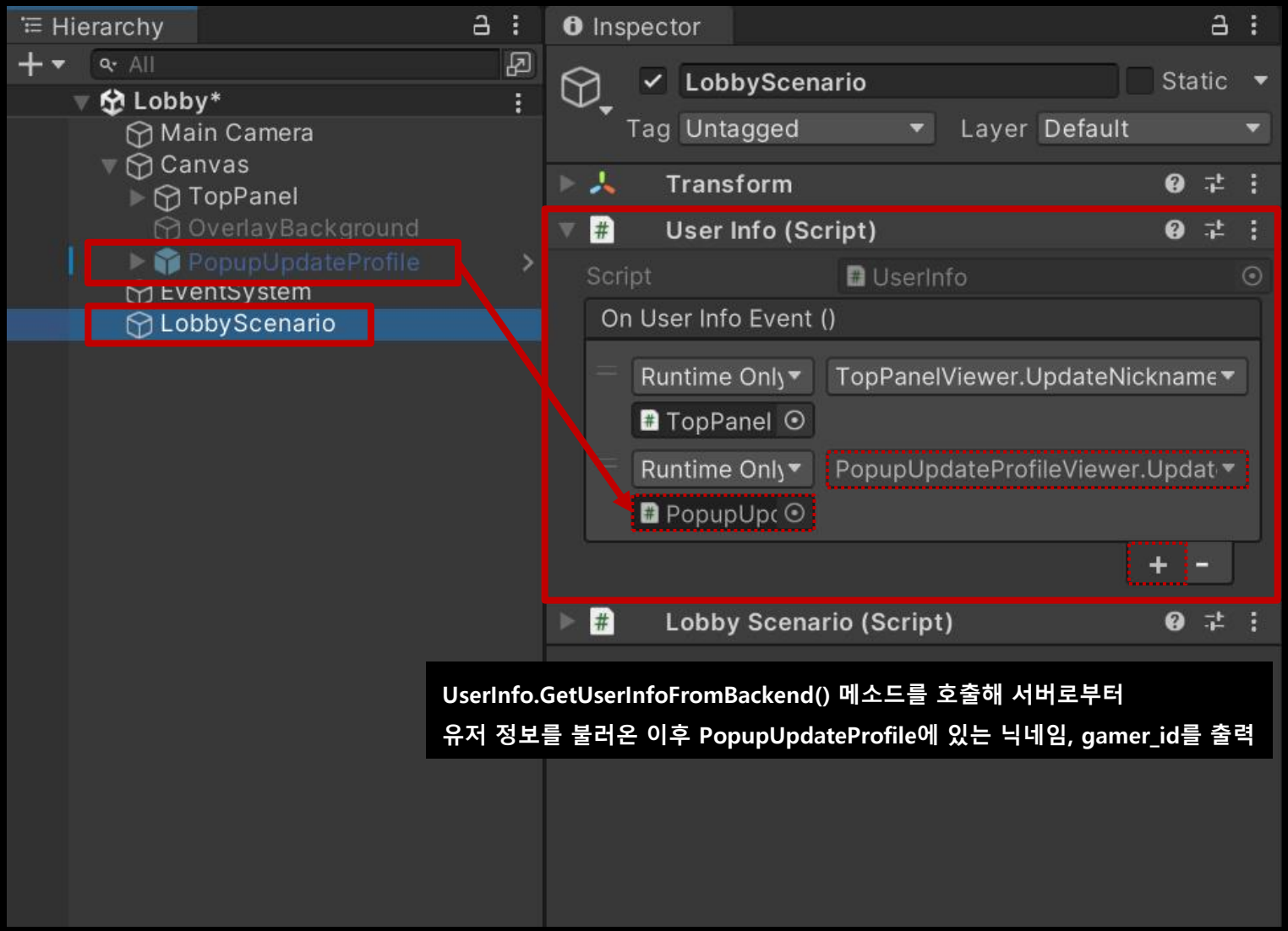
게임 유저 관리

- PopupUpdateProfile 오브젝트에 컴포넌트 추가 및 설정



게임 유저 관리

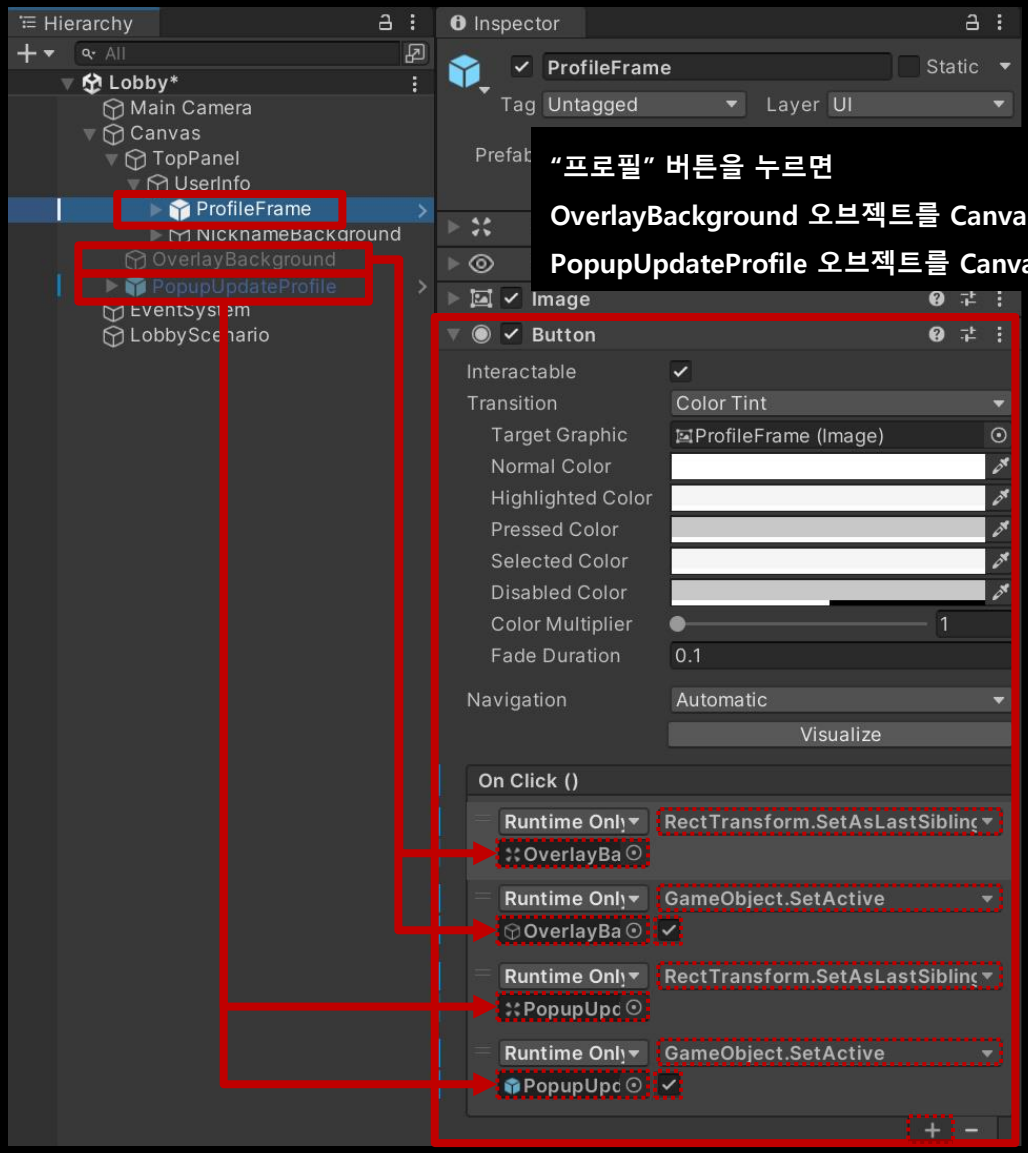
- LobbyScenario 오브젝트의 "UserInfo" 컴포넌트 변수 설정



UserInfo.GetUserInfoFromBackend() 메소드를 호출해 서버로부터
유저 정보를 불러온 이후 PopupUpdateProfile에 있는 닉네임, gamer_id를 출력

