Contents

[Analysis 3](#_Toc68098577)

[Problem Definition 3](#_Toc68098578)

[Interview with Client – 19/07/20 3](#_Toc68098579)

[Testing software that is similar to the final product 4](#_Toc68098580)

[Observation of existing system 6](#_Toc68098581)

[Advantages and limitations of the current system 8](#_Toc68098582)

[Interview with Client to discuss User Interface – 24/07/20 8](#_Toc68098583)

[Results of the Questionnaire 9](#_Toc68098584)

[Summary 11](#_Toc68098585)

[Data sources and destinations 11](#_Toc68098586)

[Objectives 13](#_Toc68098587)

[Modelling the project 14](#_Toc68098588)

[Design 16](#_Toc68098589)

[High level overview 16](#_Toc68098590)

[User interface first drafts 20](#_Toc68098591)

[Database Design and Normalisation 25](#_Toc68098592)

[Examples of SQL queries to be used 27](#_Toc68098593)

[Algorithms 28](#_Toc68098594)

[Data Structures 33](#_Toc68098595)

[Security and Integrity of Data 33](#_Toc68098596)

[Implementation 34](#_Toc68098597)

[Assets 34](#_Toc68098598)

[App.xaml 34](#_Toc68098599)

[App.xaml.cs 35](#_Toc68098600)

[ApplicationPage.cs 35](#_Toc68098601)

[ApplicationPageValueConverter.cs 35](#_Toc68098602)

[BaseValueConverter.cs 36](#_Toc68098603)

[CreateNewProject.xaml 37](#_Toc68098604)

[CreateNewProject.xaml.cs 40](#_Toc68098605)

[CurrentProjectSettings.xaml 43](#_Toc68098606)

[CurrentProjectSettings.xaml.cs 45](#_Toc68098607)

[DeleteAccount.xaml 50](#_Toc68098608)

[DeleteAccount.xaml.cs 51](#_Toc68098609)

[ExitPage.xaml 52](#_Toc68098610)

[ExitPage.xaml.cs 52](#_Toc68098611)

[ForgotPassword.xaml 53](#_Toc68098612)

[ForgotPassword.xaml.cs 54](#_Toc68098613)

[HomePage.xaml 57](#_Toc68098614)

[HomePage.xaml.cs 58](#_Toc68098615)

[HowTo.xaml 66](#_Toc68098616)

[LoginScreen.xaml 68](#_Toc68098617)

[LoginScreen.xaml.cs 69](#_Toc68098618)

[MainPage.xaml 71](#_Toc68098619)

[MainPage.xaml.cs 72](#_Toc68098620)

[MainWindow.xaml 72](#_Toc68098621)

[MainWindow.xaml.cs 73](#_Toc68098622)

[Project.xaml 73](#_Toc68098623)

[Project.xaml.cs 75](#_Toc68098624)

[Register.xaml 122](#_Toc68098625)

[Register.xaml.cs 123](#_Toc68098626)

[ResetPassword.xaml 126](#_Toc68098627)

[ResetPassword.xaml.cs 127](#_Toc68098628)

[Settings.xaml 129](#_Toc68098629)

[Settings.xaml.cs 130](#_Toc68098630)

[Testing 133](#_Toc68098631)

[Test table 133](#_Toc68098632)

[Test evidence screenshots 135](#_Toc68098633)

[Evaluation 148](#_Toc68098634)

[Objective analysis 148](#_Toc68098635)

[User feedback 150](#_Toc68098636)

[Analysis of user feedback 152](#_Toc68098637)

[Possible extensions 153](#_Toc68098638)

# Analysis

## Problem Definition

Product: Room acoustic application

Client: Colin Biggs, Technical Director of Nuaire

**Background to the problem**

Nuaire is a company producing ventilation products for domestic and industrial purposes. While the materials and structure of the fans are important, something that is often overlooked is the sound level of the ventilation.

The sound level is important as it can vary for different types of rooms. For example, the acoustic requirements of ventilation in a concert hall will be different than in a classroom. The company currently has a simple application to calculate the effect of a sound source on a person in a room. However, the options on the application, such as choosing the materials the walls of the room are made of, are limited. This means that the results shown by the application are often not accurate enough for most situations.

## Interview with Client – 19/07/20

I interviewed my client to understand their needs for the application and for me to get an overview of what I had to do.

What is your existing system of the room acoustic application?

CB: It is currently a prototype in an Excel spreadsheet in which users can enter the dimensions and details of a room and from there all the fans in the system will be displayed along with if they are acceptable for the situation.

What are the benefits of the current system?

CB: It is quick to use and it can be used by all buyers to check they are buying the correct fan for their requirements.

What are the drawbacks of the current system?

CB: It’s currently limited to standard environments such as a classroom and there are limited options for example limited materials of walls. You can also only put one sound source into the program which isn’t very practical in most situations. And the current version is in Excel, and so is not ideal to share with customers.

Which new features would be the most important to you?

CB: The addition of more materials that you can select would be useful. It would be also be nice to have the ability of having more than one sound source in your modelled room as this is more likely to be closer to the reality of the situation. A graphic display to show the zones that the sound source affects the most so like a red, amber, and green zone for each source as this would make it easier for users to understand instead of numbers. The option to add new fans into the system would be useful for whenever new fans come out.

Which existing features do you want to keep?

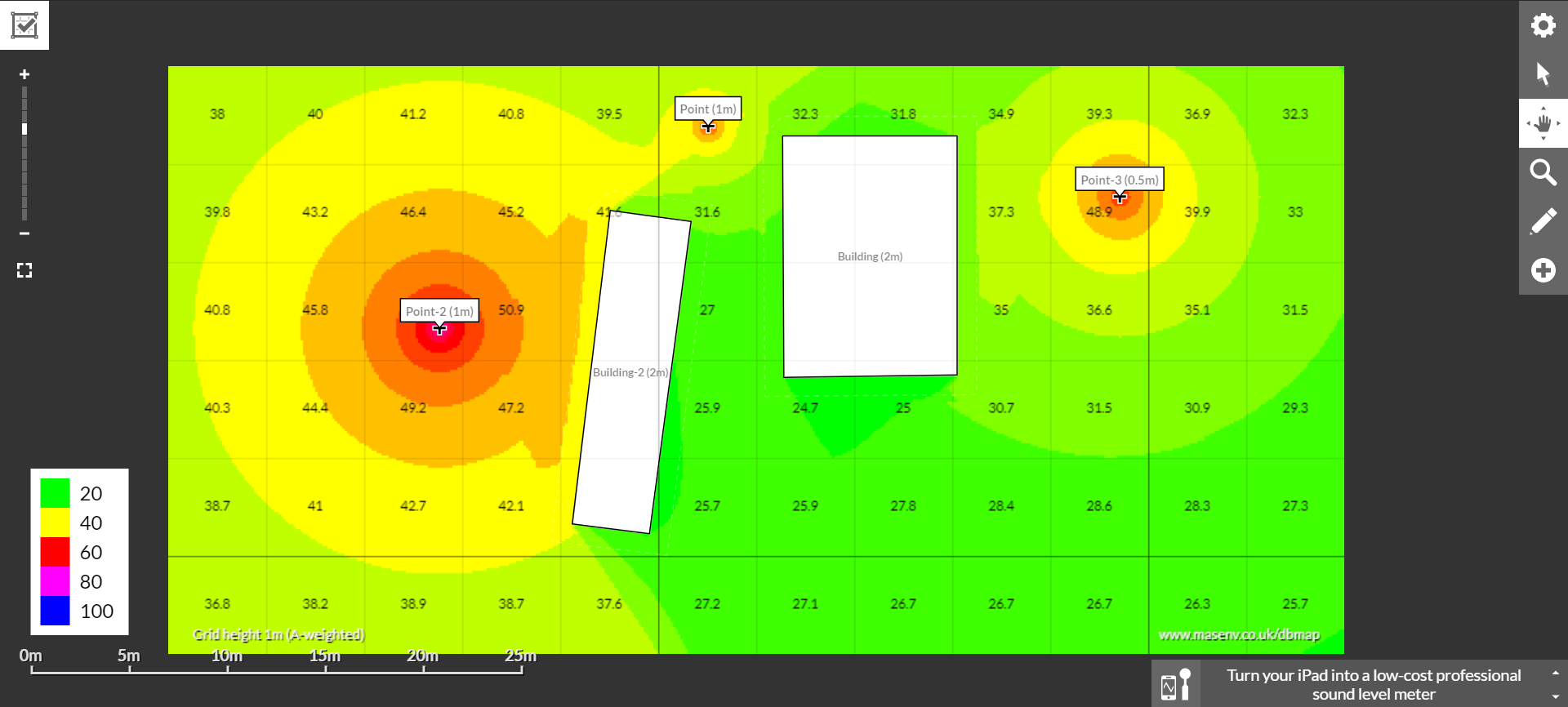
CB: The clarity of where to input the data needed in the system and the table of all the fans in the chosen fan range so you can compare them.

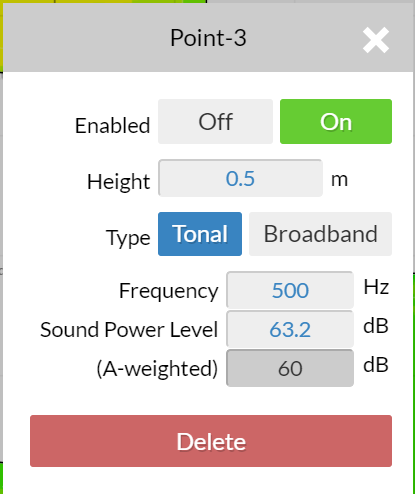
How many members of staff would use the new product?

CB: Around 30 people would use the product in the company, however there is also the possibility we would let customers use it as well so that they can do their own calculations before asking us for a quote.

## Testing software that is similar to the final product

To understand what sort of features I should add in the application as well as the ones my client mentioned, I tested a software that I thought was like what I wanted to produce. To do this, I used a product called Noise Mapping from MAS Environmental. The difference between this product and my intended product is that this one only looks at the effect of external sound sources.

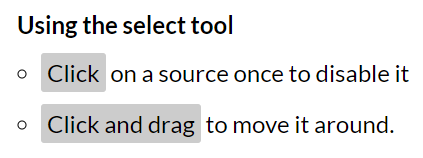


I can add point sound sources, and there is a menu to enter the relevant data about the point source:

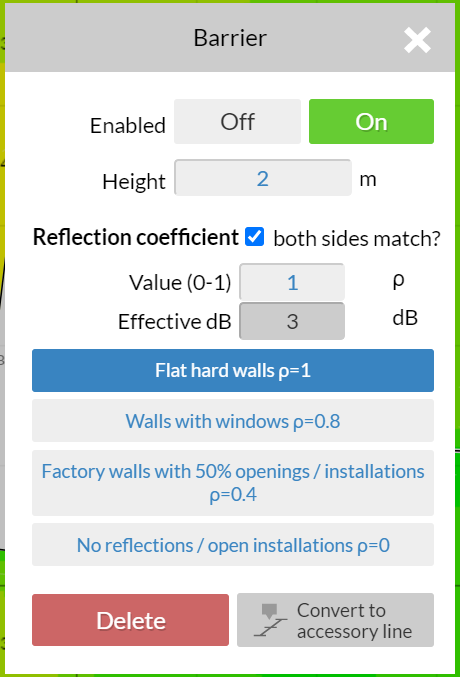
This is the same for other items I can add which include buildings and barriers.

The advantages to this product are:

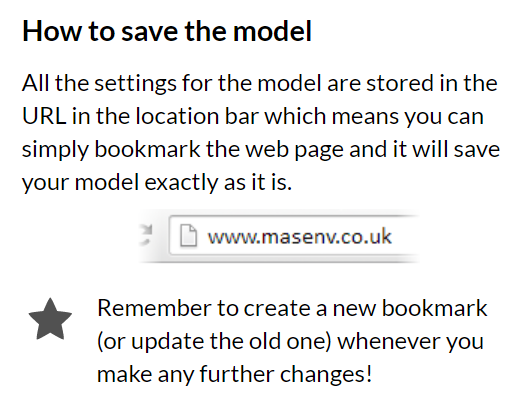
* It is easy to use as there are instructions on every button you click, e.g.:



* It gives you a graphic display, which is one of the things my client wanted to see in the finished product.
* You can also choose the height of the sound source from the ground which is important.
* It gives you options on the barrier to change the material:



There are some disadvantages to this application:

* The only way to save your model is to bookmark your current page. I don’t think this is the best way to save something as you would always have to go back to the main website to start a new model. As well as this you can end up with too many versions being saved in your browser and you could get confused between them. There is also a possibility that it would be hard to find the saved models if you already have many bookmarks. To solve this problem, I think it should be made possible to save and name a model in a designated place on the website itself.
* You also cannot change the size of the total grid. Once users create their model room in my intended product, I would like them to be able to change the size as it is possible to enter an incorrect dimension by mistake.

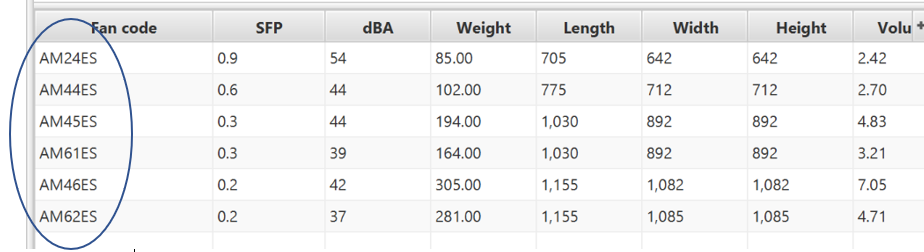
## Observation of existing system

I was lucky enough to be able to video call a worker at the company who has used the current system and they were able to talk me through the whole process. This starts with the customer making an enquiry and ended with the estimator giving back a suitable solution. The current system involves a separate app and the Excel spreadsheet that I am going to reprogram.

