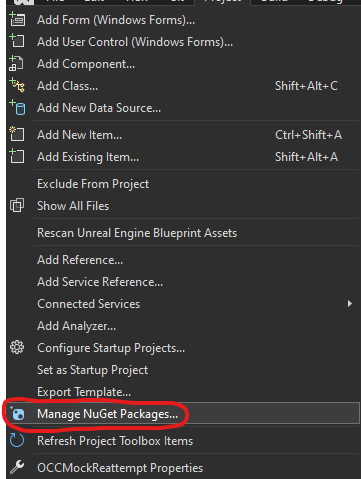
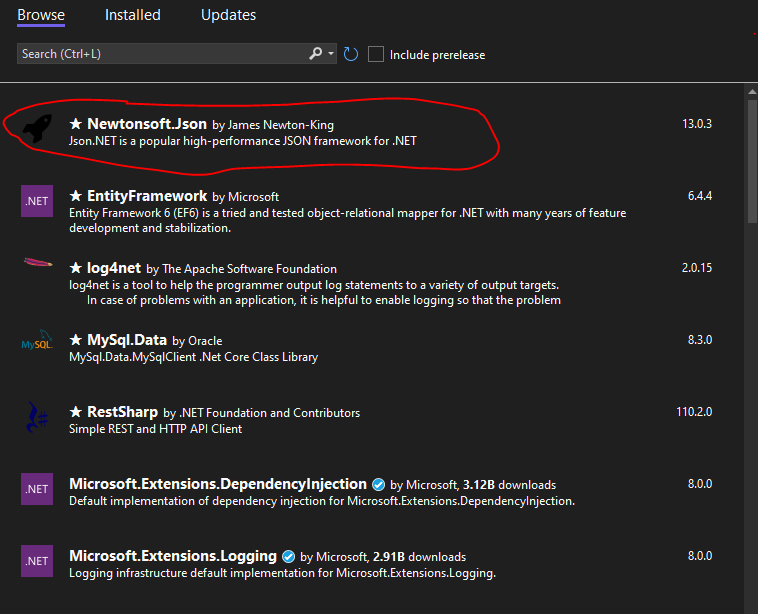
**Reattempt code:**

***Install NuGet Package:***







***Account table:***

CREATE TABLE [dbo].[Account] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Username] NVARCHAR (MAX) NULL,

[Password] NVARCHAR (MAX) NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

***Conditions table:***

CREATE TABLE [dbo].[Condition] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Name] NVARCHAR (50) NULL,

[Description] NVARCHAR (MAX) NULL,

[Symptoms] NVARCHAR (MAX) NULL,

[Treatment] NVARCHAR (MAX) NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

***Procedures:***

CREATE PROCEDURE [dbo].[GetNewAccount]

@Username nvarchar(50),

@Password nvarchar(50)

AS

BEGIN

INSERT INTO Account Values (@Username, @Password)

END

CREATE PROCEDURE [dbo].[GetAccountDetails]

AS

BEGIN

SELECT \* FROM Account

END

CREATE PROCEDURE [dbo].[UpdateAccount]

@Username nvarchar(50),

@Password nvarchar(50)

AS

BEGIN

Update Account

set Password=@Password

where Username=@Username

END

CREATE PROCEDURE [dbo].[DeleteAccount]

@Username nvarchar(50)

AS

BEGIN

DELETE FROM Account where Username = @Username

END

CREATE PROCEDURE [dbo].[GetConditionDetails]

AS

BEGIN

SELECT \* FROM Condition

END

***Account.cs:***

namespace OCCMockReattempt

{

internal class Account

{

public string username { get; set; }

public string password { get; set; }

}

}

***Condition.cs:***

namespace OCCMockReattempt

{

internal class Condition

{

public string name { get; set; }

public string description { get; set; }

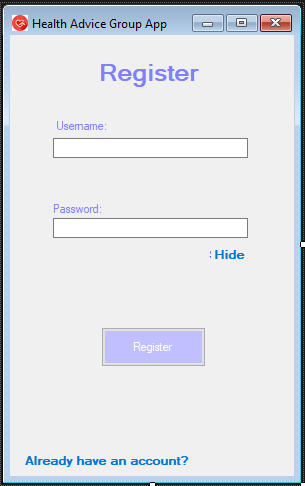
public string symptoms { get; set; }

public string treatment { get; set; }

}

}

***Form1.cs:***



namespace OCCMockReattempt

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void HAG\_register\_Load(object sender, EventArgs e)

{

//Hide register password

txtPasswordR.UseSystemPasswordChar = true;

//Clear the textboxes

txtUsernameR.Text = null;

txtPasswordR.Text = null;

}

private void btnRegister\_Click(object sender, EventArgs e)

{

//Input username and password

string usernameR = txtUsernameR.Text;

string passwordR = txtPasswordR.Text;

//Checking for null value in usernameR variable

if (string.IsNullOrEmpty(usernameR))

{

//Clear textboxes

txtUsernameR.Text = null;

txtPasswordR.Text = null;

//Display error message

MessageBox.Show("A value must be entered for the username");

}

else

{

if (string.IsNullOrEmpty(passwordR))

