```
 1 using System;
 2 using System.Data;
 3 using System.IO;
 4 using System.Linq;
 5 using System.Windows.Forms;
 6 using CASP_Standalone_Implementation.Forms;
 7 using CASP_Standalone_Implementation.Src;
 8 using System.Collections.Generic;
 9 using System.Text.RegularExpressions;
10 using Newtonsoft.Json;
11 using Newtonsoft.Json.Linq;
12
13 namespace CASP_Standalone_Implementation
14 {
15     public partial class MainForm : Form
16     {
17         public static string[] Languages
18         {
19             get
20             {
21                 string dir = ConsoleWrapper.CORE_DIR + "/cfg/";
22                 return Directory.GetFiles(dir)
23                     .Where(s => s.EndsWith(".cfg"))
24                     .Select(s => s.Replace(dir, "").Replace(".cfg", ""))
25                     .ToArray();
26             }
27         }
28
29         public static Dictionary<string, Type> Modules = new Dictionary<string, Type>() {
30             { "Analyze", typeof(CASP_AnalyzeForm) },
31             //{ "Lint", null },
32             { "Outline", typeof(CASP_OutlineForm) },
33             { "Print", typeof(CASP_PrintForm) },
34             { "Translate", typeof(CASP_TranslateForm) }
35         };
36
37         public static string TempFilename = "CASP_Temp_Src.tmp";
38
39         public string request;
40         private List<KeyValuePair<string, string>> customArgs = new List<KeyValuePair<string, string>>();
41
42
43         private void UpdateRequest()
44         {
45             try
46             {
47                 string module = ConsoleWrapper.GetArgument(ConsoleWrapper.ModuleId, ModuleCombo.SelectedItem.ToString());
48                 string srclang = ConsoleWrapper.GetArgument(ConsoleWrapper.SourceLanguage, InputLanguageCombo.SelectedItem.ToString());
49                 string code = ConsoleWrapper.GetArgument(ConsoleWrapper.CodeFile, TempFilename);
50                 //string code = ConsoleWrapper.GetArgument(ConsoleWrapper.CodeSnippet, InputTextbox.Text);
51                 List<string> requestData = customArgs.Select(kvp => ConsoleWrapper.GetArgument(kvp.Key, kvp.Value)).ToList();
52                 requestData.Insert(0, code);
53                 requestData.Insert(0, srclang);
54                 requestData.Insert(0, module);
55
56                 //request = ConsoleWrapper.GenerateRequest(module, srclang, code);
57                 request = ConsoleWrapper.GenerateRequest(requestData.ToArray());
58                 RequestTextbox.Text = "CASP " + request;
59
60             }
61             catch (Exception e)
62             {
63
64             }
65         }
66
```

```
67          private async void Execute()
68          {
69              if (!ConsoleWrapper.Running)
70              {
71                  string filename = ConsoleWrapper.CORE_DIR + "/" + TempFilename;
72                  File.WriteAllText(filename, InputTextbox.Text);
73
74                  SetExecute(true);
75                  ProgramStatus.Text = "Processing...";
76                  string output = await ConsoleWrapper.Execute(request);
77                  ProgramStatus.Text = "Ready (" + ((float)ConsoleWrapper.LastRunTime / 1000f) + "s)";
78                  SetExecute(false);
79
80                  if (ShowOutputCheckbox.Checked)
81                      new OutputForm(output).Show();
82
83                  Type T = Modules[ModuleCombo.SelectedItem.ToString()];
84                  if (T != null && T.IsSubclassOf(typeof(CASP_OutputForm)))
85                  {
86                      Regex reg = new Regex("CASP_RETURN_DATA_START(.*)CASP_RETURN_DATA_END", RegexOptions.Singleline);
87                      string jsonString = reg.Match(output).Groups[1].Value.Trim();
88                      JObject response = JsonConvert.DeserializeObject<JObject>(jsonString);
89
90                      if (response != null)
91                      {
92                          CASP_OutputForm form = (CASP_OutputForm)Activator.CreateInstance(T);
93                          form.Show();
94                          form.Set_CASP_Output(response);
95                      }
96                      else
97                      {
98                          response = JsonConvert.DeserializeObject<JObject>("{ \"Data\": {}, \"Warnings\": [], \"Errors\": [ { \"id\": -1, \"message\": \"CASP produced no valid output.\" }
99                      }
100
101                     ErrorProviderForm errorProvider = new ErrorProviderForm(response);
102                     if (errorProvider.NumErrors > 0 || errorProvider.NumWarnings > 0)
103                         errorProvider.Show();
104                     else
105                         errorProvider.Dispose();
106                 }
107
108                 File.Delete(filename);
109             }
110             else
111             {
112                 ConsoleWrapper.Kill();
113                 SetExecute(false);
114             }
115         }
116
117         public MainForm()
118         {
119             InitializeComponent();
120         }
121
122         private void MainForm_Load(object sender, EventArgs e)
123         {
124             ProgramStatus.Text = "Ready";
125
126             ModuleCombo.Items.AddRange(Modules.Select(entry => entry.Key).ToArray());
127             ModuleCombo.SelectedItem = ModuleCombo.Items[0];
128
129             InputLanguageCombo.Items.AddRange(Languages);
130             InputLanguageCombo.SelectedItem = InputLanguageCombo.Items[0];
131
132             UpdateRequest();
133         }
```

```
134
135         private void SelectedIndexChanged(object sender, EventArgs e)
136         {
137             UpdateRequest();
138         }
139
140         private void SetExecute(bool active)
141         {
142             ExecuteButton.Text = active ? "Stop" : "Execute Command";
143         }
144
145         private void ExecuteButton_Click(object sender, EventArgs e)
146         {
147             Execute();
148         }
149
150         private void InputTextbox_TextChanged(object sender, EventArgs e)
151         {
152
153         }
154
155         private void NewArgButton_Click(object sender, EventArgs e)
156         {
157             AddArgForm newArg = new AddArgForm(AddArgument);
158             newArg.Show();
159         }
160
161         private void AddArgument(string argName, string argValue)
162         {
163             customArgs.Add(new KeyValuePair<string, string>(argName, argValue));
164             UpdateArguments();
165         }
166
167         private void UpdateArguments()
168         {
169             OtherArgs.Items.Clear();
170             foreach (KeyValuePair<string, string> kvp in customArgs)
171             {
172                 OtherArgs.Items.Add(kvp.Key + ": " + kvp.Value);
173             }
174             UpdateRequest();
175         }
176
177         private void RemoveArgs_Click(object sender, EventArgs e)
178         {
179             ListBox.SelectedIndexCollection indices = OtherArgs.SelectedIndices;
180             for (int i = indices.Count - 1; i >= 0; i--)
181             {
182                 customArgs.RemoveAt(indices[i]);
183             }
184             UpdateArguments();
185         }
186     }
187 }
```