OCC Task 2 - Development Documents

CARL BAINES

Contents

APIs 2

OpenWeather APIs 2

Modules 2

Newtonsoft.JSON 2

BCrypt.Net 3

Classes 3

User Class 3

Description 3

AccessibilityHelper Class 4

Description 4

Development Process 5

Form1.cs 5

Description 6

Changes During Development 6

Form2.cs 7

Description 8

Changes During Development 9

Form3.cs 9

Description 12

Changes During Development 12

Form4.cs (SCRAPPED) 13

Form5.cs 13

Description 16

Stored Procedure 16

Changes During Development 16

Form6.cs 17

Description 18

Stored Procedure 19

Changes During Development 19

Form7.cs 20
Description 22
Stored Procedure 23
Changes During Development 23
Form8.cs 24
Description 26

APIs

OPENWEATHER APIS

To develop my app, I have used three APIs. They were all provided by Open Weather. The first API that I used was Current Weather Data. It is an API that can be called to receive access to a various amount of real-time data. Examples of data you can call from the API includes weather type, temperature, forecasts etc. The second API that I used was Open Weather's Geocoding API. This API allows you to get geographical coordinates (latitude, longitude) by using the name of a location. Finally, the third API that I used was Open Weather's Pollution API. This API pulls real-time air pollution data. Examples of data you can call from the API includes air quality index, carbon monoxide content etc.

In terms of my app's functionality, the app pulls relevant weather data from the Open Weather APIs by passing the user-inputted location into it. The geographical coordinates of the user-inputted location are pulled from the Geocoding API which is then used to call data from the weather and pollution APIs.

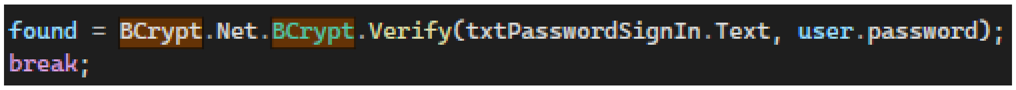
Modules

NEWTONSOFT.JSON

My app uses a module called Newtonsoft.Json. It is a popular and high-performance JSON framework for .NET. The main reason I used this module is because it makes parsing the JSON dictionary that the weather data from Open Weather is stored in very easy, as it provides JSON.NET objects such as JObject, JArray etc.

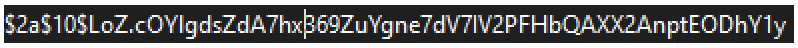
BCRYPT.NET

My app also uses a module called BCrypt.Net. It ports the jBCrypt library to C#. The main reason I used this module is because it makes hashing the passwords to be stored in the UserDetails database much easier as it requires only one line of code.



```
found = BCrypt.Net.BCrypt.Verify(txtPasswordSignIn.Text, user.password);
break;
```

This is an example of where I used the module. I used it to verify the user-inputted password by returning a hashed value of it.



```
$2a$10$LoZ.cOYlgdsZdA7hxB69ZuYgne7dV7lV2PFHbQAXX2AnptEODhY1y
```

This is an example of a hashed password which is stored in the database.

Classes

USER CLASS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Task2_Code_LL_000013680_Baines_C
{
    class User
    {
        public int Id { get; set; }

        public string username { get; set; }

        public string password { get; set; }
    }
}
```

Description

This is the code for the user class. It returns the name of the method that an ID, username and password are used in and assigns the values of the variables.

ACCESSIBILITYHELPER CLASS

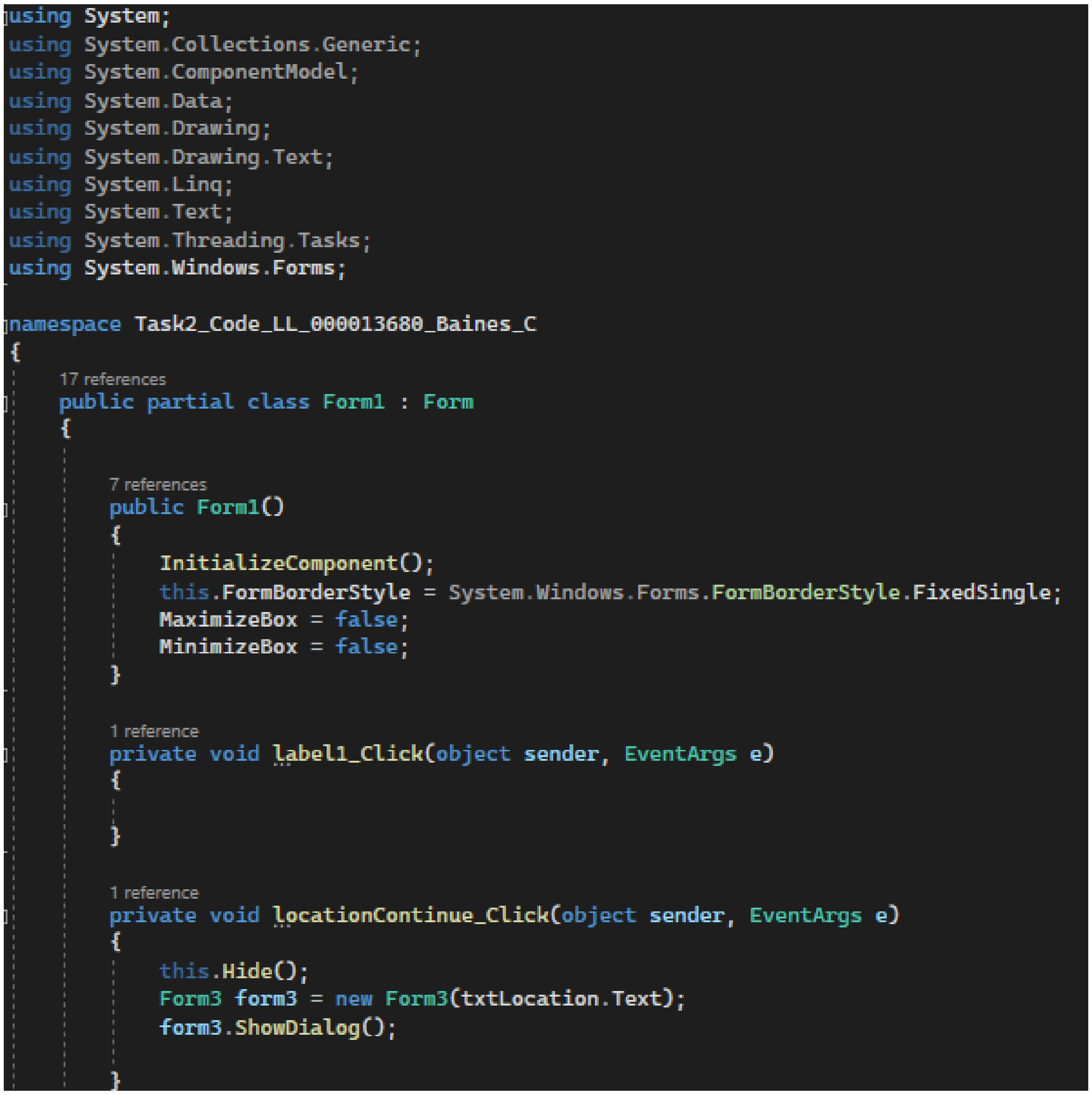
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Task2_Code_LL_000013680_Baines_C
{
    public class AccessibilityHelper
    {
        public float? fontSize { get; set; }

        public AccessibilityHelper() { }
        public AccessibilityHelper(float fontSize)
        {
            this.fontSize = fontSize;
        }

        public void UpdateFontSize(Control.ControlCollection controls)
        {
            if (fontSize == null) return;

            foreach (Control control in controls)
            {
                control.Font = new System.Drawing.Font(control.Font.Name, (float)fontSize);
            }
        }


    }
}
```