{

//Clear textboxes

txtUsernameR.Text = null;

txtPasswordR.Text = null;

//Display error message

MessageBox.Show("A value must be entered for the password");

}

else

{

if (passwordR.Length < 8)

{

//Clear textboxes

txtUsernameR.Text = null;

txtPasswordR.Text = null;

//Display error message

MessageBox.Show("The password must be at least 8 characters long");

}

else

{

//Create a list

List<Account> accountList = new List<Account>();

//Connect to database

string connectionString = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDBFilename = C:\\Users\\M2201454\\OneDrive - Middlesbrough College\\Documents\\DPY2\\OCC preparation\\OCCMockReattempt\\HAGreattempt.mdf; Integrated Security = True; Connect Timeout = 30";

SqlConnection sqlConnection = new SqlConnection(connectionString);

//Use stored procedure

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

command.CommandType = CommandType.StoredProcedure;

//Create a new SQL data adapter

SqlDataAdapter sd = new SqlDataAdapter(command);

//Create a new data table

DataTable dt = new DataTable();

//Open the SQL connection

sqlConnection.Open();

//Fill the data table

sd.Fill(dt);

//Close the SQL connection

sqlConnection.Close();

//Loop through the data table

foreach (DataRow dr in dt.Rows)

{

//Add each record to the ;list

accountList.Add(

new Account

{

//Assign the data in the Username field of the current data row to a variable

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

//Assign the data in the Password field of the current data row to a variable

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

});

}

//Assign data to variable

bool register = false;

//Looping through the accountList list

foreach (Account account in accountList)

{

//Checking if the username that was inputted already exists

if (account.username == usernameR)

{

//The user cannot register the account

register = false;

//Stopping the if statement

break;

}

else

{

//The user can register the account

register = true;

}

}

if (register == true)

{

//Connect to database

string connectionString2 = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDBFilename = C:\\Users\\M2201454\\OneDrive - Middlesbrough College\\Documents\\DPY2\\OCC preparation\\OCCMockReattempt\\HAGreattempt.mdf; Integrated Security = True; Connect Timeout = 30";

SqlConnection sqlConnection2 = new SqlConnection(connectionString2);

//Use the stored procedure

SqlCommand command2 = new SqlCommand("GetNewAccount", sqlConnection2);

command2.CommandType = CommandType.StoredProcedure;

//Assign username and password to @Username and @Password

command2.Parameters.AddWithValue("@Username", usernameR);

command2.Parameters.AddWithValue("@Password", passwordR);

//Open the SQL connection

sqlConnection2.Open();

//Execute SQL

command2.ExecuteNonQuery();

//Close SQL connection

sqlConnection2.Close();

//Display a message informing that the account has been registered.

MessageBox.Show("You have successfully registered your account");

//Hide register form

this.Hide();

//Show login form

Form2 Form2 = new Form2();

Form2.Show();

}

else

{

//Clear the textboxes

txtUsernameR.Text = null;

txtPasswordR.Text = null;

//Display an error message

MessageBox.Show("An account already has that username \nPlease enter a different username");

}

}

}

}

}

private void lblLogin\_Click(object sender, EventArgs e)

{

//Hide register form

this.Hide();

//Show login form

Form2 Form2 = new Form2();

Form2.Show();

}

private void txtPasswordR\_TextChanged(object sender, EventArgs e)

{if (txtPasswordR.Text.Length < 8)

{

string message = "Password must be at least 8 characters";

new ToolTip().Show(message,this, Cursor.Position.X - this.Location.X, Cursor.Position.Y - this.Location.Y, 1000);

}

}

private void lblShowR\_Click(object sender, EventArgs e)

{

//Show the password

txtPasswordR.UseSystemPasswordChar = false;

//Change visibility

lblShowR.Visible = false;

lblHideR.Visible = true;

}

private void lblHideR\_Click(object sender, EventArgs e)

{

//Hide the password

txtPasswordR.UseSystemPasswordChar = true;

//Change visibility

lblHideR.Visible = false;

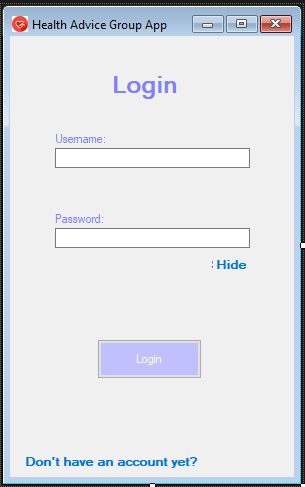
lblShowR.Visible = true;

}

}

}

***Form2.cs:***



namespace OCCMockReattempt

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

private void Form2\_Load(object sender, EventArgs e)

{

//Hide login password

txtPasswordL.UseSystemPasswordChar = true;

//Clear textboxes

txtUsernameL.Text = null;

txtPasswordL.Text = null;

}

private void btnLogin\_Click(object sender, EventArgs e)

{

//Create a list

List<Account> accountList = new List<Account>();

//Connect to database

string connectionString = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDBFilename = C:\\Users\\M2201454\\OneDrive - Middlesbrough College\\Documents\\DPY2\\OCC preparation\\OCCMockReattempt\\HAGreattempt.mdf; Integrated Security = True; Connect Timeout = 30";