The current process is as follows:

*NB: the pictures only show some of the options on the app*

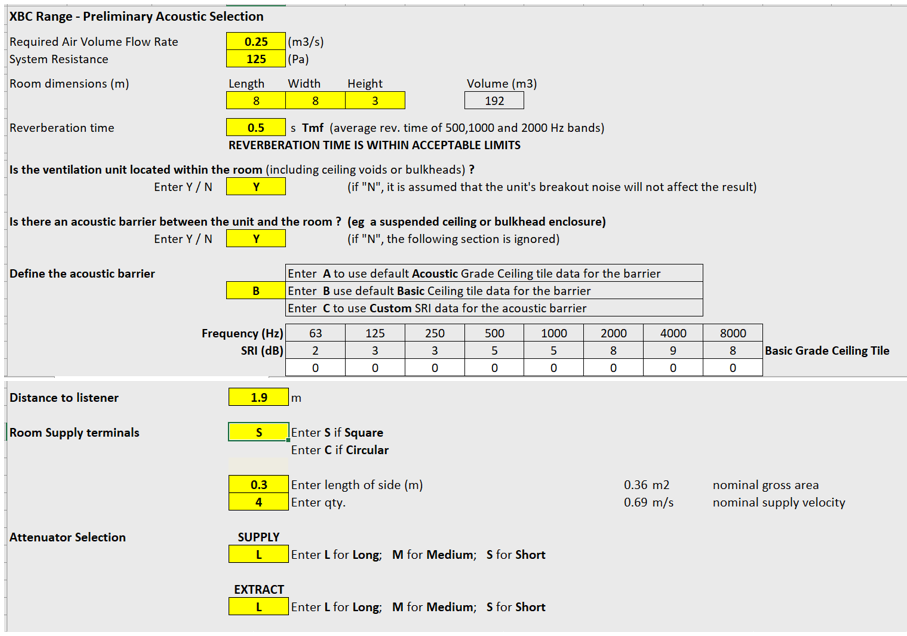
1. Customer uses fan selector (a separate app that has already been made)
   1. The fan selector is used to show the most suitable fan ranges based on the area the ventilation is being used for.

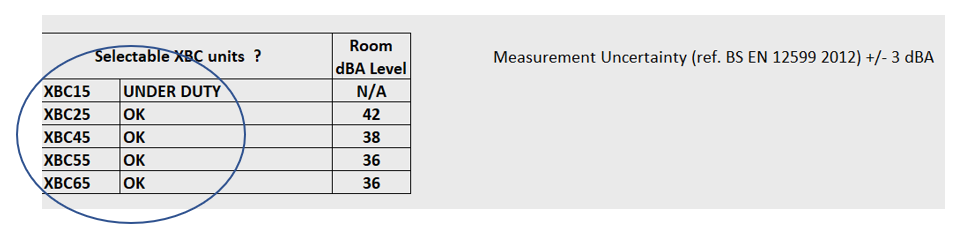
* 1. The customer then picks the most suitable fan range for them.
  2. The customer then enters the required values and the most suitable fan from their chosen fan range is selected.

The circled codes are the codes of the suitable fans.

1.4 If there is more than one selected fan, the customer chooses the one that is the most suitable for them.

1. The acoustics of the selected fan is checked (this can currently only happen if the customer in asking about a fan in the ‘XBC’ range)
   1. The customer asks an estimator in the company if the acoustic level of the fan is suitable and provides them with the necessary information to be able to check this.
   2. The estimator then enters the necessary data (yellow cells) into the Excel file.



* 1. The file calculates the suitability of the fans in the range and tells the customer if the fan they have chosen is appropriate or not.

1. Suggesting next steps: If the fan the customer originally chose is not suitable, then the estimator will advise them on another option. This could involve changing the length of the attenuator (this dampens the noise further) or changing to another fan that will fit the size of the space available and that has the lowest cost.

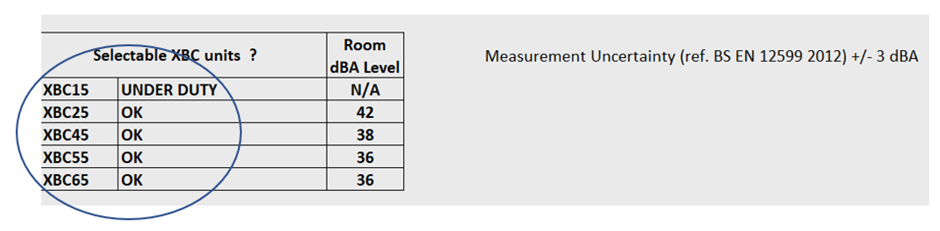
3.1 To do this, the estimator has to find the cost of each of the options from Nuaire’s main website. They also have to find the dimensions of the fan to see if they are small enough. Both of these values can be found in a separate database.

## Advantages and limitations of the current system

**Advantages**

* It is easy to enter the data as the cells are clearly coloured.
* It is easy to see which fans are suitable as they are in a clear table.

**Limitations**

* The estimator has to look at separate databases to find the required values for the cost and dimensions. It would be better to have them with the table so that you can see it straight from the table instead of having to look somewhere else.
* The client has to go back and forth with the estimators which could take some time.

## Interview with Client to discuss User Interface – 24/07/20

Would you prefer the application to be web or app based?

CB: I’d prefer it to be web based.

Would the application have different levels of user so admins and regular users?

CB: Yes, that would be ideal. So, the regular users would be the estimators and they would only be able to use the application with a limited number of rooms, for example classrooms and offices only. The admins would be able to use the full range of rooms available.

Do you have a logo or a colour scheme that you would like me to follow?

CB: Yes, and I can email the corporate ID. (font: Proxima nova, Calibri, colour: pantone 299

Do you have any other preferences relating to the appearance of the application?

CB: I’d like it to look similar to the fan selector, so similar fonts and layout. If possible, I’d also prefer it to have drop down menus instead of manually entering values so it’s harder to make an error.

While my client’s needs for the product are important, the end users will also need a say on what they would want in the application. With this in mind, I sent out a survey to the estimators in the company giving them a week’s time to reply.

## Results of the Questionnaire

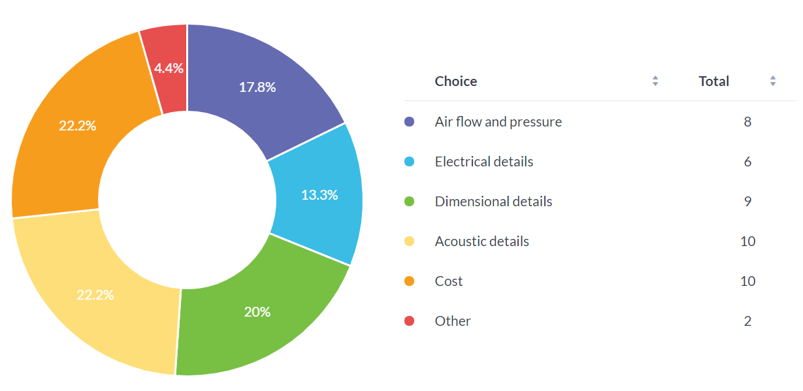
Some of the results from the questionnaire have shown that I should add some other features to the product.

1. Do you fully understand the acoustic data that the Nuaire fan selector produces?

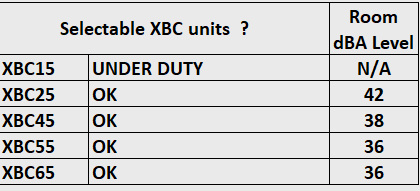


In this question, while most people have said that they understand the data, quite a high percentage also didn’t understand all the data. I think it would be worth putting a description of what each value is as a tooltip to make it easier for all users.

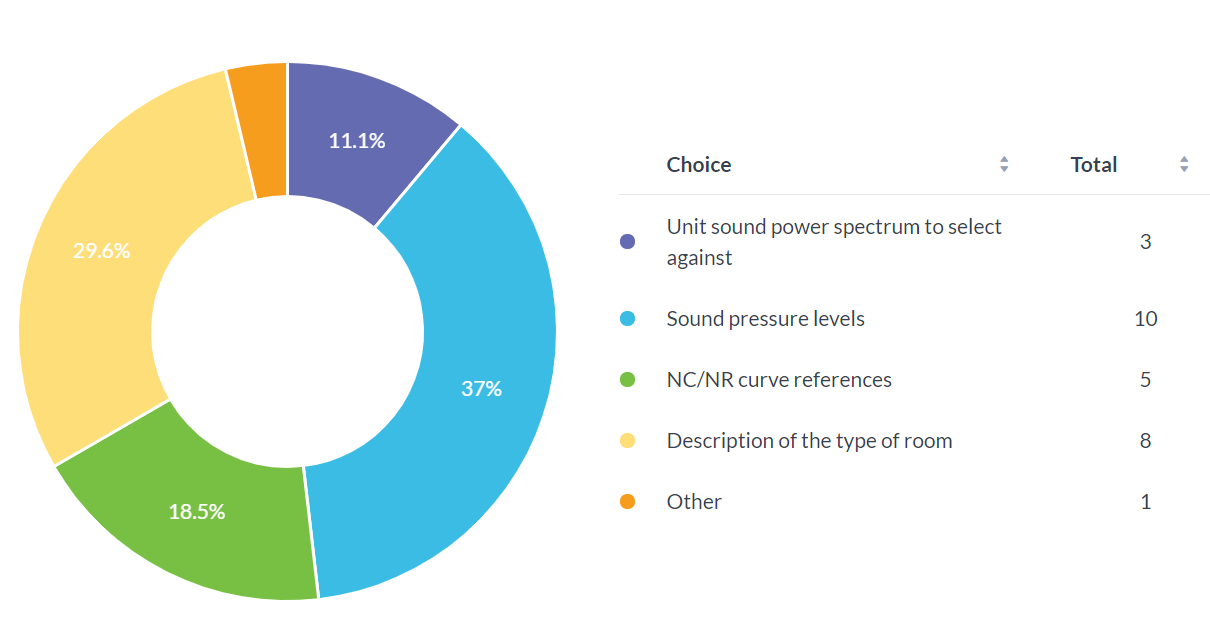
1. In selecting the suitable fan for customers, tick the following if you include those details.



There is a relatively even spread over the 5 given options, so it would be worth putting those in the final results table, that currently looks like the picture:

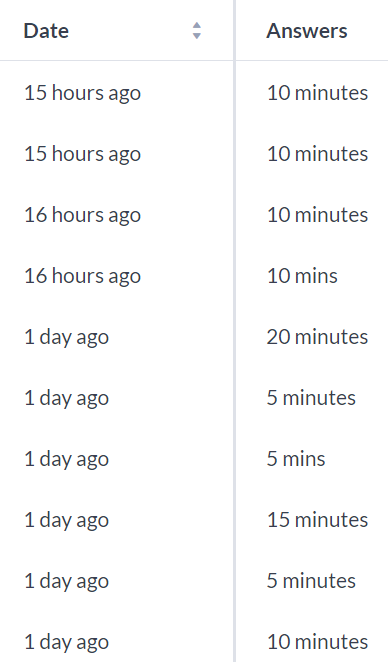


However, after discussion with my client, they said that it would only be worth putting in more acoustic details in the table. So, as well as the room dBA level, I would also display the octave band and/or overall dBa and NC/NR level.

1. What information do you get from the customer to do your calculations?

The description of the type of room and the sound pressure levels seem to have the highest votes, so to implement this I would make options relating to those two details when the estimators enter their information to make their calculations.

1. How long does it take to carry out a customer selection per location reference?

I asked this question to find the current timing, which is from 5 – 20 minutes, with the average time being 10 minutes. Using this current timing, I can compare it with when I use the product in the future to see if I need to improve its efficiency.

## Summary

**End users**

* The end users will the estimators in Nuaire, and possibly other customers.
* From the questionnaire, I now realise that certain personalisation options would be useful. After discussion with my client these would include tooltips, colour schemes, font size and units (metric or imperial).
* The input of the information into the application includes a unit sound power spectrum to select against, the sound pressure levels, NC/NR curve references and the type of room.
* Another necessary feature would be the addition of extra details to the output table. These would be the octave band and/or overall dBa and NC/NR level. NC/NR level is calculated by plotting the sound pressure and frequency, which will have to be done automatically by the application. My client has asked to make it possible to customise the output table by moving around the different columns.
* There is a senior estimator that will oversee the use of the application, as well as having their own admin access. This will allow access to more complex room types.
* A printable output page (or one that can be sent electronically) with the necessary graphs and sections would be required. The sections on this page would possibly be customisable in terms of the layout to match the layout of Nuaire’s consumer companies.

**User Needs**

* The application will need to be available online so that it is easy to download and run.
* It will need to have a graphic display to make it easier to operate and understand.
* The user will need to be able to save the diagram to their computer.
* To prevent the entering of wrong numbers there should be a dropdown of a selection of numbers.

**Limitations**

* The application will not be able to provide information on all types of rooms as it is not possible to predict all the rooms that customers will ask.
* The materials in the room will also be limited for the same reason.
* Customisations of diagram will also be limited due to complexity of the programming required.
* My knowledge of acoustic engineering is limited and so the application cannot be too complex for me to program with the resources that I have been given.
* The time constraint is that the system needs to be completed by Easter.
* The sound levels measured are for an imagined listener. Ideally, the listener can be placed anywhere in the room, however, for my time limit, it would be more ideal to place them in the centre.

## Data sources and destinations

The main data source for my program is the database with the information on each of the user’s projects. Another data source would be from the user’s personal information including email address, username and password. The other data sources would be in the C# program itself which is where I would put the required formulas and the displayed pages when the buttons are clicked. Each version of the rooms they will look at will be saved on the web server itself.

This table shows different data sources and information about each of them:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Source | Where stored | Process | User | Output | Destination |
| Email, Name | User’s database | Enters these when registering | Estimator |  |  |
| Username, Password | User’s database | Enters these when registering and logging in | Estimator |  |  |
| Formulae within the system | C# program | Uses input values during implementation | Estimator | The resulting sound levels of the source | Answer and graphics on page |
| Explanations | Program using HTML | Descriptions of the different values are given | Estimator | Descriptions of the values | Displayed as tooltips on page |
| Buttons for navigating | C# program | Once they are clicked the students will get to their desired pages. | Estimator | Shows the desired page | Displayed as page |
| Information about the room the client is enquiring | Project’s database | Enters this at the beginning of each case | Estimator | Data about the required rooms | Displayed on final output page at the end |

The data will most likely be stored on server as there can be a lot of data. Downloading it all to the user’s computer/hard-drive will make the functionality of the application slower, and in this case, my client wants the application to be as fast as possible.

The user’s name and their email address are pieces of data affected by the GDPR law. I will only be using these data to verify the users account and possibly on the welcome page. The user’s password for the application that they use to access their account will need to be secured with a hashing algorithm.