Description

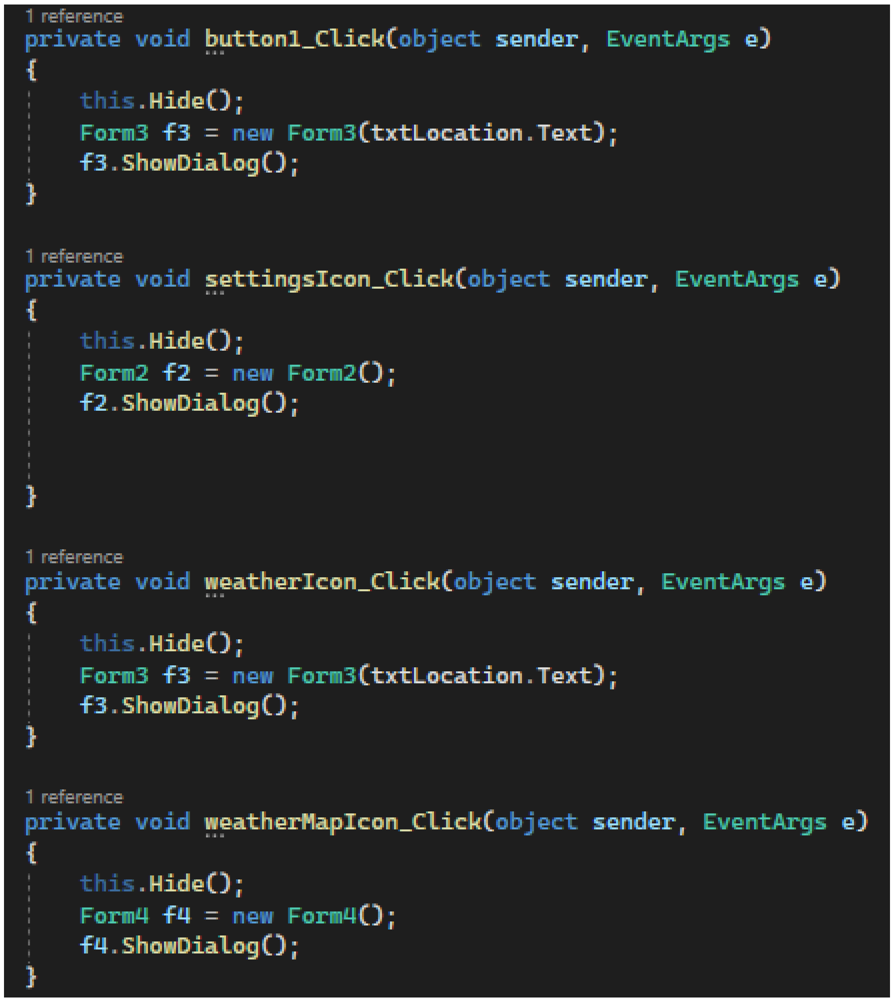
This is the code for the accessibilityHelper class. It creates a public class that can be referenced in the form's code that returns the name of a method that the variable fontSize is used in; assigns a value. The fontSize variable is then passed in the AccessibilityHelper method as a parameter and its value is updated. The UpdateFontSize method passes in the collection of control objects as a parameter and loops through the collection of control objects to assign the updated font size to them.

Development Process

FORM1.CS



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Text;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Task2_Code_LL_000013680_Baines_C
{
    17 references
    public partial class Form1 : Form
    {

        7 references
        public Form1()
        {
            InitializeComponent();
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            MaximizeBox = false;
            MinimizeBox = false;
        }

        1 reference
        private void label1_Click(object sender, EventArgs e)
        {

        }

        1 reference
        private void locationContinue_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form3 form3 = new Form3(txtLocation.Text);
            form3.ShowDialog();

        }
```

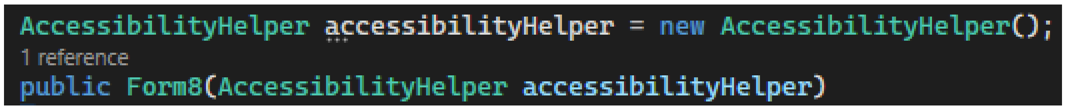


```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    this.Hide();
    Form3 f3 = new Form3(txtLocation.Text);
    f3.ShowDialog();
}

1 reference
private void settingsIcon_Click(object sender, EventArgs e)
{
    this.Hide();
    Form2 f2 = new Form2();
    f2.ShowDialog();


}

1 reference
private void weatherIcon_Click(object sender, EventArgs e)
{
    this.Hide();
    Form3 f3 = new Form3(txtLocation.Text);
    f3.ShowDialog();
}