게임 유저 관리

- ProfileFrame 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



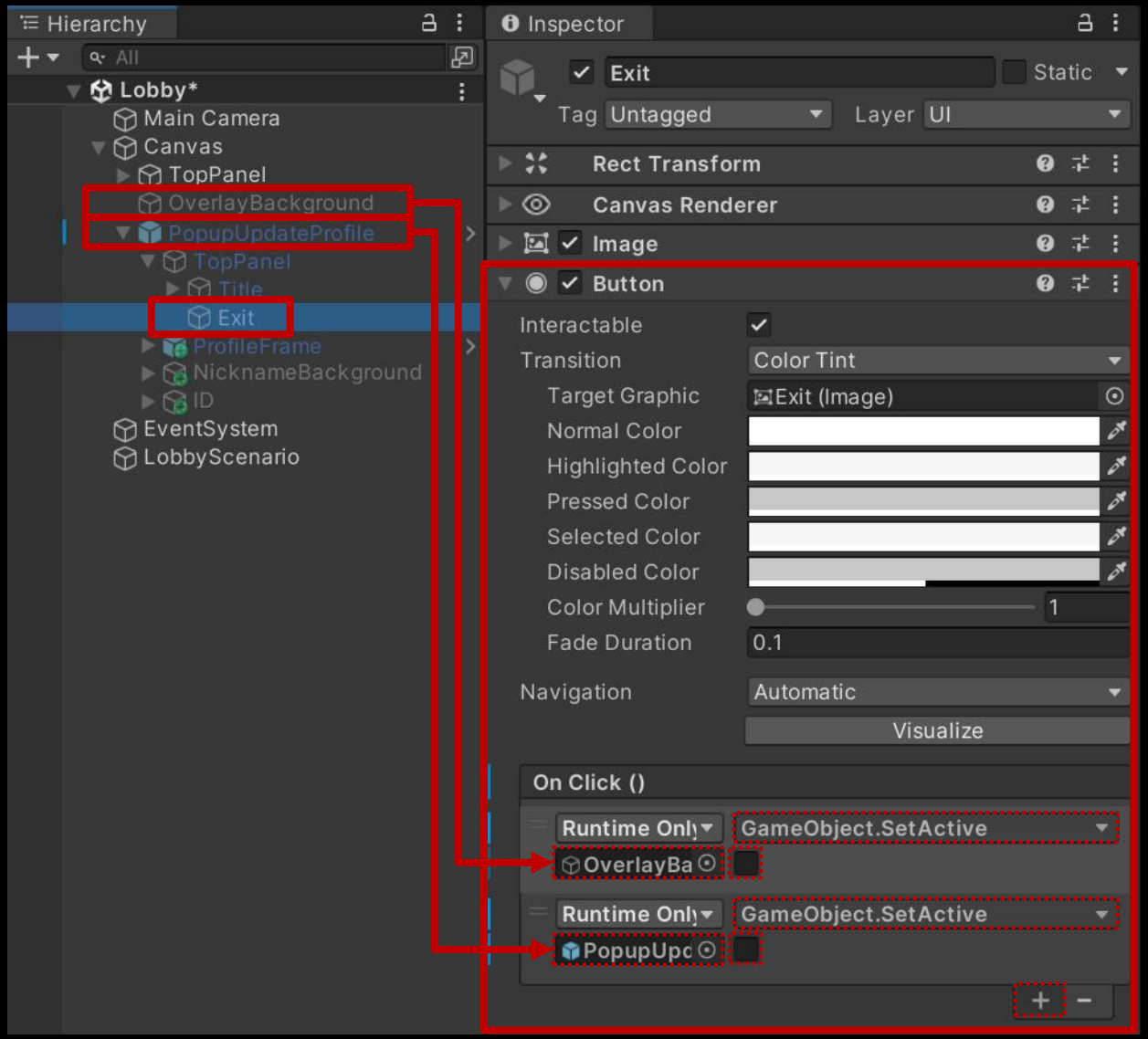
"프로필" 버튼을 누르면

OverlayBackground 오브젝트를 Canvas의 마지막 자식으로 설정한 후 활성화하고,

PopupUpdateProfile 오브젝트를 Canvas의 마지막 자식으로 설정한 후 활성화한다.