SqlConnection sqlConnection = new SqlConnection(connectionString);

//Use the stored procedure

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

command.CommandType = CommandType.StoredProcedure;

//Create a new SQL data adapter

SqlDataAdapter sd = new SqlDataAdapter(command);

//Create a new data table

DataTable dt = new DataTable();

//Open the SQL connection

sqlConnection.Open();

//Fill the data table

sd.Fill(dt);

//Close the SQL connection

sqlConnection.Close();

//Loop through the data table

foreach (DataRow dr in dt.Rows)

{

//Add each record to the ;list

accountList.Add(

new Account

{

//Assign the data in the Username field of the current data row to a variable

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

//Assign the data in the Password field of the current data row to a variable

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

});

}

//Input the username and password

string usernameL = txtUsernameL.Text;

string passwordL = txtPasswordL.Text;

//Assign data to variable

bool login = false;

//Loop through the accountList list

foreach (Account account in accountList)

{

//Checking if the inputted username is equal to the username in the list

if (usernameL == account.username)

{

//Checking if the inputted password is equal to the password in the list - only checked if the usernames are equal

if (passwordL == account.password)

{

//Clear textboxes

txtUsernameL.Text = null;

txtPasswordL.Text = null;

//User logged in

login = true;

//Stop if statement

break;

}

else

{

//User not logged in

login = false;

}

}

else

{

//User not logged in

login = false;

}

}

//Display message depending on whether the user is logged in

if (login == true)

{

//Login message

MessageBox.Show("You have successfully logged in");

//Hide login form

this.Hide();

//Show menu form

Form3 Form3 = new Form3();

Form3.Show();

}

else

{

//Clear textboxes

txtUsernameL.Text = "";

txtPasswordL.Text = "";

//Incorrect input error message

MessageBox.Show("Username or password is incorrect");

}

}

private void lblRegister\_Click(object sender, EventArgs e)

{

//Hide login form

this.Hide();

//Show register form

Form1 form1 = new Form1();

form1.Show();

}

private void lblShowL\_Click(object sender, EventArgs e)

{

//Show the password

txtPasswordL.UseSystemPasswordChar = false;

//Change visibility

lblShowL.Visible = false;

lblHideL.Visible = true;

}

private void lblHideL\_Click(object sender, EventArgs e)

{

//Hide the password

txtPasswordL.UseSystemPasswordChar = true;

//Change visibility

lblHideL.Visible = false;

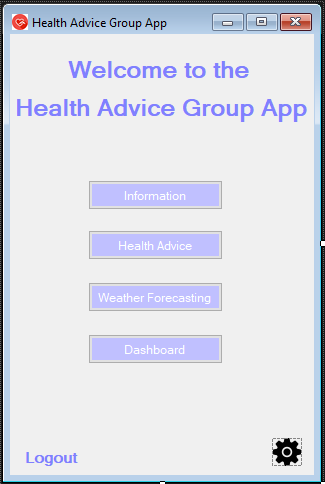
lblShowL.Visible = true;

}

}

}

***Form3.cs:***



namespace OCCMockReattempt

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

}

private void lblLogout\_Click(object sender, EventArgs e)

{

//Hide menu form

this.Hide();

//Show login form

Form2 Form2 = new Form2();

Form2.Show();

}

private void picSettings\_Click(object sender, EventArgs e)

{

//Hide menu form

this.Hide();

//Show account settings form

Form4 Form4 = new Form4();

Form4.Show();

}

private void btnInformation\_Click(object sender, EventArgs e)

{

//Hide menu form

this.Hide();

//Show information form

Form5 Form5 = new Form5();

Form5.Show();

}

private void btnAdvice\_Click(object sender, EventArgs e)

{

//Hide menu form

this.Hide();

//Show advice form

Form6 Form6 = new Form6();

Form6.Show();

}

private void btnWeatherForecasting\_Click(object sender, EventArgs e)

{

//Hide menu form

this.Hide();

//Show advice form

Form7 Form7 = new Form7();

Form7.Show();

}

private void btnDashboard\_Click(object sender, EventArgs e)

{

//Hide menu form

this.Hide();

//Show advice form

Form8 Form8 = new Form8();

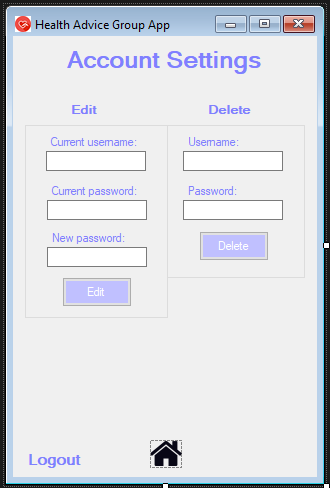
Form8.Show();

}

}

}

***Form4.cs:***



namespace OCCMockReattempt

{

public partial class Form4 : Form

{

public Form4()

{

InitializeComponent();

}

private void lblLogout\_Click(object sender, EventArgs e)

{

//Hide account settings form

this.Hide();

//Show login form

Form2 Form2 = new Form2();

