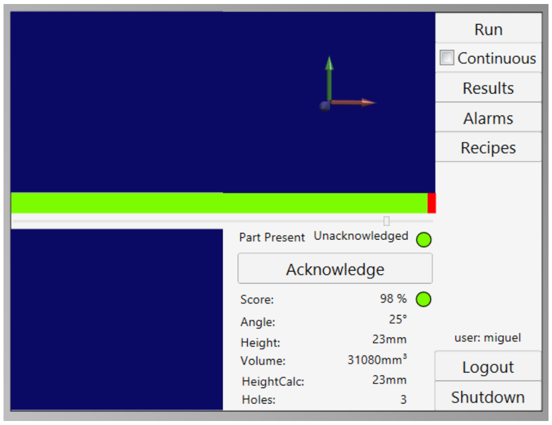
Cognex Designer Advanced – Section 3
Recipes Lab
Approximate Duration: 30 minutes

EXPECTED OUTCOMES:

- Create Recipe to allow for tolerance change between parts

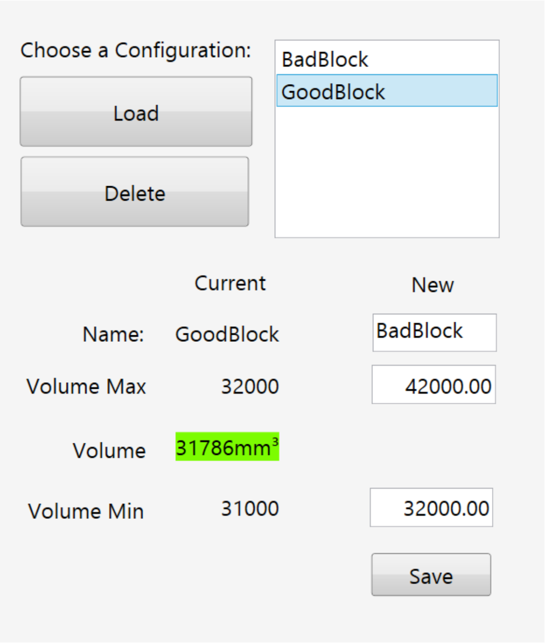
EXPECTED VISUAL RESULT:



Part Present Unacknowledged ●

Score:	98 % ●
Angle:	25°
Height:	23mm
Volume:	$31080mm^3$
HeightCalc:	23mm
Holes:	3

user: miguel



Choose a Configuration:

BadBlock
GoodBlock

	Current	New
Name:	GoodBlock	BadBlock
Volume Max	32000	42000.00
Volume	$31786mm^3$	
Volume Min	31000	32000.00

OUTLINE OF LAB:

1. **Allow other configurations to PASS with system.**
2. **Create Recipe – ability to add recipes**
   a. **Add Tags**
   b. **Add page and scripting**
3. **Integrate Recipe tags into application**
   a. **Switch in the new Tolerances**
   b. **Save application**

Challenge 1: Make use of the default $System.Info tags for some of the information. Also, add a confirmation dialog to the Delete to keep users from deleting configurations inadvertently.

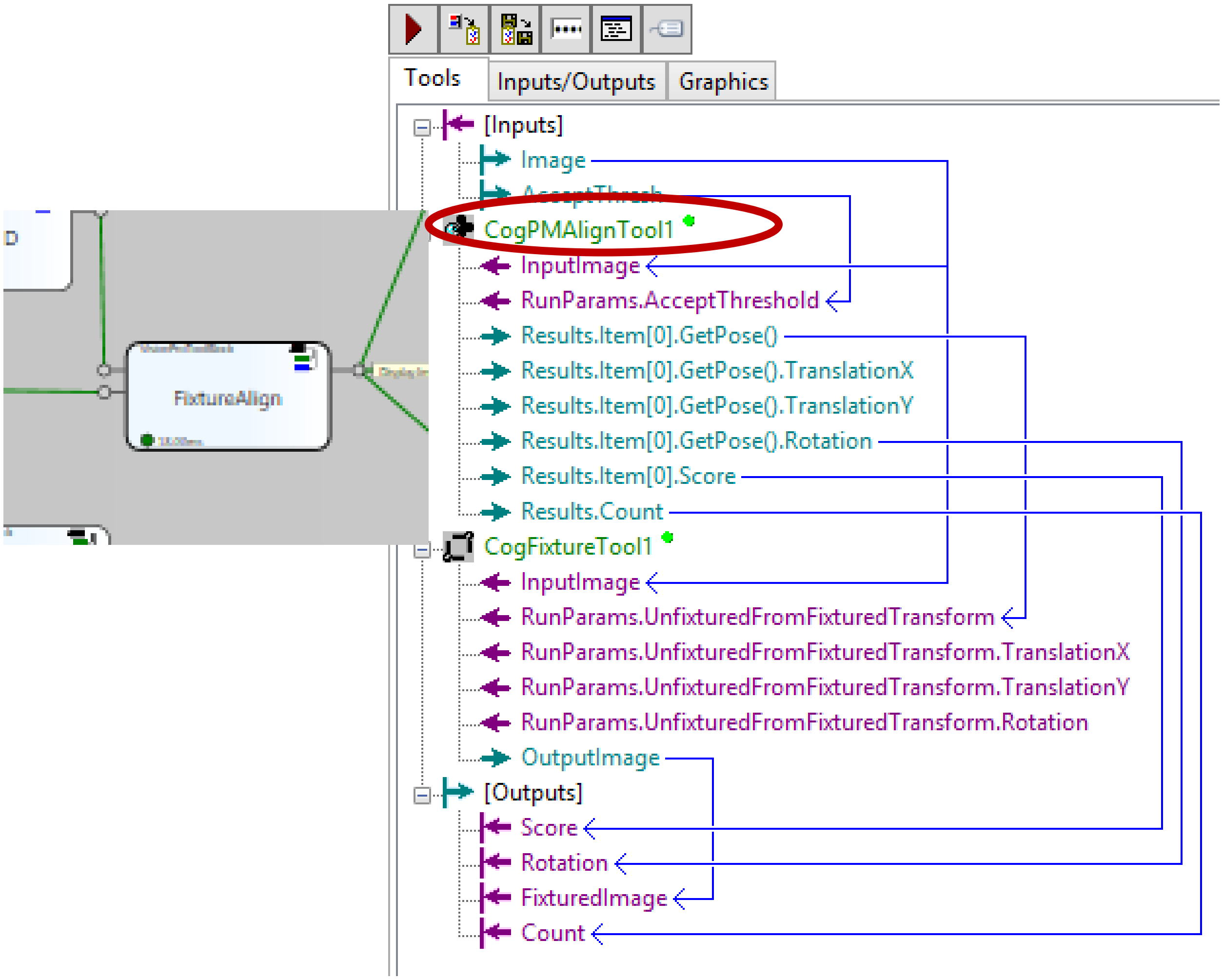
Challenge 2: Use the example above to add support for checking Height tolerance.

