**Task 2 write up, Declan Hitchcock (S2000939)**

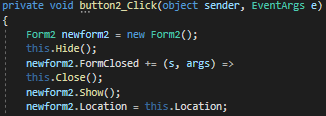
**Form 1 – Login page**

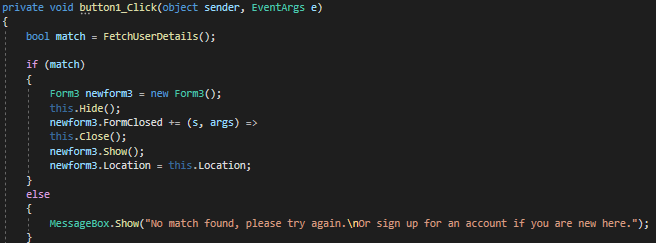
****

In form 1 I created a login page. This is where a user will enter their user details to log onto the program. To kickstart the program the user would click on the create account button that will take you to form 2. That is the page where users will enter user details for the first time to be stored on a database. These details can then be used in form 1 to allow the user to gain access and log in to the application and have their own personalised account.

The create account button changes the form whilst hiding the one that had just been in use. It then makes the next form open in the exact same place where the other is to keep the program neat and tidy.

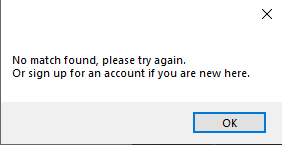
The GibJohn logo was also designed by me on a website where you can personalise and create your own logos for a business or organisation. I wanted to keep it plain so I kept the illustrations black and made the writing white. This makes it more apealing and easier to read for a user with potential sight problems or reading difficulties. I also photoshopped a mortarboard hat to place on the background to theme the website in an educational manner.



****

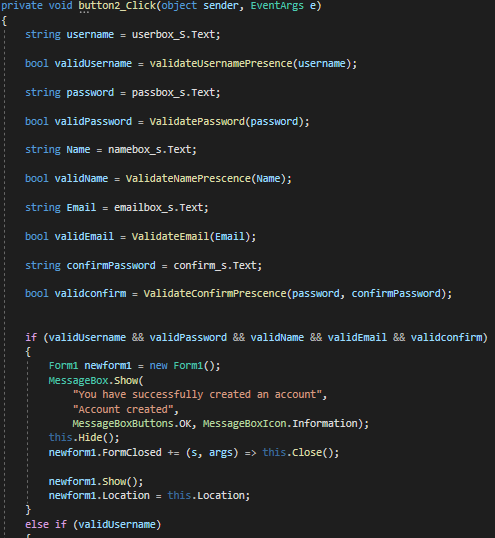
If user details are correct, then the menu form will then be opened, this is by linking the form to the SQL server, (**page 6** explains how I set up a database). If not, the message box will open telling you that it is invalid

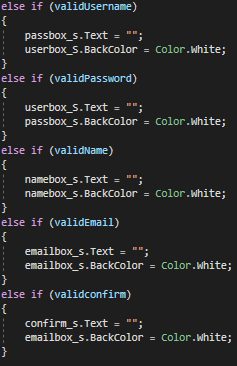
The Graphical user Interface is simple with nothing but what the user is required to do. The username and password box. The SQL server will check the user inputs to see if the user is valid or not. If the details are not valid, then the program will ask you to try again or create an account.

****

**Form 2 – sign up page**

This form is the create an account page, where the user inputs will be stored on the SQL database as a new valid user. The user details consist of Name, Username, Email, password and confirm password. There are some validators that have been put in place to ensure that user details are accurate and meet the requirements. If any of the boxes don’t meet the requirements or are empty, the program will tell the user where the details have gone wrong and will make them retry until they are valid.

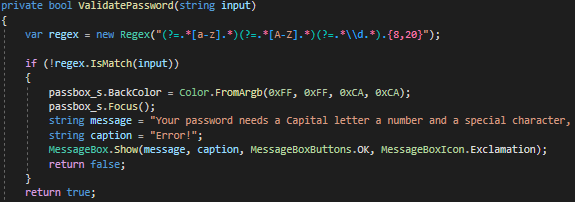


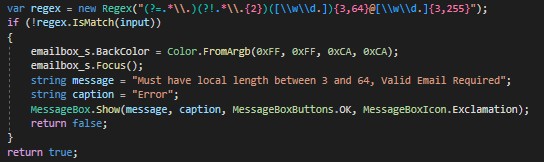


At the beginning of the program, when the continue button is clicked then it will ensure all the textboxes are filled with user inputs. I did this by assigning every required detail to string variable for a user input, and then a Boolean after to be true when the input is valid and false if it is invalid. the variables that I set to each one will then be validated separately. If all the bool variables are true, then the program can carry on and store the user details to allow them to log in.