1 reference
private void weatherMapIcon_Click(object sender, EventArgs e)
{
    this.Hide();
    Form4 f4 = new Form4();
    f4.ShowDialog();
}
```

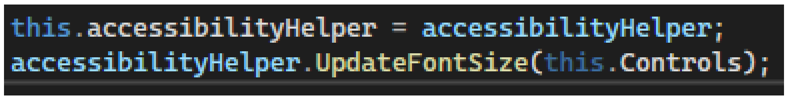
Description

This is the code for the home page on the app. The first part of the code sets the style for the window the app is displayed in, creating a fixed single border. This so the window cannot be resized. As well as this, I set ‘MaximiseBox’ to false to ensure that the window cannot be resized. For all the button clicks, I made it so that they redirect to their designated pages and close the previous page e.g., clicking the weather icon will redirect the user to the weather forecast page.

Changes During Development

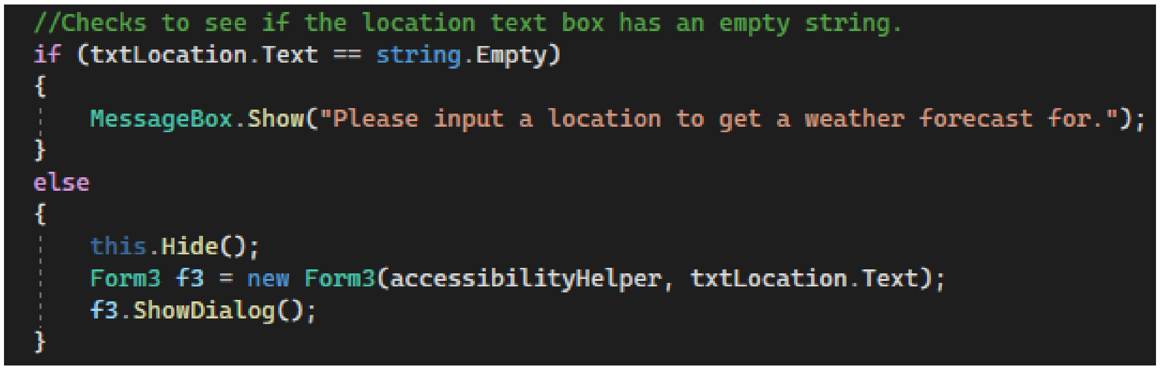


```
AccessibilityHelper accessibilityHelper = new AccessibilityHelper();
1 reference
public Form8(AccessibilityHelper accessibilityHelper)
```



```
this.accessibilityHelper = accessibilityHelper;
accessibilityHelper.UpdateFontSize(this.Controls);
```

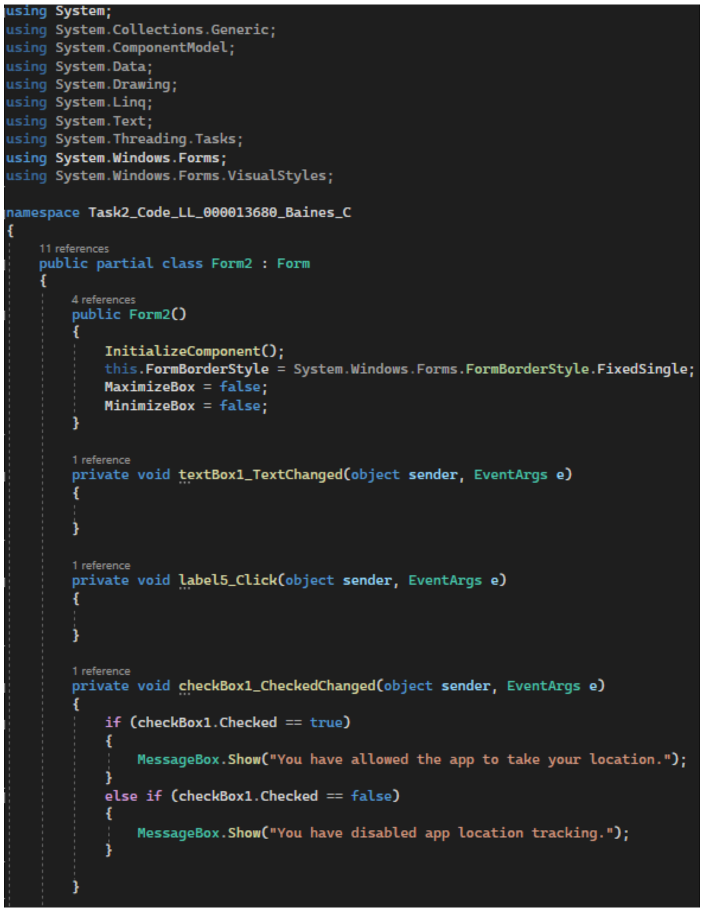
I added references to the AccessibilityHelper class later in the code's development so that the font size of the controls on the form could be changed based on whether the user changed the font size within the settings menu.



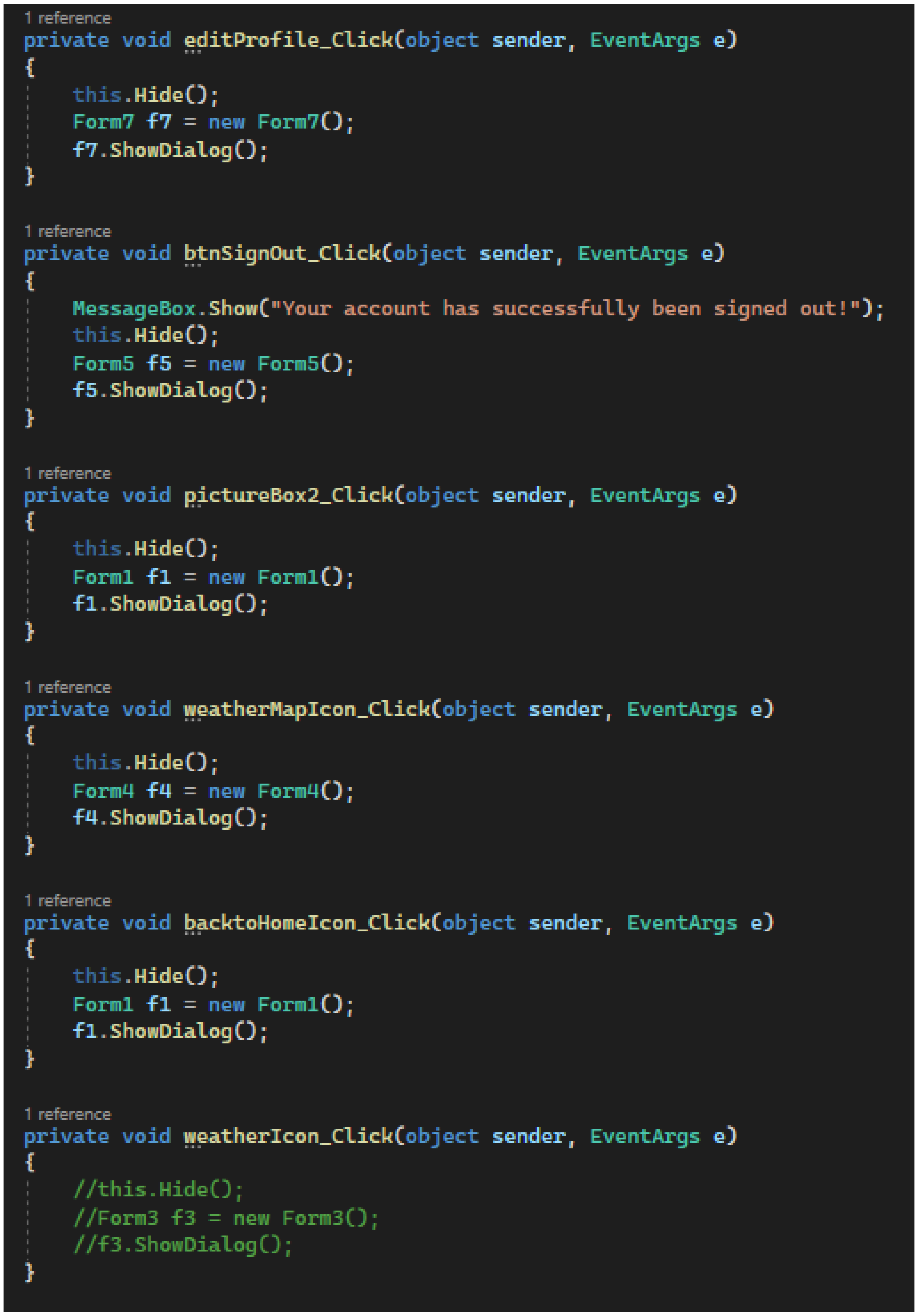
```
//Checks to see if the location text box has an empty string.
if (txtLocation.Text == string.Empty)
{
    MessageBox.Show("Please input a location to get a weather forecast for.");
}
else
{
    this.Hide();
    Form3 f3 = new Form3(accessibilityHelper, txtLocation.Text);
    f3.ShowDialog();
}
```

I also added validation make sure the user enters a location in the location text box before navigating to the weather forecast page. It checks to see if the text in the text box an empty string is. I added the same code for the icon and read more button clicks in order to ensure that the user cannot get to the weather forecast page without an inputted location.

FORM2.CS



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.VisualStyles;

namespace Task2_Code_LL_000013680_Baines_C
{
    11 references
    public partial class Form2 : Form
    {
        4 references
        public Form2()
        {
            InitializeComponent();
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            MaximizeBox = false;
            MinimizeBox = false;
        }

        1 reference
        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        1 reference
        private void label5_Click(object sender, EventArgs e)
        {

        }

        1 reference
        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked == true)
            {
                MessageBox.Show("You have allowed the app to take your location.");
            }
            else if (checkBox1.Checked == false)
            {
                MessageBox.Show("You have disabled app location tracking.");
            }

        }
```

