Health Advice Group - Code Documentation

GitHub HAG Project - https://github.com/DOOMswat/Health-Advice-Group

Contents

[Version 1 2](#_Toc153890224)

[Login Window 2](#_Toc153890225)

[Window Design 2](#_Toc153890226)

[Backend Code 3](#_Toc153890227)

[Overview 3](#_Toc153890228)

[Register Window 4](#_Toc153890229)

[Window Design 4](#_Toc153890230)

[Backend Code 5](#_Toc153890231)

[Overview 6](#_Toc153890232)

[Homepage Window 6](#_Toc153890233)

[Window Design 6](#_Toc153890234)

[Backend Code 7](#_Toc153890235)

[Overview 8](#_Toc153890236)

[Version 2 8](#_Toc153890237)

[Login Window 8](#_Toc153890238)

[Window Design 8](#_Toc153890239)

[Backend Code 9](#_Toc153890240)

[Overview 9](#_Toc153890241)

[Version 3 10](#_Toc153890242)

[Version 4 10](#_Toc153890243)

[MySQL Workbench Log 10](#_Toc153890244)

[Tables 10](#_Toc153890245)

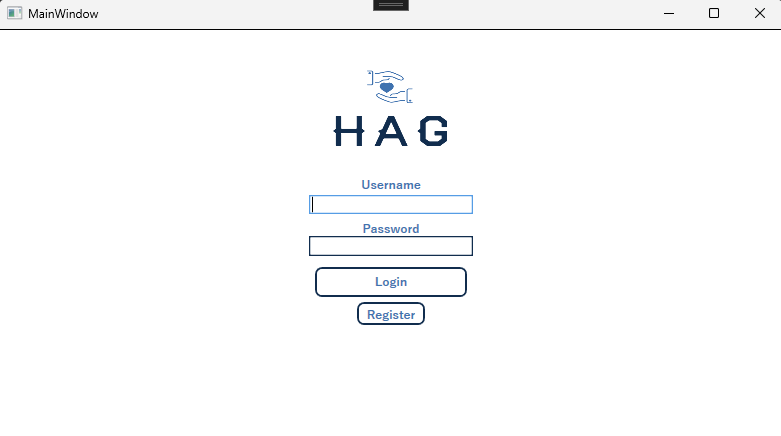
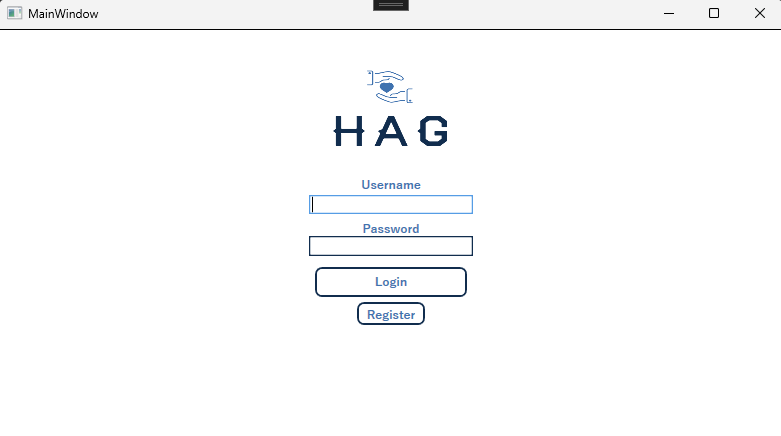
[Data 11](#_Toc153890246)

# Version 1

Version 1 consists of the initial designs and basic functionalities in the program. This allows the digital solutions to have a base for the initial designs provided which allows for more features to be implemented later if needed. This also gives the client a visual idea on where the buttons would redirect them to.

## Login Window

### Window Design



This is not visible however, there is a grid which makes it always centre in the middle allowing the user to resize the window without the objects being everywhere.

The image is stored on a rectangle which can be resizable.

There is a label and text box. Anything stored in the text box will be stored as a username which is indicated by the label.

The password box allows the user to type in their password but also mitigates attacks. The password box will hash any text entered into the box.