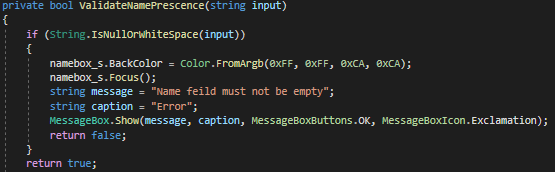
It also sets all the text boxes to white when there is no text inside of them (at the start of the program) this is because the boxes will turn a different colour when the input is wrong. This means that there will be no bugs in the system where the textboxes are the wrong colour when the program is started. Once every variable is valid, then a text box will appear on screen letting you know that an account has been successfully created and will then take you to the log in form.

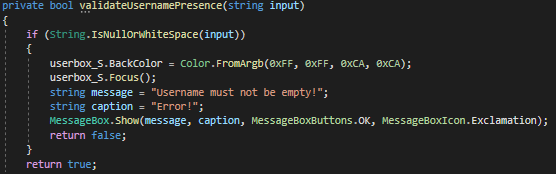
**Validating using REGEX**



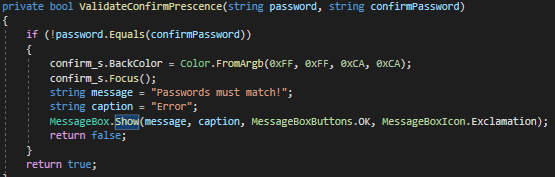
To validate the password I used Regex, this is where the user input is validated to ensure it meets the requirements for what the field is asking. In my regex validation, there needs to be letters from a-z in lowercase and A-Z in capitals, whilst also containing a special character and being between 8 and 20. If the user input does not meet those requirements, then the textbox would turn red, and an error message will pop up on the users screen explaining the requirements.

I used the same principal when validating the email but instead used different regex rules such as the email needing to be no longer than 255 characters and no less than 3, while also keeping the domain lower than 64 and higher than 3. There also needs to be only 1 dot before the @ sign and 1 to 2 afterwards as every email is in those requirements. Once again if the requirements are not met, then the textbox will turn red, and the email requirements will show up with an error message explaining what is needed and telling the user that the email Is invalid

**Validating the rest of the user inputs**

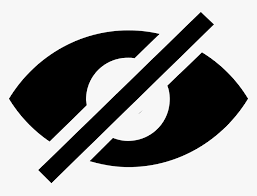
****

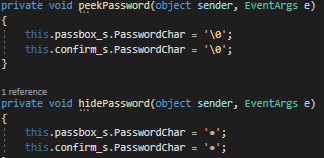
The name and username boxes are validated by there being text there or not. If there is no text, then an error message will show telling the user that the name field or username field must not be empty whilst turning the text boxes red. If there is any kind of text there, then it will be valid as anyone can choose whatever name and username they like. I understand the flaws with this, such as students creating inappropriate names, and typos being made, but I am trying to figure out a method where the user can change their user details on request and create a system where it filters out inappropriate words.



To confirm that the passwords match then the password user input needs to equal the confirm password if they don’t match, then an error message appears on the screen telling the user that the passwords must match whilst also turning the textbox red.

**Small features that the form contains**

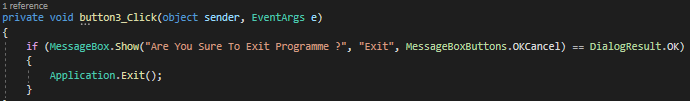
****In my program I inserted a peek password feature, that allows the user to be able to peak at their password and make sure its correct, revealing the text from the password characters and then returning to a password state once the user has done. To do this I created a button with a peek pass logo I retrieved from the internet

I then created private voids called peekPassword and hidePassword.

when the mouse is touching the peek password sign, then it will execute the code to show the password by setting it to \0 (null) which will show the text in both boxes. once the mouse leaves, then it will return the values to the original characters which is a bullet point. This is done by setting **MouseHover** to **peekPassword** and **MouseLeave** to **hidePassword.** This is put in place so that people do not make mistakes when entering their password and confirm password.

