

“Tools 23 – QR Code Generator” Document

Step by step guide to use this package:

After importing the package, you will see a folder named “Tools 23 – QR Code Generator” inside your “Assets” folder. In the Figure 2 you can see the third-party plugins that have been used in this project.

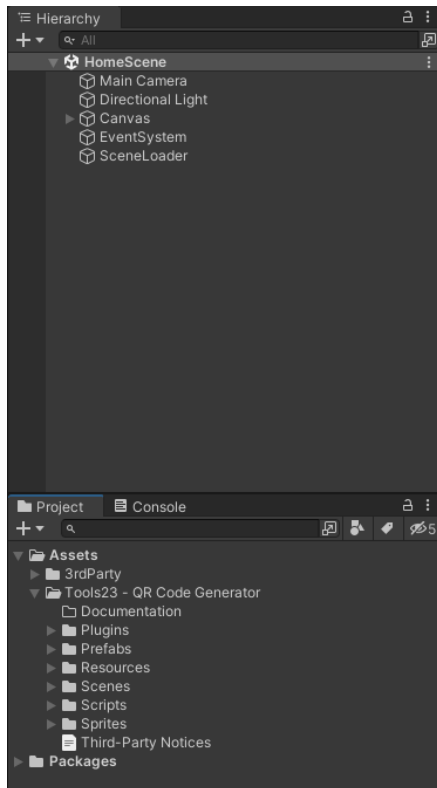


Figure 1

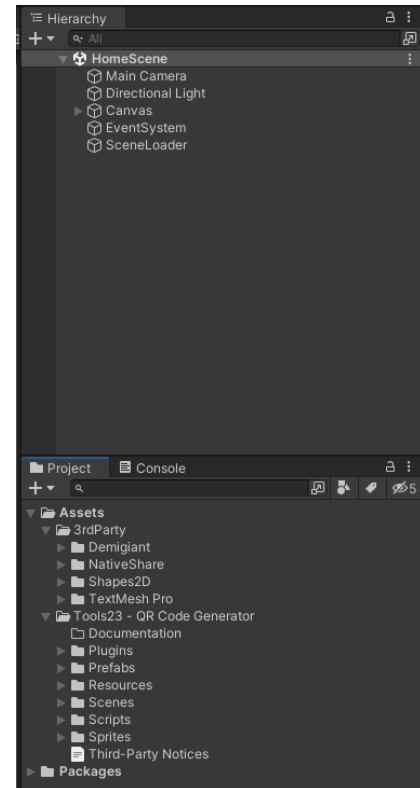


Figure 2

The links to these third-party plugins are given below:

DOTween (HOTween v2): [DOTween \(HOTween v2\) | Animation Tools | Unity Asset Store](#)

Shapes2D – Procedural sprites and UI: [Shapes2D - Procedural sprites and UI | Sprite Management | Unity Asset Store](#)

Native Share for Android & iOS: [Native Share for Android & iOS | Integration | Unity Asset Store](#)

Rest of the package you can download and import from the Package Manager of Unity editor.

In the Figure 3, you can see that our package has only two sprites that's been used. “backButton” sprite is used for both the back button and the home button. “close” sprite is used as the error sign.