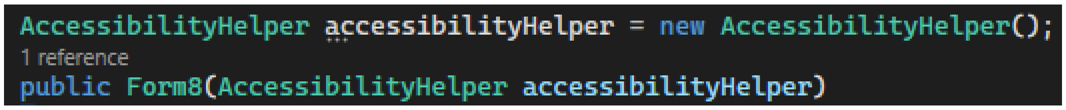


```
1 reference
private void editProfile_Click(object sender, EventArgs e)
{
    this.Hide();
    Form7 f7 = new Form7();
    f7.ShowDialog();
}

1 reference
private void btnSignOut_Click(object sender, EventArgs e)
{
    MessageBox.Show("Your account has successfully been signed out!");
    this.Hide();
    Form5 f5 = new Form5();
    f5.ShowDialog();
}

1 reference
private void pictureBox2_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 f1 = new Form1();
    f1.ShowDialog();
}

1 reference
private void weatherMapIcon_Click(object sender, EventArgs e)
{
    this.Hide();
    Form4 f4 = new Form4();
    f4.ShowDialog();
}

1 reference
private void backtoHomeIcon_Click(object sender, EventArgs e)
{
    this.Hide();
    Form1 f1 = new Form1();
    f1.ShowDialog();
}

1 reference
private void weatherIcon_Click(object sender, EventArgs e)
{
    //this.Hide();
    //Form3 f3 = new Form3();
    //f3.ShowDialog();
}
```

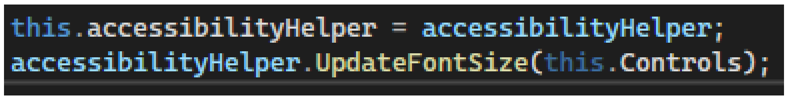
Description

This is the code for the settings page of the app. The first part of the code styles the window the app is displayed in identical to the home page. The checkbox part of the code is a design feature in the settings menu that displays messages based on the app's location tracking. However, the app does not have the functionality to track the user's location, therefore it is just there as an exhibit. The final part of the code is for the button clicks where I made it so that they redirect to their designated pages. The sign out button is coded so that on click, the user is redirected back to the sign in page.

Changes During Development



```
AccessibilityHelper accessibilityHelper = new AccessibilityHelper();
1 reference
public Form8(AccessibilityHelper accessibilityHelper)
```



```
this.accessibilityHelper = accessibilityHelper;
accessibilityHelper.UpdateFontSize(this.Controls);
```

I added references to the AccessibilityHelper class later in the code's development so that the font size of the controls on the form could be changed based on whether the user changed the font size within the settings menu.