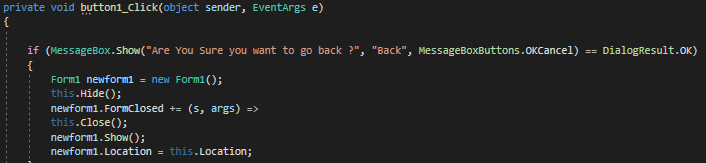


I also have an **X** button on every form that allows the user to log out and will also give the user a safety box to verify that they wanted to log out and that it wasn’t just a miss click.

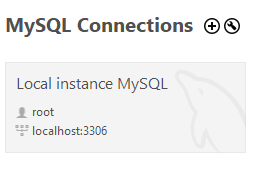


 This is the button that I designed through visual studio. By placing the letter X X on a red rectangular background.

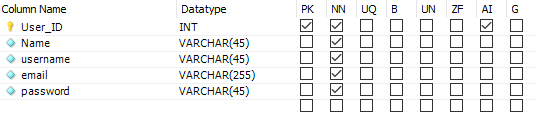
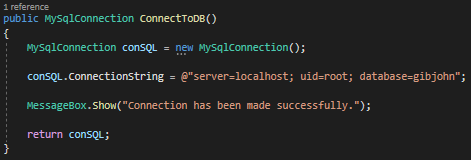
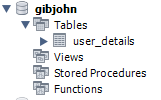
I also added a back button to allow the user to go cack to the log in form, this could be because they mis clicked the create account button or forgot they already had an account created. It is not a vital feature, but it always keeps options open for users and can make a small mistake easier to solve. If this was not in place, then the user would have to restart the program if they clicked onto the wrong form.



 i I retrieved a photo of an arrow on the internet and I put it in a box in front of t the same colour as the background.

**Setting up an SQL database**

I used MySQL workbench for setting up my database as it provides a visual interface for easily operating MySQL installations and improving database accessibility. It's a multi-tab editor, which means you can have many Queries open at the same time whilst simply being able navigate between them all.  It's also possible to connect it to programming software’s such as Visual Studio simple and effectively.

****

I created a new schema and named is as **gibjohn** and I created a new table and I called it **user\_details**. In the table it contains a User\_ID which is the primary key. The primary key uniquely identifies every single row in the table so that there will be faster access to each one. Every value is set to **NN** (Not Null) including the primary key, which means that every detail has to have an input and 0 values would not be valid. The primary key is also set to **AI** (auto increment). This means that when a new record is entered into a table, a unique number will be produced automatically. This means that every user that creates an account will have a large inimitable number assigned to them, so they can be identified more accurately. Furthermore, if users check in with the same information, such as their date of birth, name, and possibly password. It doesn't matter because they have an identity to prove who they are. The User\_ID is also set to an integer as it is the easiest to manage due to its size. The rest are varchar as that means a string that holds as many characters as you assign it to. I set all my values to default, except for the email where I assigned it to 255. This is due to the email max limit being 255 characters.

I inserted a using directive at the top of the page which allows you to use SQL through your program.

I made a connection between the database and the windows forms by giving the program the server id, the unique identifier, and the database name. this will retrieve the chosen gib john database that I created and when it has been found, it will then give the user a message box saying that a connection has been made successfully. I will take the verification box away once the program is done. But I am keeping it in place at the moment so that testing is easier and so that I can be told clearly when a link has been made.

**SQL - creating a new user**



I started off by inserting the details one by one from the user\_details table on My SQL workbench and assigned them to variables of the same name.

Once I did that, I used the SQL insert and added the variables as parameters and set them to each text box that users will enter there details into, This how the database retrieves and stores user inputs for each column.

I then wrote the SQL connection statements so that the user details can be stored in the database, when this is executing a message box will show telling the user that a connection has been made and that the information is being inserted into user\_details. I then wrote insertSQL.Close(); to keep the application neat.

At the end I programmed it so that if any of the values were null. Then the connection will fail, and the user will be notified. The validation has already been done for the details to ensure they are eligible. But this is just to ensure that no wrong user details are entered.

When the continue button is pressed then.