Steps for the Lab:

1. **Get new part to "work" with system.**
   - We will use the "bad" demo block for this lab. This is so we have a part that we can use to check for tolerances. Put the part under the camera and take an image.

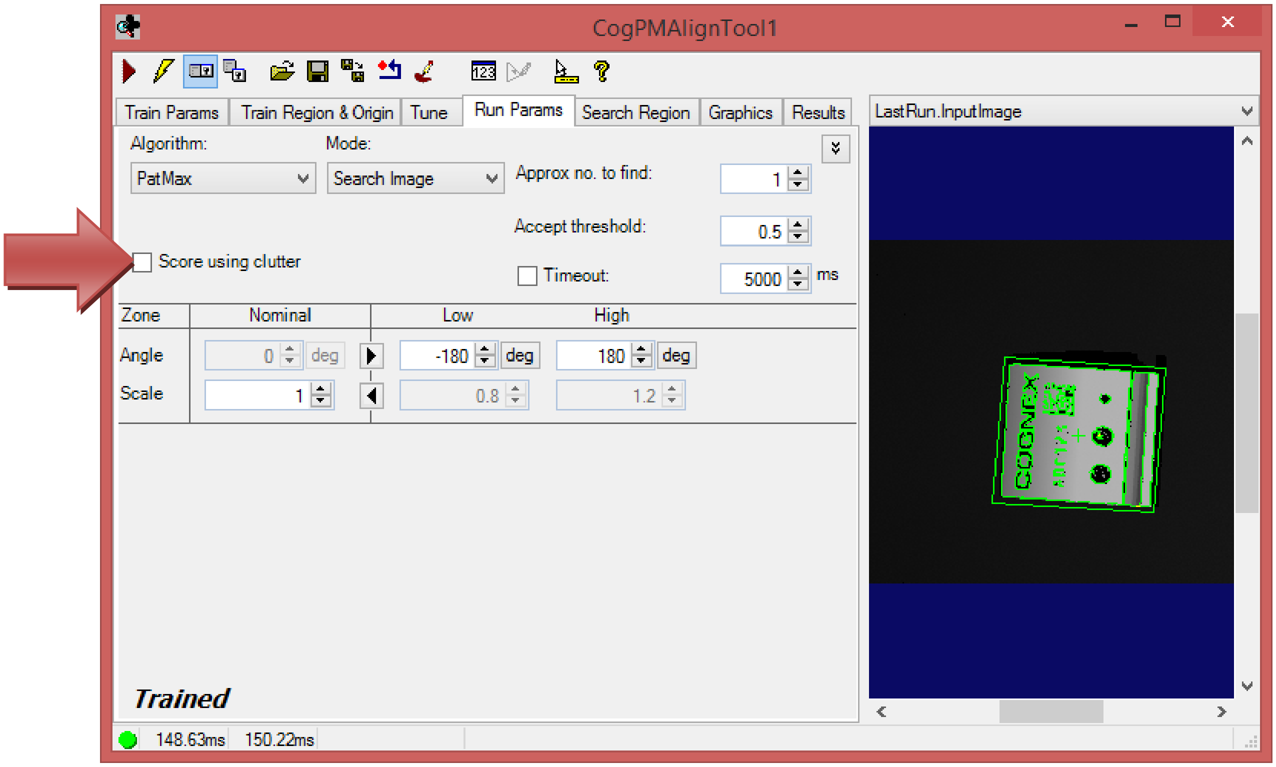
   a. **Change PatMax to see part**

      i. Go to the PMAlignTool Toolblock and open it so that you can get access to the PatMax tool.



```
[Inputs]
  Image
  AcceptThresh
CogPMAlignTool1
  InputImage
  RunParams.AcceptThreshold
  Results.Item[0].GetPose()
  Results.Item[0].GetPose().TranslationX
  Results.Item[0].GetPose().TranslationY
  Results.Item[0].GetPose().Rotation
  Results.Item[0].Score
  Results.Count
CogFixtureTool1
  InputImage
  RunParams.UnfixturedFromFixturedTransform
  RunParams.UnfixturedFromFixturedTransform.TranslationX
  RunParams.UnfixturedFromFixturedTransform.TranslationY
  RunParams.UnfixturedFromFixturedTransform.Rotation
  OutputImage
[Outputs]
  Score
  Rotation
  FixturedImage
  Count
```

ii. Open your PatMax tool and go to the RunParams tab and deselect Score using Clutter if you haven’t already done so.