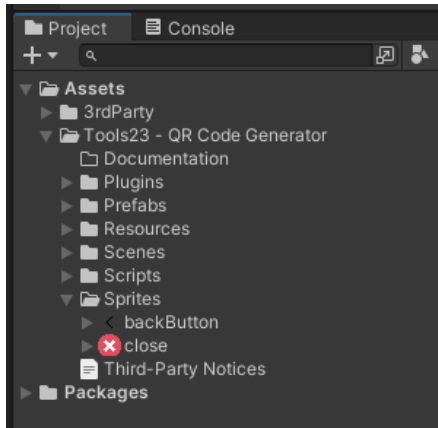


Figure 3

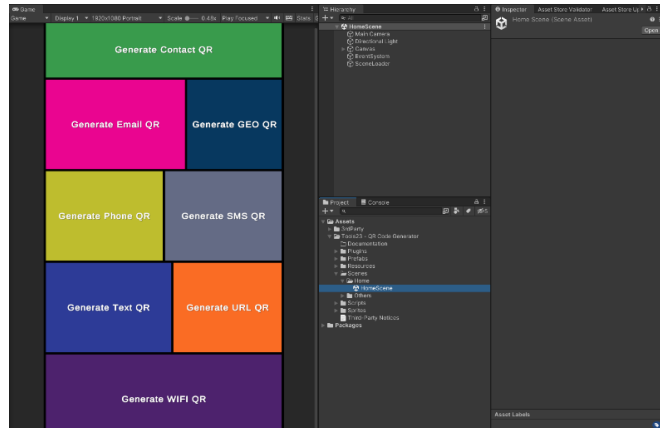


Figure 4

To open the “HomeScene” follow the Figure 4. As shown in the Figure 5, ButtonPanel can be expanded and you can see all the buttons responsible to switch from “HomeScene” to all the other QR Code generating scenes.

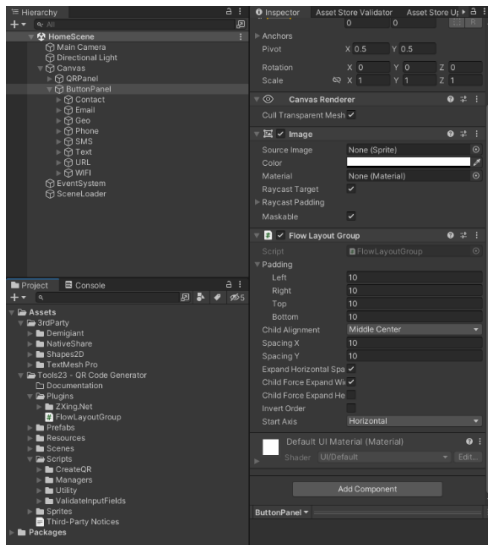


Figure 5

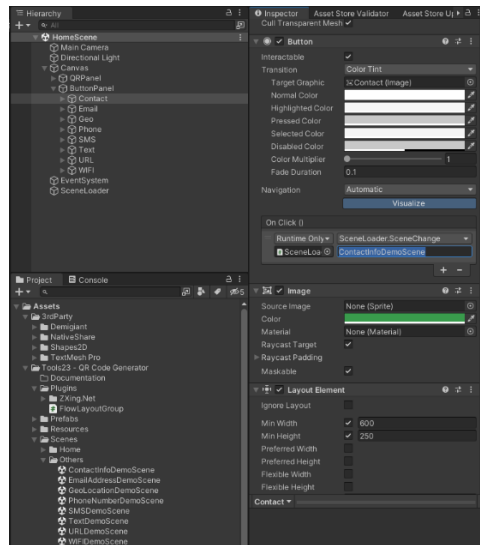


Figure 6

For example, you can select the “Contact” button, add “OnClick()”, then drag and drop “SceneLoader” and choose “SceneChange” method. “SceneChange” method takes parameter which is a string that represents scene name. (Figure 5 – 7)

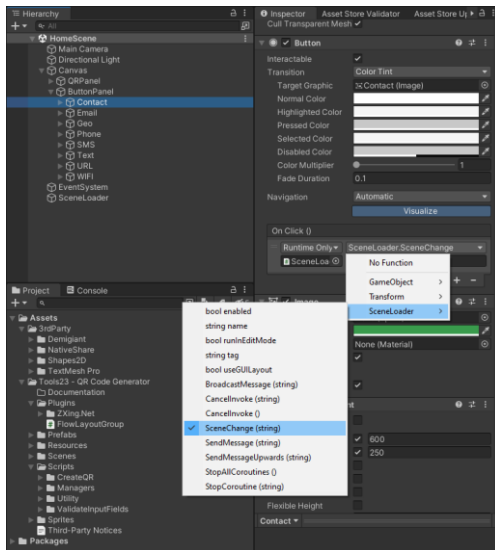


Figure 7

If you want to setup a Contact QR Generator UI, then go to the “Others” folder and drag “ContactInfoDemoScene” to the “Canvas” (Figure 8).