Form2.Show();

}

private void picHome\_Click(object sender, EventArgs e)

{

//Hide account settings form

this.Hide();

//Show menu form

Form3 Form3 = new Form3();

Form3.Show();

}

private void btnEdit\_Click(object sender, EventArgs e)

{

//Input data

string currentUsername = txtCurrentUsername.Text;

string currentPassword = txtCurrentPassword.Text;

string newPassword = txtNewPassword.Text;

//Create a list

List<Account> accountList = new List<Account>();

//Connect to database

string connectionString = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDBFilename = C:\\Users\\M2201454\\OneDrive - Middlesbrough College\\Documents\\DPY2\\OCC preparation\\OCCMockReattempt\\HAGreattempt.mdf; Integrated Security = True; Connect Timeout = 30";

SqlConnection sqlConnection = new SqlConnection(connectionString);

//Use the stored procedure

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

command.CommandType = CommandType.StoredProcedure;

//Create a new SQL data adapter

SqlDataAdapter sd = new SqlDataAdapter(command);

//Create a new data table

DataTable dt = new DataTable();

//Open the SQL connection

sqlConnection.Open();

//Fill the data table

sd.Fill(dt);

//Close the SQL connection

sqlConnection.Close();

//Loop through the data table

foreach (DataRow dr in dt.Rows)

{

//Add each record to the ;list

accountList.Add(

new Account

{

//Assign the data in the Username field of the current data row to a variable

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

//Assign the data in the Password field of the current data row to a variable

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

});

}

//Assign data to variable

bool login = false;

//Loop through the accountList list

foreach (Account account in accountList)

{

//Checking if the inputted username is equal to the username in the list

if (currentUsername == account.username)

{

//Checking if the inputted password is equal to the password in the list - only checked if the usernames are equal

if (currentPassword == account.password)

{

//User logged in

login = true;

//Stop if statement

break;

}

else

{

//User not logged in

login = false;

}

}

else

{

//User not logged in

login = false;

}

}

if (login == true)

{

if (newPassword.Length >= 8)

{

//Connect to database

string connectionString2 = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDBFilename = C:\\Users\\M2201454\\OneDrive - Middlesbrough College\\Documents\\DPY2\\OCC preparation\\OCCMockReattempt\\HAGreattempt.mdf; Integrated Security = True; Connect Timeout = 30";

SqlConnection sqlConnection2 = new SqlConnection(connectionString2);

//Use the stored procedure

SqlCommand command2 = new SqlCommand("UpdateAccount", sqlConnection2);

command2.CommandType = CommandType.StoredProcedure;

//Assign username and password to @Username and @Password

command2.Parameters.AddWithValue("@Username", currentUsername);

command2.Parameters.AddWithValue("@Password", newPassword);

//Open SQL connection

sqlConnection2.Open();

//Execute SQL

command2.ExecuteNonQuery();

//Close SQL connection

sqlConnection2.Close();

//Clear textboxes

txtCurrentUsername.Text = null;

txtCurrentPassword.Text = null;

txtNewPassword.Text = null;

}

}

}

private void txtNewPassword\_TextChanged(object sender, EventArgs e)

{

if (txtNewPassword.Text.Length < 8)

{

string message = "Password must be at least 8 characters";

new ToolTip().Show(message, this, Cursor.Position.X - this.Location.X, Cursor.Position.Y - this.Location.Y, 1000);

}

}

private void btnDelete\_Click(object sender, EventArgs e)

{

//Input data

string username = txtUsernameDelete.Text;

string password = txtPasswordDelete.Text;

//Create a list

List<Account> accountList = new List<Account>();

//Connect to database

string connectionString = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDBFilename = C:\\Users\\M2201454\\OneDrive - Middlesbrough College\\Documents\\DPY2\\OCC preparation\\OCCMockReattempt\\HAGreattempt.mdf; Integrated Security = True; Connect Timeout = 30";

SqlConnection sqlConnection = new SqlConnection(connectionString);

//Use the stored procedure

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

command.CommandType = CommandType.StoredProcedure;

//Create a new SQL data adapter

SqlDataAdapter sd = new SqlDataAdapter(command);

//Create a new data table

DataTable dt = new DataTable();

//Open the SQL connection

sqlConnection.Open();

//Fill the data table

sd.Fill(dt);

//Close the SQL connection

sqlConnection.Close();

//Loop through the data table

foreach (DataRow dr in dt.Rows)

{

//Add each record to the ;list

accountList.Add(

new Account

{

//Assign the data in the Username field of the current data row to a variable

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

//Assign the data in the Password field of the current data row to a variable

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

});

}

//Assign data to variable

bool login = false;

//Loop through the accountList list

foreach (Account account in accountList)

{

//Checking if the inputted username is equal to the username in the list

if (username == account.username)

{

//Checking if the inputted password is equal to the password in the list - only checked if the usernames are equal

if (password == account.password)

{

//User logged in

login = true;

//Stop if statement

break;

}

else

{

//User not logged in

login = false;

}

}

else

{

//User not logged in

login = false;

}

}

if (login == true)