## Objectives

1. Create databases
   1. There will be a table where all the users’ projects are stored.
   2. Each type of room would have its NR value stored in a table.
2. Create a web application
   1. It must start up within 5 seconds.
   2. The web application will have the colour scheme and font that my client requested and will have to look similar to the company’s current website.
   3. Must be easy to use
   4. There will be a ‘Login’ tab
      1. Each user will have a password.
      2. There will be a register link for people who are using the website for the first time.
      3. The login page must have a blank form and simple form of email address and password.
      4. The register page must ask for the user’s name, email address, and let them choose their password.
      5. There must also be an option to change their password.
      6. A user’s account has to be verified by sending them an email.
      7. The logins are stored in a hashtable.
   5. There will be a ‘Home’ tab
      1. On the home page, the user would see all their saved projects.
      2. The projects have to have their last edited dates displayed as well as the project name.
      3. There must be a button to create a new project.
   6. The projects shown in the home tab will lead to another display
      1. The user will have to be able to save their projects, so that they are re-editable.
      2. There must be a button that provides an output of all the relevant information.
   7. There will be a ‘How To’ tab

2.7.1 The ‘How to’ tab will have an explanation of how to use the application.

* 1. There will be a ‘Settings’ tab
     1. Users must be able to choose the units they use (imperial or metric)

Extra things to add if I have the time

Ex1. User can be an admin

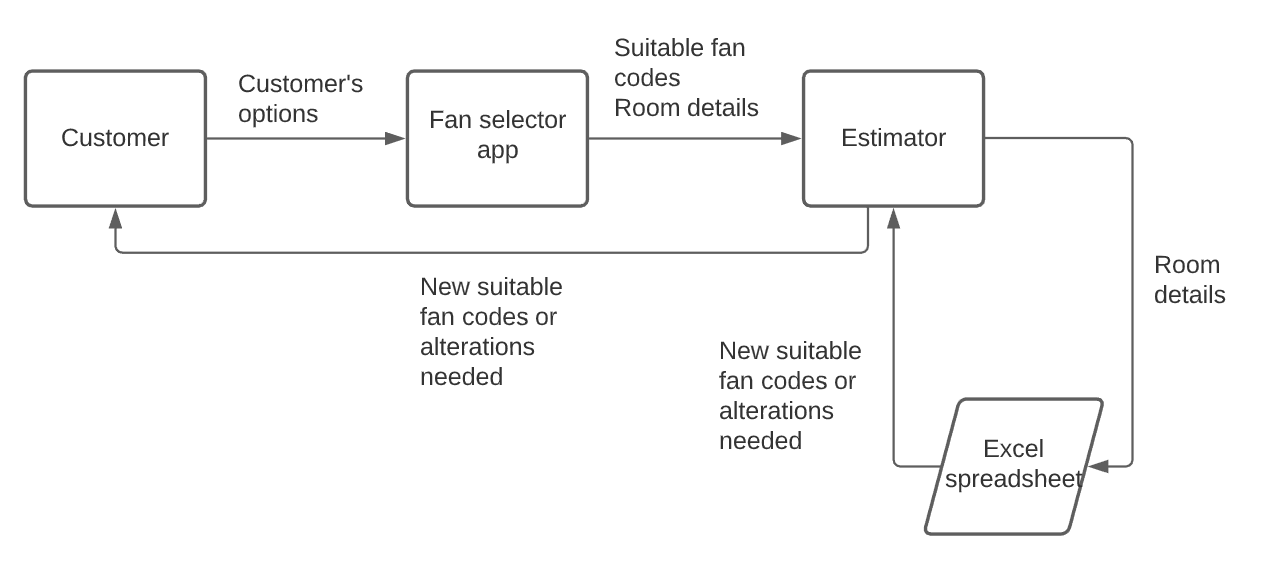
* 1. Being an admin allows you to see everyone’s projects and their details.
  2. An admin cannot edit other people’s projects.

Ex2. The user can personalise what is in the output summary, they might choose to not have certain sections.

Ex3. The resulting application that I create will be put as a link on to Nuaire’s company website.

## Modelling the project

**Data flow diagram**



**Hierarchy chart**

Main program

Output data

Process data

Get room details

Do the relevant calculations and see if the acoustic levels are suitable

Output the fans in the range

Output which fans are suitable

**Potential solutions for the project**

**Manual solution**

This solution would take up a lot of time from the estimators in the company as they have to do all the calculations with the relevant formulas themselves. The required graphs would also need to be manually plotted and this would take some time due to the accuracy required. Therefore this solution is not the most ideal given the number of enquiries made per day.

**Using an existing web/desktop application**

Having researched this previously to further understand what was required of me, I realised that there were very few solutions that often required payment. This is most likely because of the limited information on acoustics. The company is looking to spend the least money possible and so this is probably not the best option. As the company is also debating on letting customers use the application, it looks more professional if it is their own.

**Chosen solution: Programmed in a known programming language**

This would require a lower cost while also getting rid of the manual time-related limitations. The user would enter the required information, be able to move the sound sources around to where they want them, then get an output sheet at the end. This is easier than what the company has now, which requires the estimators to look through the whole Excel spreadsheet (which is quite confusing) and manually find the graphs required. It is also hard to visualise a specific situation on the spreadsheet, which the graphic element of the project would make much easier. This is also the most feasible for me as it fits my time limits.

# Design

## High level overview

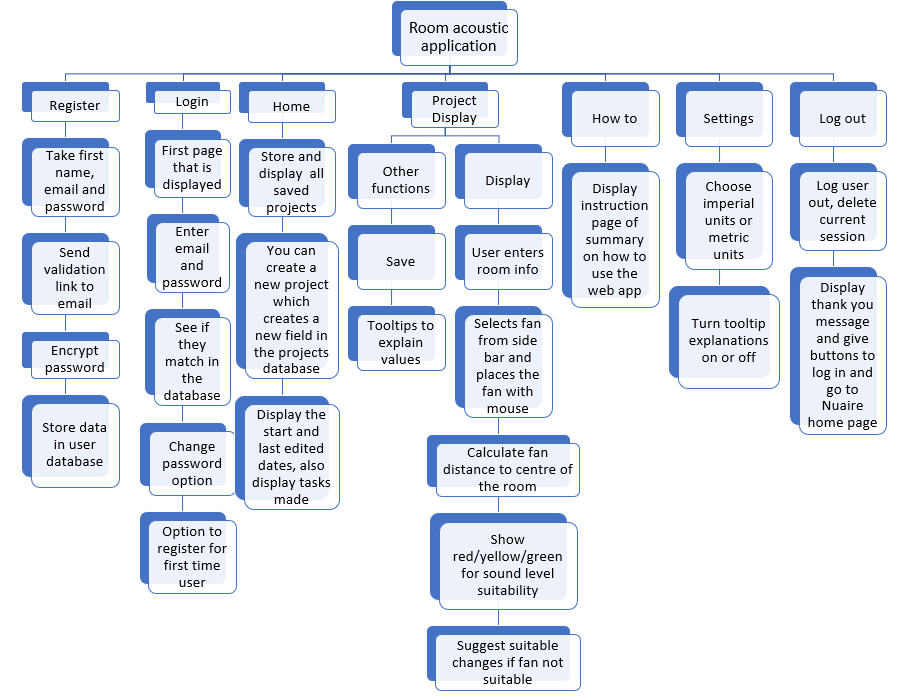
I am going to be programming a web application that tests the acoustic level of fan sources for my client’s company. The program will have a graphic display for the user to move around the fan’s sources and it will also show if the fan used is suitable or not.

**IPSO chart**

This chart shows what happens to the data in the new system in terms of input, output, processing and storage.

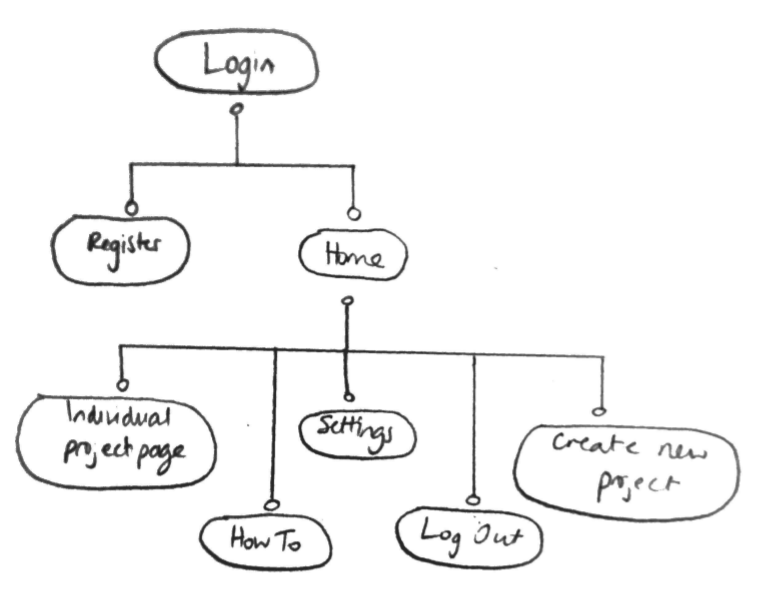
|  |  |  |  |
| --- | --- | --- | --- |
| Input | Process | Storage | Output |
| Email, Name, Password | Validate the login  Encrypt the password  When registering | User database | Send email with validation link to the user, which outputs a message saying validation successful |
| Formulae within the system | Uses input values during implementation | Program | The resulting sound levels of the source |
| Explanations | Descriptions of the different values are given | In a database | Descriptions of the values |
| Buttons for navigating | Once they are clicked the users will get to their desired pages. | Destinations are stored in C# program | Shows the desired page |
| Information about the room the client is enquiring | Formulae that are in the system are used to calculate the values and whether the fan is suitable | In a database that has all details about all the projects | Outputs the relevant data required by the user |
| Location of the fan | Calculates the distance from the listener (usually centre of the room unless specified), and so calculates | Saved together with the project, most likely in a database | Areas will be red, yellow, or green depending on the sound level for the listener |

**Hierarchy chart**

This allows me to see a top down design of the web app which splits the project up into sections. This makes it easier for me to know what I need to do for the project. This is different to the existing system as the chart is based on the web pages of the web application, which the current system does not have.

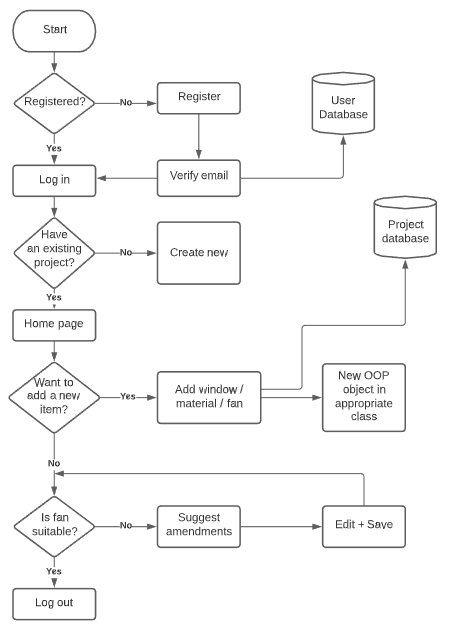
**Page navigation**

This will help me when designing my user interface as I know what pages I need to make.



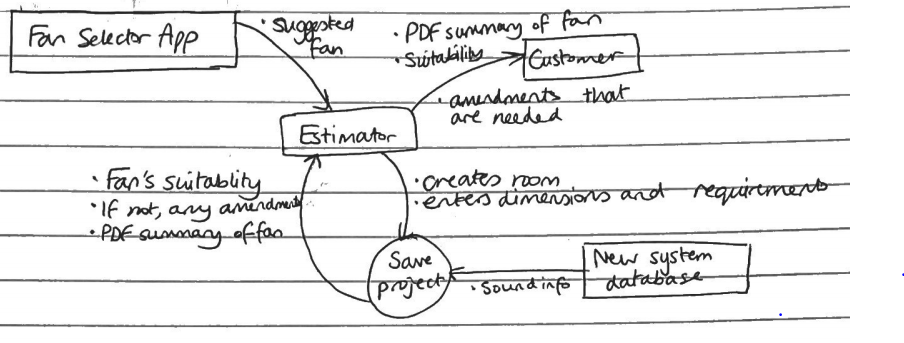
**System flowchart**

This chart outlines how the new system should operate. It looks slightly more complex than the current system however this mainly because of the registering section, which is not currently required, and the new interface adds some complexity too. The data is also saved to a separate database instead of being kept in the same spreadsheet.





**Data flow diagram**



The data flow diagram is not that different from the current system’s apart from the fact that the estimator needs to draw out the room. While this is more work on the estimator’s side, it allows a more accurate judgement on the suitability of the fan, which is what my client required.

## User interface first drafts

The font I will be using is Montserrat from Google Fonts. This is the href link: <https://fonts.googleapis.com/css2?family=Montserrat&display=swap>.

The main colour is #00a3e0:

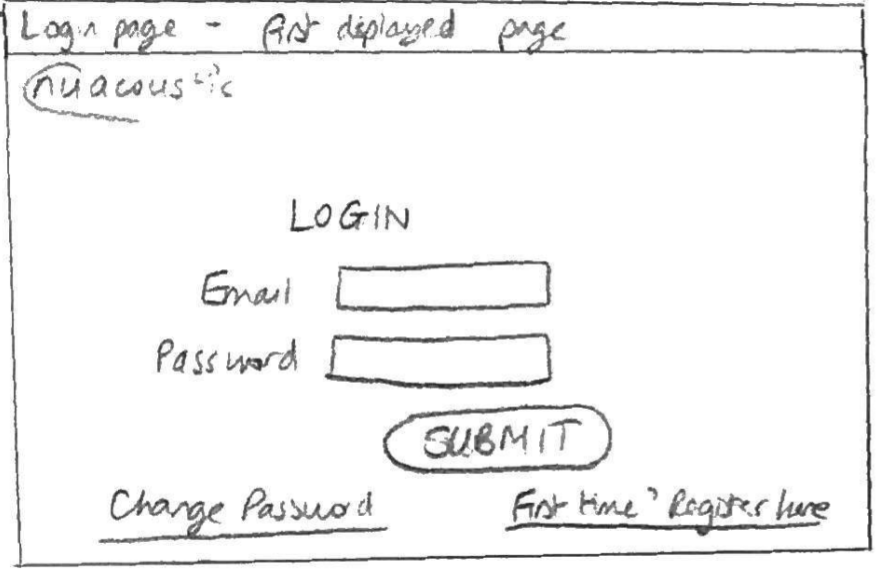
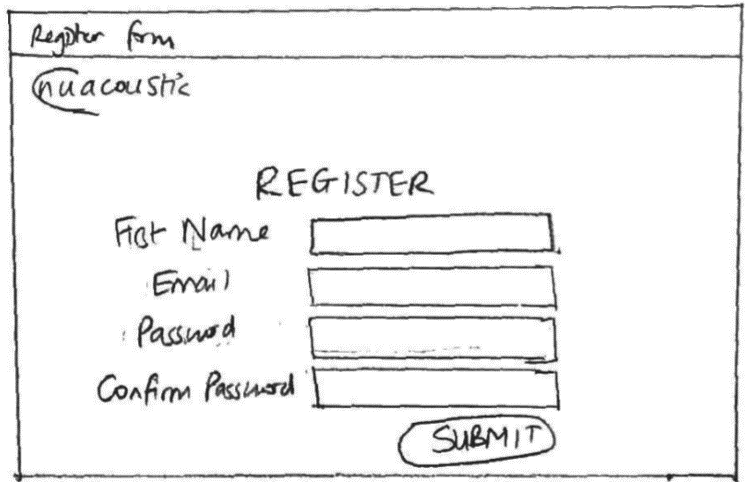
The screen background will be white.