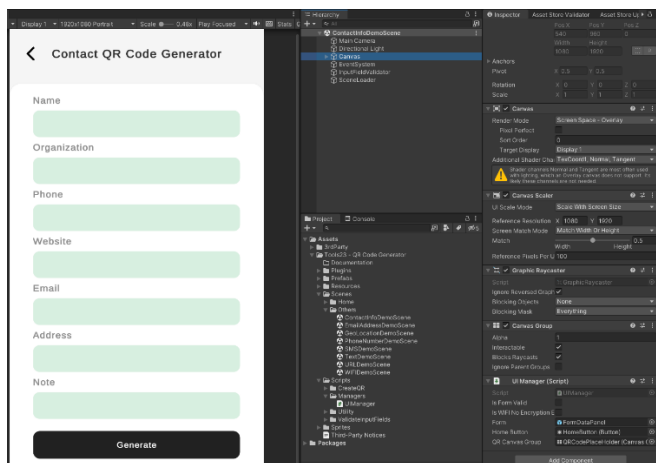


Figure 8

As shown in the Figure 9, add “UIManager” component in the “Canvas” (or you can just create an empty object and do the same in your scene). Drag “FormDataPanel”, “HomeButton” and “QRCodePlaceHolder” to the “Form”, “HomeButton” and “QR Canvas Group” field of the UIManager script respectively.

According to the Figure 10, you can create an empty gameobject called “InputFieldValidator” and add “ContactInfoInputFieldValidations” script as a component.

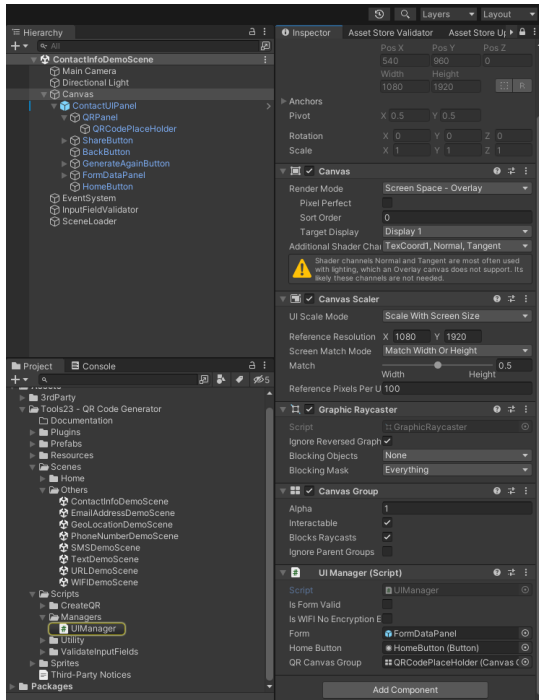


Figure 9

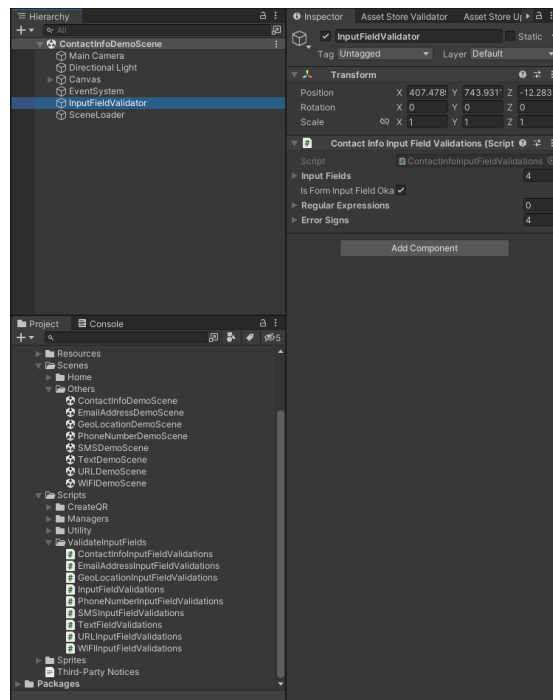


Figure 10

As shown in the Figure 11, drag and drop the “Name”, “Telephone”, “URL” and “Email” input field to the “Input Fields” list under the “Contact Info Input Field Validations” script. The mentioned input field has their corresponding “Label” with an image used as Error sign. Drag those images to the “Error Signs” list.