Algorithm: PatMax

Mode: Search Image

Approx no. to find: 1

Accept threshold: 0.5

□ Score using clutter

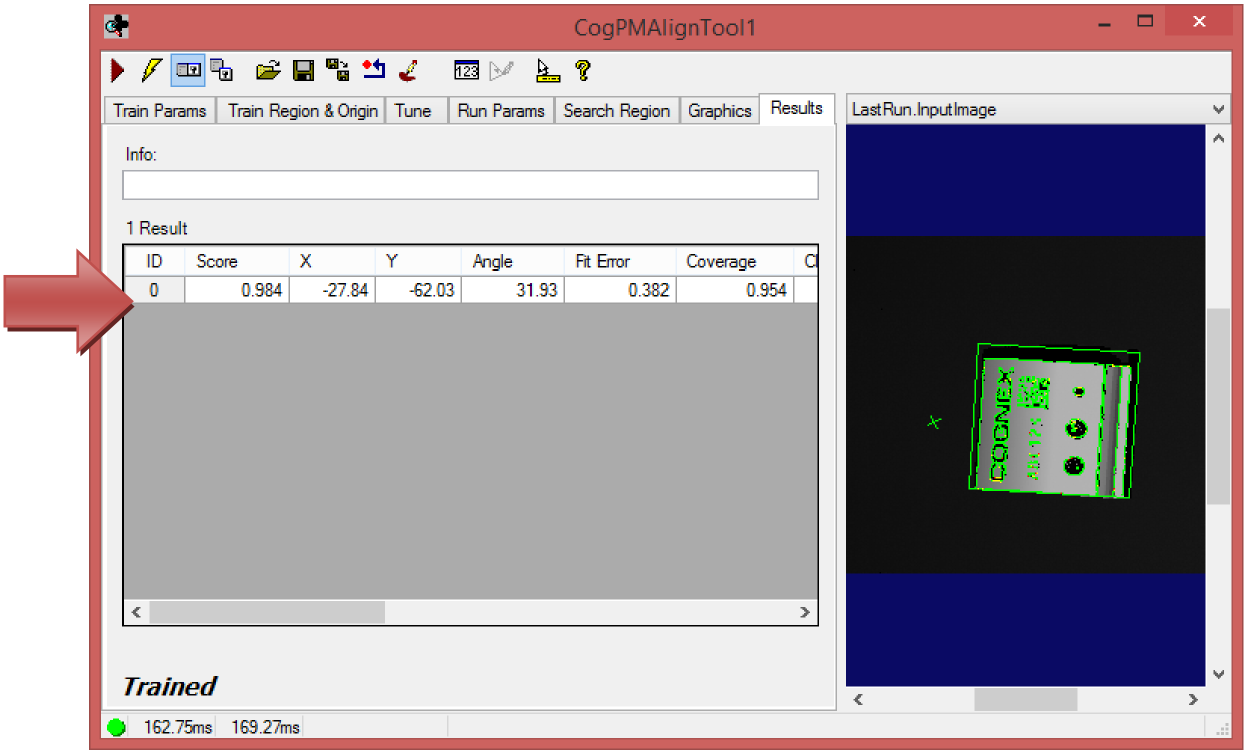
□ Timeout: 5000 ms

Zone	Nominal	Low	High
Angle	0 deg	-180 deg	180 deg
Scale	1	0.8	1.2

Trained

148.63ms 150.22ms

iii. Execute the sequence to make sure that both types of demo blocks can be found by the PMAlign tool.



Info:

1 Result

ID	Score	X	Y	Angle	Fit Error	Coverage	Cl
0	0.984	-27.84	-62.03	31.93	0.382	0.954	

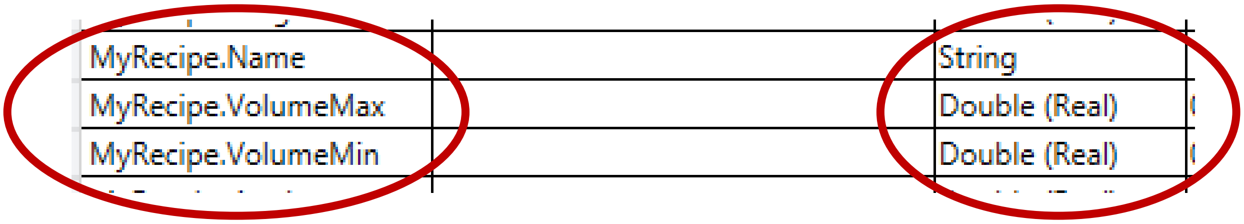
Trained

162.75ms 169.27ms

Create Recipe

b. Add Tags

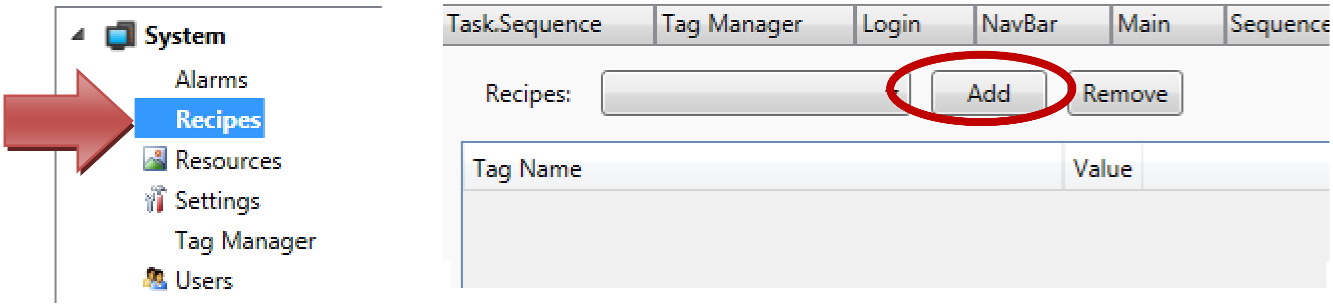
i. In the tag manager, add tags to deal with the varying tolerances of the parts. You will need a tag for VolumeMin, ValumeMax, and Name.



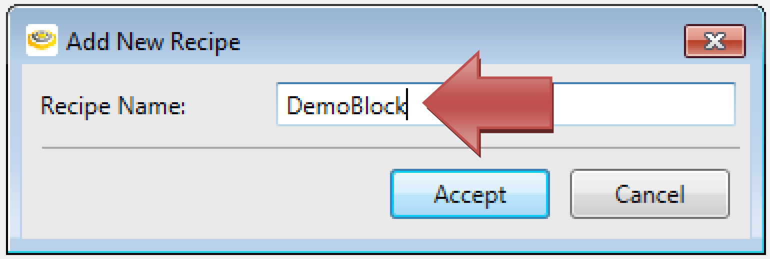
MyRecipe.Name		String
MyRecipe.VolumeMax		Double (Real)
MyRecipe.VolumeMin		Double (Real)

c. Create Recipe

i. Go to Recipes and Add a new Recipe called DemoBlock



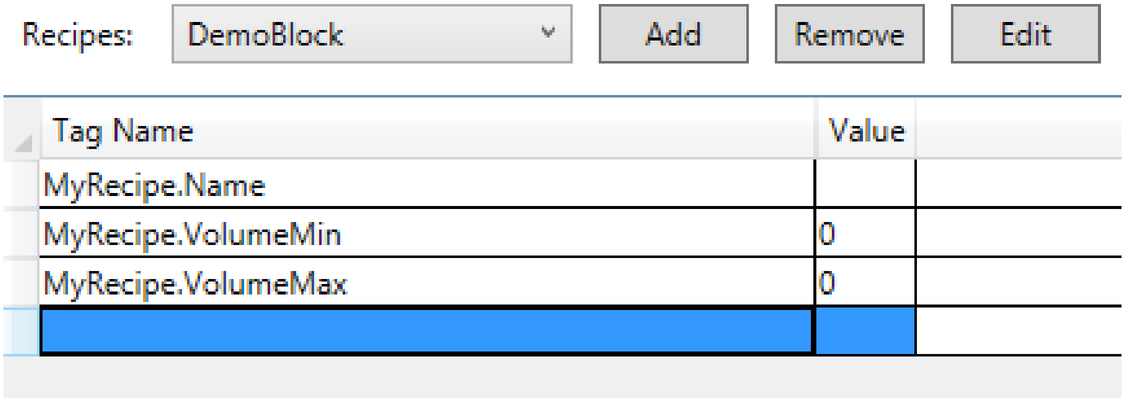
ii. Name it DemoBlock



Add New Recipe

Recipe Name: DemoBlock

iii. Select it from the pull-down and add the tags that were created in an earlier step to this DemoBlock recipe.