{

//Connect to database

string connectionString2 = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDBFilename = C:\\Users\\M2201454\\OneDrive - Middlesbrough College\\Documents\\DPY2\\OCC preparation\\OCCMockReattempt\\HAGreattempt.mdf; Integrated Security = True; Connect Timeout = 30";

SqlConnection sqlConnection2 = new SqlConnection(connectionString2);

//Use the stored procedure

SqlCommand command2 = new SqlCommand("DeleteAccount", sqlConnection2);

command2.CommandType = CommandType.StoredProcedure;

//Assign username to @Username

command2.Parameters.AddWithValue("@Username", username);

//Open SQL connection

sqlConnection2.Open();

//Execute SQL

command2.ExecuteNonQuery();

//Close SQL connection

sqlConnection2.Close();

//Clear textboxes

txtUsernameDelete = null;

txtPasswordDelete = null;

//Hide account settings form

this.Hide();

//Show register form

Form1 Form1 = new Form1();

Form1.Show();

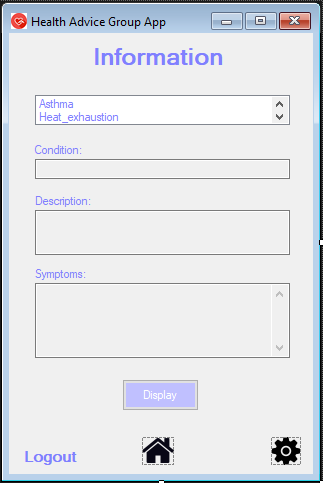
}

}

}

}

***Form5.cs:***



namespace OCCMockReattempt

{

public partial class Form5 : Form

{

public Form5()

{

InitializeComponent();

}

private void Form5\_Load(object sender, EventArgs e)

{

//Clear textboxes

txtCondition.Text = null;

txtDescription.Text = null;

txtSymptoms.Text = null;

}

private void lblLogout\_Click(object sender, EventArgs e)

{

//Hide account settings form

this.Hide();

//Show login form

Form2 Form2 = new Form2();

Form2.Show();

}

private void picHome\_Click(object sender, EventArgs e)

{

//Hide account settings form

this.Hide();

//Show menu form

Form3 Form3 = new Form3();

Form3.Show();

}

private void picSettings\_Click(object sender, EventArgs e)

{

//Hide menu form

this.Hide();

//Show account settings form

Form4 Form4 = new Form4();

Form4.Show();

}

private void btnDisplayInformation\_Click(object sender, EventArgs e)

{

//Create a list

List<Condition> conditionList = new List<Condition>();

//Connect to database

string connectionString = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDBFilename = C:\\Users\\M2201454\\OneDrive - Middlesbrough College\\Documents\\DPY2\\OCC preparation\\OCCMockReattempt\\HAGreattempt.mdf; Integrated Security = True; Connect Timeout = 30";

SqlConnection sqlConnection = new SqlConnection(connectionString);

//Use the stored procedure

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

command.CommandType = CommandType.StoredProcedure;

//Create a new SQL data adapter

SqlDataAdapter sd = new SqlDataAdapter(command);

//Create a new data table

DataTable dt = new DataTable();

//Open the SQL connection

sqlConnection.Open();

//Fill the data table

sd.Fill(dt);

//Close the SQL connection

sqlConnection.Close();

//Loop through the data table

foreach (DataRow dr in dt.Rows)

{

//Add each record to the ;list

conditionList.Add(

new Condition

{

//Assign the data in the Name field of the current data row to a variable

name = Convert.ToString(dr["Name"]),

//Assign the data in the Description field of the current data row to a variable

description = Convert.ToString(dr["Description"]),

//Assign the data in the Symptoms field of the current data row to a variable

symptoms = Convert.ToString(dr["Symptoms"]),

//Assign the data in the Treatment field of the current data row to a variable

treatment = Convert.ToString(dr["Treatment"]),

});

}

//Choose health condition

string healthCondition = liInformation.Text;

//Selecting the case equal to healthondition

switch (healthCondition)

{

//The case where Asthma is chosen

case "Asthma":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtDescription textbox

txtDescription.Text = condition.description;

//Assigning data to the txtSymptoms textbox

txtSymptoms.Text = condition.symptoms;

}

}

//Stops switch statement

break;

//The case where Heat\_exhaustion is chosen

case "Heat\_exhaustion":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtDescription textbox

txtDescription.Text = condition.description;

//Assigning data to the txtSymptoms textbox

txtSymptoms.Text = condition.symptoms;

}

}

//Stops switch statement

break;

//The case where Dehydration is chosen

case "Dehydration":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtDescription textbox

txtDescription.Text = condition.description;

//Assigning data to the txtSymptoms textbox

txtSymptoms.Text = condition.symptoms;

}

}

//Stops switch statement

break;

//The case where Frostbite is chosen

case "Frostbite":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtDescription textbox

txtDescription.Text = condition.description;

//Assigning data to the txtSymptoms textbox

txtSymptoms.Text = condition.symptoms;

}

}

//Stops switch statement

break;

//The case where Hay\_fever is chosen

case "Hay\_fever":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtDescription textbox

txtDescription.Text = condition.description;