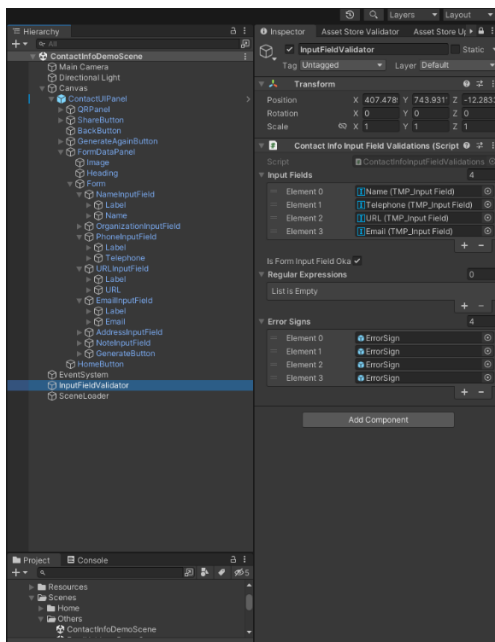


Figure 11

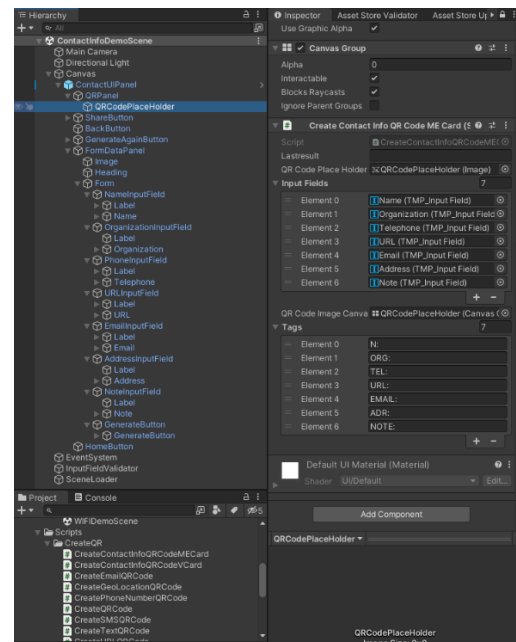


Figure 12

You can set the functionality of the other buttons as shown in the (Figure 13-17).