Buttons:

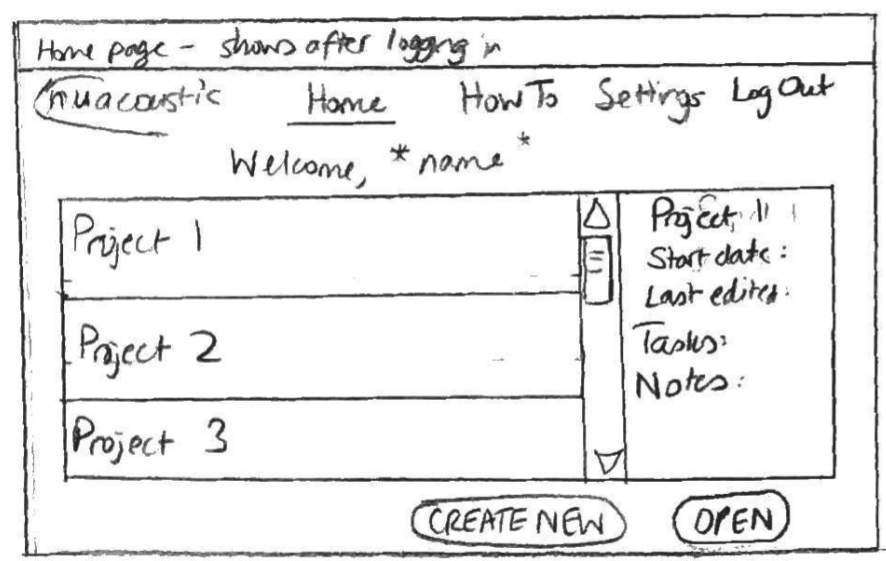
My client wants the interface to be similar to the company’s existing website, so I will be using the same or similar format for the buttons. Here is an example:



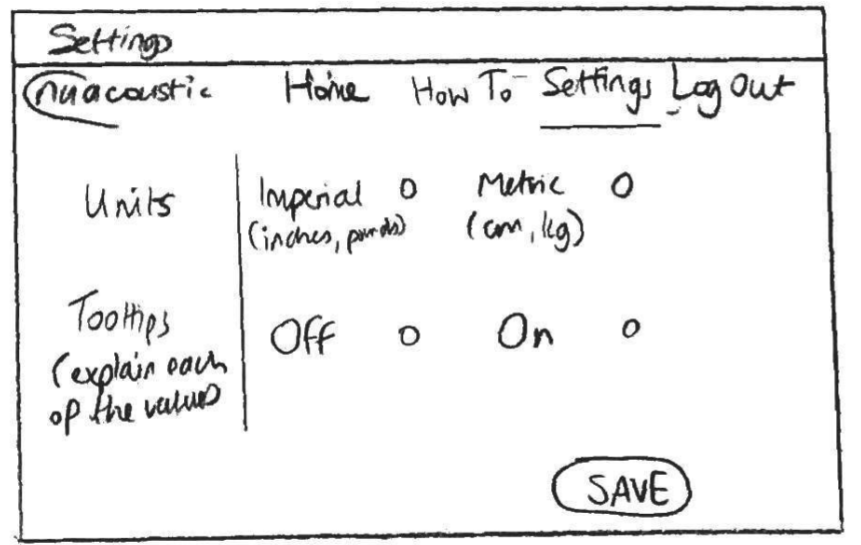
**Login form Register form (accessed by register link on login)**



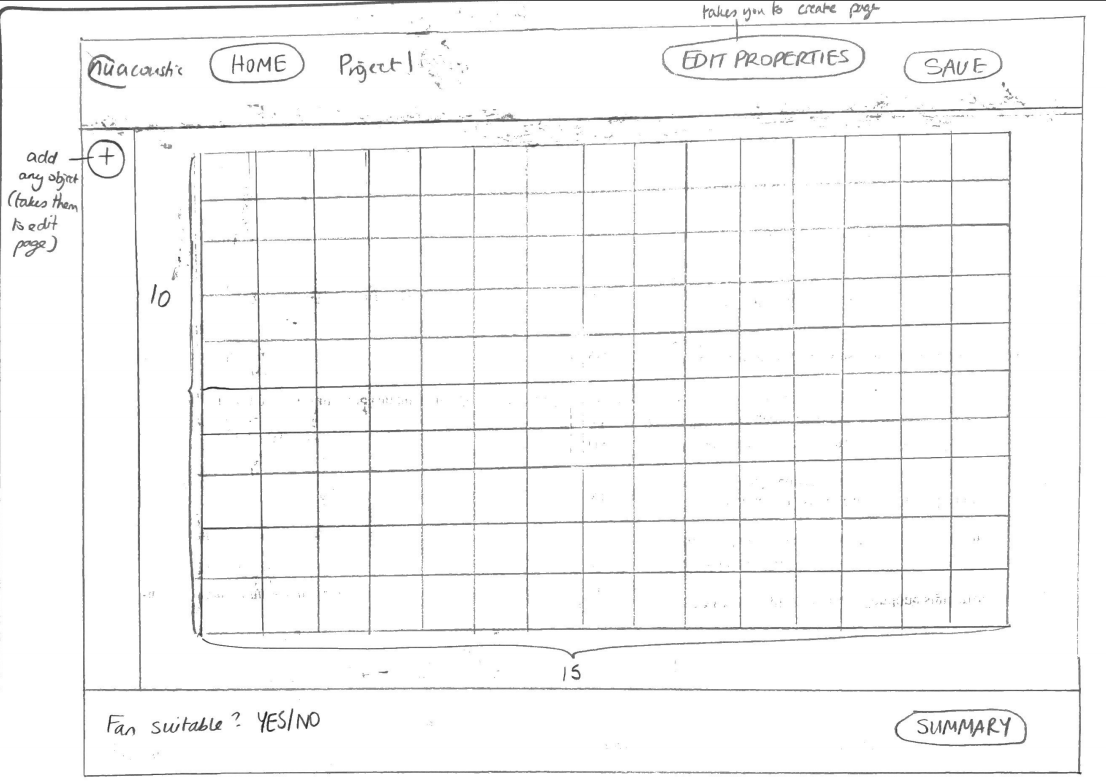
**Home page (displays the user’s project and allow them to create a new one)**

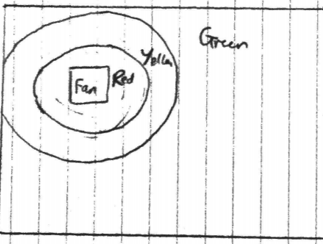


My client has previously stated that different companies use different units, partly because of the country they are in, so I thought it would be useful to have an option to change units.

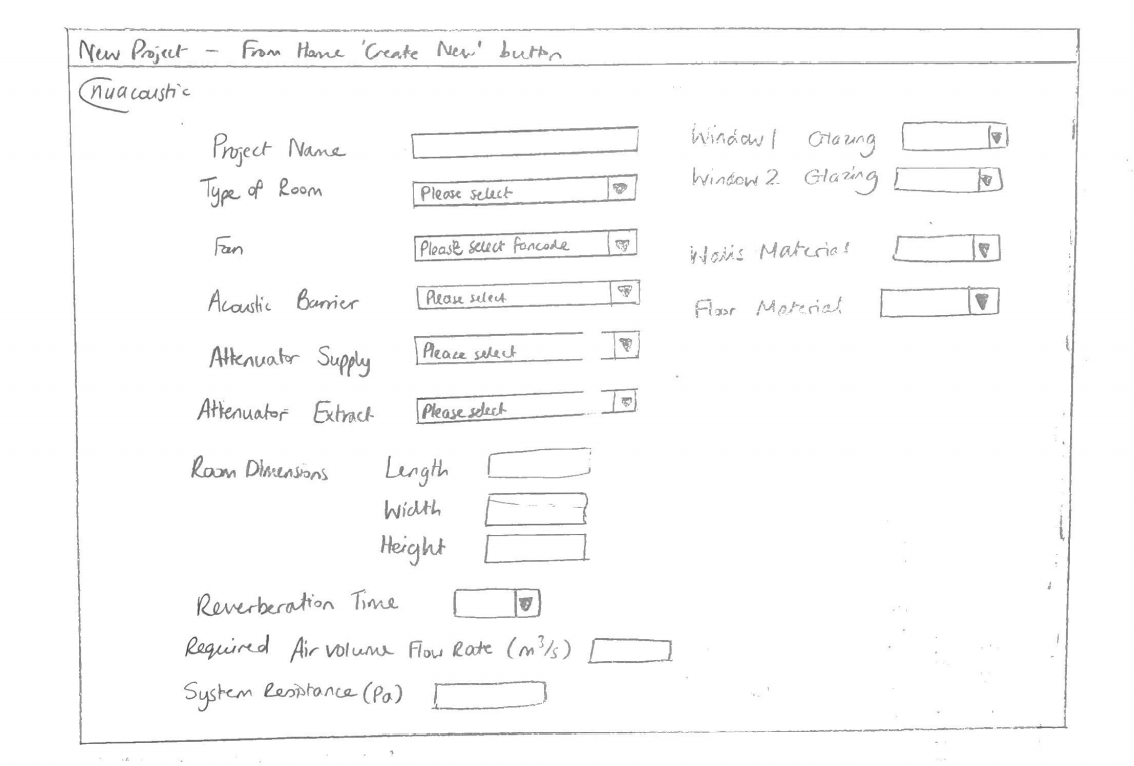


This is for the main project page where users will edit their products. On the left hand column, prothey can add new items by dragging the boxes. When each added object is hovered over, they can be assigned names and values.

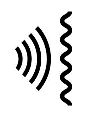
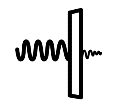


The diagram on the left shows the colour scheme for the fan suitability. Depending on if it is an acceptable noise level, the area will turn green, yellow or red. For a fan to be suitable for its use, the whole room will need to be green.

**New project page**



**Symbols on the application**

I plan to have images as symbols that the user can drag.

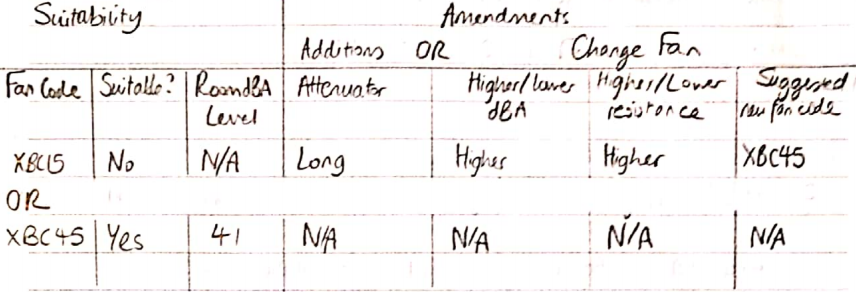
Fan: Attenuator: Acoustic Barrier:

When a user starts a project, they will start by drawing a rectangle with their cursor, much like on Microsoft Word.

**System outputs**

The main output of the web app is whether the chosen fan gives a suitable acoustic level.

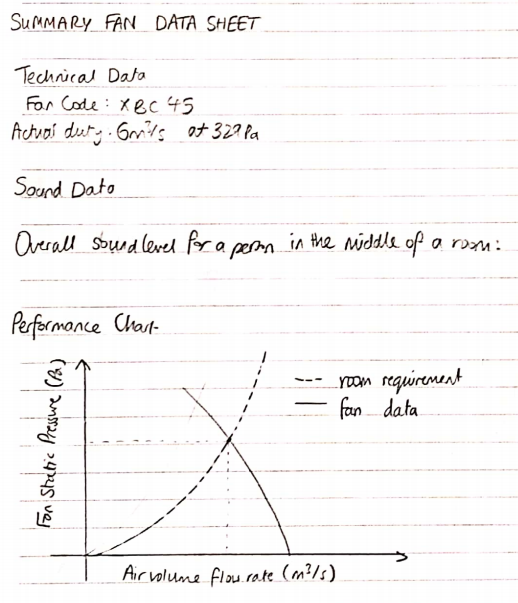
This would be output in a table as shown below:



The first row with fan code XBC15 shows the example of a fan that is not suitable. The table will say what needs to be changed, i.e. in this case the user needs to either put a long attenuator in the ventilation system or they need to change the fan. A user would have to change a fan because either the dBA or resistance of the fan is not right for the situation. Instead of letting the user pick out a new fan for themselves, the app will suggest one for them that would be suitable.

The second row with fan code XBC45 shows another scenario where the fan is suitable. Here, a value for the room dBA level is given as this is only possible when the fan is suitable. The data in the amendments part of the table all show N/A as there is no need to amend anything as the fan is already at a suitable sound level.

Another part of the output is the summary fan data sheet, which I plan to look like this:



Headings like this will be blue

Other data including the project name, the client and the date will also be put into the technical data section.

I want this data sheet to be a simple summary of the chosen fan. On the sheet there is a technical data section, a sound data section and a performance chart section. I plan to make this sheet a PDF file that can be accessed as a link on the table part of the output. The PDF format would allow the sheet to be printed and shared easily.

## Database Design and Normalisation

To base the new system around a database, it needs to be normalised to third normal form to avoid duplicated data and eliminate repeating attributes.