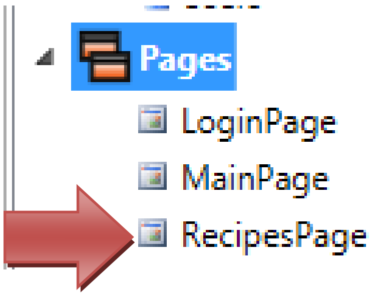


Recipes: DemoBlock

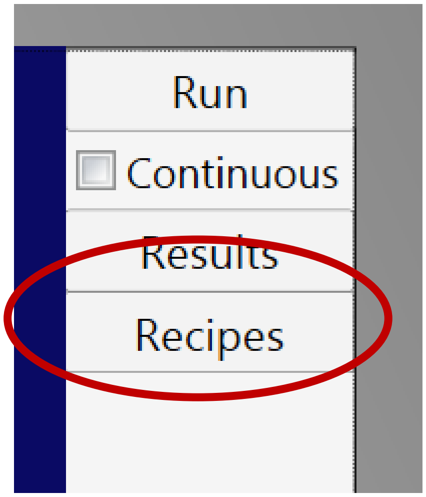
Tag Name	Value
MyRecipe.Name	
MyRecipe.VolumeMin	0
MyRecipe.VolumeMax	0

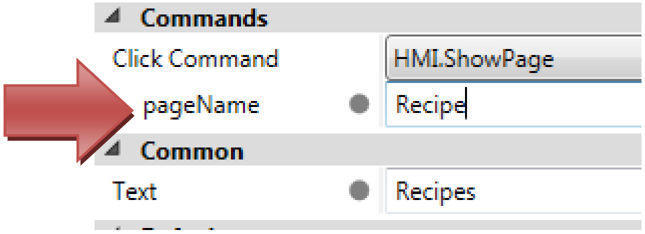
d. Add page and scripting

i. Create a new page and name it RecipesPage



ii. Create buttons on the MainPage and RecipePage to go back and forth between the two pages.





Commands

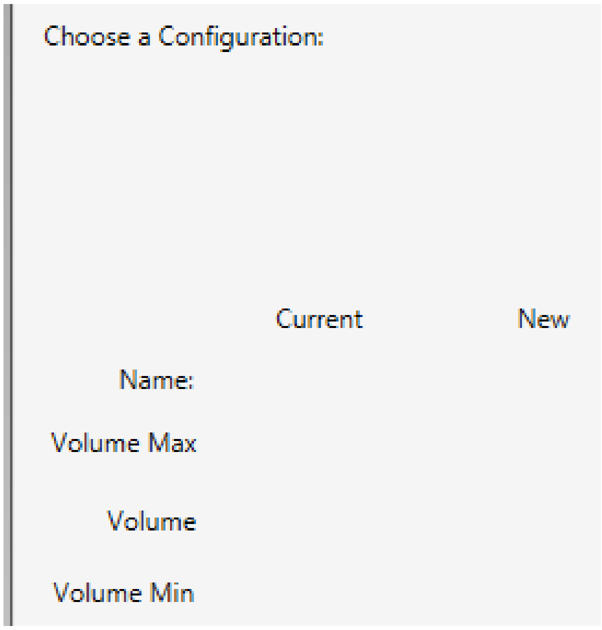
Click Command: HMI.ShowPage

pageName: Recipe

Common

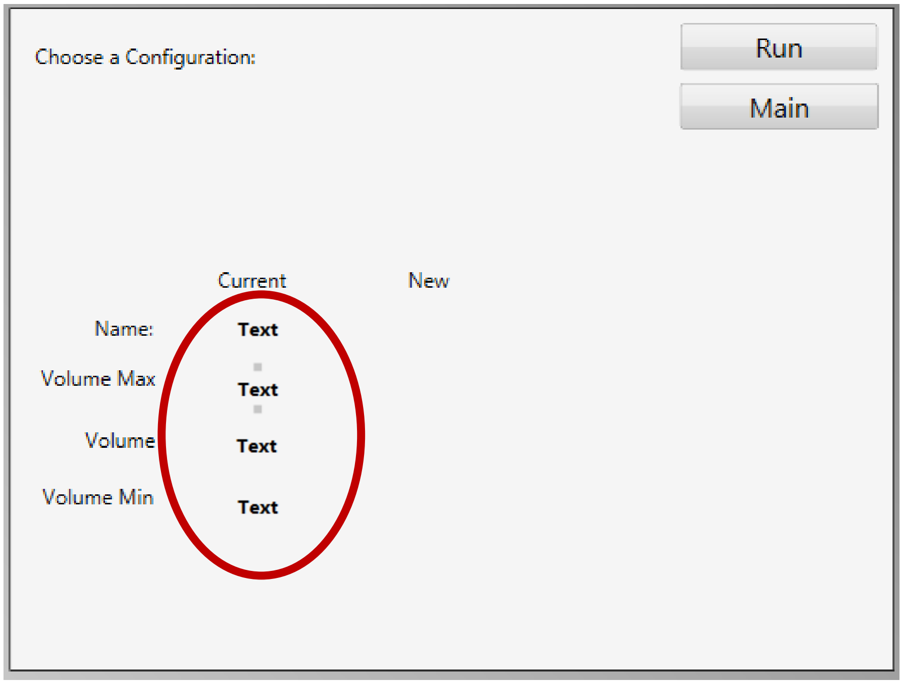
Text: Recipes

iii. On the RecipesPage, add some labels to help you navigate the user interface.

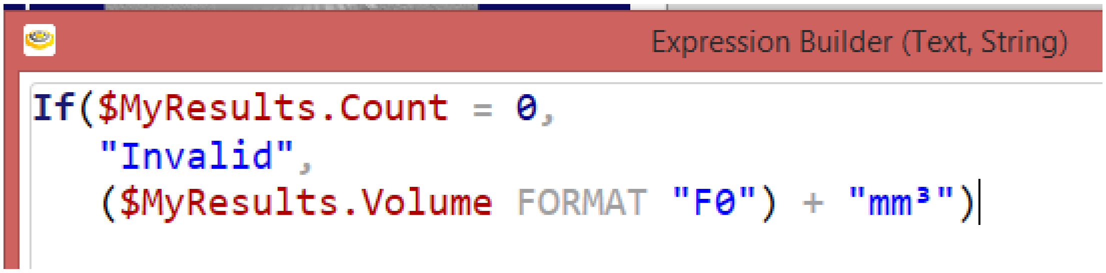


iv. Under Current column, add labels that reference the recipe tags MyRecipe.VolumeMax, MyRecipe.VolumeMin, and MyRecipe.Name.