//Assigning data to the txtSymptoms textbox

txtSymptoms.Text = condition.symptoms;

}

}

//Stops switch statement

break;

//The case where Spring\_allergies is chosen

case "Spring\_allergies":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtDescription textbox

txtDescription.Text = condition.description;

//Assigning data to the txtSymptoms textbox

txtSymptoms.Text = condition.symptoms;

}

}

//Stops switch statement

break;

//The case where Summer\_allergies is chosen

case "Summer\_allergies":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtDescription textbox

txtDescription.Text = condition.description;

//Assigning data to the txtSymptoms textbox

txtSymptoms.Text = condition.symptoms;

}

}

//Stops switch statement

break;

//The case where Autumn\_allergies is chosen

case "Autumn\_allergies":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtDescription textbox

txtDescription.Text = condition.description;

//Assigning data to the txtSymptoms textbox

txtSymptoms.Text = condition.symptoms;

}

}

//Stops switch statement

break;

//The case where Winter\_allergies is chosen

case "Winter\_allergies":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtDescription textbox

txtDescription.Text = condition.description;

//Assigning data to the txtSymptoms textbox

txtSymptoms.Text = condition.symptoms;

}

}

//Stops switch statement

break;

//Runs when none of the cases are fulfilled

default:

//Error message explaining that an option must be chosen

MessageBox.Show("Please choose one of the options");

//Stops switch statement

break;

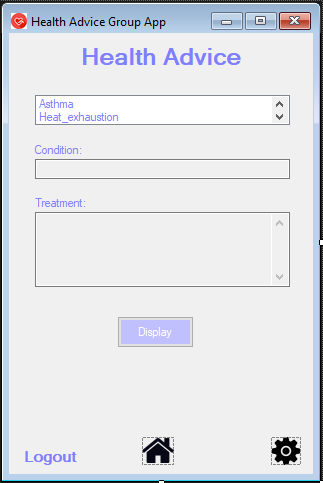
}

}

}

}

***Form6.cs:***



namespace OCCMockReattempt

{

public partial class Form6 : Form

{

public Form6()

{

InitializeComponent();

}

private void Form6\_Load(object sender, EventArgs e)

{

//Clear textboxes

txtCondition.Text = null;

txtTreatment.Text = null;

}

private void lblLogout\_Click(object sender, EventArgs e)

{

//Hide account settings form

this.Hide();

//Show login form

Form2 Form2 = new Form2();

Form2.Show();

}

private void picHome\_Click(object sender, EventArgs e)

{

//Hide account settings form

this.Hide();

//Show menu form

Form3 Form3 = new Form3();

Form3.Show();

}

private void picSettings\_Click(object sender, EventArgs e)

{

//Hide menu form

this.Hide();

//Show account settings form

Form4 Form4 = new Form4();

Form4.Show();

}

private void btnDisplayInformation\_Click(object sender, EventArgs e)

{

//Create a list

List<Condition> conditionList = new List<Condition>();

//Connect to database

string connectionString = "Data Source = (LocalDB)\\MSSQLLocalDB; AttachDBFilename = C:\\Users\\M2201454\\OneDrive - Middlesbrough College\\Documents\\DPY2\\OCC preparation\\OCCMockReattempt\\HAGreattempt.mdf; Integrated Security = True; Connect Timeout = 30";

SqlConnection sqlConnection = new SqlConnection(connectionString);

//Use the stored procedure

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

command.CommandType = CommandType.StoredProcedure;

//Create a new SQL data adapter

SqlDataAdapter sd = new SqlDataAdapter(command);

//Create a new data table

DataTable dt = new DataTable();

//Open the SQL connection

sqlConnection.Open();

//Fill the data table

sd.Fill(dt);

//Close the SQL connection

sqlConnection.Close();

//Loop through the data table

foreach (DataRow dr in dt.Rows)

{

//Add each record to the ;list

conditionList.Add(

new Condition

{

//Assign the data in the Name field of the current data row to a variable

name = Convert.ToString(dr["Name"]),

//Assign the data in the Description field of the current data row to a variable

description = Convert.ToString(dr["Description"]),

//Assign the data in the Symptoms field of the current data row to a variable

symptoms = Convert.ToString(dr["Symptoms"]),

//Assign the data in the Treatment field of the current data row to a variable

treatment = Convert.ToString(dr["Treatment"]),

});

}

//Choose health condition

string healthCondition = liAdvice.Text;

//Selecting the case equal to healthondition

switch (healthCondition)

{

//The case where Asthma is chosen

case "Asthma":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtTreatment textbox

txtTreatment.Text = condition.treatment;

}

}

//Stops switch statement

break;

//The case where Heat\_exhaustion is chosen

case "Heat\_exhaustion":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtTreatment textbox

txtTreatment.Text = condition.treatment;

}

}

//Stops switch statement

break;

//The case where Dehydration is chosen

case "Dehydration":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtTreatment textbox

txtTreatment.Text = condition.treatment;

}

}

//Stops switch statement

break;

//The case where Frostbite is chosen

case "Frostbite":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtTreatment textbox

txtTreatment.Text = condition.treatment;

}

}

//Stops switch statement

break;