**tblUser**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Data Type** | **Validation** |
| UserID | Primary | Integer | None |
| FirstName |  | String | 1-20 |
| Email |  | String | >5, include ‘@’ |
| Password |  | String | 1-20, include lower, upper, number |
| Admin |  | Boolean |  |

**tblProject**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Data Type** | **Validation** |
| ProjectID | Primary | Integer | None |
| UserID | Foreign | Integer | None |
| ProjectName |  | String | 1-20 |
| LastEditDate |  | Date | Before today |
| FanCode | Foreign | String | None |
| TypeID | Foreign | String |  |
| Volume |  | Real | >0&(width\*length\*height) |
| FloorMaterial |  | String | >0 |
| WallsMaterial |  | String | >0 |
| XCoordCentre |  | Real |  |
| YCoordCentre |  | Real |  |
| RoomLength |  |  |  |
| RoomWidth |  |  |  |
| ReqAirVolRate |  | Real | >0 |
| SystemRes |  | Real | >0 |
| ReverbTime |  | Real | 0-0.8 |

**tblFan**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Data Type** | **Validation** |
| FanCode | Primary | String | None |
| ProjectID | Foreign | Integer | None |
| UserID | Foreign | Integer | None |
| XCoord |  | Real |  |
| YCoord |  | Real |  |
| dBA |  | Real | >0 |
| AirVolRate |  | Real | >0 |

**tblWindow**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Data Type** | **Validation** |
| WindowID | Primary | Integer | None |
| ProjectID | Foreign | Integer | None |
| UserID | Foreign | Integer | None |
| Double |  | Boolean |  |
| XCoord |  | Real |  |
| YCoord |  | Real |  |

**tblAcousticBarrier**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Data Type** | **Validation** |
| BarrierID | Primary | Integer | None |
| ProjectID | Foreign | Integer | None |
| UserID | Foreign | Integer | None |
| MaterialName | Foreign | String | >0 |
| XCoord |  | Real |  |
| YCoord |  | Real |  |

**tblAttenuator**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Data Type** | **Validation** |
| AttenuatorID | Primary | Integer | None |
| ProjectID | Foreign | Integer | None |
| UserID | Foreign | Integer | None |
| SupplyLength |  | String | >0 |
| ExtractLength |  | String |  |

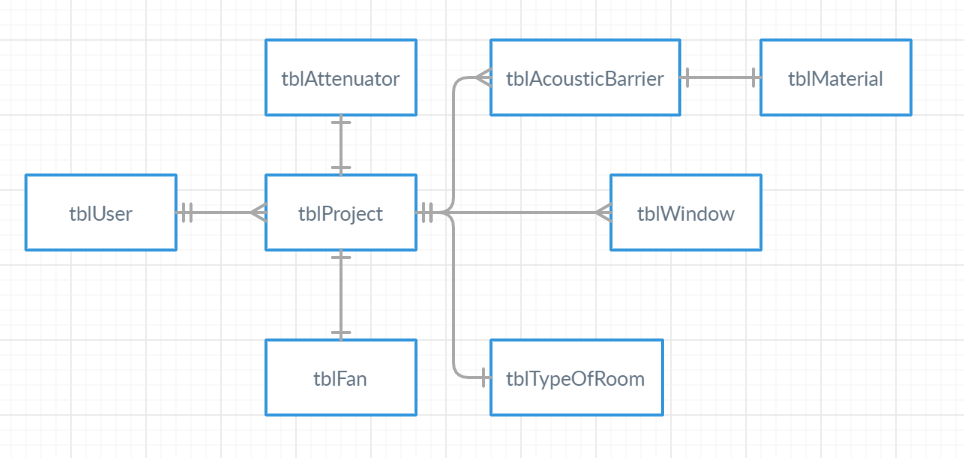
**tblMaterial**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Data Type** | **Validation** |
| MaterialName | Primary | String | None |
| DecibelsTaken |  | Real | >0 |

**tblTypeOfRoom**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Key** | **Data Type** | **Validation** |
| TypeID | Primary | String | None |
| ProjectID | Foreign |  |  |
| UserID | Foreign |  |  |
| NRVal |  | Real | >0 |

**ER diagram**



The diagram above the relationships between the entities of the database. A user can create many projects. A project represents a room which can have many fans, many walls and acoustic barriers (which could be any furniture). A wall can have many windows. The acoustic barrier and wall will have a certain material that they are made of.

## Examples of SQL queries to be used

I have only included certain SQL queries here, as others are quite similar, e.g. updating the value of Double to TRUE on tblWindow is similar to the query for deleting the user’s email, as it is just updating the table.

|  |
| --- |
| Making a new project |
| INSERT INTO tblProject(ProjectID, UserID, ProjectName, LastEditDate, FanCode, Volume, FloorMaterial, ReqAirVolRate, SystemRes, ReverbTime)  VALUES(?, ?, ?, Today, ?, ?, ?, ?, ?, ?) |

|  |
| --- |
| Registering new user |
| INSERT INTO tblUser(UserID, FirstName, Email, Password)  VALUES(?, ?, ?, ?) |

|  |
| --- |
| Deleting a project |
| DELETE FROM tblProject  WHERE ProjectID=? |

|  |
| --- |
| Displaying all the user’s projects on the home page |
| SELECT ProjectName, LastEditDate  FROM tblProject  WHERE ProjectID=? AND UserID=? |

|  |
| --- |
| Deleting a user’s email to comply with GDPR rules |
| UPDATE tblUser  SET Email = “”  WHERE UserID=? |

## Algorithms

|  |
| --- |
| Dragging images |
| The user will drag 4 images that represent the location of the fan’s inlets and outlets. |
| Procedure DragImage  draggedImage 🡨 Image  mousePosition 🡨 Point  IF no image is selected THEN  position 🡨 GetPosition  offset 🡨 position – mousePosition  mousePosition 🡨 position  draggedImage.Left = draggedImage.Left + offset.X  draggedImage.Top = draggedImage.Top + offset.Y  End Procedure |

|  |
| --- |
| Finding the coordinates of the centre of the room |
| These coordinates will be used when finding the distance of the fan from the centre of the room. |
| Procedure GetCentreCoordinates  double rectangle.CentreX = (LocationX1Y1.X + RoomLength)/2  double rectangle.CentreY = (LocationX1Y1.Y + RoomWidth)/2  End Procedure |

|  |
| --- |
| Getting coordinates of fan (or any object) |
| I will be putting the symbol for the objects into textboxes, as Visual Studio has a function that allows you to see the coordinates of the top left corner of the textbox |
| Procedure GetFanCoordinates  double fan.x = textboxFan.Location.X  double fan.y = textboxFan.Location.Y  End Procedure |

|  |
| --- |
| Calculating fan distance from listener |
| This will be used when calculating the sound level of the fan from the listener. |
| Function GetFanDistance  double fanxdistance = |fan.x – listener.x|  double fanydistance = |fan.y – listener.y|  double Xpixel = RoomLength / 600  double Ypixel = RoomWidth / 320  double horizontal = Xpixel \* fanxdistance  double vertical = Ypixel \* fanydistance  double fandistancetolistener = sqrt((horiztonal)^2 + (vertical)^2))  return fandistancetolistener  End Function |

|  |
| --- |
| Calculating acoustic barrier/window (called object) distance from fan |
| This will be used when calculating the sound level of the fan from the listener. |
| Procedure GetObjectDistance  double object.xdistanceFan = |fan.x – object.x|  double object.ydistanceFan = |fan.y – object.y|  double object.distanceFan = sqr((object.xdistanceFan)^2 + (object.ydistanceFan)^2)  End Procedure |

|  |
| --- |
| Write output textfile |
| This takes the required information and puts it in a textfile. |
| Procedure WriteProjectName(FanCode, ActualDuty, OveralldBA)  ProjectName as String  Save Filename = SummaryOutput.txt  ProjectName = from database  OPEN Filename for APPEND  WRITE ProjectName, FanCode, ActualDuty, OveralldBA  CLOSE Filename  End Procedure |

|  |
| --- |
| Hashing passwords |
| I will be using the SHA256 hashing algorithm to hash the users’ passwords. The passwords are converted into a 64 character string. |
| Function CreateSHA256(string rawData)  Sha256 = SHA256.Create  byte[] bytes = ComputeHash(rawData)  int count 🡨 0  FOR count < bytes.Length  stringBytes = Convert.ToString(bytes[count])  count 🡨 count + 1  string hash 🡨 stringBytes  return hash  End Function |

|  |
| --- |
| Finding hashtable key |
| Each of the user’s logins are put in to a hashtable. The hash key is calculated from their email string. |
| Function GetHashtableKey(string email)  int countEmail 🡨 0  int arrayLength = email.Length  array emailstr[arrayLength]  int total 🡨 0  for each char c in email  emailstr[countemail] = Convert.ToInt(c)  total = emailstr[countemail] + total  countemail = countemail + 1  int key 🡨 total % 997  return key  End Function |

|  |
| --- |
| Checking contents of the hashtable when a user tries to log in. |
| To find the hashtable key of the password in the hashtable, it is calculated with the previous procedure above. The contents at this location are then compared with the hash of the password provided by the user. |
| Procedure LogIn(string email)  key 🡨 GetHashtablekey(string email)  hashedPassword 🡨 CreateSHA256(string password)  IF key exists in hashtable AND Logins[key] == hashedPassword THEN  Go to home page  ELSE IF key does not exist in hashtable THEN  Output “Your email or password is incorrect.”  ELSE  Output “Your email or password is incorrect.”  End Procedure |

|  |
| --- |
| Sending a verification code to the user. |
| I use this subroutine twice in the program, one for when registering and when the user forgets their password and they would like to reset it. The only difference between the two is the content of the message. This version is for when the user is registering. Here, I have used the SMTP protocol to send an email to the user’s email address. I have created an email for this purpose – “nuacoustic.nuaire@gmail.com” |
| Procedure SendVCode  string from, pass, messageBody 🡨 “0”  string randomCode 🡨 generate a random number from 0 to 999999 and convert to a string  string to 🡨 user’s email  from 🡨 “nuacoustic.nuaire@gmail.com”  pass 🡨 “!testPass123!”  messageBody 🡨 “Your verification code is “ + randomCode + “.”  message.To.Add(to)  message.From 🡨 new MailAddress(from)  message.Body 🡨 messageBody  message.Subject 🡨 “Email Verification”  smtp Client 🡨 “smtp.gmail.com”  smtp.EnableSsL 🡨 True  smtp.Port = 587  smtp.Credentials 🡨 NetworkCredentials(from, pass)  smtp.Send(message)  End Procedure |

|  |
| --- |
| Calculating the graduations on the grid |
| To help users, I put a button that would show the user the graduations of the grid to make it easier for them to model where the fans went. As each user has the option to choose either meters or feet as their unit, this takes that into account by displaying the graduations in the user’s chosen unit. |
| Procedure Graduations  decimal roomLength 🡨 get from database  decimal roomWidth 🡨 get from database  decimal horizontalGradfull 🡨 roomLength / 15  decimal horizontalGraduation 🡨 round horizontalGradfull to 2dp  decimal verticalGradfull 🡨 roomWidth / 10  decimal verticalGradfull 🡨 round verticalGradfull to 2dp  string units 🡨 get from database  IF units = “Feet“ THEN  decimal convertToFeet 🡨 3.28  horizontalGraduation 🡨 (horizontalGradfull \* convertToFeet) all rounded to 2dp  verticalGraduation 🡨 (verticalGradfull \* convertToFeet) all rounded to 2dp  OUTPUT “Horizontal: starting from 0 on the left, each graduation is “ + horizontalGraduation + “ft. Vertical: starting from 0 at the bottom, each graduation is “ + verticalGraduation + “ft.“  ELSE  OUTPUT “Horizontal: starting from 0 on the left, each graduation is “ + horizontalGraduation + “m. Vertical: starting from 0 at the bottom, each graduation is “ + verticalGraduation + “m.“  End IF  End Procedure |

|  |
| --- |
| Solving a cubic equation |
| To find the fan flow rate, which is the amount of air that passes through a fan in a given time, a cubic equation needs to be solved. The equation is different depending on which fan the user decides to use. The equation is in the form Ax^3 + Bx^2 + Cx + D, where the capitals correspond with the parameters of the function. This uses Cardano’s method to solve a cubic equation. |
| Function SolvingACubic(decimal A, decimal B, decimal C, decimal D)  decimal temp1 = ((3 \* A \* C) – B^2) / (9 \* A^2)  decimal temp2 = ((9 \* A \* B \* C) – (27 \* A^2 \* D) – (2 \* B^3)) / (54 \* A^3)  decimal temp3 = temp2 + (temp1^3 + temp2^2)^(1/2)  decimal temp4 = temp3 ^ (1/3)  decimal temp5 = temp2 – (temp1^3 + temp2^2)^(1/2)  decimal temp6 = temp5 ^ (1/3)  decimal solution = temp4 + temp6 – (B / (3\*A))  return solution  End Function |

|  |
| --- |
| Finding the new sound value/SWL (sound power level) in decibels |
| To do this, we find the amount by which the sound decreases as a result of absorption then subtract this by the initial fan volume which is stored in the database. |
| Function FindSWL(decimal FanFlowRate)  decimal RequiredAirVolumeFlowRate 🡨 entered by user so can be accessed via the database  decimal speedReductionRatio 🡨 RequiredAirVolumeFlowRate / FanFlowRate  decimal DeductionIndB 🡨 50log(speedReductionRatio)  decimal InitialVolume 🡨 from database  decimal SWL 🡨 InitialVolume - DeductionIndB  return SWL  End Function |

|  |
| --- |
| Finding the SPL using Schultz’s formula |
| This is used for different octave bands and different fan inlets and outlets (breakout, supply and extract). The SWL is calculated from the above function FindSWL. The distanceFromListener and RoomVolume parameters are both entered by the user and are accessed via the database. In total there are 8 octave bands, which are different frequencies that the calculations are performed at and the SPL is calculated for each octave band. |
| Function FindingSPL(SWL, distanceFromListener, RoomVolume, octaveband)  decimal distancevariable 🡨 10log(distanceFromListener)  decimal volumevariable 🡨 5log(RoomVolume)  decimal SPL = SWL – distancevariable – volumevariable – 3log(octaveband) + 12  return SPL  End Function |

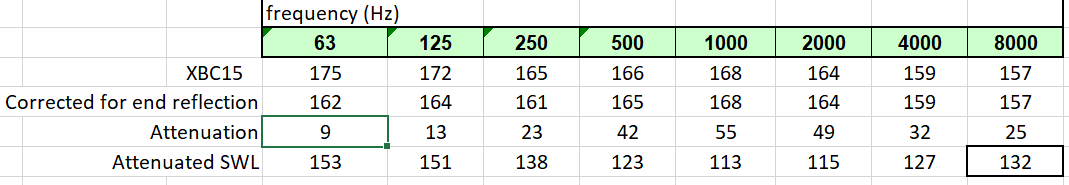
|  |
| --- |
| Finding the suitability of the fan at each octave band. |
| NR curveThe graph below is called an NR curve. Each type of room the user can pick from (e.g. concert hall, living room etc.) has an NR value, which is represented on the right of the graph. If a room has an NR value of 60 (as shown with the arrow), that line is the room’s NR curve. At a particular octave band, which would be one of the values on the x-axis, it has a sound pressure level, found the y-axis. For example, a room with an NR value of 60 at the octave band 1000 Hz has a sound pressure level of 60dB. This means that 60dB is the loudest sound level possible that can be used in the room that is comfortable for the human ear.  🡨  So, to find out if the fan is suitable at that octave band, the SPL for that band has to be lower than the one found on the graph. We then use this information to see if the fan is suitable overall, which is only possible if it is suitable at every octave band. |
| Function FindIfSuitable(SPLCalculated, NRSPL)  boolean suitable = false  IF (SPLCalculated >= NRSPL)  suitable = true  End IF  return suitable  End Function |

## Data Structures

I will be using a hashtable ‘Logins’ to store the hashed users’ passwords which will keep them secure as they cannot be converted back into their original passwords.