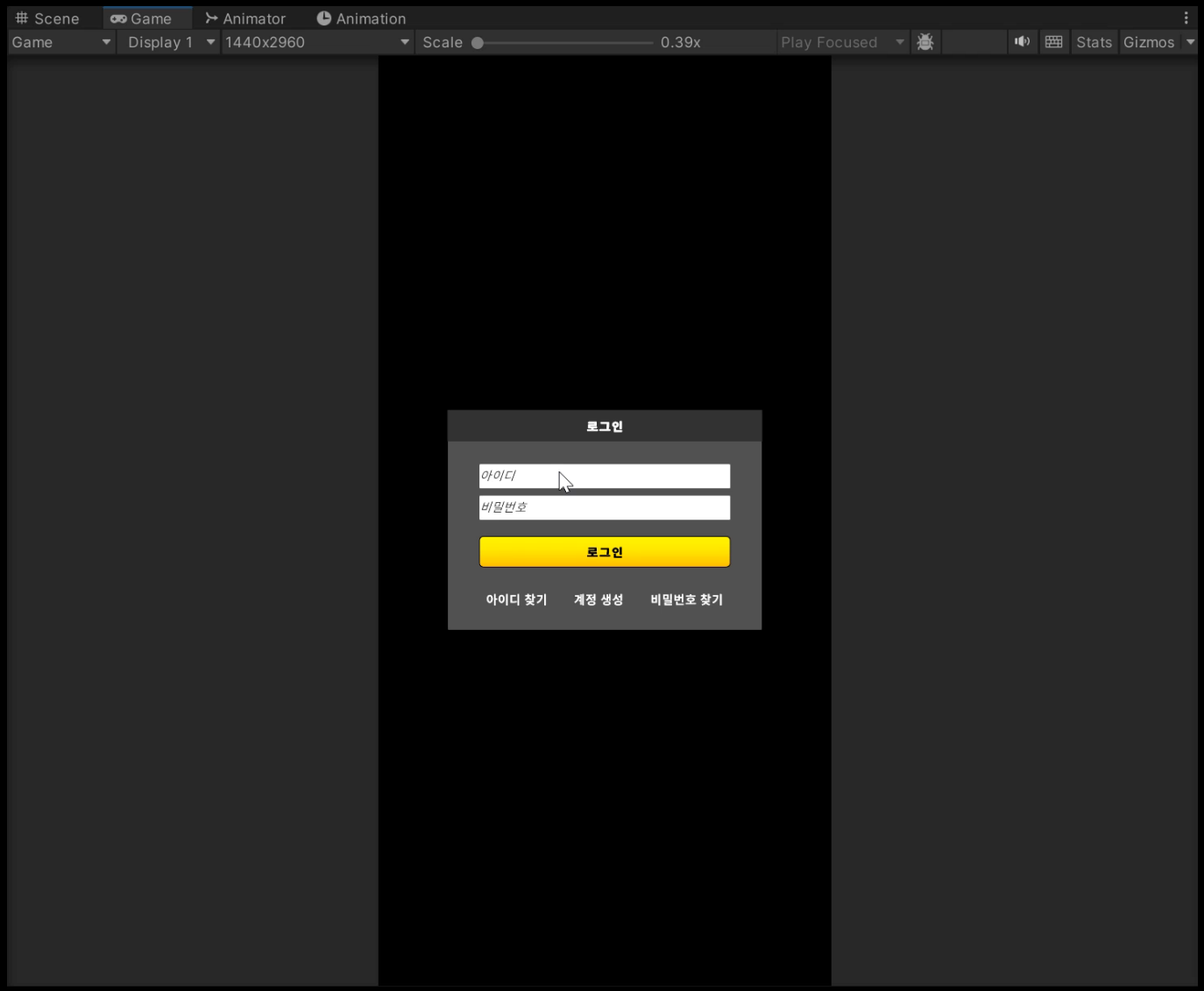
게임 유저 관리

- Exit 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



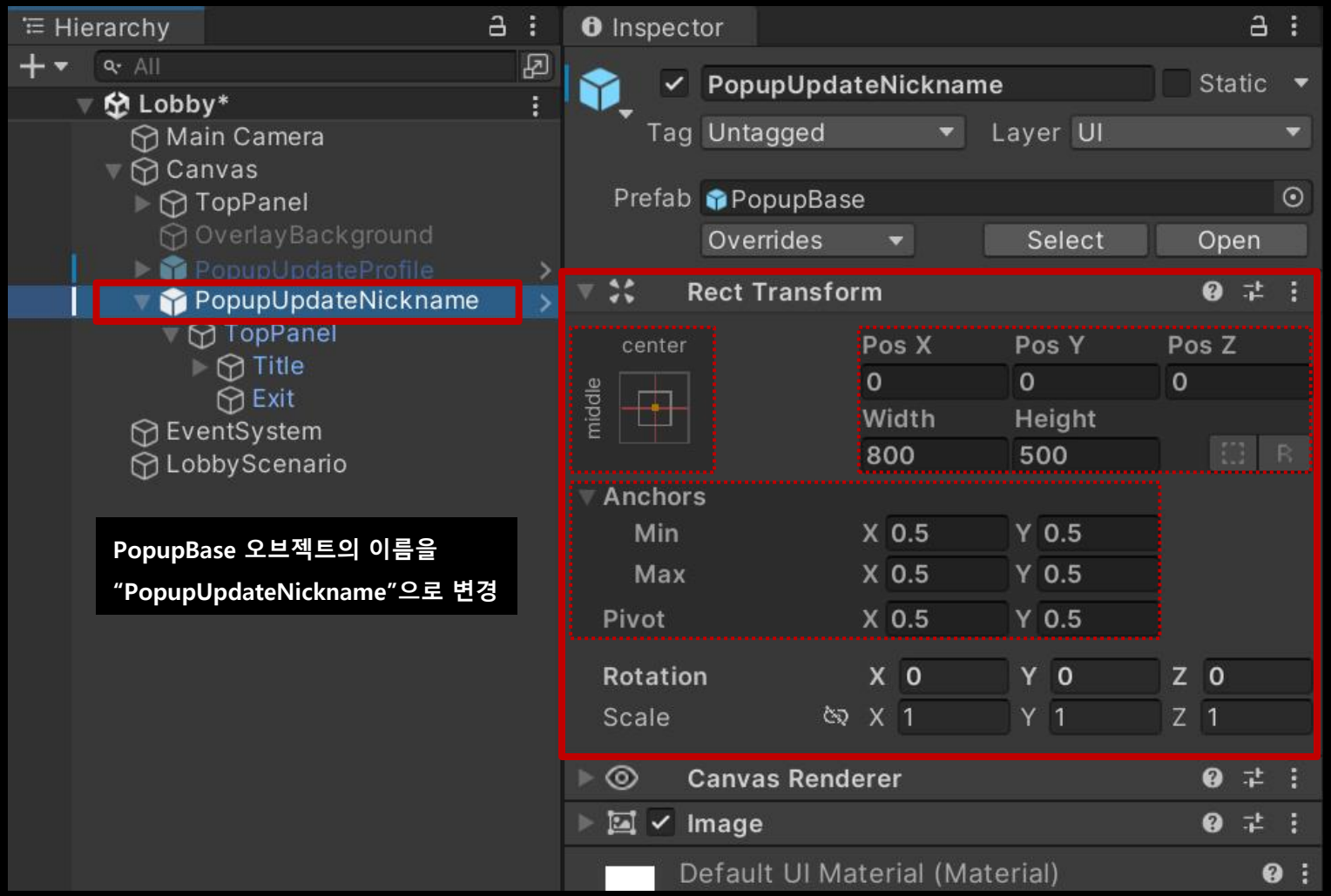
게임 유저 관리

- 결과 화면


Scene
Game
Animator
Animation
Game
Display 1
1440x2960
Scale
0.39x
Play Focused
Stats
Gizmos
로그인
아이디
비밀번호
로그인
아이디 찾기
계정 생성
비밀번호 찾기

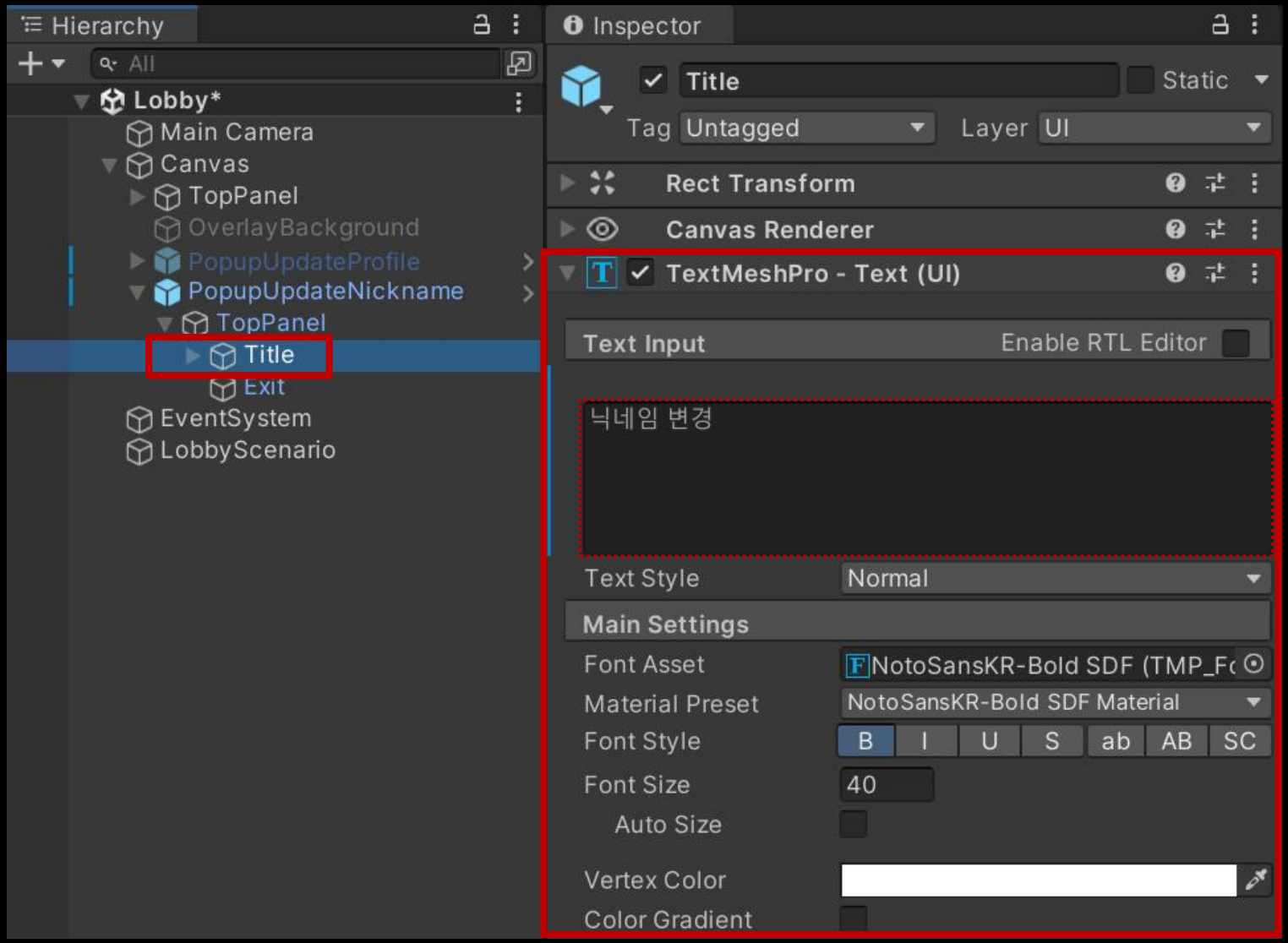
게임 유저 관리

- 닉네임 변경 팝업 윈도우 생성 및 설정