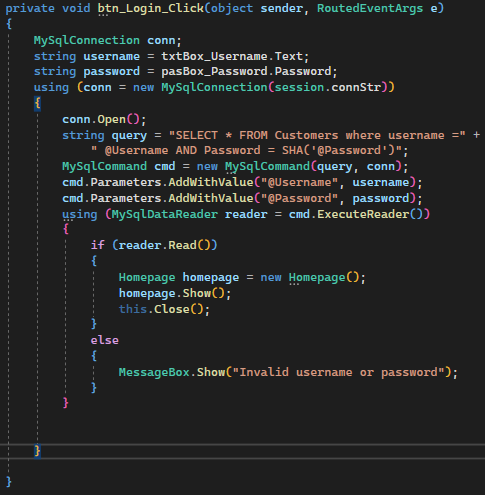
The login button is supposed to get the variables username and password, read both into the database and if it is correctly read, it will redirect the user to the homepage.

The register button currently has no function yet.

### Backend Code

The line “MySqlConnection conn;” establishes the connection allowing it to be connected when needed.

This is the event of when the button is clicked.



This will display a message box saying the string inside.

This will open a new window, display it on the users screen and then close the Login window.

When it has outputted successfully from the DB it will go through this if statement

This is a using statement will execute the cmd called reader.

This is parametrising the variables which mitigates man in the middle attacks and SQL injections.

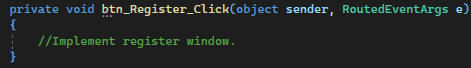
This is putting the query and connection string in a string called command which will be used to be executed later.

This is declaring a string which is a mySQL query.

This will open the connection.

This is a using statement which will properly dispose the instance. This will initialize the connection with the connection string which is stored in conn

This is declaring the username and password boxes as strings allowing them to be used later.



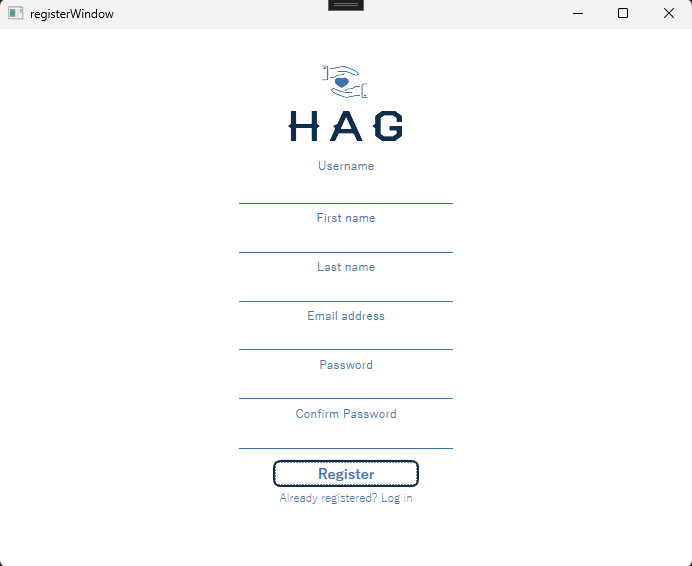
This is the event of when the register button is clicked.

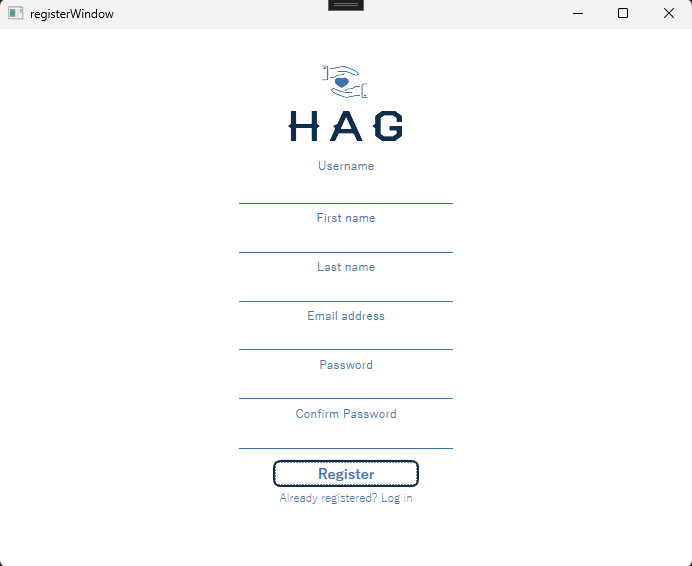
### Overview

The login page is supposed to allow the user to securely log in or be redirected to the navigation page. Suggestions I would make is I will update the backend code to allow the user to press the register button to redirected to the register page. Another thing to implement is to add the user information into a session to allow easy passing through the program with ease. These suggestions will be implemented in future versions.