There will be a series of JPEG images that I am using for the symbols and the logo and these will be put in textboxes on the screen.

I also plan to be using a 2D array to store the fan’s acoustic details, e.g.:



The output summary will be a text file – as it is specific to a project, it will only be used once, so the formatting is not as important. If I am to use text files however, I will need to make sure it is easy to read instead of just a whole block of text that will make it difficult for the user to find what they need quickly.

## Security and Integrity of Data

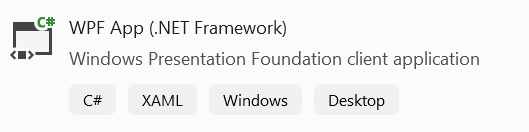
The personal data stored about users are their email addresses and their names. These will not be hashed; however, they will be deleted if a user deletes their account. The user will also have the option to remove their email address from the database if they want as it is only used for verification. The passwords the users choose will be put into the database so they are secured. I want to keep the security as simple but as effective as possible, as if there are too many stages it will become difficult and time consuming for users to log on.

To ensure the integrity of the stored data, all the data that is entered will be controlled by validation rules. The data will be entered by selecting from drop-down menus where possible. In other cases, the user will only be able to enter a certain range of values. This stops errors that would cause incorrect data to be stored or even the system to crash. Time is also saved as there are a set number of responses that the system will be prepared for.

# Implementation

## Assets

I used Visual Studio software to program this application and decided to use the languages C# and XAML. The type of app was WPF as this allowed me to fit my client’s needs.

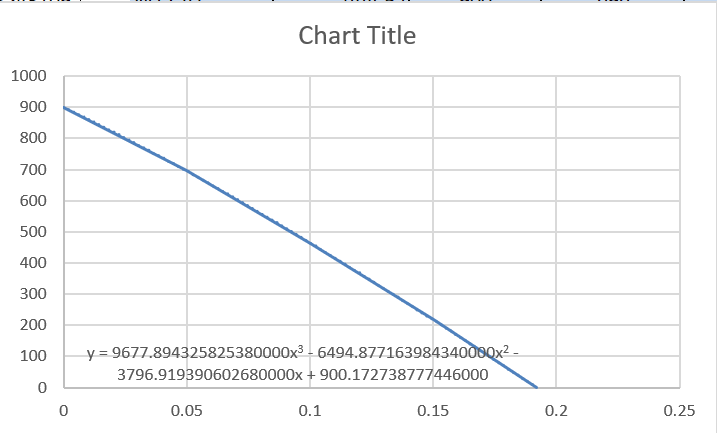


To store the required data, I used Microsoft SQL Server Management Studio 18. This software allowed me to create SQL procedures within the software. It also allowed me to relate my tables to each other.

Certain information, e.g. data values that were needed, were found from the following source:

Cory, B 2005, *Fans & Ventilation: A Practical Guide*, Elsevier (page 237)

I also used my client’s previous version for other data. I used Excel’s trendline function to find the equations relating key values in the calculations.



## 

## App.xaml

This is where the application’s fonts are defined.

<Application x:Class="Nuacoustic.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:Nuacoustic"

StartupUri="MainWindow.xaml">

<Application.Resources>

<FontFamily x:Key="MontserratRegular">./Fonts/Montserrat-Regular.ttf#Montserrat</FontFamily>

<FontFamily x:Key="MontserratItalic">./Fonts/NotoSans-Italic.tts#Noto Sans</FontFamily>

</Application.Resources>

</Application>

## App.xaml.cs

using System.Windows;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for App.xaml

/// </summary>

public partial class App : Application

{

}

}

The next three sections are where the application’s pages are defined, this is used to allow the application to make links between pages so the user can move between them.

## 

## ApplicationPage.cs

namespace Nuacoustic

{

// references all the pages

public enum ApplicationPage

{

MainPage = 0,

LoginScreen = 1,

Register = 2,

ForgotPassword = 3,

ResetPassword = 4,

HomePage = 5,

Settings = 6,

ExitPage = 7,

DeleteAccount = 8,

Project = 9,

CreateNewProject = 10,

HowTo = 11,

CurrentProjectSettings = 12,

}

}

## ApplicationPageValueConverter.cs

using System;

using System.Globalization;

using System.Diagnostics;

namespace Nuacoustic.ValueConverters

{

class ApplicationPageValueConverter : BaseValueConverter<ApplicationPageValueConverter>

{

public override object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

switch ((ApplicationPage)value)

{

case ApplicationPage.MainPage:

return new MainPage();

case ApplicationPage.LoginScreen:

return new LoginScreen();

case ApplicationPage.Register:

return new Register();

case ApplicationPage.ForgotPassword:

return new ForgotPassword();

case ApplicationPage.ResetPassword:

return new ResetPassword();

case ApplicationPage.HomePage:

return new HomePage();

case ApplicationPage.Settings:

return new Settings();

case ApplicationPage.ExitPage:

return new ExitPage();

case ApplicationPage.DeleteAccount:

return new DeleteAccount();

case ApplicationPage.Project:

return new Project();

case ApplicationPage.CreateNewProject:

return new CreateNewProject();

case ApplicationPage.HowTo:

return new HowTo();

case ApplicationPage.CurrentProjectSettings:

return new CurrentProjectSettings();

default:

Debugger.Break();

return null;

}

}

public override object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

## BaseValueConverter.cs

using System;

using System.Windows.Data;

using System.Windows.Markup;

using System.Globalization;

namespace Nuacoustic.ValueConverters

{

public abstract class BaseValueConverter<T> : MarkupExtension, IValueConverter

where T : class, new()

{

private static T mConverter = null;

public override object ProvideValue(IServiceProvider serviceProvider)

{

return mConverter ?? (mConverter = new T());

}

public abstract object Convert(object value, Type targetType, object parameter, CultureInfo culture);

public abstract object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture);

}

}

## 

## CreateNewProject.xaml

This page sets the display of the ‘Create New Project’ page. Here the user defines characteristics of the room by either typing in a value or selecting from drop down lists.

<Page x:Class="Nuacoustic.CreateNewProject"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="CreateNewProject">

<Grid Background="White">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<Button Name="btnSave" Content="SAVE" FontWeight="Bold" FontSize="18" Grid.Row="12" Grid.Column="5" Margin="5" Click="btnSave\_Click"/>

<Button Name="btnBack" Content="BACK" FontWeight="Bold" FontSize="18" Grid.Row="1" Grid.Column="0" Margin="5" Click="btnBack\_Click"/>

<TextBlock Text="Project Name" Grid.Row="1" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtProjectName" Grid.Row="1" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Type of Room" Grid.Row="2" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtTypeOfRoom" Grid.Column="2" Grid.Row="2" Background="#cccccc" FontSize="13" Margin="5">

<ComboBoxItem IsSelected="True">Concert Hall</ComboBoxItem>

<ComboBoxItem >Bedroom</ComboBoxItem>

<ComboBoxItem>Living room</ComboBoxItem>

<ComboBoxItem>Small office</ComboBoxItem>

<ComboBoxItem>Large open office</ComboBoxItem>

<ComboBoxItem>Canteens</ComboBoxItem>

</ComboBox>

<TextBlock Text="Fan" Grid.Row="3" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtFanCode" Grid.Column="2" Grid.Row="3" Background="#cccccc" FontSize="13" Margin="5">

<ComboBoxItem IsSelected="True">XBC15</ComboBoxItem>

<ComboBoxItem >XBC25</ComboBoxItem>

<ComboBoxItem>XBC45</ComboBoxItem>

<ComboBoxItem>XBC55</ComboBoxItem>

<ComboBoxItem>XBC65</ComboBoxItem>

</ComboBox>

<TextBlock Text="Acoustic Barrier" Grid.Row="4" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtBarrierType" Grid.Column="2" Grid.Row="4" Background="#cccccc" FontSize="13" Margin="3.333,5.333,7,5.333">

<ComboBoxItem IsSelected="True">Basic</ComboBoxItem>

<ComboBoxItem >Acoustic Grade</ComboBoxItem>

</ComboBox>

<TextBlock Text="Attenuator Supply" Grid.Row="5" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtAttenuatorSupply" Grid.Column="2" Grid.Row="5" Background="#cccccc" FontSize="13" Margin="5">

<ComboBoxItem IsSelected="True">None</ComboBoxItem>

<ComboBoxItem >Short</ComboBoxItem>

<ComboBoxItem >Medium</ComboBoxItem>

<ComboBoxItem>Long</ComboBoxItem>

</ComboBox>

<TextBlock Text="Attenuator Extract" Grid.Row="6" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtAttenuatorExtract" Grid.Column="2" Grid.Row="6" Background="#cccccc" FontSize="13" Margin="5">

<ComboBoxItem IsSelected="True">None</ComboBoxItem>

<ComboBoxItem >Short</ComboBoxItem>

<ComboBoxItem >Medium</ComboBoxItem>

<ComboBoxItem>Long</ComboBoxItem>

</ComboBox>

<TextBlock Text="Room Dimensions" Grid.Row="7" Grid.Column="0" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBlock Text="Length" Grid.Row="7" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtRoomLength" Grid.Row="7" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Width" Grid.Row="8" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtRoomWidth" Grid.Row="8" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Height" Grid.Row="9" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtRoomHeight" Grid.Row="9" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Number of people in room (put digits)" FontSize="13" Grid.Row="10" VerticalAlignment="Center" HorizontalAlignment="Center" Grid.Column="0" Grid.ColumnSpan="3" Margin="41,9.667,129,4.667" Width="230" Height="23"/>

<TextBox Name="txtNumberOfPeople" FontSize="13" Grid.Row="10" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Air Volume Flow Rate (m^3\*s^-1)" Grid.Row="11" Grid.Column="0" Grid.ColumnSpan="2" FontSize="13" HorizontalAlignment="Right" VerticalAlignment="center"/>

<TextBox Name="txtAirVolFlowRate" Grid.Row="11" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="System Resistance (Pa)" Grid.Row="2" Grid.Column="3" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtSysRes" Grid.Row="2" Grid.Column="4" Background="Lightgray" Margin="5"/>

<TextBlock Text="Room Terminal Side Length" Grid.Row="1" Grid.Column="3" FontSize="11" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtRoomTerminalSideLength" Grid.Column="4" Grid.Row="1" Background="#cccccc" FontSize="13" Margin="5"/>

<TextBlock Text="Email" Grid.Row="2" Grid.Column="0" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtEmail" Grid.Column="0" Grid.Row="3" Background="#cccccc" FontSize="13" Margin="5"/>

<TextBlock Text="Units for lengths" Background="Transparent" Grid.Row="3" Grid.Column="3" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtUnits" Grid.Column="4" Grid.Row="3" Background="Transparent" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center" Margin="4.667,7,4.667,7.333" Width="124">

<ComboBoxItem IsSelected="True">Metres</ComboBoxItem>

<ComboBoxItem>Feet</ComboBoxItem>

</ComboBox>

</Grid>

</Page>

## 

## CreateNewProject.xaml.cs

This section saves the user’s choices to the database after they have clicked ‘Save’. This is done with the SQL statement INSERT INTO… VALUES.

using System;

using System.Windows;

using System.Windows.Controls;

using System.Data;

using System.Data.SqlClient;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for CreateNewProject.xaml

/// </summary>

public partial class CreateNewProject : Page

{

public CreateNewProject()

{

InitializeComponent();

txtEmail.Text = (Application.Current.MainWindow as MainWindow).email;

}

// Collecting values from this page to send forwards so users do not need to keep entering the same details.

private void DisplayEmail()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.email = txtEmail.Text;

}

private void DisplayProjectName()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.projectname = txtProjectName.Text;

}

private void btnSave\_Click(object sender, RoutedEventArgs e)

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

String query = "SELECT COUNT(1) FROM tblUser WHERE Email=@Email";

SqlCommand sqlCmd = new SqlCommand(query, sqlCon);

sqlCmd.CommandType = CommandType.Text;

sqlCmd.Parameters.AddWithValue("@Email", txtEmail.Text);

int count = Convert.ToInt32(sqlCmd.ExecuteScalar());

if (count != 1)

{

MessageBox.Show("Email is not found. Please try again.");

}

// Inserting details of the project into the database.

using (sqlCon)

{

string sql = "insert into dbo.tblProject ([ProjectName], [LastEditDate], [SystemRes], [ReqAirVolRate], " +

"[RoomLength], [RoomWidth], [RoomHeight], [FanCode], [TypeOfRoom], [NumberOfPeople], " +

"[SupplyAttenuatorType], [ExtractAttenuatorType], [AcousticBarrierType], [RoomTerminalSideLength], [Units]," +

" [XCoordListener], [YCoordListener], [XCoordFan], [YCoordFan], [Email], [XCoordSupply], [YCoordSupply]," +

"[XCoordExtract], [YCoordExtract]) values(@ProjectName,@LastEditDate,@SystemRes,@ReqAirVolRate," +

"@RoomLength,@RoomWidth,@RoomHeight,@FanCode,@TypeOfRoom,@NumberOfPeople,@SupplyAttenuatorType," +

"@ExtractAttenuatorType,@AcousticBarrierType,@RoomTerminalSideLength,@Units,@XCoordListener,@YCoordListener," +

"@XCoordFan,@YCoordFan,@Email,@XCoordSupply,@YCoordSupply,@XCoordExtract,@YCoordExtract)";

// Create the connection

try

{

// Open the connection to the database.

// This is the first critical step in the process.