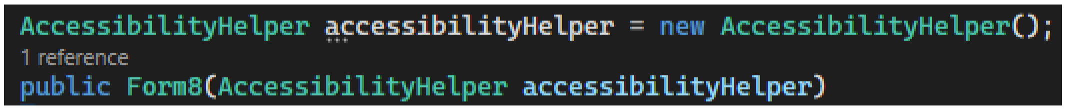
FORM3.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Newtonsoft.Json.Linq;

namespace Task2_Code_LL_000013680_Baines_C
{
    public partial class Form3 : Form
    {
        public string location { get; set; }
        public HttpClient httpClient { get; set; }
        //API key for OpenWeather APIs
        public string APIKey = "b1fe4659e300363e368f4ce2cf007fac";

        public Form3(string location)
        {
            InitializeComponent();
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            MaximizeBox = false;
            MinimizeBox = false;
            txtForecastLocation.Text = "Location: " + location.ToString().ToUpper();

            //Creates a new http client in its class.
            httpClient = new HttpClient();
            this.location = location;
        }

        private void lblWeatherMap_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form4 form4 = new Form4();
            form4.ShowDialog();

        }

        public async Task<List<float>> GetCoordinates(string region)
        {
            //Pulls data from the geolocation API to get the coordinates of the user-inputted location.
            HttpResponseMessage response = await
httpClient.GetAsync($"http://api.openweathermap.org/geo/1.0/direct?q={region}&limit=5&appid={APIKey}");
            string payload = await response.Content.ReadAsStringAsync();

            JArray payloadObject = JArray.Parse(payload);

            return new List<float>()
            {
                (float)payloadObject[0]["lat"],
                (float)payloadObject[0]["lon"]
            };
        }

        private async void button1_Click(object sender, EventArgs e)
        {
            List<float> coordinates = await GetCoordinates(this.location);

            HttpResponseMessage response = await
httpClient.GetAsync($"https://api.openweathermap.org/data/2.5/weather?lat={coordinates[0]}&lon={coordina
tes[1]}&appid={APIKey}");
            string payload = await response.Content.ReadAsStringAsync();

            JObject payloadObject = JObject.Parse(payload);

            //Gets main weather type from API dictionary and outputs it e.g. sunny, clouds.
            txtWeather.Text = (string)payloadObject["weather"][0]["description"];
            MessageBox.Show((string)payloadObject["weather"][0]["description"]);

        }

        private void weatherIcon_Click(object sender, EventArgs e)
        {

        }
        private void weatherMapIcon_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form4 f4 = new Form4();
            f4.ShowDialog();
        }

        private void slidersIcon_Click(object sender, EventArgs e)
        {

        }

        private void homeIcon_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 f1 = new Form1();
            f1.ShowDialog();
        }

        private void settingsIcon_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form2 f2 = new Form2();
            f2.ShowDialog();
        }

        private void Form3_Load(object sender, EventArgs e)
        {

        }

        private async void button2_Click(object sender, EventArgs e)
        {
            List<float> coordinates = await GetCoordinates(this.location);

            HttpResponseMessage response = await
httpClient.GetAsync($"https://api.openweathermap.org/data/2.5/weather?lat={coordinates[0]}&lon={coordina
tes[1]}&appid={APIKey}&units=metric");
            string payload = await response.Content.ReadAsStringAsync();

            JObject payloadObject = JObject.Parse(payload);

            //Gets temperature in celcius from API dictionary.
            txtTemperature.Text = (string)payloadObject["main"]["temp"] + "°c";
            MessageBox.Show((string)payloadObject["main"]["temp"]);
        }


    }
}
```

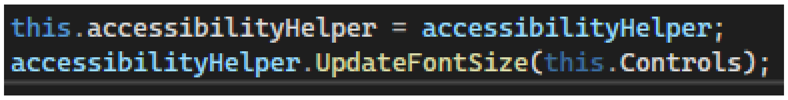
Description

This code is for the weather forecast page of the app. The first part of the code uses a public class to get the user-inputted location from form1 which is the homepage of the app. It also creates a HTTP client which is used to host APIs, the API key for the Open Weather APIs used later in the code is also assigned. The next part of the code passes in the location as a parameter and styles the form the app id displayed in. Then, the code pulls data from Open Weather's geolocation API to get the coordinates of the user-inputted location from form 1. It returns the latitude and longitude as a new list. The button1 and button2 clicks use the Open Weather API to pull weather data e.g., temperature, main type of weather and outputs it to be displayed on the app.

Changes During Development



```
AccessibilityHelper accessibilityHelper = new AccessibilityHelper();
1 reference
public Form8(AccessibilityHelper accessibilityHelper)
```



```
this.accessibilityHelper = accessibilityHelper;
accessibilityHelper.UpdateFontSize(this.Controls);
```

I added references to the AccessibilityHelper class later in the code's development so that the font size of the controls on the form could be changed based on whether the user changed the font size within the settings menu.

```
private async void btnAirQuality_Click(object sender, EventArgs e)
    {
        List<float> coordinates = await GetCoordinates(this.location);

        HttpResponseMessage response = await
httpClient.GetAsync($"https://api.openweathermap.org/data/2.5/air_pollution?lat={coordinates[0]}&lon={coo
rdinates[1]}&appid={APIKey}&units=metric");
        string payload = await response.Content.ReadAsStringAsync();

        string payload2 = await response.Content.ReadAsStringAsync();

        JObject payloadObject = JObject.Parse(payload);

        JObject payloadObject2 = JObject.Parse(payload2);

        //Gets air quality index from API dictionary.
        txtAirQuality.Text = ("Air Quality Index: " + (string)payloadObject["list"][0]["main"]["aqi"] + "\t" +
"Carbon Monoxide: " + (string)payloadObject2["list"][0]["components"]["co"]);
        MessageBox.Show("AQI: " + (string)payloadObject["list"][0]["main"]["aqi"] + "\n" + "CO: " +
(string)payloadObject2["list"][0]["components"]["co"]);

    }
```

Later in development, I added a button click event to get the air quality index and carbon monoxide content from Open Weather's pollution API.

FORM4.CS (SCRAPPED)

Form4 was scrapped during development. It was supposed to be a weather map page, but I had difficulty finding a weather map API that was free and I also did not know how it could be integrated into a windows form application.

FORM5.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace Task2_Code_LL_000013680_Baines_C
{
    public partial class Form5 : Form
    {
        public Form5()
        {
            InitializeComponent();
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            MaximizeBox = false;
            MinimizeBox = false;
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void btnSignin_Click(object sender, EventArgs e)
        {

            //Tells the code where the database is
            string connectionString = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDbFilename =
C:\\Users\\M2202766\\OneDrive - Middlesbrough College\\Documents\\Year 2\\Week 16\\OSC Mock\\Task
```