  - PopupBase 프리팹을 Hierarchy View로 Drag & Drop



게임 유저 관리

- 닉네임 변경 팝업 윈도우 상단 이름 설정



게임 유저 관리

- "닉네임을.." 텍스트를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"

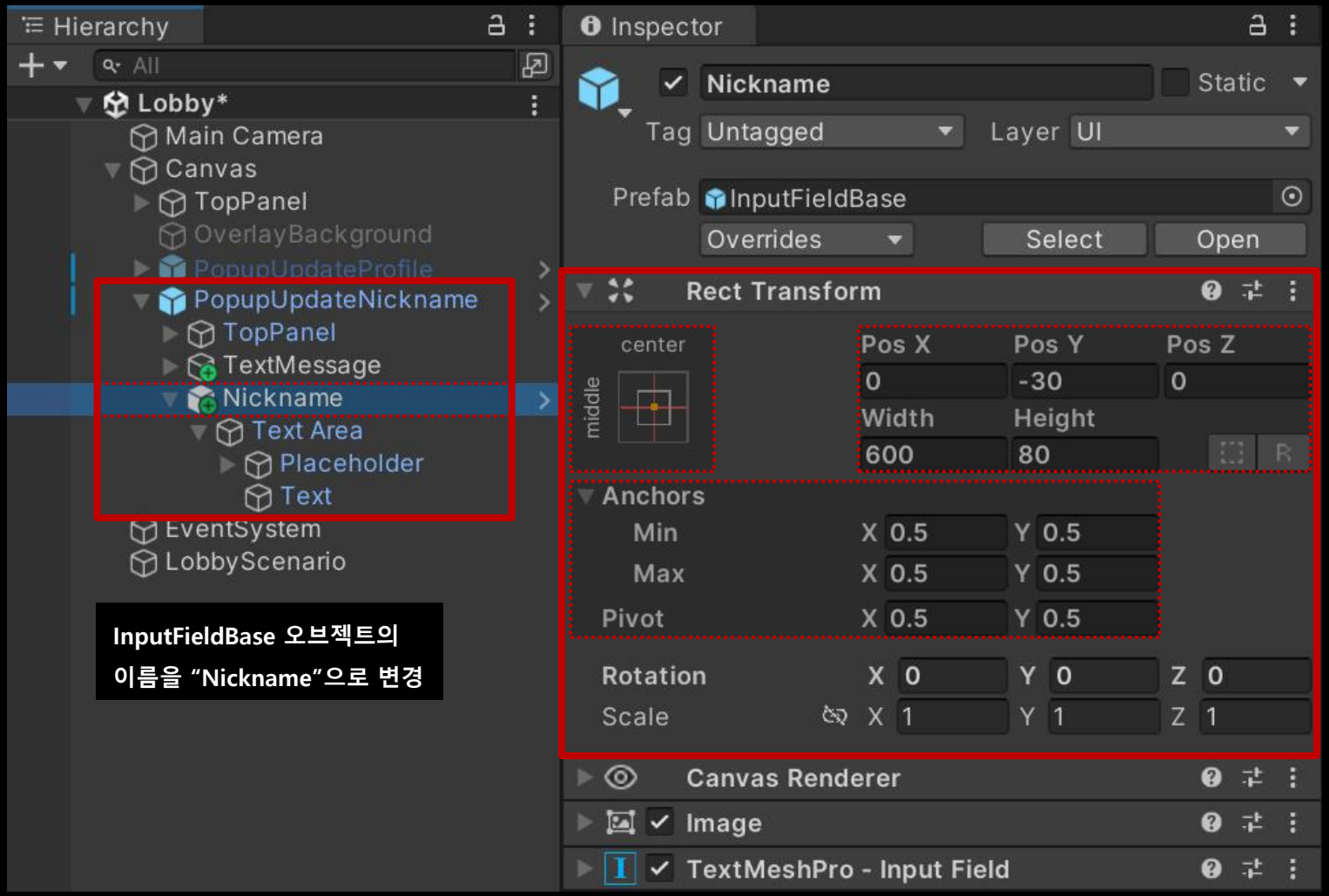
Hierarchy
All
Lobby*
Main Camera
Canvas
TopPanel
OverlayBackground
PopupUpdateProfile
PopupUpdateNickname
TopPanel
TextMessage
EventSystem
LobbyScenario