// If we cannot reach the db then we have connectivity problems

// Prepare the command to be executed on the db

using (SqlCommand cmd = new SqlCommand(sql, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@LastEditDate", SqlDbType.NVarChar).Value = DateTime.Now.ToString("yyyy-dd-mm");

cmd.Parameters.Add("@SystemRes", SqlDbType.NVarChar).Value = txtSysRes.Text.Trim();

cmd.Parameters.Add("@ReqAirVolRate", SqlDbType.NVarChar).Value = txtAirVolFlowRate.Text.Trim();

cmd.Parameters.Add("@RoomLength", SqlDbType.NVarChar).Value = txtRoomLength.Text.Trim();

cmd.Parameters.Add("@RoomWidth", SqlDbType.NVarChar).Value = txtRoomWidth.Text.Trim();

cmd.Parameters.Add("@RoomHeight", SqlDbType.NVarChar).Value = txtRoomHeight.Text.Trim();

cmd.Parameters.Add("@FanCode", SqlDbType.NVarChar).Value = txtFanCode.Text.Trim();

cmd.Parameters.Add("@TypeOfRoom", SqlDbType.NVarChar).Value = txtTypeOfRoom.Text.Trim();

cmd.Parameters.Add("@NumberOfPeople", SqlDbType.NVarChar).Value = txtNumberOfPeople.Text.Trim();

cmd.Parameters.Add("@SupplyAttenuatorType", SqlDbType.NVarChar).Value = txtAttenuatorSupply.Text.Trim();

cmd.Parameters.Add("@ExtractAttenuatorType", SqlDbType.NVarChar).Value = txtAttenuatorExtract.Text.Trim();

cmd.Parameters.Add("@AcousticBarrierType", SqlDbType.NVarChar).Value = txtBarrierType.Text.Trim();

cmd.Parameters.Add("@RoomTerminalSideLength", SqlDbType.NVarChar).Value = txtRoomTerminalSideLength.Text.Trim();

cmd.Parameters.Add("@Units", SqlDbType.NVarChar).Value = txtUnits.Text.Trim();

cmd.Parameters.Add("@XCoordListener", SqlDbType.NVarChar).Value = "0";

cmd.Parameters.Add("@YCoordListener", SqlDbType.NVarChar).Value = "0";

cmd.Parameters.Add("@XCoordFan", SqlDbType.NVarChar).Value = "0";

cmd.Parameters.Add("@YCoordFan", SqlDbType.NVarChar).Value = "0";

cmd.Parameters.Add("@XCoordSupply", SqlDbType.NVarChar).Value = "0";

cmd.Parameters.Add("@YCoordSupply", SqlDbType.NVarChar).Value = "0";

cmd.Parameters.Add("@XCoordExtract", SqlDbType.NVarChar).Value = "0";

cmd.Parameters.Add("@YCoordExtract", SqlDbType.NVarChar).Value = "0";

cmd.Parameters.Add("@Email", SqlDbType.NVarChar).Value = txtEmail.Text.Trim();

cmd.ExecuteNonQuery();

}

}

catch (Exception ex)

{

// We should log the error somewhere,

// for this example let's just show a message

MessageBox.Show("ERROR:" + ex.Message);

}

}

MessageBox.Show("The information has been saved.");

DisplayProjectName();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.Project)

{

mainWindow.CurrentPage = ApplicationPage.Project;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private void btnBack\_Click(object sender, RoutedEventArgs e)

{

DisplayEmail();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.HomePage)

{

mainWindow.CurrentPage = ApplicationPage.HomePage;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

}

}

## CurrentProjectSettings.xaml

This is the display of the ‘Current Project Settings’ page. This is quite similar to the ‘Create New Project’ page, the main difference between that the buttons update the project instead of creating a new record.

<Page x:Class="Nuacoustic.CurrentProjectSettings"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="CurrentProjectSettings">

<Grid Background="White">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<Button Name="btnBack" Content="BACK" FontWeight="Bold" FontSize="15" Grid.Row="1" Grid.Column="0" Margin="5" Click="btnBack\_Click"/>

<Button Name="btnUpdate" Content="UPDATE" FontWeight="Bold" FontSize="15" Grid.Row="12" Grid.Column="5" Margin="5" Click="btnUpdate\_Click"/>

<Button Name="btnProjectPage" Content="PROJECT PAGE" FontWeight="Bold" FontSize="15" Grid.Row="12" Grid.Column="3" Margin="5" Click="btnProjectPage\_Click"/>

<TextBlock Text="Project Name" Grid.Row="1" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtProjectName" Grid.Row="1" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Type of Room" Grid.Row="2" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtTypeOfRoom" Grid.Column="2" Grid.Row="2" Background="#cccccc" FontSize="13" Margin="5">

<ComboBoxItem IsSelected="True">Concert Hall</ComboBoxItem>

<ComboBoxItem >Bedroom</ComboBoxItem>

<ComboBoxItem>Living room</ComboBoxItem>

<ComboBoxItem>Small office</ComboBoxItem>

<ComboBoxItem>Large open office</ComboBoxItem>

<ComboBoxItem>Canteens</ComboBoxItem>

</ComboBox>

<TextBlock Text="Fan" Grid.Row="3" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtFanCode" Grid.Column="2" Grid.Row="3" Background="#cccccc" FontSize="13" Margin="5">

<ComboBoxItem IsSelected="True">XBC15</ComboBoxItem>

<ComboBoxItem >XBC25</ComboBoxItem>

<ComboBoxItem>XBC45</ComboBoxItem>

<ComboBoxItem>XBC55</ComboBoxItem>

<ComboBoxItem>XBC65</ComboBoxItem>

</ComboBox>

<TextBlock Text="Acoustic Barrier" Grid.Row="4" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtBarrierType" Grid.Column="2" Grid.Row="4" Background="#cccccc" FontSize="13" Margin="3.333,5.333,7,5.333">

<ComboBoxItem IsSelected="True">Basic</ComboBoxItem>

<ComboBoxItem >Acoustic Grade</ComboBoxItem>

</ComboBox>

<TextBlock Text="Attenuator Supply" Grid.Row="5" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtAttenuatorSupply" Grid.Column="2" Grid.Row="5" Background="#cccccc" FontSize="13" Margin="5">

<ComboBoxItem IsSelected="True">None</ComboBoxItem>

<ComboBoxItem >Short</ComboBoxItem>

<ComboBoxItem >Medium</ComboBoxItem>

<ComboBoxItem>Long</ComboBoxItem>

</ComboBox>

<TextBlock Text="Attenuator Extract" Grid.Row="6" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtAttenuatorExtract" Grid.Column="2" Grid.Row="6" Background="#cccccc" FontSize="13" Margin="5">

<ComboBoxItem IsSelected="True">None</ComboBoxItem>

<ComboBoxItem >Short</ComboBoxItem>

<ComboBoxItem >Medium</ComboBoxItem>

<ComboBoxItem>Long</ComboBoxItem>

</ComboBox>

<TextBlock Text="Room Dimensions" Grid.Row="7" Grid.Column="0" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBlock Text="Length" Grid.Row="7" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtRoomLength" Grid.Row="7" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Width" Grid.Row="8" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtRoomWidth" Grid.Row="8" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Height" Grid.Row="9" Grid.Column="1" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtRoomHeight" Grid.Row="9" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Number of people in room (put digits)" FontSize="13" Grid.Row="10" VerticalAlignment="Center" HorizontalAlignment="Center" Grid.Column="0" Grid.ColumnSpan="3" Margin="41,9.667,129,4.667" Width="230" Height="23"/>

<TextBox Name="txtNumberOfPeople" FontSize="13" Grid.Row="10" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="Air Volume Flow Rate (m^3\*s^-1)" Grid.Row="11" Grid.Column="0" Grid.ColumnSpan="2" FontSize="13" HorizontalAlignment="Right" VerticalAlignment="center"/>

<TextBox Name="txtAirVolFlowRate" Grid.Row="11" Grid.Column="2" Background="Lightgray" Margin="5"/>

<TextBlock Text="System Resistance (Pa)" Grid.Row="2" Grid.Column="3" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtSysRes" Grid.Row="2" Grid.Column="4" Background="Lightgray" Margin="5"/>

<TextBlock Text="Room Terminal Side Length" Grid.Row="1" Grid.Column="3" FontSize="11" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtRoomTerminalSideLength" Grid.Column="4" Grid.Row="1" Background="#cccccc" FontSize="13" Margin="5"/>

<TextBlock Text="Email" Grid.Row="2" Grid.Column="0" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtEmail" Grid.Column="0" Grid.Row="3" Background="#cccccc" FontSize="13" Margin="5"/>

<TextBlock Text="Units for lengths" Background="Transparent" Grid.Row="3" Grid.Column="3" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<ComboBox Name="txtUnits" Grid.Column="4" Grid.Row="3" Background="Transparent" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center" Margin="4.667,5,4.667,4.667" Width="124" Height="25">

<ComboBoxItem IsSelected="True">Metres</ComboBoxItem>

<ComboBoxItem>Feet</ComboBoxItem>

</ComboBox>

<TextBlock Text="Last Edited" Grid.Row="4" Grid.Column="3" FontSize="13" HorizontalAlignment="center" VerticalAlignment="center"/>

<TextBox Name="txtLastEdit" Grid.Column="4" Grid.Row="4" Background="#cccccc" FontSize="13" Margin="5"/>

</Grid>

</Page>

## CurrentProjectSettings.xaml.cs

This updates the user’s project in the database after they click ‘Update’. When this page begins to load, the existing values in the database are fetched with ‘SELECT… FROM… WHERE’, and the values are displayed on the page. When the user is ready to update, this is done with the SQL statement ‘UPDATE… SET’.

using System;

using System.Windows;

using System.Windows.Controls;

using System.Data;

using System.Data.SqlClient;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for CurrentProjectSettings.xaml

/// </summary>

public partial class CurrentProjectSettings : Page

{

public CurrentProjectSettings()

{

InitializeComponent();

txtEmail.Text = (Application.Current.MainWindow as MainWindow).email;

txtProjectName.Text = (Application.Current.MainWindow as MainWindow).projectname;

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

String queryLastDate = "SELECT LastEditDate FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND tblProject.Email=@Email";

SqlCommand sqlCmdLastDate = new SqlCommand(queryLastDate, sqlCon);

sqlCmdLastDate.CommandType = CommandType.Text;

sqlCmdLastDate.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmdLastDate.Parameters.AddWithValue("@Email", txtEmail.Text);

txtLastEdit.Text = Convert.ToString(sqlCmdLastDate.ExecuteScalar());

String querySysRes = "SELECT SystemRes FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND tblProject.Email=@Email";

SqlCommand sqlCmdSysRes = new SqlCommand(querySysRes, sqlCon);

sqlCmdSysRes.CommandType = CommandType.Text;

sqlCmdSysRes.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmdSysRes.Parameters.AddWithValue("@Email", txtEmail.Text);

txtSysRes.Text = Convert.ToString(sqlCmdSysRes.ExecuteScalar());

String queryReqAirVolRate = "SELECT ReqAirVolRate FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmdReqAirVolRate = new SqlCommand(queryReqAirVolRate, sqlCon);

sqlCmdReqAirVolRate.CommandType = CommandType.Text;

sqlCmdReqAirVolRate.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmdReqAirVolRate.Parameters.AddWithValue("@Email", txtEmail.Text);

txtAirVolFlowRate.Text = Convert.ToString(sqlCmdReqAirVolRate.ExecuteScalar());

String queryRoomLength = "SELECT RoomLength FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmdRoomLength = new SqlCommand(queryRoomLength, sqlCon);

sqlCmdRoomLength.CommandType = CommandType.Text;

sqlCmdRoomLength.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmdRoomLength.Parameters.AddWithValue("@Email", txtEmail.Text);

txtRoomLength.Text = Convert.ToString(sqlCmdRoomLength.ExecuteScalar());

String queryRoomWidth = "SELECT RoomWidth FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmdRoomWidth = new SqlCommand(queryRoomWidth, sqlCon);

sqlCmdRoomWidth.CommandType = CommandType.Text;

sqlCmdRoomWidth.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmdRoomWidth.Parameters.AddWithValue("@Email", txtEmail.Text);

txtRoomWidth.Text = Convert.ToString(sqlCmdRoomWidth.ExecuteScalar());

String queryRoomVolume = "SELECT RoomHeight FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmdRoomVolume = new SqlCommand(queryRoomVolume, sqlCon);

sqlCmdRoomVolume.CommandType = CommandType.Text;

sqlCmdRoomVolume.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmdRoomVolume.Parameters.AddWithValue("@Email", txtEmail.Text);

txtRoomHeight.Text = Convert.ToString(sqlCmdRoomVolume.ExecuteScalar());

String queryFanCode = "SELECT FanCode FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmdFanCode = new SqlCommand(queryFanCode, sqlCon);

sqlCmdFanCode.CommandType = CommandType.Text;

sqlCmdFanCode.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmdFanCode.Parameters.AddWithValue("@Email", txtEmail.Text);

txtFanCode.Text = Convert.ToString(sqlCmdFanCode.ExecuteScalar());

String queryTypeOfRoom = "SELECT TypeOfRoom FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmdTypeOfRoom = new SqlCommand(queryTypeOfRoom, sqlCon);

sqlCmdTypeOfRoom.CommandType = CommandType.Text;

sqlCmdTypeOfRoom.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmdTypeOfRoom.Parameters.AddWithValue("@Email", txtEmail.Text);

txtTypeOfRoom.Text = Convert.ToString(sqlCmdTypeOfRoom.ExecuteScalar());

String queryPeople = "SELECT NumberOfPeople FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmdPeople = new SqlCommand(queryPeople, sqlCon);

sqlCmdPeople.CommandType = CommandType.Text;

sqlCmdPeople.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmdPeople.Parameters.AddWithValue("@Email", txtEmail.Text);

txtNumberOfPeople.Text = Convert.ToString(sqlCmdPeople.ExecuteScalar());

String query1 = "SELECT SupplyAttenuatorType FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmd1 = new SqlCommand(query1, sqlCon);

sqlCmd1.CommandType = CommandType.Text;

sqlCmd1.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmd1.Parameters.AddWithValue("@Email", txtEmail.Text);

txtAttenuatorSupply.Text = Convert.ToString(sqlCmd1.ExecuteScalar());

String query2 = "SELECT ExtractAttenuatorType FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmd2 = new SqlCommand(query2, sqlCon);

sqlCmd2.CommandType = CommandType.Text;

sqlCmd2.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmd2.Parameters.AddWithValue("@Email", txtEmail.Text);

txtAttenuatorExtract.Text = Convert.ToString(sqlCmd2.ExecuteScalar());

String query3 = "SELECT AcousticBarrierType FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmd3 = new SqlCommand(query3, sqlCon);

sqlCmd3.CommandType = CommandType.Text;

sqlCmd3.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmd3.Parameters.AddWithValue("@Email", txtEmail.Text);

txtBarrierType.Text = Convert.ToString(sqlCmd3.ExecuteScalar());

String query4 = "SELECT RoomTerminalSideLength FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmd4 = new SqlCommand(query4, sqlCon);

sqlCmd4.CommandType = CommandType.Text;

sqlCmd4.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmd4.Parameters.AddWithValue("@Email", txtEmail.Text);

txtRoomTerminalSideLength.Text = Convert.ToString(sqlCmd4.ExecuteScalar());

String query5 = "SELECT Units FROM tblProject WHERE tblProject.ProjectName=@ProjectName AND [tblProject.Email=@Email](mailto:tblProject.Email=@Email)";