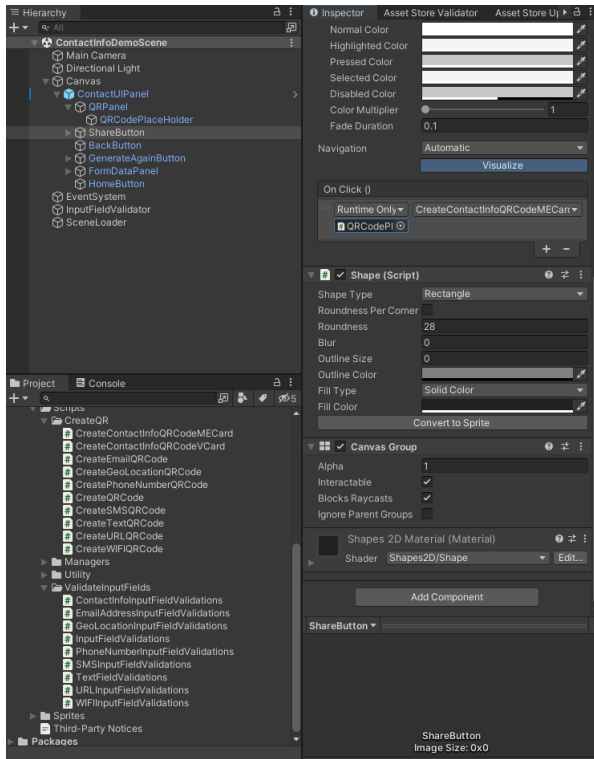


Figure 13

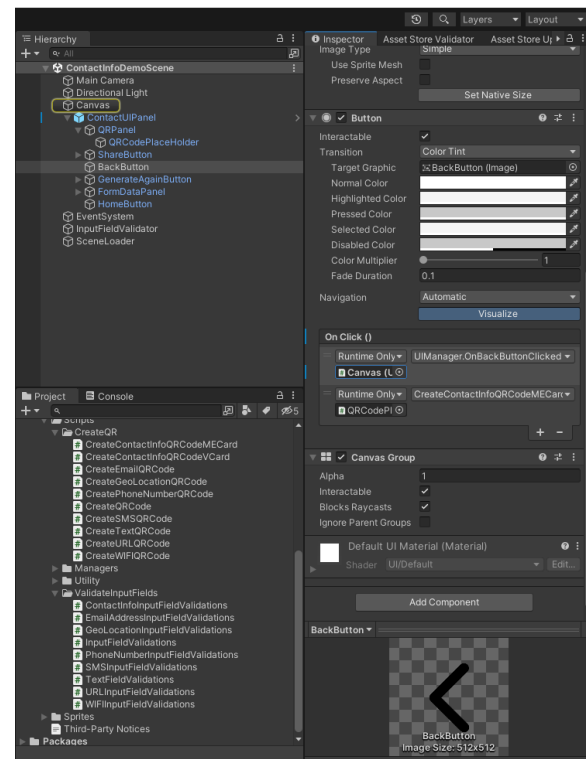


Figure 14

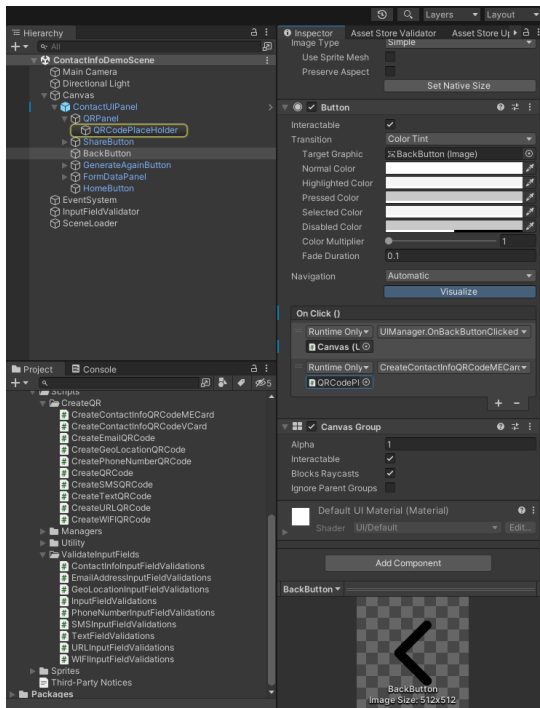


Figure 15

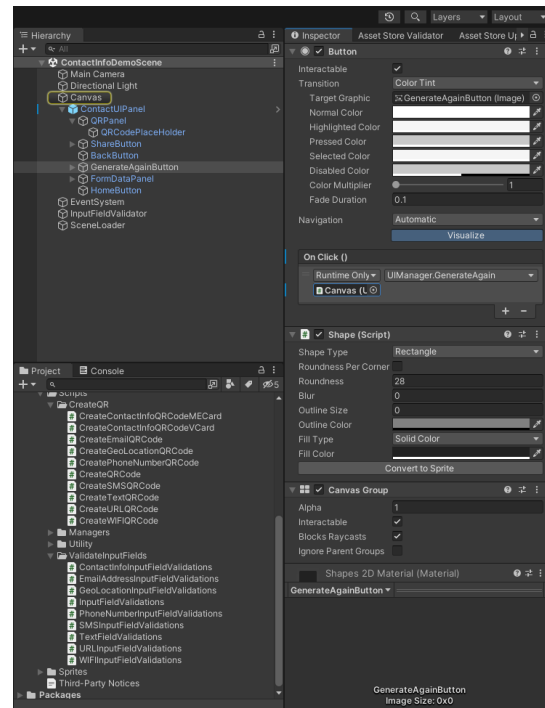


Figure 16

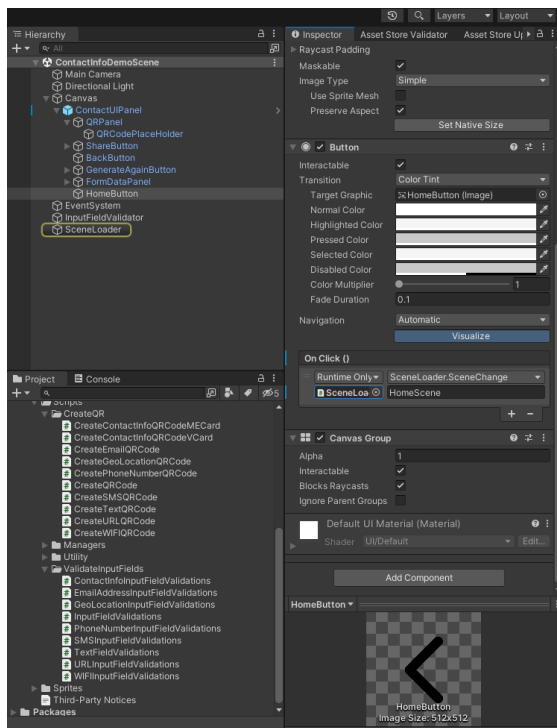


Figure 17

Scripts:

CreateQRCode: This is the main script that is responsible for creating the actual QR code (Figure 18 - 20).

```

9 namespace QRCodeGenerator23
10 {
11     // Unity Script 19 references
12     public class CreateQRCode : MonoBehaviour
13     {
14         public string Lastresult;
15         public static Texture2D encoded;
16         public Image QRCodePlaceholder;
17         public List<TMP_InputField> inputFields;
18         public CanvasGroup QRCodeImageCanvas;
19     }

```

Figure 18

```

40 // For generating the QRCode Image
41 // References
42 public void GenerateQROutput()
43 {
44     if (UIManager.Instance.IsValid)
45     {
46         encoded = new Texture2D(312, 312);
47         var textForEncoding = Lastresult;
48         if (textForEncoding != null)
49         {
50             var color32 = Encode(textForEncoding, encoded.width, encoded.height);
51             encoded.SetPixels32(color32);
52             encoded.Apply();
53         }