## Register Window

### Window Design





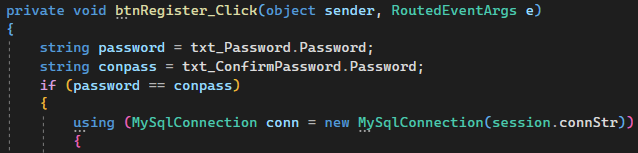
When clicked, the button will redirect the user back to the log in page.

When clicked, this button will check the input data in the text boxes and insert it into the database while redirecting the user the homepage.

Write above the blue lines are text boxes which are under the labels

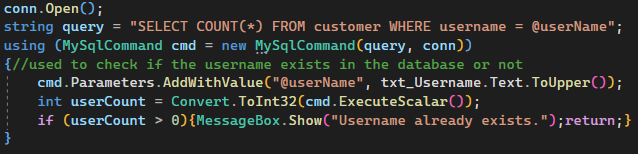
### Backend Code

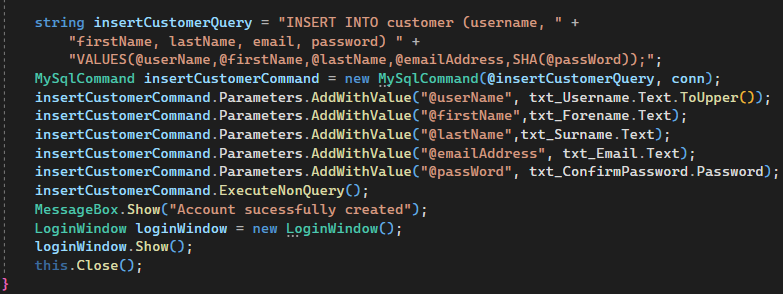
This is declaring the strings of the entered password



This chunk of code will open the connection and check if the username the person has entered in the username textbox has already been made. This is done by checking the count on how many usernames are there of the entered username. If the count is bigger than 0 it will display a message that will tell the client, the username already exists. If the count is not bigger than 0 it will return.

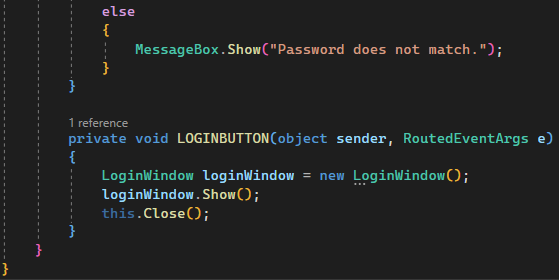
If these passwords match it will go on to the next function in the bracket.





Opens the Log in window back up to allow the user to log back in.

This will insert the inputted data from the textboxes and input them into the query which is then executed and inserted into the database. The inputted username will always be all uppercase. After this is done, the program will display a message letting the know they have made an account.



Redirect the user to the login page.

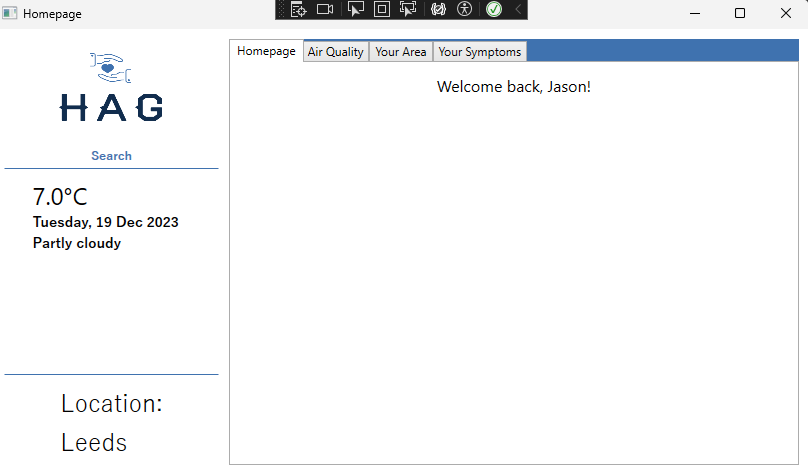
This message box will only display if the password does not match.