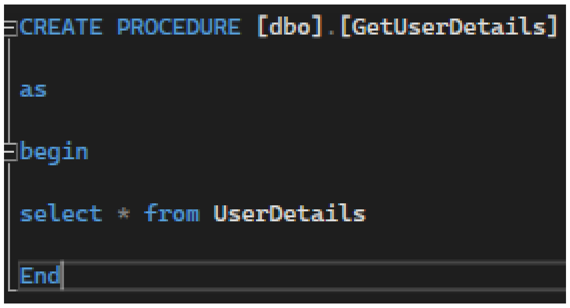
```
2\\Task2_Code_LL-000013680_Baines_C\\Task2_Code_LL-000013680_Baines_C\\userDetails.mdf; Integrated
Security = True; Connect Timeout = 30";

        SqlConnection sqlConnection = new SqlConnection(connectionString);

        //Creates an empty list
        List<User> userDetailsList = new List<User>();

        //Sets up stored procedure
        SqlCommand cmd = new SqlCommand("GetUserDetails", sqlConnection);

        cmd.CommandType = CommandType.StoredProcedure;

        //Executes stored procedure
        SqlDataAdapter sd = new SqlDataAdapter(cmd);

        DataTable dt = new DataTable();


       sqlConnection.Open();

        sd.Fill(dt);

        sqlConnection.Close();


        //Gets every field out of the person record and puts it into the people list.
        foreach (DataRow dr in dt.Rows)

        {

          userDetailsList.Add(

          new User()
          {

            Id = Convert.ToInt32(dr["Id"]),

            username = Convert.ToString(dr["Username"]),

            password = Convert.ToString(dr["Password"])



          });

        }

        //Take username and password inputs from form.

        string UserInputtedUsername = txtUsernameSignIn.Text;

        string UserInputtedPassword = txtPasswordSignIn.Text;
```

```
            //Checks to see if username and password are in the userDetails list.

            //Sets up a temporary boolean for search purposes.
            bool found = false;

            //Loops through each user in the database
            foreach (User user in userDetailsList)
            {
                //Checks if the user inputted username and password are equal to those stored in the database
                if (UserInputtedUsername == user.username && UserInputtedPassword == user.password)
                {
                    found = true; break;


                }
            }

            if (found)
            {
                MessageBox.Show("You have successfully signed in");

                //Hides sign in page and loads the app's homepage
                this.Hide();
                Form1 f1 = new Form1();
                f1.ShowDialog();
            }

            else
            {
                MessageBox.Show("Invalid login details. Please try again!");
                //Clears the input fields so the user can retry entering valid login details.
                txtUsernameSignIn.Clear();
                txtPasswordSignIn.Clear();
            }

        }

        private void lblSignUp_Click(object sender, EventArgs e)
        {
            //Navigates the user to the sign up page and closes the sign in page.
            this.Hide();
            Form6 f6 = new Form6();
            f6.ShowDialog();
        }

        private void Form5_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Description

This is the code for the sign in page of the app. The first part of the code sets the style for the window the app is displayed in. The second part of the code assigns a connection string to connect the code to a database. It then creates an empty list to store user details e.g., their username and password, creates an empty data table and sets up a stored procedure which is executed. The stored procedure runs an SQL query statement to select all user details from the database. The next part of the code executes a for loop which loops through the rows in the connected database and adds the data to the empty list created earlier. The data is converted from the database to their required data types as a new user in the user class. Then, the code takes the username and password inputs from the sign in page and loops through the user details list to check if the user inputs are equal to any stored in the database. If the usernames and passwords are found, the code redirects the user to the home page, otherwise it produces an error message. From this page, the code also allows the user to navigate to the sign-up page.

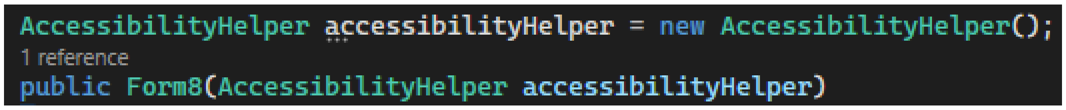
Stored Procedure



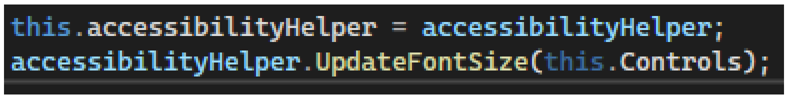
```
CREATE PROCEDURE [dbo].[GetUserDetails]

as

begin

select * from UserDetails

End
```

The stored procedure runs an SQL query statement to select all user details from the database.

Changes During Development

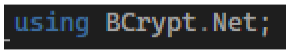


```
AccessibilityHelper accessibilityHelper = new AccessibilityHelper();
1 reference
public Form8(AccessibilityHelper accessibilityHelper)
```

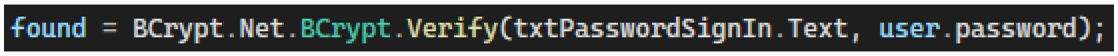


```
this.accessibilityHelper = accessibilityHelper;
accessibilityHelper.UpdateFontSize(this.Controls);
```

I added references to the AccessibilityHelper class later in the code's development so that the font size of the controls on the form could be changed based on whether the user changed the font size within the settings menu.