54         QRCodePlaceholder.sprite =
55             Sprite.Create(encoded, new Rect(0, 0, encoded.width, encoded.height), Vector2.zero);
56     }
57 }

```

Figure 19

```

57
58
59 //On back button click this function allows the user to edit the form
60 public void EnableEditFunction()
61 {
62     Lastresult = "";
63     QRCodeImageCanvas.alpha = 0;
64 }
65
66 //used to share the image on web
67 public void ShareImage()
68 {
69     Texture2D ss = encoded;
70
71     string filePath = Path.Combine(Application.temporaryCachePath, "shared.jpg.png");
72     File.WriteAllBytes(filePath, ss.EncodeToPNG());
73
74     // to avoid memory leaks
75     Destroy(ss);
76
77     new NativeShare().AddFile(filePath)
78     .SetSubject("subject: pass here").SetText("hello world!")
79     .SetUrl("https://github.com/passatola/UnityNativeShare")
80     .SetCallback((result, shareTarget) =>
81     {
82         Debug.Log("Share result: " + result + ", selected app: " + shareTarget));
83     }).Share();
84
85 }

```

Figure 20

The “GenerateQROutput” creates the QR code. The “EnableEditFunction” enables the user to edit the form on clicking the back button. The “ShareImage” uses the “NativeShare” to allow the user to share the QR image on available platforms.