Inspector
TextMessage
Static
Tag Untagged
Layer UI
Rect Transform
stretch
middle
Left 0
Pos Y 80
Pos Z 0
Right 0
Height 100
Anchors
Min X 0 Y 0.5
Max X 1 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
TextMeshPro - Text (UI)
NotoSansKR-Bold SDF Material (Material)
Shader TextMeshPro/Distance Fi
Edit...
Add Component

TextMeshPro - Text (UI)
Text Input
Enable RTL Editor
닉네임을 입력하세요
Text Style Normal
Main Settings
Font Asset NotoSansKR-Bold SDF (TMP_Fc
Material Preset NotoSansKR-Bold SDF Material
Font Style B I U S ab AB SC
Font Size 36
Auto Size
Vertex Color
Color Gradient
Override Tags
Spacing Options (em) Character 0 Word 0
Line 0 Paragraph 0
Alignment
Wrapping Enabled
Overflow Overflow
Horizontal Mapping Character
Vertical Mapping Character

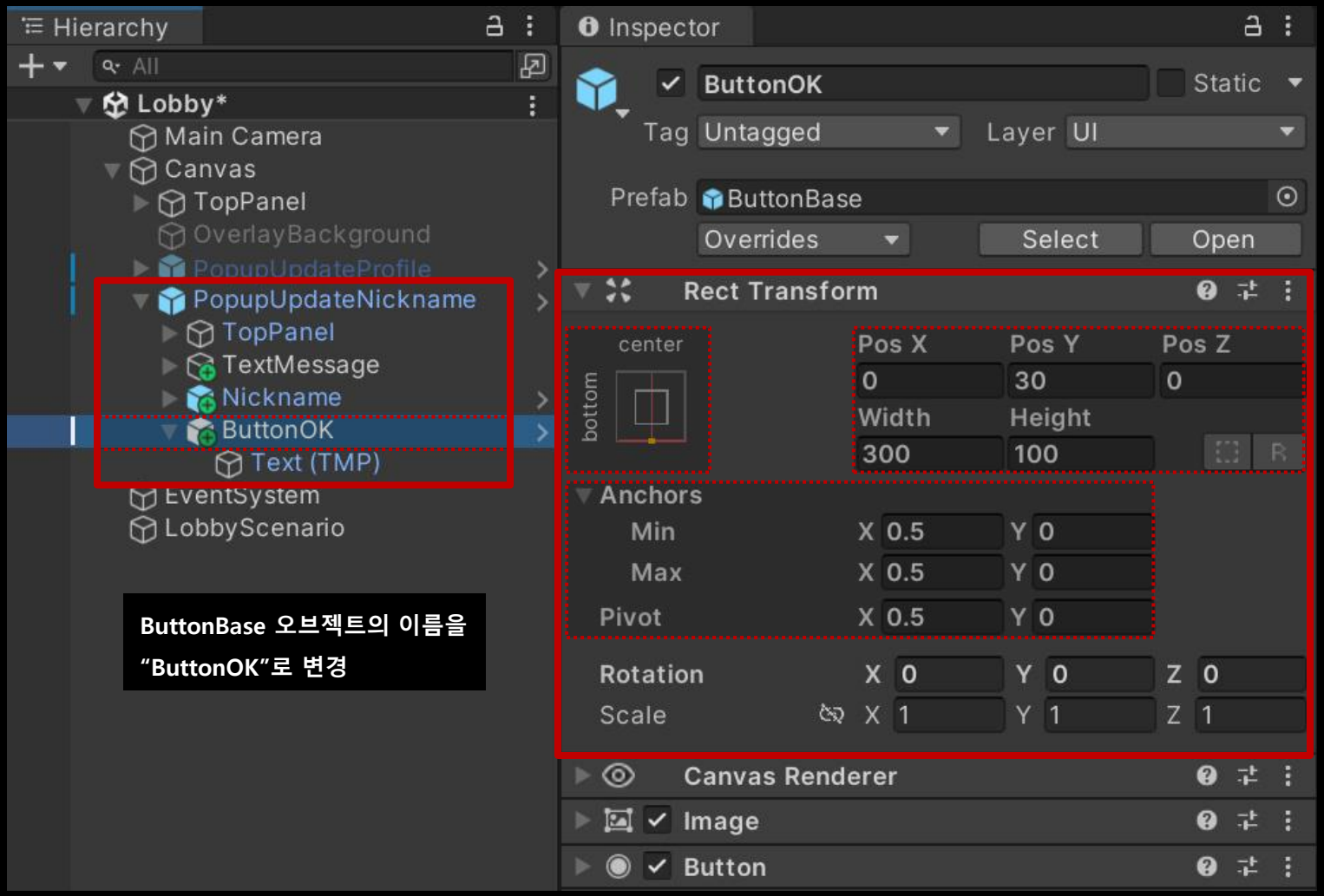
게임 유저 관리

- 닉네임 입력 필드 생성 및 설정
  - InputFieldBase 프리팹을 Hierarchy View로 Drag & Drop



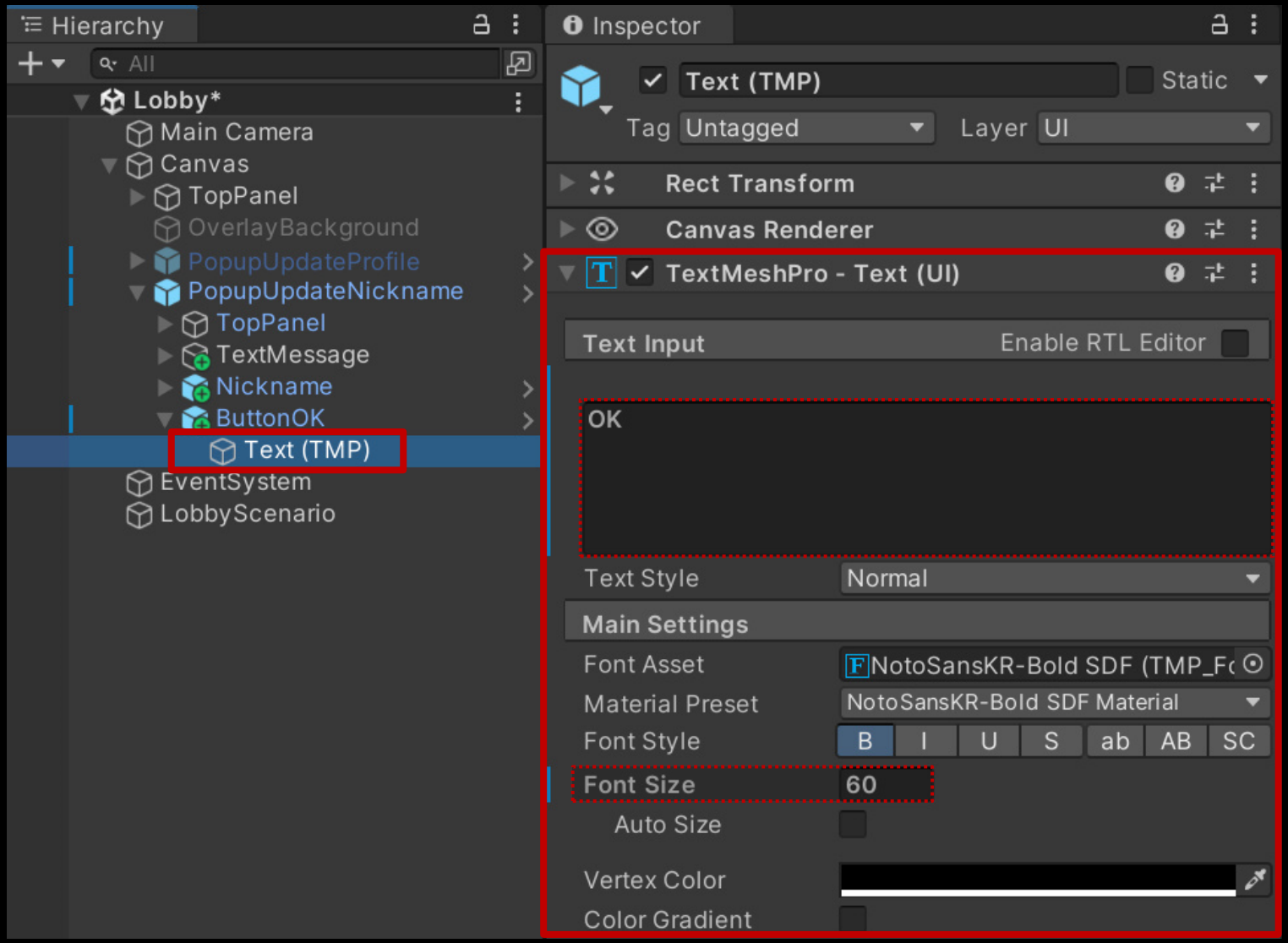
게임 유저 관리

- "닉네임 변경" 버튼 생성 및 설정
  - ButtonBase 프리팹을 Hierarchy View로 Drag & Drop



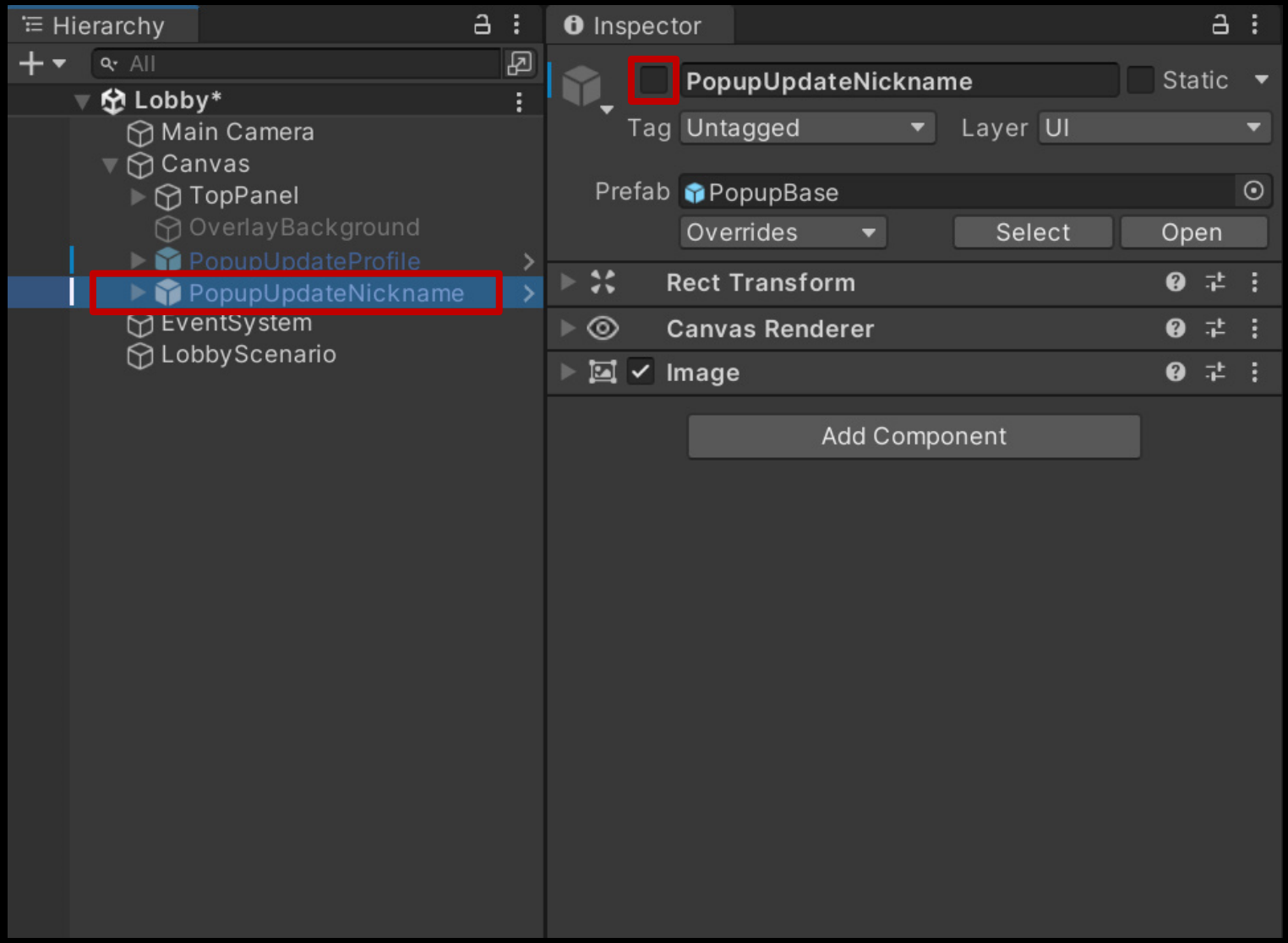
게임 유저 관리

- "닉네임 변경" 버튼 생성 및 설정 (계속)



게임 유저 관리

- PopupUpdateNickname 오브젝트 비활성화



게임 유저 관리

- 뒤끝 서버와 연동해 닉네임을 변경하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "Nickname"으로 변경

```
1  using UnityEngine;
2  using UnityEngine.UI;
3  using TMPro;
4  using BackEnd;
5
6  public class Nickname : LoginBase
7  {
8      [System.Serializable]
9      public class NicknameEvent : UnityEngine.Events.UnityEvent { }
10     public NicknameEvent        onNicknameEvent = new NicknameEvent();
11
12     [SerializeField]
13     private Image               imageNickname;      // 닉네임 필드 색상 변경
14     [SerializeField]
15     private TMP_InputField      inputFieldNickname; // 닉네임 필드 텍스트 정보 추출
16
17     [SerializeField]
18     private Button              btnUpdateNickname;  // "닉네임 설정" 버튼 (상호작용 가능/불가능)
19
```

게임 유저 관리

- 뒤끝 서버와 연동해 닉네임을 변경하는 스크립트 생성 및 작성 (계속)

```
20      private void OnEnable()
21      {
22          // 닉네임 변경에 실패해 에러 메시지를 출력한 상태에서
23          // 닉네임 변경 팝업을 닫았다가 열 수 있기 때문에 상태를 초기화
24          ResetUI(imageNickname);
25          SetMessage("닉네임을 입력하세요");
26      }
27
28      public void OnClickUpdateNickname()
29      {
30          // 매개변수로 입력한 InputField UI의 색상과 Message 내용 초기화
31          ResetUI(imageNickname);
32
33          // 필드 값이 비어있는지 체크
34          if ( IsFieldDataEmpty(imageNickname, inputFieldNickname.text, "Nickname") ) return;
35
36          // "닉네임 변경" 버튼의 상호작용 비활성화
37          btnUpdateNickname.interactable = false;
38          SetMessage("닉네임 변경중입니다..");
39
40          // 뒤끝 서버 닉네임 변경 시도
41          UpdateNickname();
42      }
43
```

게임 유저 관리

- 뒤끝 서버와 연동해 닉네임을 변경하는 스크립트 생성 및 작성 (계속)

```
44      private void UpdateNickname()
45      {
46          // 닉네임 설정
47          Backend.BMember.UpdateNickname(inputFieldNickname.text, callback =>
48          {
49              // "닉네임 변경" 버튼의 상호작용 활성화
50              btnUpdateNickname.interactable = true;
51
52              // 닉네임 변경 성공
53              if ( callback.IsSuccess() )
54              {
55                  SetMessage($"{inputFieldNickname.text}(으)로 닉네임이 변경되었습니다.");
56
57                  // 닉네임 변경에 성공했을 때 onNicknameEvent에 등록되어 있는 이벤트 메소드 호출
58                  onNicknameEvent?.Invoke();
59              }
60              // 닉네임 변경 실패
61              else...
80          });
81      }
82  }
```