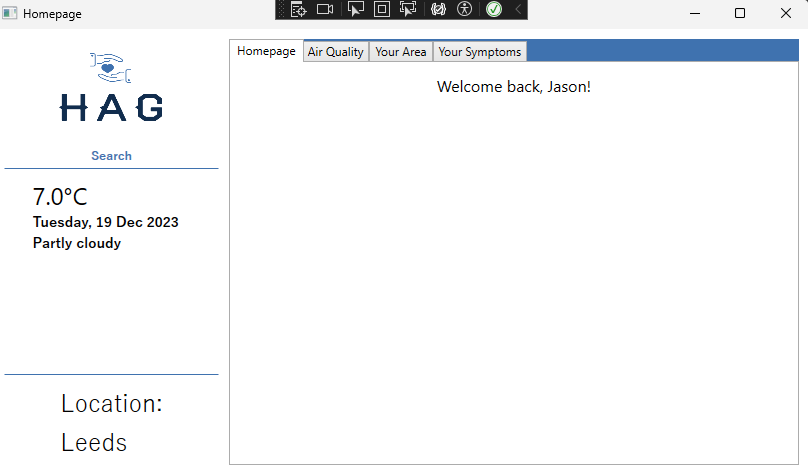
### Overview

The register window is supposed to allow the user to insert the data they would like to register in the correct boxes and allow the user to create an account. This version of the register page needs some more validation.

## Homepage Window

### Window Design





This should display the current location of where the given temperature and other information

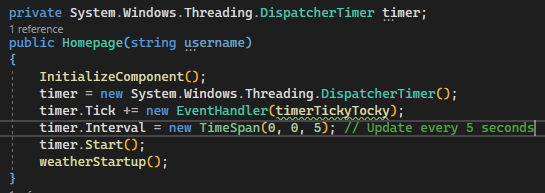
Textbox allowing the user to search.

This is displaying the current temperature, date and message. This will update every 5 seconds which always gets the latest update.

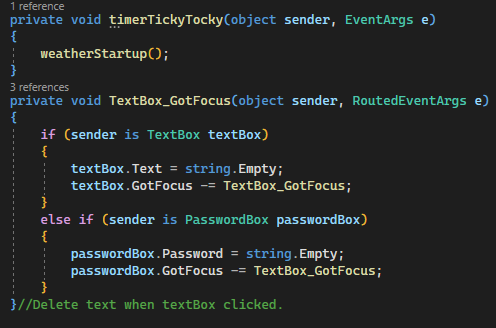
Tab control allowing the user to change tabs when clicking the buttons.

Displaying the welcome text of the users first name from database

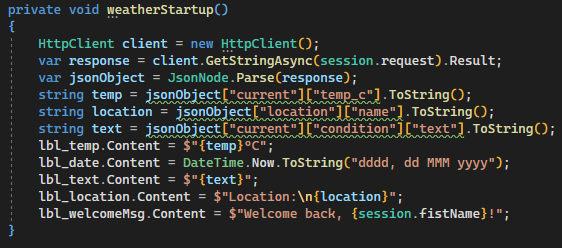
### Backend Code



This is a type of iteration where it would tick every 5 seconds when allowing it to get the latest information.



A function that makes the initial text delete when the user clicks on the textbox.



This chunk of code will get a JSON file from the request and pinpoint what is needed into the string which is then updated on to the window.

### Overview

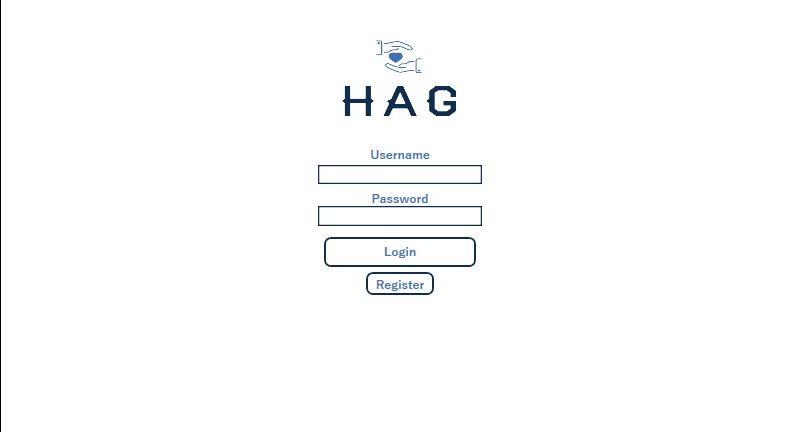
The first version of this homepage only displays the location of Leeds, a few interactive tabs, weather temperature in Leeds, the full date, and a bit of text to display. This version is used to let the user in a homepage window after they have successfully logged in. Some suggestions I would make is, validating the program making sure it is robust and does not crash.