```
using BCrypt.Net;
```



```
found = BCrypt.Net.BCrypt.Verify(txtPasswordSignIn.Text, user.password);
```

I also added a method of validation to allow the user-inputted password to be hashed to be compared against the hashed password stored in the database. This uses the module BCrypt.

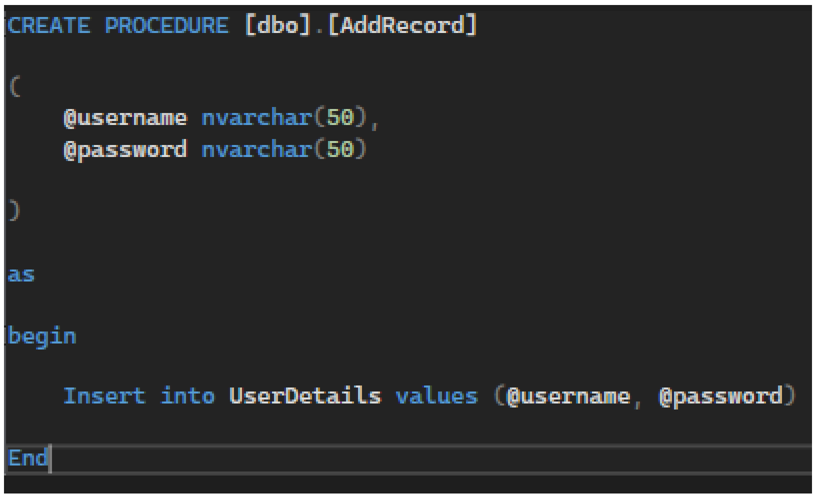
FORM6.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Security.Principal;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Task2_Code_LL_000013680_Baines_C
{
    public partial class Form6 : Form
    {
        public Form6()
        {
            InitializeComponent();
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            MaximizeBox = false;
            MinimizeBox = false;
        }

        private void lblBacktoSignin_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form5 f5 = new Form5();
            f5.ShowDialog();
        }

        private void btnSignUp_Click(object sender, EventArgs e)
        {
            //Tells the code where the database is
            string connectionString = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDbFilename =
C:\\Users\\M2202766\\OneDrive - Middlesbrough College\\Documents\\Year 2\\Week 16\\OSC Mock\\Task
2\\Task2_Code_LL-000013680_Baines_C\\Task2_Code_LL-000013680_Baines_C\\userDetails.mdf; Integrated
Security = True; Connect Timeout = 30";

            SqlConnection sqlConnection = new SqlConnection(connectionString);

            //Use stored procedure
            SqlCommand command = new SqlCommand("AddRecord", sqlConnection);

            command.CommandType = CommandType.StoredProcedure;

            //Takes username and password from the sign up form.
            string username = txtUsernameSignUp.Text;

            string password = txtPasswordSignUp.Text;

            //Call stored procedure and pass in username and password as parameters

            command.Parameters.AddWithValue("@Username", username);

            command.Parameters.AddWithValue("@Password", password);

            //Opens connection to the database, executes stored procedure and closes the connection.
            sqlConnection.Open();

            command.ExecuteNonQuery();

            sqlConnection.Close();

            MessageBox.Show("Account successfully created");

            //Opens the home screen if the user successfully creates an account.

            this.Hide();
            Form1 f1 = new Form1();

            f1.ShowDialog();
        }

        private void Form6_Load(object sender, EventArgs e)
        {

        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Description

This code is for the sign-up page of the app. It functions almost identically to the sign-in page apart from the fact that it adds a user-inputted username and password to the database instead of searching for them. It opens the home screen if the user successfully creates an account.

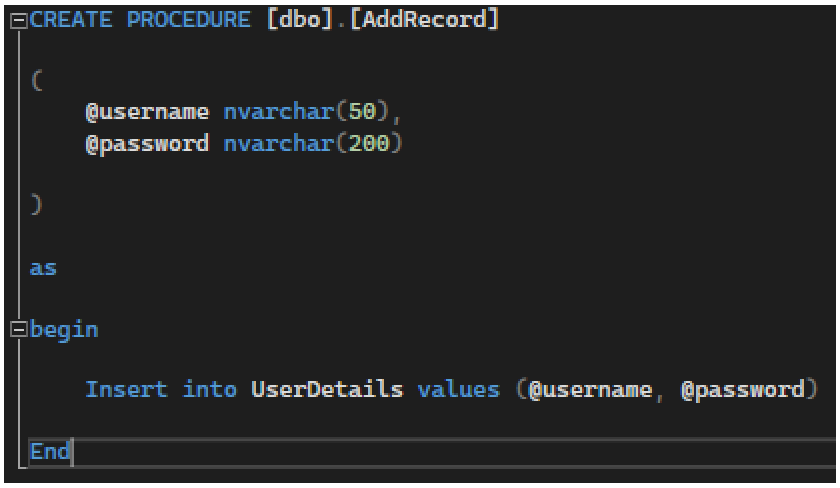
Stored Procedure



```
CREATE PROCEDURE [dbo].[AddRecord]

(
    @username nvarchar(50),
    @password nvarchar(50)

)

as

begin

    Insert into UserDetails values (@username, @password)

End
```

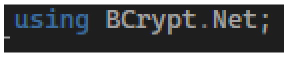
The stored procedure sets a username and password as parameters. They are assigned as strings with a max length of 50 characters. They are inserted into the UserDetails database.

Changes During Development

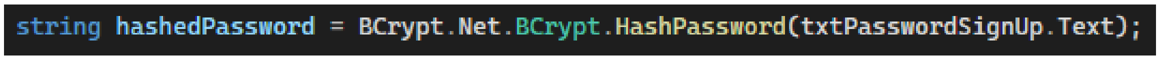


```
CREATE PROCEDURE [dbo].[AddRecord]

(
    @username nvarchar(50),
    @password nvarchar(200)

)

as

begin

    Insert into UserDetails values (@username, @password)

End
```

During development, I changed the length of the password parameter to have a max length of 200 characters. I did this after I made a hash function to allow for the password to be hashed and stored in the database.