1. Name subject points to $MyRecipe.Name
2. Volume Max subject points to $MyRecipe.VolumeMax
3. Volume subject points to $MyResults.Volume
4. Volume Min subject points to $MyRecipe.VolumeMin

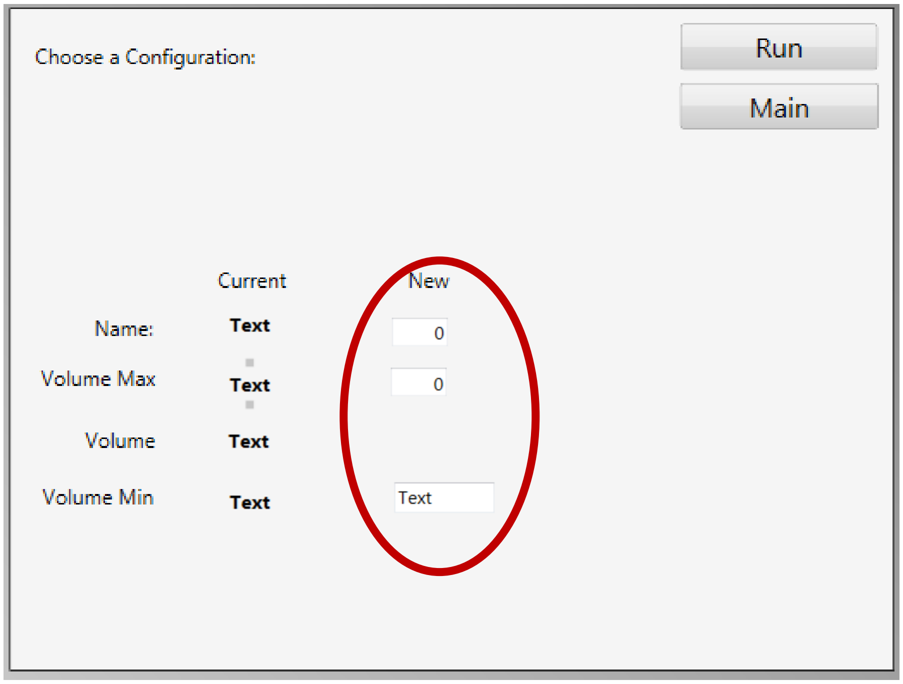


i. Set the actual Volume to display only if you actually have a valid result.

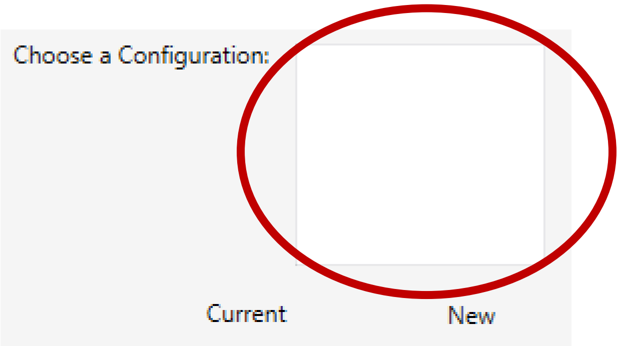

Expression Builder (Text, String)

```
If($MyResults.Count = 0,
   "Invalid",
   ($MyResults.Volume FORMAT "F0") + "mm³")
```

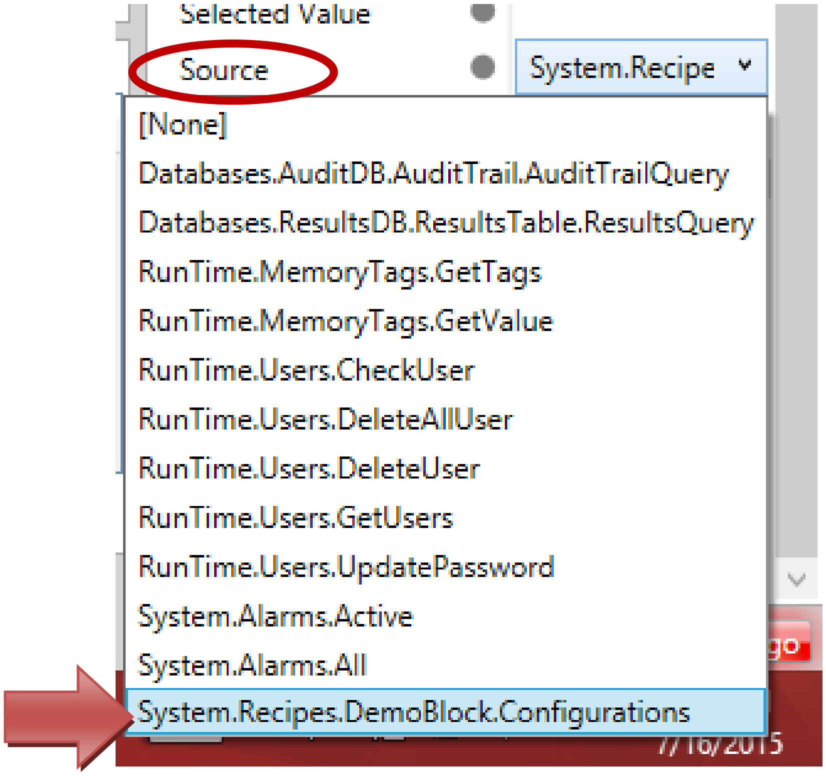
ii. Under the new column, add NumericEntry boxes and a TextBox that will be used to give your new configuration a name.



iii. Add a Listbox control to display a list of configurations you will be able to create. Stretch it into a square shape rather than the default single line shape so you can display more than one configuration option as a time.



i. Set the Source property to System.Recipes.DemoBlock.Configurations.

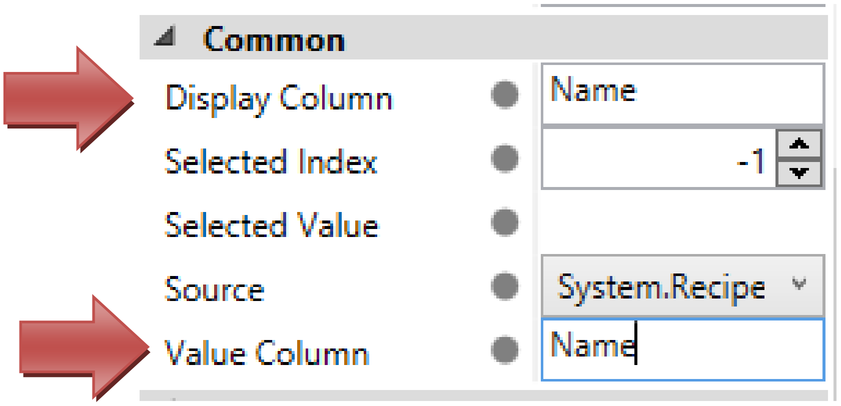


Selected Value

Source: System.Recipe

[None]
Databases.AuditDB.AuditTrail.AuditTrailQuery
Databases.ResultsDB.ResultsTable.ResultsQuery
RunTime.MemoryTags.GetTags
RunTime.MemoryTags.GetValue
RunTime.Users.CheckUser
RunTime.Users.DeleteAllUser
RunTime.Users.DeleteUser
RunTime.Users.GetUsers
RunTime.Users.UpdatePassword
System.Alarms.Active
System.Alarms.All
System.Recipes.DemoBlock.Configurations
7/16/2015

ii. Set the Listbox’s Display Column Property and the Value Column Property to “Name”. This will set it to display the name property of the Recipe’s configuration.



