```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.ComponentModel;
 4  using System.Data;
 5  using System.Drawing;
 6  using System.Linq;
 7  using System.Text;
 8  using System.Threading.Tasks;
 9  using System.Threading;
10  using System.Windows.Forms;
11  using static SmokeScreen.Modules.Cryptography;
12
13  /* TODO
14   * OneWay Hash Password - Complete
15   * Login user / pass    - Complete
16   * store in array/DB    - Complete
17   * Diffee/Hellemen exc  - Complete <- Check SKIP 1024 bits
18   * 3 encryptions algor  - Complete
19   * File Transfer 2way   - TODO
20   */
21
22
23  namespace SmokeScreen
24  {
25      public partial class AuthForm : Form
26      {
27          public static string SymmetricKey = string.Empty;
28          public static string PublicKey = string.Empty;
29
30          private static readonly Color Default = Color.White;
31          private static readonly Color Warning = Color.FromArgb(245, 144, 66);
32
33          private readonly AsynchronousClient asyncClient;
34
35          public AuthForm()
36          {
37              InitializeComponent();
38              algBox.DataSource = Enum.GetValues(typeof(Algorithm));
39              SetStyle(ControlStyles.SupportsTransparentBackColor, true);
40              TransparentBackColor(
41                  authenicationLabel,
42                  passwordLabel,
43                  usernameLabel,
44                  usernameError,
45                  passwordError,
46                  exceptionError
47              );
48              signinButton.BackColor = Color.Azure;
49              asyncClient = new AsynchronousClient();
50          }
51
52          private void CreateAccountButton_Click(object sender, EventArgs e)
```

```
53          {
54              ClearErrors(usernameError, passwordError, exceptionError);
55              ClearErrors(usernameInput, passwordInput);
56              GetSymmetricKey();
57
58              bool isInvalid = false;
59
60              if (usernameInput.Text.Length < 8)
61              {
62                  usernameError.Text = "Username must be 8 chars in length";
63                  usernameInput.BackColor = Warning;
64                  isInvalid = true;
65              }
66              if (passwordInput.Text.Length < 8)
67              {
68                  passwordError.Text = "Password must be 8 chars in length";
69                  passwordInput.BackColor = Warning;
70                  isInvalid = true;
71              }
72              if (isInvalid)
73              {
74                  exceptionError.Text = $"Unable to Create an Account... Invalid
                      Input Detected";
75              }
76              else
77              {
78                  int truth = asyncClient.CreateAccount((Algorithm)
                      algBox.SelectedItem, SymmetricKey, PublicKey,
                      usernameInput.Text, HashPassword(passwordInput.Text));
79
80
81                  if (truth == 1)
82                  {
83                      usernameInput.Text = string.Empty;
84                      passwordInput.Text = string.Empty;
85                      exceptionError.Text = $"Account Created, You may now
                          authenicate...";
86                  }
87                  else if (truth == 2)
88                  {
89                      usernameError.Text = "Username must be 8 chars in length";
90                      usernameInput.BackColor = Warning;
91                  }
92                  else if (truth == 3)
93                  {
94                      usernameError.Text = "Username already Exists";
95                      usernameInput.BackColor = Warning;
96                  }
97                  else if (truth == 4)
98                      exceptionError.Text = $"Unable to Create an Account...
                          Database Error";
99                  else if (truth == 5)
```

```
100                     exceptionError.Text = $"Unable to Create an Account...
                         Transaction Error";
101                 else
102                 {
103                     exceptionError.Text = $"Unable to Create an Account...
                         Exception Occurred";
104                 }
105             }
106
107         }
108
109         private void SigninButton_Click(object sender, EventArgs e)
110         {
111             try
112             {
113                 ClearErrors(usernameError, passwordError, exceptionError);
114                 ClearErrors(usernameInput, passwordInput);
115                 GetSymmetricKey();
116
117                 int truth = asyncClient.Authenicate((Algorithm)
                      algBox.SelectedItem, SymmetricKey, PublicKey,
                      usernameInput.Text, HashPassword(passwordInput.Text));
118
119                 if (truth == 1)
120                 {
121                     LogIn((Algorithm)algBox.SelectedItem, usernameInput.Text,
                         SymmetricKey, PublicKey);
122                 }
123                 else if (truth == 2)
124                 {
125                     usernameError.Text = "Invalid Username";
126                     usernameInput.BackColor = Warning;
127                 }
128                 else if (truth == 3)
129                 {
130                     passwordError.Text = "Invalid Password";
131                     passwordInput.BackColor = Warning;
132                 }
133                 else if (truth == 4)
134                     exceptionError.Text = $"A Connection Issue has occured...