뒷장

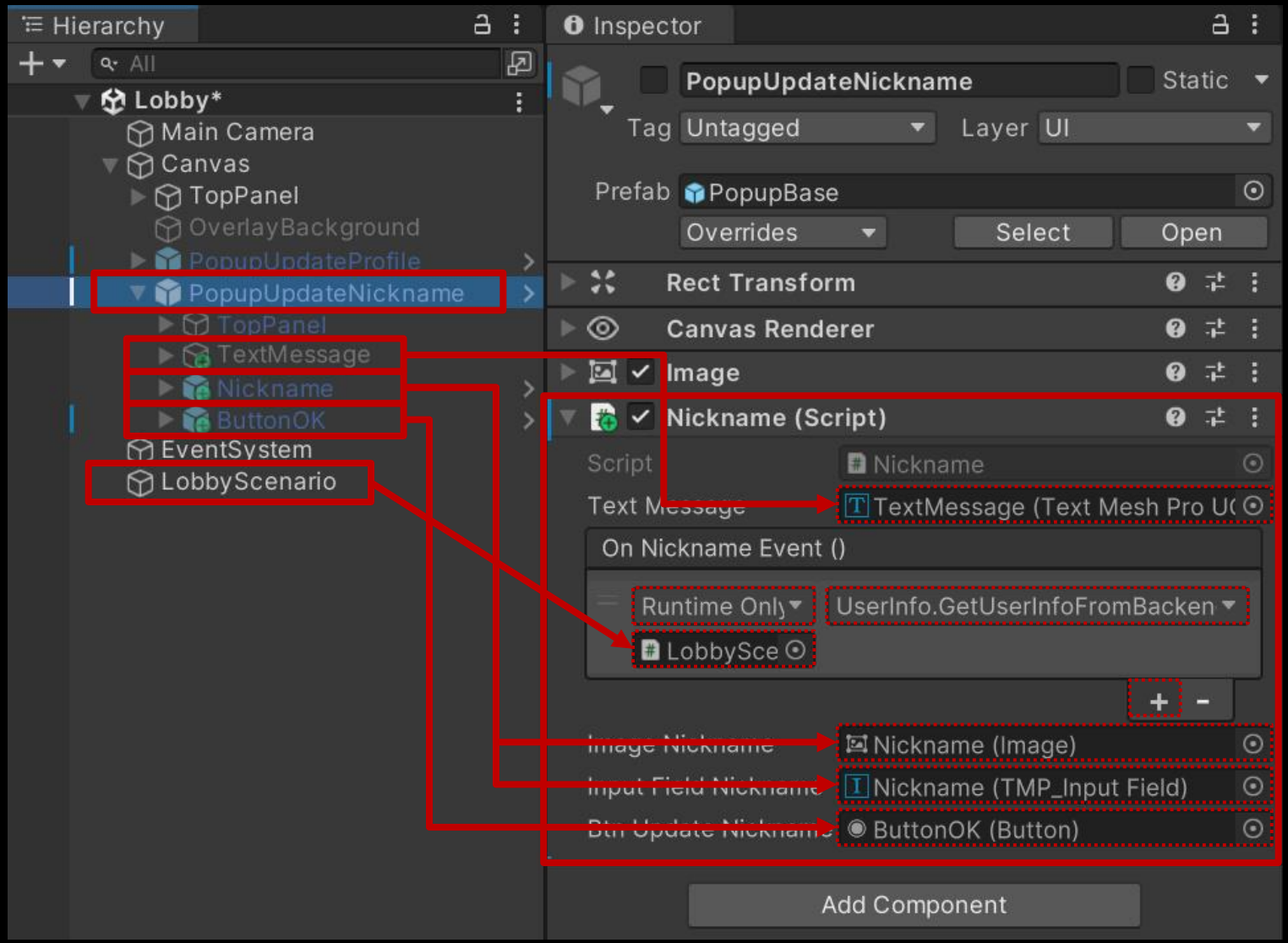
게임 유저 관리

- 뒤끝 서버와 연동해 닉네임을 변경하는 스크립트 생성 및 작성 (계속)

```
60          // 닉네임 변경 실패
61          else
62          {
63              string message = string.Empty;
64
65              switch ( int.Parse(callback.GetStatusCode()) )
66              {
67                  case 400:   // 빈 닉네임 혹은 string.Empty, 20자 이상의 닉네임, 닉네임 앞/뒤에 공백이 있는 경우
68                      message = "닉네임이 비어있거나 | 20자 이상 이거나 | 앞/뒤에 공백이 있습니다.";
69                      break;
70                  case 409:   // 이미 중복된 닉네임이 있는 경우
71                      message = "이미 존재하는 닉네임입니다.";
72                      break;
73                  default:
74                      message = callback.GetMessage();
75                      break;
76              }
77
78              GuideForIncorrectlyEnteredData(imageNickname, message);
79          }
```