# Version 2

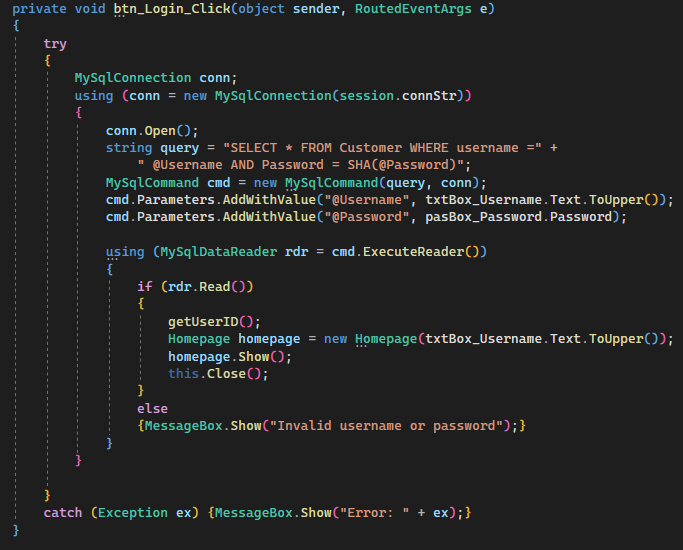
Version 2 is mostly implementing validation and functionality. This version will focus mostly on functionality and less design overall. This is to meet the overall functionality of the digital solution after the version one base is created.

## Login Window

### Window Design



### Backend Code

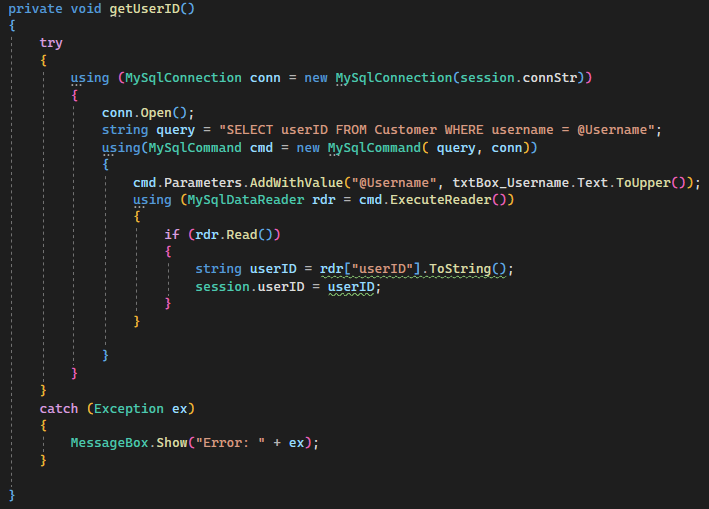


This is a sub-routine that will get the userID and put it in the session. This is done only after the user has successfully logged in.

I have replaced the declared strings with the text boxes for optimisation. On the txtBox\_Username I have set it to always uppercase which means the username box is no longer case sensitive.

The catch statement allows the program to still function if it does crashes. If the program does crash, it will display a pop up message on the clients screen.

This is validating what is between the curly brackets.



This will pinpoint the userID from rdr which was read from the database and place the userID in a class which can be used at any time later.

This query is used to find the userID from the username entered.

### Overview

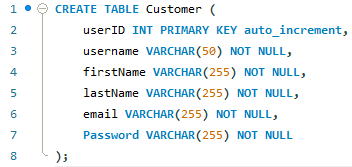
Version 2 of the login window focuses on validation which allows the program to be more secure allowing the program not to crash. I have also implemented another subroutine which when ran, it will get the userID from the username entered and then place the userID in a public class allowing the userID to be accessed anywhere within the code.

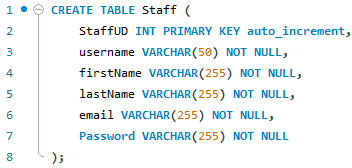
# Version 3

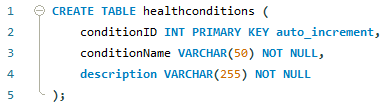
# Version 4

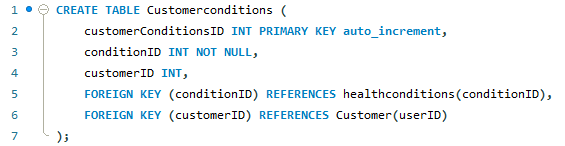
# MySQL Workbench Log

## Tables









## Data