```
using BCrypt.Net;
```



```
string hashedPassword = BCrypt.Net.BCrypt.HashPassword(txtPasswordSignUp.Text);
```

I also added a method of validation to allow the user-inputted password to be hashed when stored in the UserDetails database. This uses the module BCrypt.

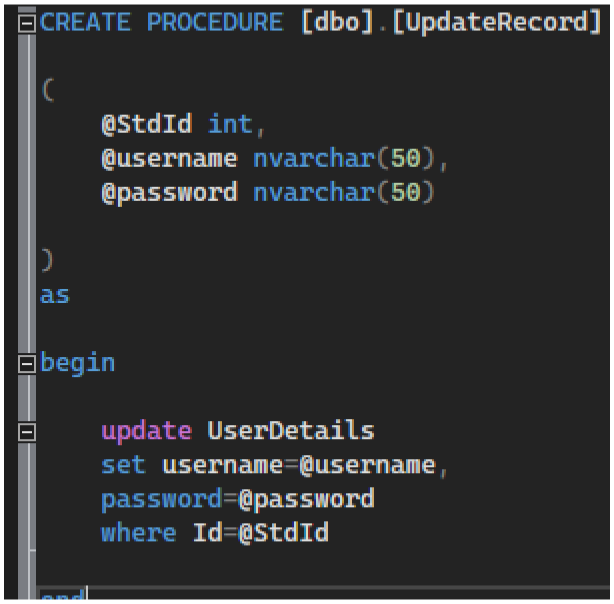
FORM7.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace Task2_Code_LL_000013680_Baines_C
{
    public partial class Form7 : Form
    {
        AccessibilityHelper accessibilityHelper = new AccessibilityHelper();

        public string location;
        public Form7(AccessibilityHelper accessibilityHelper)
        {
            InitializeComponent();
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            MaximizeBox = false;
            MinimizeBox = false;

            this.accessibilityHelper = accessibilityHelper;
            accessibilityHelper.UpdateFontSize(this.Controls);
        }

        private void weatherMapIcon_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form4 f4 = new Form4(accessibilityHelper);
            f4.ShowDialog();
        }

        private void backtoHomeIcon_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 f1 = new Form1(accessibilityHelper);
            f1.ShowDialog();
        }

        private void weatherIcon_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form3 f3 = new Form3(accessibilityHelper, location);
            f3.ShowDialog();
        }

        private void Form7_Load(object sender, EventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            //Tells the code where the database is
            string connectionString = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDbFilename =
C:\\Users\\M2202766\\OneDrive - Middlesbrough College\\Documents\\Year 2\\Week 16\\OSC Mock\\Task
2\\Task2_Code_LL-000013680_Baines_C\\Task2_Code_LL-000013680_Baines_C\\userDetails.mdf; Integrated
Security = True; Connect Timeout = 30";

            SqlConnection sqlConnection = new SqlConnection(connectionString);

            //Use stored procedure

            SqlCommand command = new SqlCommand("UpdateRecord", sqlConnection);

            command.CommandType = CommandType.StoredProcedure;

            //Takes the changed username and password from the form

            string id = txtID.Text;
            string updatedUsername = txtChangePassword.Text;
            string updatedPassword = txtChangePassword.Text;

            command.Parameters.AddWithValue("@StdId", id);
            command.Parameters.AddWithValue("@Username", updatedUsername);
            command.Parameters.AddWithValue("@Password", updatedPassword);

            sqlConnection.Open();

            command.ExecuteNonQuery();

            MessageBox.Show("Your username and password have successfully been changed.");

            sqlConnection.Close();









        }

        private void backIcon_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form2 f2 = new Form2(accessibilityHelper);
            f2.ShowDialog();
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

Description

This is the code for the second settings page of the app that allows the user to change their username and password. The first part of the code creates a new form of the AccessibilityHelper class.

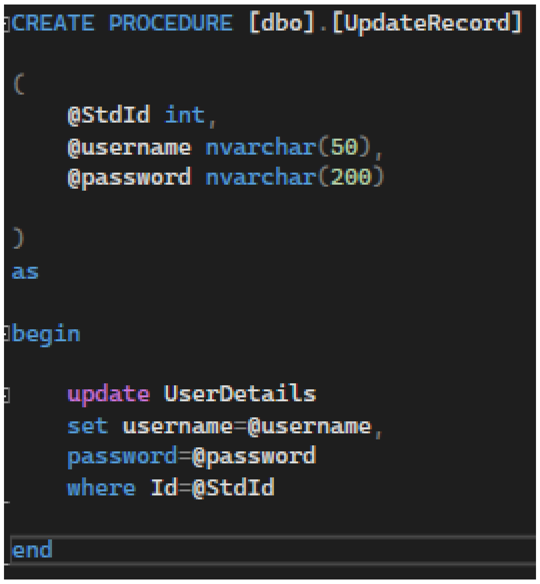
Stored Procedure



```
CREATE PROCEDURE [dbo].[UpdateRecord]

(
    @StdId int,
    @username nvarchar(50),
    @password nvarchar(50)

)
as

begin

    update UserDetails
    set username=@username,
    password=@password
    where Id=@StdId

end
```

The stored procedure takes the currentID, username and password as parameters. It assigns an integer value to the currentID, and strings with a max length of fifty characters for both the username and password. It updates the UserDetails database by setting the username and password column to their updated values where the ID is equal to currentID.

Changes During Development



```
CREATE PROCEDURE [dbo].[UpdateRecord]

(
    @StdId int,
    @username nvarchar(50),
    @password nvarchar(200)

)
as

begin

    update UserDetails
    set username=@username,
    password=@password
    where Id=@StdId

end
```

I changed the stored procedure to change the length of the password parameter to a max length of 200 characters. I did this to allow for hashing in case the length of the hash keys stored would be too big.

```
//Takes the changed username and password from the form

          string id = txtID.Text;
          string updatedUsername = txtChangeUsername.Text;
          string updatedPassword = txtChangePassword.Text;

          //temporary boolean for search purposes
          bool found = false;

          //Checks to see if any of the fields are empty.
          if (txtID.Text == string.Empty || txtChangeUsername.Text == string.Empty || txtChangePassword.Text ==
string.Empty)
          {
             found = true;
          }

          if (found)
          {
             MessageBox.Show("Please enter a valid ID, username and password");
          }

          else
          {
             command.Parameters.AddWithValue("@StdId", id);
             command.Parameters.AddWithValue("@Username", updatedUsername);
             command.Parameters.AddWithValue("@Password", updatedPassword);

             sqlConnection.Open();

             command.ExecuteNonQuery();

             MessageBox.Show("Your username and password have successfully been changed. Please sign back
in");

             sqlConnection.Close();

             this.Hide();
             Form5 f5 = new Form5(accessibilityHelper);
             f5.ShowDialog();

          }
```

Later in development, I added validation to the middle section of the code. I did this to check if any of the fields were empty in which the user had to enter details to update a record in the userDetails database.

FORM8.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Task2_Code_LL_000013680_Baines_C
{
    public partial class Form8 : Form
    {
        AccessibilityHelper accessibilityHelper = new AccessibilityHelper();
        public Form8(AccessibilityHelper accessibilityHelper)
        {
            InitializeComponent();
            MaximizeBox = false;
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

            this.accessibilityHelper = accessibilityHelper;
            accessibilityHelper.UpdateFontSize(this.Controls);
        }

        private void Form8_Load(object sender, EventArgs e)
        {

        }

        private void homeIcon_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 f1 = new Form1(accessibilityHelper);
            f1.ShowDialog();
        }

        private void backToPage_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form2 f2 = new Form2(accessibilityHelper);
            f2.ShowDialog();
        }

        private void weatherIcon_Click(object sender, EventArgs e)
        {
            //this.Hide();
            //Form3 f3 = new Form3();
            //f3.ShowDialog();
        }

        private void weatherMapIcon_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form4 f4 = new Form4(accessibilityHelper);
            f4.ShowDialog();
        }

        private void btnNewFontSize_Click(object sender, EventArgs e)
        {
            accessibilityHelper.fontSize = float.Parse(txtChangeFontSize.Text);
            accessibilityHelper.UpdateFontSize(this.Controls);
        }
    }
}
```

Description

This code is for the third settings page of the app where it gives the user an accessibility option of changing the font size of the app's content. The first part of the code creates a new form of the AccessibilityHelper class and sets the style of the form the app is displayed on. It also calls the UpdateFontSize method stored in the AccessibilityHelper class and passes in the collection of controls in the form as a parameter. For all the button clicks, I made it so that they redirect to their designated pages and close the previous page e.g., clicking the weather icon will redirect the user to the weather forecast page. The exception was for the new font size button click. It takes the user's input from the text box in the form and uses it to pass a parameter into the UpdateFontSize method to be ran.