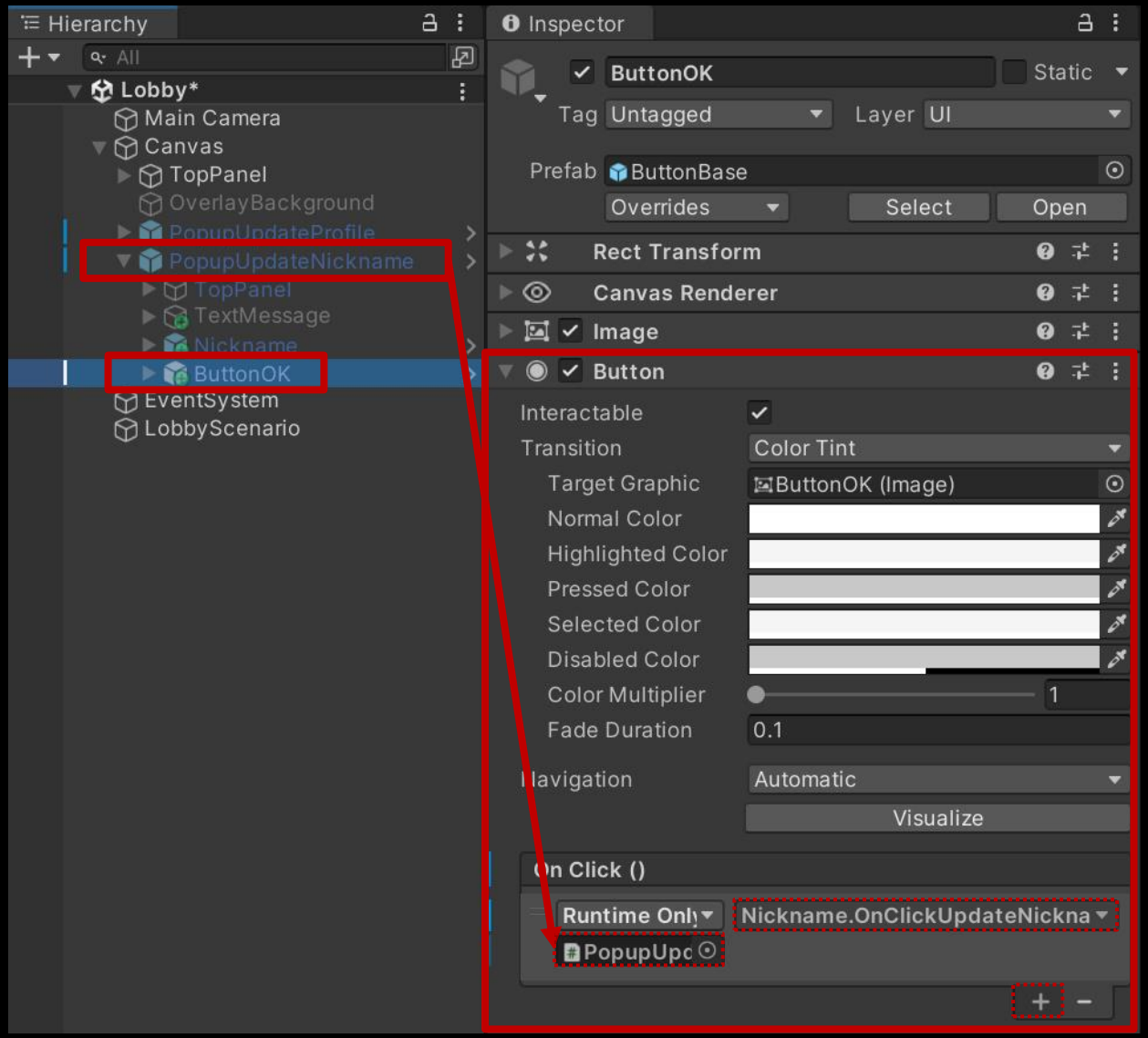
게임 유저 관리

- PopupUpdateNickname 오브젝트에 "Nickname" 컴포넌트 추가 및 설정



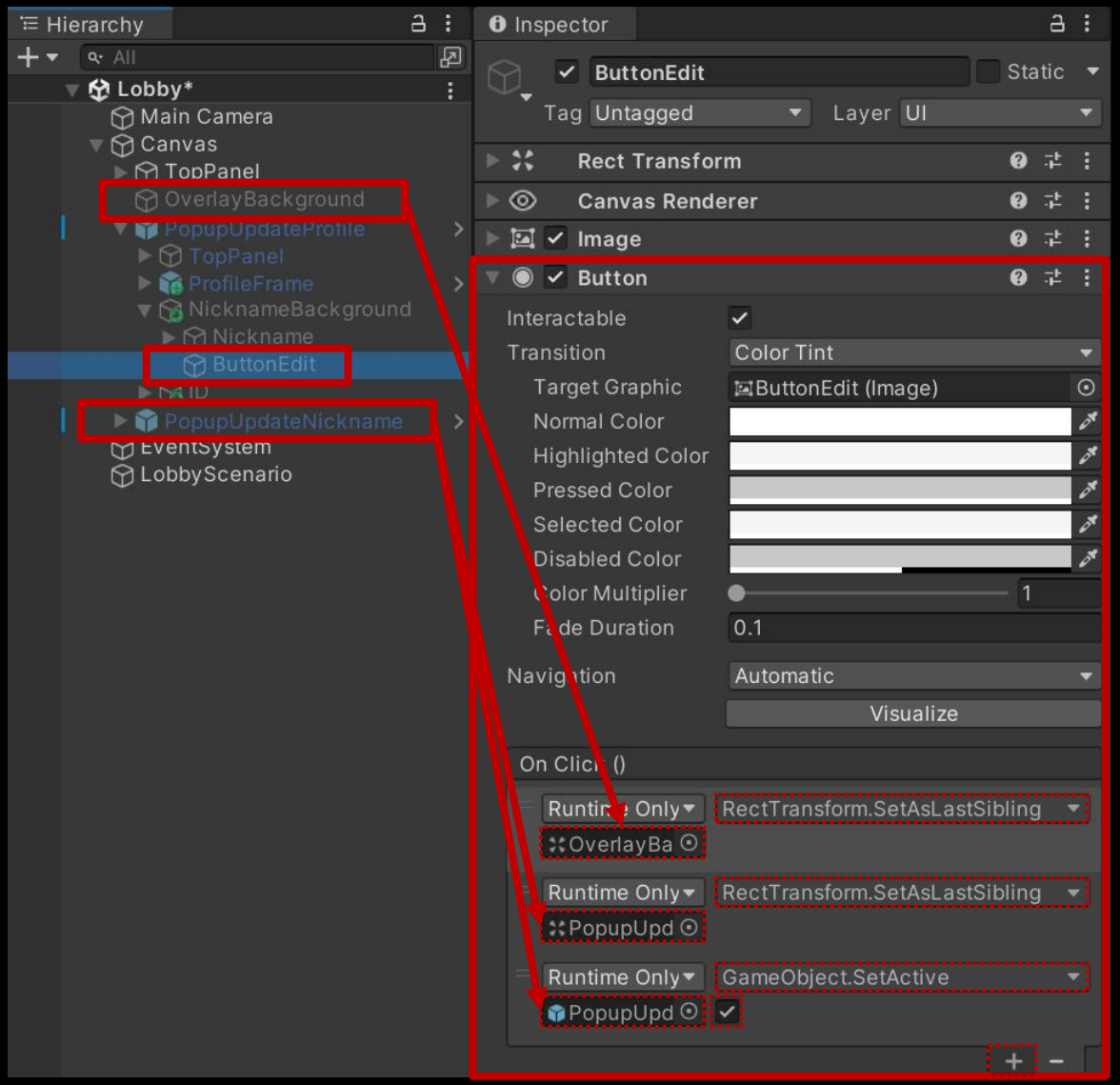
게임 유저 관리

- ButtonOK 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



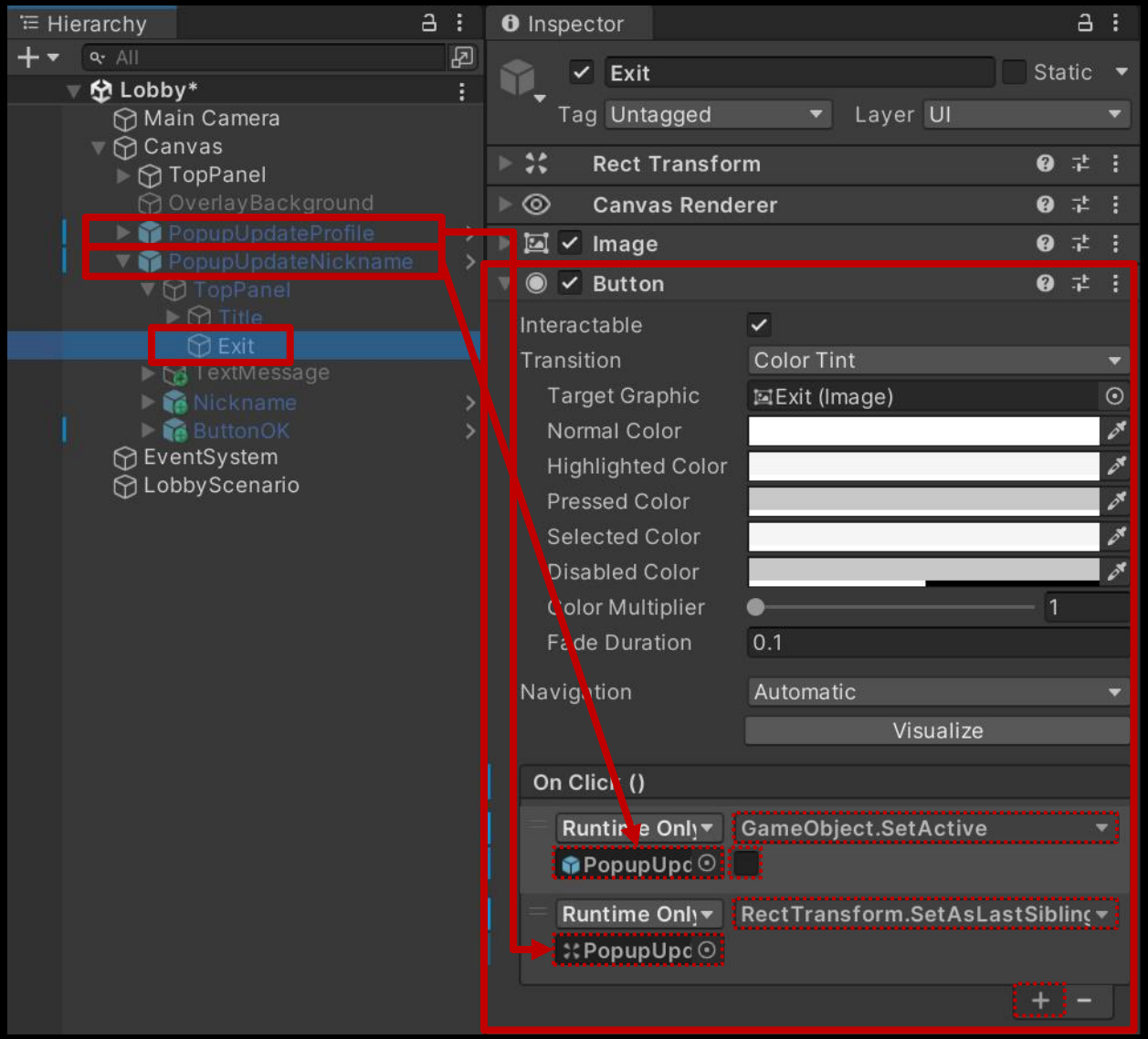
게임 유저 관리

- ButtonEdit 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



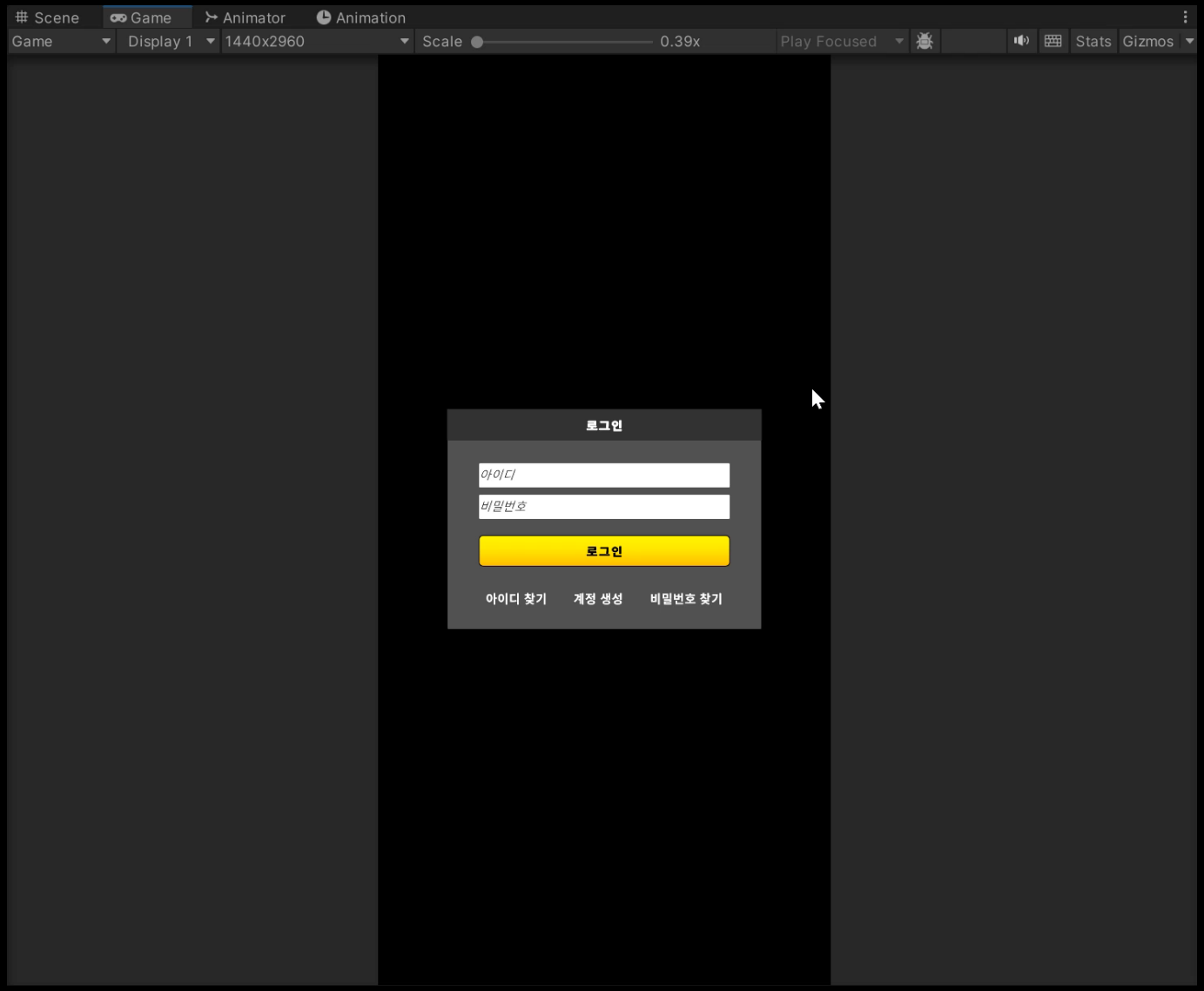
게임 유저 관리

- Exit 오브젝트의 "Button" 컴포넌트 OnClick() 이벤트 설정



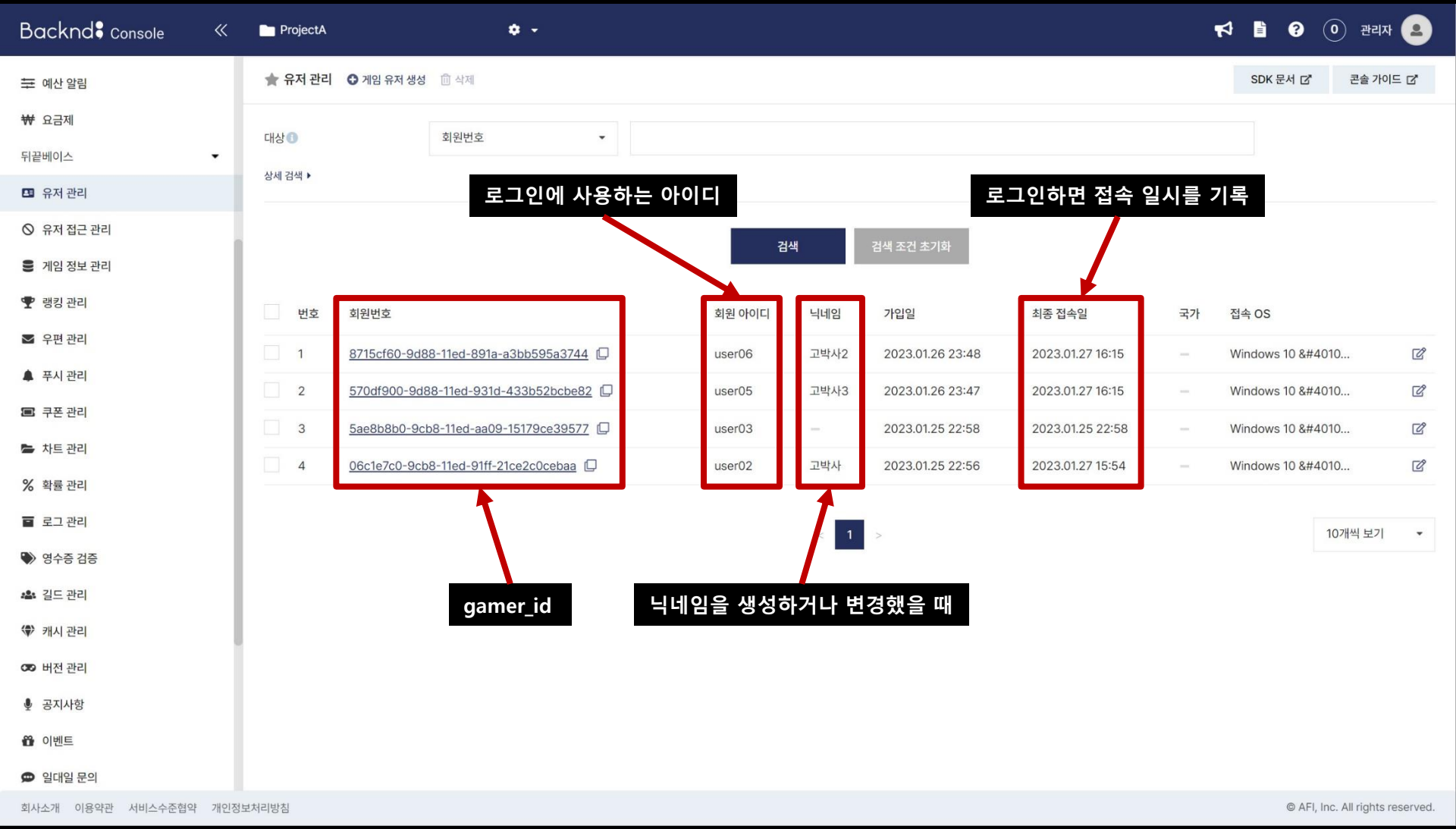
게임 유저 관리

- 결과 화면


Scene
Game
Animator
Animation
Game
Display 1
1440x2960
Scale
0.39x
Play Focused
Stats
Gizmos
로그인
아이디
비밀번호
로그인
아이디 찾기
계정 생성
비밀번호 찾기

게임 유저 관리

- 결과 화면


Backnd Console
ProjectA
관리자
예산 알림
요금제
뒤끝베이스
유저 관리
유저 접근 관리
게임 정보 관리