//The case where Hay\_fever is chosen

case "Hay\_fever":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtTreatment textbox

txtTreatment.Text = condition.treatment;

}

}

//Stops switch statement

break;

//The case where Spring\_allergies is chosen

case "Spring\_allergies":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtTreatment textbox

txtTreatment.Text = condition.treatment;

}

}

//Stops switch statement

break;

//The case where Summer\_allergies is chosen

case "Summer\_allergies":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtTreatment textbox

txtTreatment.Text = condition.treatment;

}

}

//Stops switch statement

break;

//The case where Autumn\_allergies is chosen

case "Autumn\_allergies":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtTreatment textbox

txtTreatment.Text = condition.treatment;

}

}

//Stops switch statement

break;

//The case where Winter\_allergies is chosen

case "Winter\_allergies":

//Looping through every condition in the list

foreach (Condition condition in conditionList)

{

//Checking if the name of the current condition is equal to the condition that was chosen

if (condition.name == healthCondition)

{

//Assigning data to the txtCondition textbox

txtCondition.Text = condition.name;

//Assigning data to the txtTreatment textbox

txtTreatment.Text = condition.treatment;

}

}

//Stops switch statement

break;

//Runs when none of the cases are fulfilled

default:

//Error message explaining that an option must be chosen

MessageBox.Show("Please choose one of the options");

//Stops switch statement

break;

}

}

}

}

***Form7.cs:***



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net.Http;

using Newtonsoft.Json.Linq;

//Go to Project – Manage Nuget Pacckages – Browse – Newton.Json and install it

namespace OCCMockReattempt

{

public partial class Form7 : Form

{

public string location { get; set; } = "Middlesbrough";

public HttpClient httpClient { get; set; }

//API key for the OpenWeather oraganisation APIs

public string APIKey = "b1fe4659e300363e368f4ce2cf007fac";

public async Task<List<float>> GetCoordinates(string region)

{

//Gets the latitude and longitude of the chosen 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 btnWeatherForecasting\_Click(object sender, EventArgs e)

{

List<float> coordinates = await GetCoordinates(this.location);

//Use the api to get some weather data

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

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

//MessageBox.Show(payload);

//Parse (read) the data itemds from the JSON object

JObject payloadObject = JObject.Parse(payload);

//Get individual items from the JSON object

//MessageBox.Show((string)payloadObject["weather"][0]["description"]);

//MessageBox.Show((string)payloadObject["main"]["temp"]);

//MessageBox.Show((string)payloadObject["visibility"]);

//MessageBox.Show((string)payloadObject.GetValue("main").ToString());

//MessageBox.Show((string)payloadObject.Count.ToString());

//MessageBox.Show((string)payloadObject.First.ToString());

//Change visibility

grpForecast.Visible = true;

//Display individual items

string location = (string)payloadObject["name"];

//Location

lblLocation.Text = location;

string description = (string)payloadObject["weather"][0]["description"];

//Description

lblDescriptionOutcome.Text = description;

double windSpeed = (double)payloadObject["wind"]["speed"];

//Wind speed

lblWindSpeedOutcome.Text = windSpeed.ToString();

double temperatureK = (double)payloadObject["main"]["temp"];

double temperatureC = temperatureK - 273.15;

double temperatureF = temperatureC \* 9 / 5 + 32;

//Temperature

lblTempKOutcome.Text = temperatureK.ToString();

lblTempFOutcome.Text = temperatureF.ToString();

lblTempCOutcome.Text = temperatureC.ToString();

double feelsLikeK = (double)payloadObject["main"]["feels\_like"];

double feelsLikeC = feelsLikeK - 273.15;

double feelsLikeF = feelsLikeC \* 9 / 5 + 32;

//Feels like

lblFeelsLikeKOutcome.Text = feelsLikeK.ToString();

lblFeelsLikeFOutcome.Text = feelsLikeF.ToString();

lblFeelsLikeCOutcome.Text = feelsLikeC.ToString();

double pressure = (double)payloadObject["main"]["pressure"];

double humidity = (double)payloadObject["main"]["humidity"];

//Pressure

lblPressureOutcome.Text = pressure.ToString();

//Humidity

lblHumidityOutcome.Text = humidity.ToString();

}

public Form7()

{

InitializeComponent();

//Creates a new form of the http client class.

httpClient = new HttpClient();

this.location = location;

}

private void Form7\_Load(object sender, EventArgs e)

{

//Change visibility

grpForecast.Visible = false;

}

private void lblLogout\_Click(object sender, EventArgs e)

{

//Hide account settings form

this.Hide();

//Show login form

Form2 Form2 = new Form2();

Form2.Show();

}

private void picHome\_Click(object sender, EventArgs e)

{

//Hide account settings form

this.Hide();

//Show menu form

Form3 Form3 = new Form3();

Form3.Show();

}

private void picSettings\_Click(object sender, EventArgs e)

{

//Hide menu form

this.Hide();

//Show account settings form

Form4 Form4 = new Form4();