CreateContactInfoQRCodeMECard: This script is inherited from the “CreateQRCode” script.

```

14 namespace QRCodeGenerator23
15 {
16     // Unity Script (1 must reference) | 0 references
17     public class CreateContactInfoQRCodeMECard : CreateQRCode
18     {
19         public List<string> Tags = new List<string> { "N:", "ORG:", "TEL:", "URL:", "EMAIL:", "ADR:", "NOTE:" };
20
21         // 0 references
22         public void GenerateTextToConvert()
23         {
24             if (UIManager.Instance.isFormValid) GenerateText();
25         }
26
27         //The QR generation depends on the result of the GenerateText() method
28         // 2 references
29         public override string GenerateText()
30         {
31             Lastresult += "MECARD:";
32             for (int i = 0; i < inputFields.Count; i++)
33             {
34                 if (inputFields[i].text != null)
35                 {
36                     Lastresult += (Tags[i] + inputFields[i].text + ";");
37                 }
38             }
39
40             Lastresult += ":";
41             Debug.Log(Lastresult);
42             return Lastresult;
43         }
44     }
45 }

```

Figure 21

The “GenerateTextToConvert” function uses the “GenerateText” method and returns a string that is further used to create the QR Code.

InputFieldValidation: This is an abstract class and responsible for validating the input field values.

```

1 using System.Collections.Generic;
2 using System.Text.RegularExpressions;
3 using TMPro;
4 using UnityEngine;
5
6 namespace QRCodeGenerator23
7 {
8     // 0 must script | 1 reference
9     public abstract class InputFieldValidations : MonoBehaviour
10     {
11         public List<TMP_InputField> InputFields;
12         [SerializeField] protected bool isFormInputFieldKey = true;
13         public List<string> RegularExpressions;
14         public List<string> ErrorSigns;
15
16         // 0 references
17         public abstract void ValidateInputFields();
18     }
19 }

```

Figure 22

```

16 protected void ValidateFields()
17 {
18     for (int i = 0; i < InputFields.Count; i++)
19     {
20         isFormInputFieldOk =
21             (isFormInputFieldOk) && Regex.IsMatch(InputFields[i].text, RegularExpressions[i]);
22         if (!isFormInputFieldOk)
23         {
24             ErrorSigns[i].gameObject.SetActive(true);
25             Handheld.Vibrate();
26             break;
27         }
28         else
29         {
30             ErrorSigns[i].gameObject.SetActive(false);
31         }
32     }
33
34     if (isFormInputFieldOk) UIManager.Instance.isFormValid = true;
35     isFormInputFieldOk = true;
36 }
37

```

Figure 23

ContactInputFieldValidation: This class is inherited from the “InputFieldValidations” class.

```

1 namespace QRCodeGenerator23
2 {
3     public class ContactInfoInputFieldValidations : InputFieldValidations
4     {
5         // Regex pattern for validating contact information
6         private string namePattern = @"^[A-Za-z]+(\s)?([A-Za-z])+$";
7         private string phonePattern = @"^[0-9]{0,3}[ -]?[0-9]{3}[ -]?[0-9]{3}[ -]?[0-9]{4}$";
8
9         private string urlPattern =
10             @"^(https?|ftp)://?(www\.)?([a-zA-Z0-9_-]+)?(\.[a-zA-Z]{2,})?(\.[a-zA-Z]{2,})?(/[w\.-\^?@&=]*)?$";
11
12         private string emailPattern = @"^[a-zA-Z0-9_+]+@[a-zA-Z0-9_-]+\.[a-zA-Z]{2,}$";
13
14         public override void ValidateInputFields()
15         {
16             RegularExpressions.Add(namePattern);
17             RegularExpressions.Add(phonePattern);
18             RegularExpressions.Add(urlPattern);
19             RegularExpressions.Add(emailPattern);
20             ValidateFields();
21         }
22     }
23 }
24
25

```

Figure 24

There are other scripts similar to the above mentioned scripts. You also need to follow the same process to setup the different QR code generating UI and get your desired output.