Common

Display Column: Name

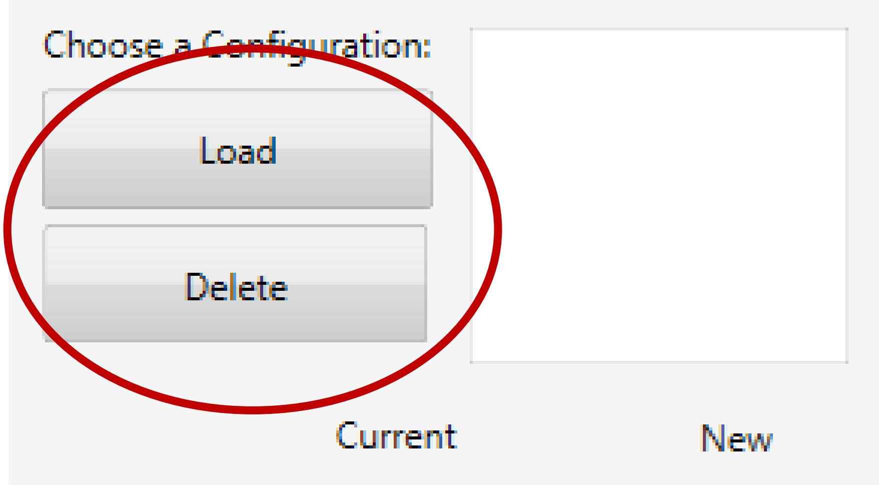
Selected Index: -1

Selected Value

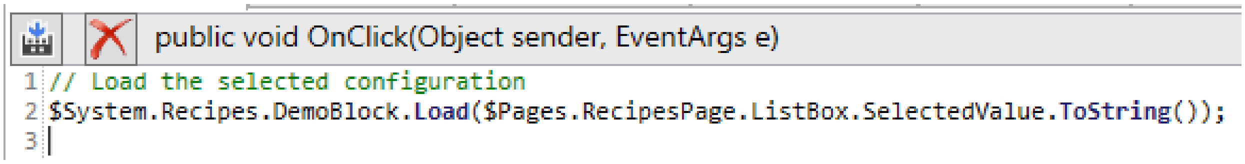
Source: System.Recipe

Value Column: Name

iii. Add a Load and Delete buttons to help manage the currently loaded configuration or delete unwanted configurations.

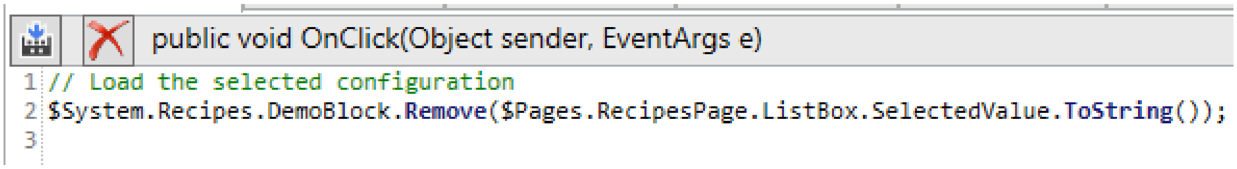


iv. Type in the following for the script in the Load button's OnClick script.



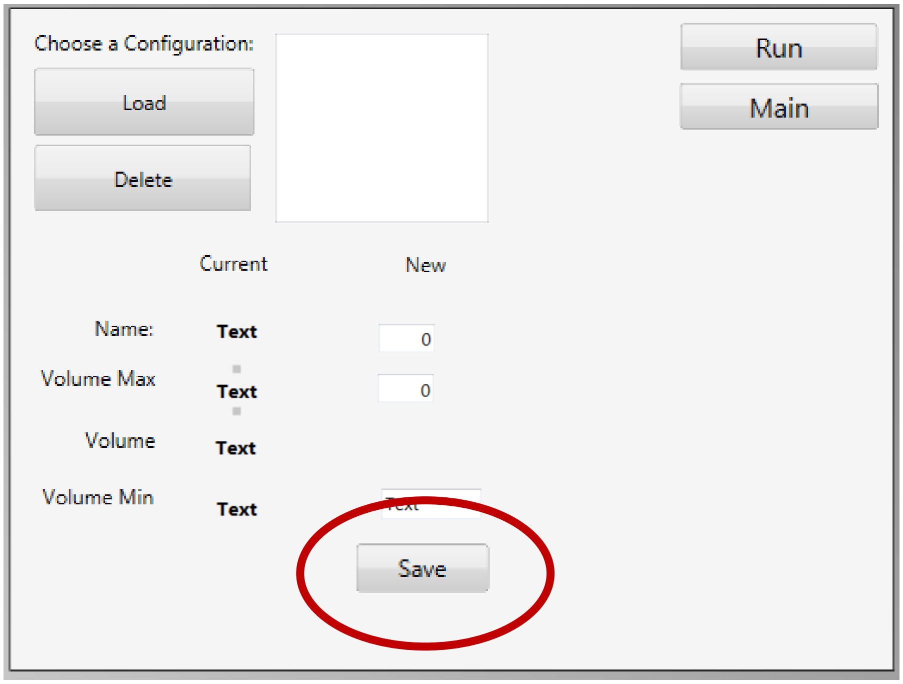
```
public void OnClick(Object sender, EventArgs e)
// Load the selected configuration
$System.Recipes.DemoBlock.Load($Pages.RecipesPage.ListBox.SelectedValue.ToString());

```

v. Type in the following for the script in the Delete button's OnClick script.