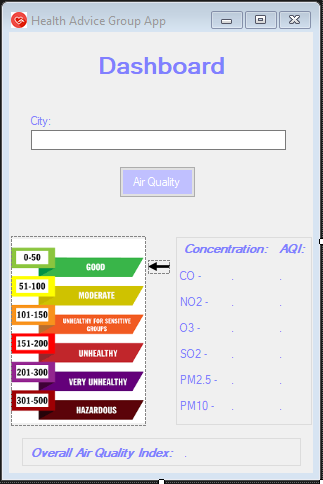
Form4.Show();

}

}

}

***Form8.cs:***



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Net.Http;

using Newtonsoft.Json.Linq;

//Go to Project – Manage Nuget Pacckages – Browse – Newton.Json and install it

namespace OCCMockReattempt

{

public partial class Form8 : Form

{

public Form8()

{

InitializeComponent();

}

private void Form8\_Load(object sender, EventArgs e)

{

//Change visibility

picScale.Visible = false;

picArrow.Visible = false;

grpAirQuality.Visible = false;

grpOverallAQI.Visible = false;

//Clear textbox

txtCity.Text = null;

}

private async void btnWeatherForecasting\_Click(object sender, EventArgs e)

{

//Exception handling

try

{

//Assign data from textbox to location variable

string city = txtCity.Text;

//Air quality API

var client = new HttpClient();

var request = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = new Uri("https://air-quality-by-api-ninjas.p.rapidapi.com/v1/airquality?city=" + city),

Headers =

{

{ "X-RapidAPI-Key", "99f2852ca5msh4a728f4e7f97df9p1649aajsndfe1bf581f06" },

{ "X-RapidAPI-Host", "air-quality-by-api-ninjas.p.rapidapi.com" },

},

};

using (var response = await client.SendAsync(request))

{

response.EnsureSuccessStatusCode();

var body = await response.Content.ReadAsStringAsync();

JObject payloadObject = JObject.Parse(body);

Console.WriteLine(body);

//MessageBox.Show(body);

//Change visibility

grpAirQuality.Visible = true;

grpOverallAQI.Visible = true;

//Display individual values

double coConcentration = (double)payloadObject["CO"]["concentration"];

double coAQI = (double)payloadObject["CO"]["aqi"];

//CO

lblCOConcentration.Text = coConcentration.ToString();

lblCOAQI.Text = coAQI.ToString();

double no2Concentration = (double)payloadObject["NO2"]["concentration"];

double no2AQI = (double)payloadObject["NO2"]["aqi"];

//NO2

lblNO2Concentration.Text = no2Concentration.ToString();

lblNO2AQI.Text = no2AQI.ToString();

double o3Concentration = (double)payloadObject["O3"]["concentration"];

double o3AQI = (double)payloadObject["O3"]["aqi"];

//O3

lblO3Concentration.Text = o3Concentration.ToString();

lblO3AQI.Text = o3AQI.ToString();

double so2Concentration = (double)payloadObject["SO2"]["concentration"];

double so2AQI = (double)payloadObject["SO2"]["aqi"];

//SO2

lblSO2Concentration.Text = so2Concentration.ToString();

lblSO2AQI.Text = so2AQI.ToString();

double pm2Concentration = (double)payloadObject["PM2.5"]["concentration"];

double pm2AQI = (double)payloadObject["PM2.5"]["aqi"];

//PM2.5

lblPM2Concentration.Text = pm2Concentration.ToString();

lblPM2AQI.Text = pm2AQI.ToString();

double pm10Concentration = (double)payloadObject["PM10"]["concentration"];

double pm10AQI = (double)payloadObject["PM10"]["aqi"];

//PM10

lblPM10Concentration.Text = pm10Concentration.ToString();

lblPM10AQI.Text = pm10AQI.ToString();

double overallAQI = (double)payloadObject["overall\_aqi"];

//Overall AQI

lblOverallAQIOutcome.Text = overallAQI.ToString();

//Displaying arrow on air quality index scale

if (overallAQI <= 50)

{

//Change visibility + move arrow

picScale.Visible = true;

picArrow.Visible = true;

picArrow.Location = new Point(139, 228);

}

else

{

if (overallAQI <= 100)

{

//Change visibility + move arrow

picScale.Visible = true;

picArrow.Visible = true;

picArrow.Location = new Point(139, 258);

}

else

{

if (overallAQI <= 150)

{

//Change visibility + move arrow

picScale.Visible = true;

picArrow.Visible = true;

picArrow.Location = new Point(139, 286);

}

else

{

if (overallAQI <= 200)

{

//Change visibility + move arrow

picScale.Visible = true;

picArrow.Visible = true;

picArrow.Location = new Point(139, 315);

}

else

{

if (overallAQI <= 300)

{

//Change visibility + move arrow

picScale.Visible = true;

picArrow.Visible = true;

picArrow.Location = new Point(139, 343);

}

else

{

if (overallAQI <= 500)

{

//Change visibility + move arrow

picScale.Visible = true;

picArrow.Visible = true;

picArrow.Location = new Point(139, 372);

}

else

{

MessageBox.Show("The overall AQI is past the scale.");

}

}

}

}

}

}

}

}

//Catch exception

catch (Exception ex)

{

//Output exception message

MessageBox.Show("There is an exception: " + ex.Message);

}

}

}

}