                         Transaction Error";
135                 else
136                 {
137                     exceptionError.Text = $"A Connection Issue has occured...
                         Exception Occured";
138
139                 }
140             }
141             catch(Exception ex)
142             {
143                 exceptionError.Text = ex.Message;
144             }
```

```
145
146         }
147
148         private void LogIn(Algorithm algorithm, string username, string
              symmetricKey, string publicKey)
149         {
150             using (MainForm mainForm = new MainForm(asyncClient, algorithm,
                  username, symmetricKey, publicKey))
151             {
152                 Hide();
153                 mainForm.ShowDialog();
154                 Close();
155             }
156         }
157
158         private static void ClearErrors(params Label[] list)
159         {
160             foreach (Label label in list)
161             {
162                 label.Text = "";
163             }
164         }
165
166         private static void ClearErrors(params TextBox[] list)
167         {
168             foreach (TextBox textBox in list)
169             {
170                 textBox.BackColor = Default;
171             }
172         }
173
174         private static void TransparentBackColor(params Label[] list)
175         {
176             foreach(Label label in list)
177             {
178                 label.BackColor = Color.Transparent;
179                 label.ForeColor = Color.Azure;
180             }
181         }
182
183         private void GetSymmetricKey()
184         {
185             if (SymmetricKey == string.Empty)
186             {
187                 SymmetricKey = asyncClient.SymmetricKeyExchange((Algorithm)
                      algBox.SelectedItem, out PublicKey);
188             }
189             if (SymmetricKey == null)
190             {
191                 throw new Exception("Cannot Communicate With the Server...
                      Invalid Symmmetric Key");
192             }
```

```
193         }
194
195         private string HashPassword(string password)
196         {
197             if (password != string.Empty)
198             {
199                 password = Sha256Hash.Generate(passwordInput.Text);
200                 return password;
201             }
202             return string.Empty;
203         }
204
205     }
206
207 }
208
209 //NOTE: You should be able to dynamically add new user/password combinations
      during the demonstration
210
211 /*
212 Once users are authenticated, they wish to be able to exchange an arbitrary
      number of communications with the manufacturer using symmetric encryption.  The
       following protocol has been agreed upon to determine an encryption algorithm
      and secret key.  You should be able to dynamically add new user/password
      combinations during the demonstration.
213
214 Protocol:
215 1)  Using a socket, customers contact the server (manufacturer) and login in for
      validation using their account name and password.
216 2)  Upon validation, the manufacturer and customer determine a secret session key
       via a two part process.  First they determine an initial secret key based on
      the work of Diffie-Helleman.  It has been decided to use the SKIP protocol with
       a 1024 bit key.  You may use a different protocol than Diffie-Helleman with
      permission from the instructor.  Once the initial key has been determined, you
      may utilize hashing or another technique of your choice to reduce it to a size
      appropriate (number of bytes) to initialize the symmetric key encryption
      algorithm.  The rest of the communications are completed using symmetric
      encryption.
217 3)  The client should determine the symmetric algorithm to be used and
      communicate their choice to the server.  The server should support at least
      three symmetric encryption Algorithm.  You are free to take advantage of code
      from examples used in class as long as you document the source (do not
      plagiarize).  You may not use code from other sources!
218
219 Implement the client and server.  Both client and server should print sufficient
      material to show they have:
220 1)  Properly completed the login protocol.
221 2)  Generated the same initial key using SKIP (or alternate technique).
222 3)  The client must encrypt a file and transmit it to the server.  The server
      must then decrypt the file.  You must convenience me this communication has
      been accomplished during a live demonstration.
223 4)  The server must encrypt a file (with different contents) and transmit it to
```

```
      the client.  The client must then decrypt the file.  You must convenience me
      this communication has been accomplished during a live demonstration.
224  */
225
```