랭킹 관리
우편 관리
푸시 관리
쿠폰 관리
차트 관리
확률 관리
로그 관리
영수증 검증
길드 관리
캐시 관리
버전 관리
공지사항
이벤트
일대일 문의
유저 관리
게임 유저 생성
삭제
SDK 문서
콘솔 가이드
대상
회원번호
상세 검색
로그인에 사용하는 아이디
로그인하면 접속 일시를 기록
검색
검색 조건 초기화
번호
회원번호
회원 아이디
닉네임
가입일
최종 접속일
국가
접속 OS
1 8715cf60-9d88-11ed-891a-a3bb595a3744 user06 고박사2 2023.01.26 23:48 2023.01.27 16:15 Windows 10 &#4010...
2 570df900-9d88-11ed-931d-433b52bcbe82 user05 고박사3 2023.01.26 23:47 2023.01.27 16:15 Windows 10 &#4010...
3 5ae8b8b0-9cb8-11ed-aa09-15179ce39577 user03 2023.01.25 22:58 2023.01.25 22:58 Windows 10 &#4010...
4 06c1e7c0-9cb8-11ed-91ff-21ce2c0cebaa user02 고박사 2023.01.25 22:56 2023.01.27 15:54 Windows 10 &#4010...
10개씩 보기
gamer_id
닉네임을 생성하거나 변경했을 때
회사소개
이용약관
서비스수준협약
개인정보처리방침
© AFI, Inc. All rights reserved.