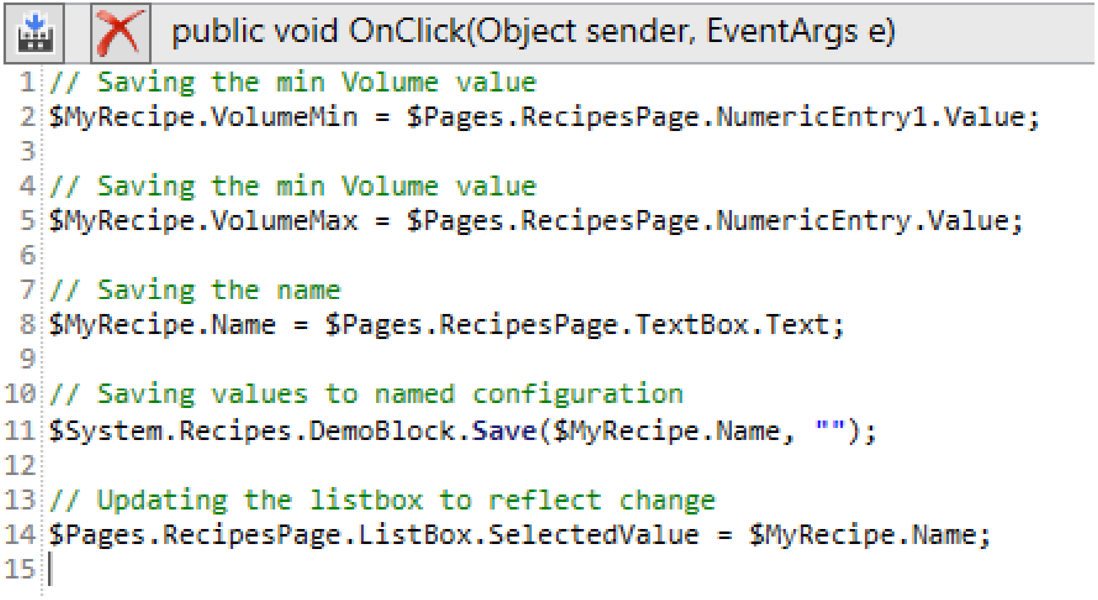


```
public void OnClick(Object sender, EventArgs e)
// Load the selected configuration
$System.Recipes.DemoBlock.Remove($Pages.RecipesPage.ListBox.SelectedValue.ToString());

```

vi. Add another button named Save and go to the Script for "OnMouseClick"

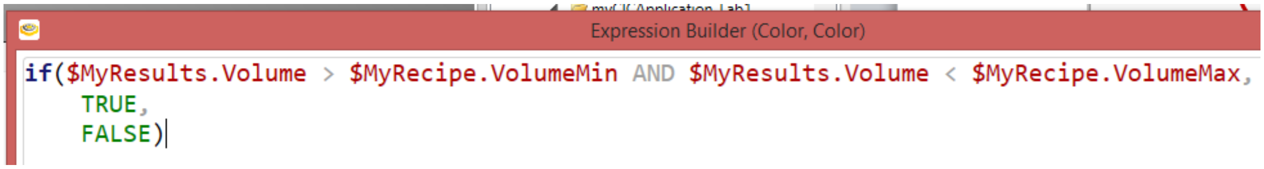


i. Type in the following for the script in the Save button’s OnClick script.

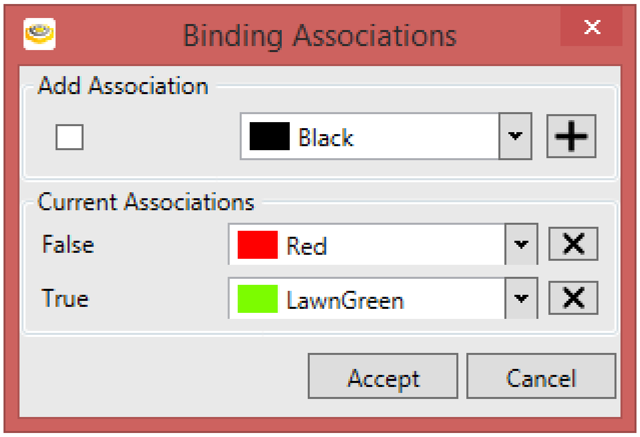


```
// Saving the min Volume value
$MyRecipe.VolumeMin = $Pages.RecipesPage.NumericEntry1.Value;

// Saving the min Volume value
$MyRecipe.VolumeMax = $Pages.RecipesPage.NumericEntry.Value;

// Saving the name
$MyRecipe.Name = $Pages.RecipesPage.TextBox.Text;

// Saving values to named configuration
$System.Recipes.DemoBlock.Save($MyRecipe.Name, "");

// Updating the listbox to reflect change
$Pages.RecipesPage.ListBox.SelectedValue = $MyRecipe.Name;

```

ii. Set the Volume label’s Background Color so that it turns green when the Volume value is within tolerance and red when it is out of tolerance. Remember to set the association properly.



```
if($MyResults.Volume > $MyRecipe.VolumeMin AND $MyResults.Volume < $MyRecipe.VolumeMax,
    TRUE,
    FALSE)
```

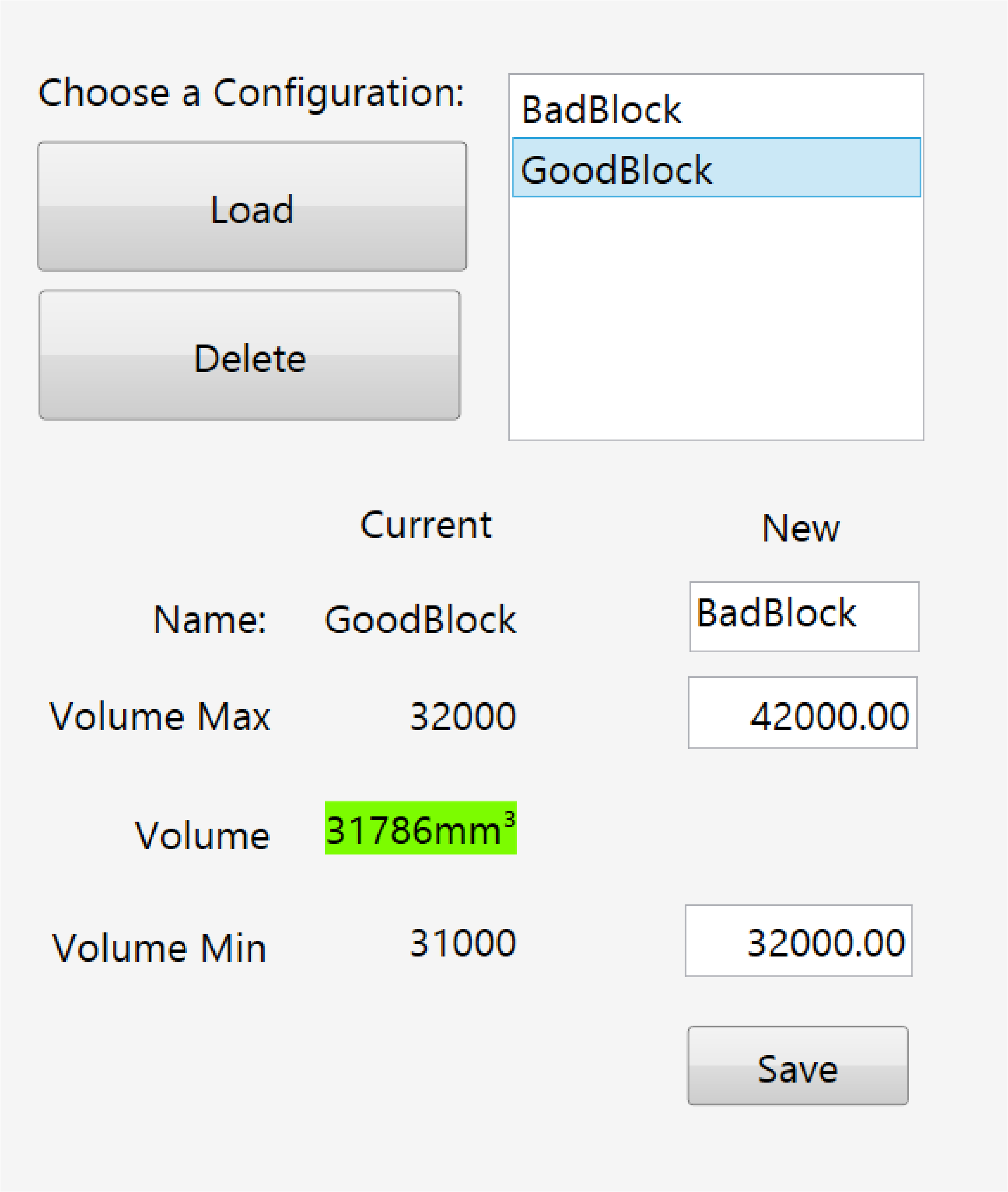


□ Black

False Red

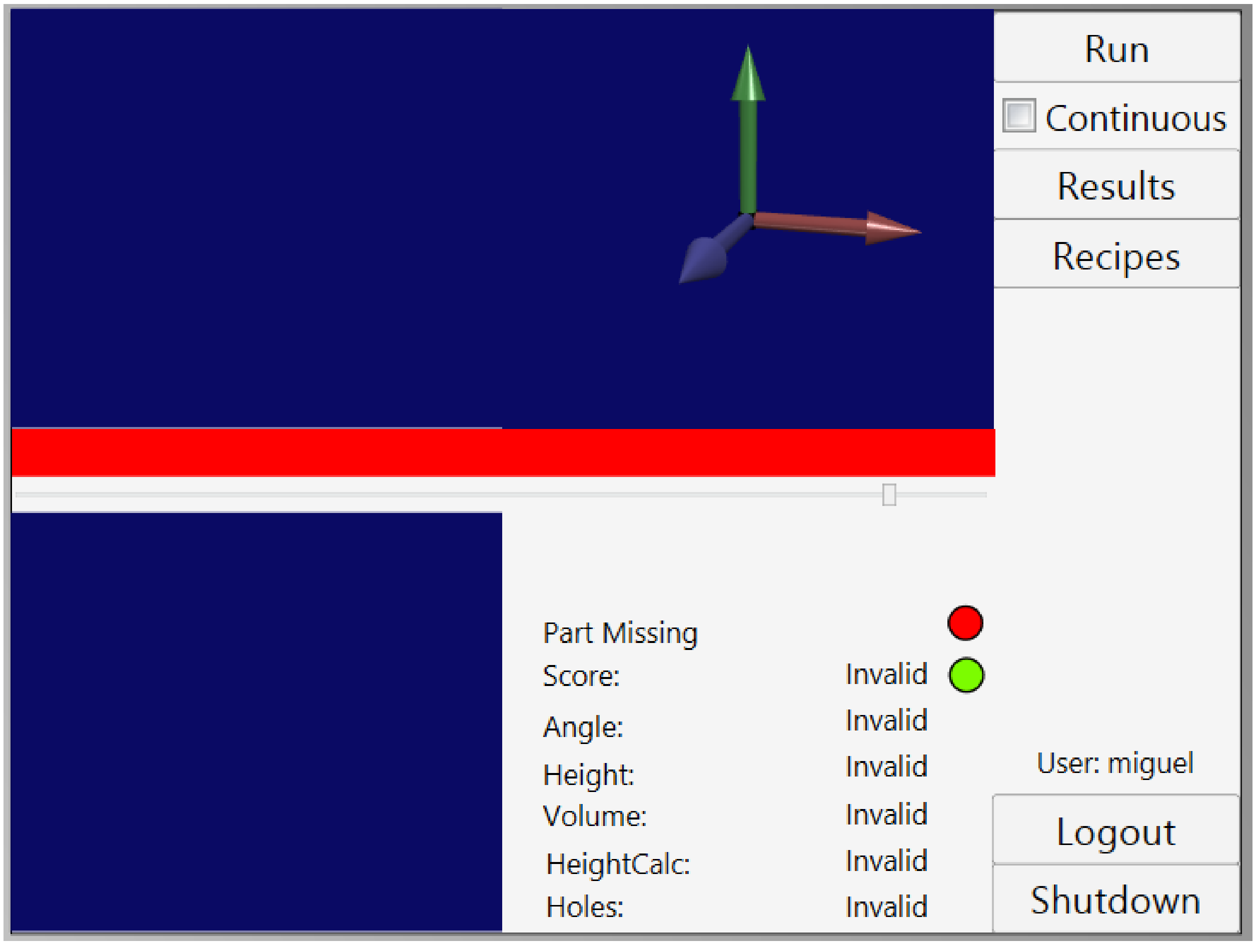
True LawnGreen

iii. Go into Test Mode and see how it works. Put meaningful values in for each part.



	Current	New
Name:	GoodBlock	BadBlock
Volume Max	32000	42000.00
Volume	31786mm^3	
Volume Min	31000	32000.00

v. Go to Test Mode and see if you get better results



Part Missing ●
Score: Invalid ●
Angle: Invalid
Height: Invalid
Volume: Invalid
HeightCalc: Invalid
Holes: Invalid

User: miguel

e. Save application

Challenge 1: Make use of the default $System.Info tags for some of the information. Also, add a confirmation dialog to the Delete to keep users from deleting configurations inadvertently.

Challenge 2: Use the example above to add support for checking Height tolerance.