SqlCommand sqlCmd5 = new SqlCommand(query5, sqlCon);

sqlCmd5.CommandType = CommandType.Text;

sqlCmd5.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

sqlCmd5.Parameters.AddWithValue("@Email", txtEmail.Text);

txtUnits.Text = Convert.ToString(sqlCmd5.ExecuteScalar());

}

// Collecting values from this page to send forwards so users do not need to keep entering the same details.

private void DisplayEmail()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.email = txtEmail.Text;

}

private void DisplayProjectName()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.projectname = txtProjectName.Text;

}

private void btnBack\_Click(object sender, RoutedEventArgs e)

{

DisplayEmail();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.HomePage)

{

mainWindow.CurrentPage = ApplicationPage.HomePage;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private void btnUpdate\_Click(object sender, RoutedEventArgs e)

{

// Saving new values for a project in the database.

if (txtProjectName.Text == "" || txtEmail.Text == "")

{

MessageBox.Show("Please enter a valid project name and email.");

}

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

string sql = "UPDATE tblProject SET SystemRes=@SystemRes, LastEditDate=@LastEditDate, ReqAirVolRate=@ReqAirVolRate," +

"RoomLength=@RoomLength, RoomWidth=@RoomWidth, RoomHeight=@RoomHeight, TypeOfRoom=@TypeOfRoom, FanCode=@FanCode," +

"NumberOfPeople=@NumberOfPeople, SupplyAttenuatorType=@SupplyAttenuatorType, ExtractAttenuatorType=@ExtractAttenuatorType," +

"AcousticBarrierType=@AcousticBarrierType, RoomTerminalSideLength=@RoomTerminalSideLength, Units=@Units WHERE ProjectName = @ProjectName AND Email = @Email";

using (SqlCommand cmd = new SqlCommand(sql, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@Email", SqlDbType.NVarChar).Value = txtEmail.Text.Trim();

cmd.Parameters.Add("@SystemRes", SqlDbType.NVarChar).Value = txtSysRes.Text.Trim();

cmd.Parameters.Add("@LastEditDate", SqlDbType.NVarChar).Value = DateTime.Now;

cmd.Parameters.Add("@ReqAirVolRate", SqlDbType.NVarChar).Value = txtAirVolFlowRate.Text.Trim();

cmd.Parameters.Add("@RoomLength", SqlDbType.NVarChar).Value = txtRoomLength.Text.Trim();

cmd.Parameters.Add("@RoomWidth", SqlDbType.NVarChar).Value = txtRoomWidth.Text.Trim();

cmd.Parameters.Add("@RoomHeight", SqlDbType.NVarChar).Value = txtRoomHeight.Text.Trim();

cmd.Parameters.Add("@TypeOfRoom", SqlDbType.NVarChar).Value = txtTypeOfRoom.Text.Trim();

cmd.Parameters.Add("@FanCode", SqlDbType.NVarChar).Value = txtFanCode.Text.Trim();

cmd.Parameters.Add("@NumberOfPeople", SqlDbType.NVarChar).Value = txtNumberOfPeople.Text.Trim();

cmd.Parameters.Add("@SupplyAttenuatorType", SqlDbType.NVarChar).Value = txtAttenuatorSupply.Text.Trim();

cmd.Parameters.Add("@ExtractAttenuatorType", SqlDbType.NVarChar).Value = txtAttenuatorExtract.Text.Trim();

cmd.Parameters.Add("@AcousticBarrierType", SqlDbType.NVarChar).Value = txtBarrierType.Text.Trim();

cmd.Parameters.Add("@RoomTerminalSideLength", SqlDbType.NVarChar).Value = txtRoomTerminalSideLength.Text.Trim();

cmd.Parameters.Add("@Units", SqlDbType.NVarChar).Value = txtUnits.Text.Trim();

cmd.ExecuteNonQuery();

}

MessageBox.Show("Your project information has been updated.");

}

private void btnProjectPage\_Click(object sender, RoutedEventArgs e)

{

DisplayProjectName();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.Project)

{

mainWindow.CurrentPage = ApplicationPage.Project;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

}

}

## DeleteAccount.xaml

This page is where users delete their data and their account. The user enters their email and then press ‘Delete’.

<Page x:Class="Nuacoustic.DeleteAccount"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="DeleteAccount">

<Grid Background="White">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<TextBlock Text="DELETE ACCOUNT" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="30" FontWeight="Bold" Margin="10" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3"/>

<TextBlock Text="All your data, including your projects, will be deleted when you delete your account." Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="2" Margin="36,9.667,10,10" Grid.ColumnSpan="3"/>

<TextBlock Text="Email" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="3" Margin=" 210 10 160 10" Grid.ColumnSpan="2"/>

<TextBox x:Name="txtEmail" Grid.Column="1" Grid.Row="3" Background ="#cccccc" Margin="150 10 150 10" Grid.ColumnSpan="2" FontSize="28"/>

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="5" BorderThickness="1" CornerRadius="25" Margin="0"/>

<Button Name="btnDelete" Grid.Column="1" Grid.Row="5" Content="DELETE" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="21" Margin="8" Click="btnDelete\_Click"/>

</Grid>

</Page>

## DeleteAccount.xaml.cs

After the user presses ‘Delete’, their account and the projects they have made are deleted.

using System;

using System.Windows;

using System.Windows.Controls;

using System.Data;

using System.Data.SqlClient;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for DeleteAccount.xaml

/// </summary>

public partial class DeleteAccount : Page

{

public DeleteAccount()

{

InitializeComponent();

}

// Deletes both user account and the projects they created

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

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

try

{

string sql = "DELETE FROM tblUser WHERE Email=@Email";

using (SqlCommand cmd = new SqlCommand(sql, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@Email", SqlDbType.NVarChar).Value = txtEmail.Text.Trim();

cmd.ExecuteNonQuery();

}

string sql1 = "DELETE FROM tblProject WHERE Email=@Email";

using (SqlCommand cmd = new SqlCommand(sql1, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@Email", SqlDbType.NVarChar).Value = txtEmail.Text.Trim();

cmd.ExecuteNonQuery();

}

MessageBox.Show("Your information has been deleted.");

}

catch(Exception ex)

{

MessageBox.Show(Convert.ToString(ex));

}

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.ExitPage)

{

mainWindow.CurrentPage = ApplicationPage.ExitPage;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

MessageBox.Show("Your account has been deleted.");

sqlCon.Close();

}

}

}

These next two sections are a display page from which the user would shut the application down.

## ExitPage.xaml

<Page x:Class="Nuacoustic.ExitPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ExitPage">

<StackPanel Background="White">

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0" Height="52" Margin="0,0,656,0"/>

<Button x:Name="btnHome" Content="HOME" FontWeight="Bold" FontSize="22" Margin="637,0,36,0" Height="42" Click="btnHome\_Click" />

<TextBlock Text="THANK YOU" FontSize="26" FontWeight="Bold" HorizontalAlignment="Center" VerticalAlignment="Center"/>

<Separator></Separator>

<TextBlock Text="Thank you for using Nuacoustic, you have been logged out." FontSize="26" FontWeight="Bold" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="15"/>

<TextBlock Text="Please press the exit button to leave the application." FontSize="26" FontWeight="Bold" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="15"/>

</StackPanel>

</Page>

## ExitPage.xaml.cs

using System.Windows;

using System.Windows.Controls;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for ExitPage.xaml

/// </summary>

public partial class ExitPage : Page

{

public ExitPage()

{

InitializeComponent();

}

private void btnHome\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.HomePage)

{

mainWindow.CurrentPage = ApplicationPage.HomePage;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

}

}

## ForgotPassword.xaml

<Page x:Class="Nuacoustic.ForgotPassword"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ForgotPassword">

<Grid Background="white">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<TextBlock Text="FORGOT PASSWORD" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="30" FontWeight="Bold" Margin="10" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3"/>

<!--email section-->

<TextBlock Text="Email" Foreground="black" FontSize="30" Grid.Column="0" Grid.Row="2" Margin=" 210 10 160 10" Grid.ColumnSpan="2"/>

<TextBox x:Name="txtEmail" Grid.Column="1" Grid.Row="2" Background ="#cccccc" Margin="150 10 150 10" Grid.ColumnSpan="2" FontSize="28" />

<!--verification code section-->

<TextBlock Text="Verification Code" Foreground="black" FontSize="30" Grid.Column="0" Grid.Row="4" Margin="141,10.333,160.333,10.333" Grid.ColumnSpan="2"/>

<TextBox x:Name="txtVerificationCode" Grid.Column="1" Grid.Row="4" Background ="#cccccc" Margin="150 10 150 10" Grid.ColumnSpan="2" FontSize="28"/>

<!--buttons-->

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="3" BorderThickness="1" CornerRadius="25" Margin="10"/>

<Button Name="btnSendCode" Grid.Column="1" Grid.Row="3" Content="SEND CODE" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19" Click="btnSendCode\_Click"/>

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="2" Grid.Row="0" BorderThickness="1" CornerRadius="25" Margin="10"/>

<Button Name="btnClickToRegister" Grid.Column="2" Grid.Row="0" Content="First time? Register here" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="15" Margin="18.5"/>

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="5" BorderThickness="1" CornerRadius="25" Margin="10"/>

<Button Name="btnVerify" Grid.Column="1" Grid.Row="5" Content="VERIFY" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="21" Margin="18.5" Click="btnVerify\_Click"/>

<Button Name="btnBack" Margin="15" Background="white" FontSize="26" FontWeight="Bold" Grid.Column="0" Grid.Row="5" Content="BACK" Click="btnBack\_Click"/>

</Grid>

</Page>

## ForgotPassword.xaml.cs

If a user forgets their password, they enter their email, and this is searched for in a ‘Logins’ hashtable. If it is found, then they are sent a verification code allowing them to update their password.

using System;

using System.Windows;

using System.Windows.Controls;

using System.Net;

using System.Net.Mail;

using System.Data;

using System.Data.SqlClient;

using System.Collections;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for ForgotPassword.xaml

/// </summary>

public partial class ForgotPassword : Page

{

string randomCode;

public static string to;

public Hashtable Logins;

public ForgotPassword()

{

InitializeComponent();

txtEmail.Text = (Application.Current.MainWindow as MainWindow).email;

Logins = (Application.Current.MainWindow as MainWindow).Logins;

}

private void ShareHashtable()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.Logins = Logins;

}

// Sends code to users email in order to verify it is them.

private void btnSendCode\_Click(object sender, RoutedEventArgs e)

{

string from, pass, messageBody;

Random rand = new Random();

randomCode = (rand.Next(999999)).ToString();

MailMessage message = new MailMessage();

to = (txtEmail.Text).ToString();

from = "nuacoustic.nuaire@gmail.com";

pass = "!testPass123!";

messageBody = "Your reset code is " + randomCode + ".";

message.To.Add(to);

message.From = new MailAddress(from);

message.Body = messageBody;

message.Subject = "Password reset";

SmtpClient smtp = new SmtpClient("smtp.gmail.com");

smtp.EnableSsl = true;

smtp.Port = 587;

smtp.DeliveryMethod = SmtpDeliveryMethod.Network;

smtp.Credentials = new NetworkCredential(from, pass);

// checking if the email the user enters is registered in the database

int total = GetHashtableTotal();

int hashlocation = total % 1000;

if (Logins.ContainsKey(hashlocation) == true)

{

try

{

smtp.Send(message);

MessageBox.Show("Code sent successfully.");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

{

MessageBox.Show("Please enter an already registered email.");

}

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

String query = "select count(Email) as Email from tblUser where Email = '" + txtEmail.Text + "'";

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

SqlCommand cmd = new SqlCommand(query, sqlCon);

cmd.CommandType = CommandType.Text;

SqlDataReader dr;

dr = cmd.ExecuteReader();

while (dr.Read())

{

string iemail= dr["Email"].ToString();

if (iemail != "0")

{

try

{

smtp.Send(message);

MessageBox.Show("Code sent successfully.");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

{

MessageBox.Show("Please enter an already registered email.");

}

}

}

private void btnVerify\_Click(object sender, RoutedEventArgs e)

{

if(randomCode == (txtVerificationCode.Text).ToString())

{

to = txtEmail.Text;

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.LoginScreen)

{

mainWindow.CurrentPage = ApplicationPage.LoginScreen;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

else

{

MessageBox.Show("Wrong code");

}

}

private void btnBack\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.MainFrame.NavigationService.GoBack();

}

int GetHashtableTotal()

{

string email = Convert.ToString(txtEmail.Text);

int countemail = 0;

int arrayLength = email.Length;

int[] emailstr = new int[arrayLength];

int total = 0;

foreach (char c in email)

{

emailstr[countemail] = System.Convert.ToInt32(c);

total = emailstr[countemail] + total;

countemail++;

}

return total;

}

}

}

## HomePage.xaml

<Page x:Class="Nuacoustic.HomePage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="HomePage">

<Grid Background="white">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<!-- top for all 'logged in' pages -->

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<Button Name="btnHome" Grid.Column="1" Grid.Row="0" Content="HOME" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19" Click="btnHome\_Click"/>

<Button Name="btnHowTo" Grid.Column="2" Grid.Row="0" Content="HOW TO" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19" Click="btnHowTo\_Click"/>

<Button Name="btnSettings" Grid.Column="3" Grid.Row="0" Content="SETTINGS" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19" Click="btnSettings\_Click"/>

<Button Name="btnLogOut" Grid.Column="4" Grid.Row="0" Content="LOG OUT" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19" Click="btnLogOut\_Click"/>

<TextBlock Text="HOME" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="30" FontWeight="Bold" Margin="35,0,34,72.667" Grid.Row="1" RenderTransformOrigin="-1.29,0.441"/>

<TextBlock Text="Project Name" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="22" FontWeight="Bold" Margin="10,29.333,9,53.333" Grid.Column="0" Grid.Row="2"/>

<TextBox Name="txtProjectName" VerticalAlignment="Center" Grid.Row="2" Background="#cccccc" FontSize="22" Margin="10,58.333,85,10.333" Grid.ColumnSpan="2" RenderTransformOrigin="0.553,1.435" Height="44"/>

<TextBlock Text="Email" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="22" FontWeight="Bold" Margin="11,54,92,28.667" Grid.Column="0" Grid.Row="1"/>

<TextBox Name="txtEmail" VerticalAlignment="Center" Grid.Row="1" Background="#cccccc" FontSize="22" Margin="10,88,85,88.333" Grid.ColumnSpan="2" RenderTransformOrigin="0.553,1.435" Height="49" Grid.RowSpan="2"/>

<TextBlock Text="Project Details" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="22" FontWeight="Bold" Margin="80,45.667,75,23" Grid.Column="1" Grid.Row="2" Grid.ColumnSpan="2" Height="44" Width="165"/>

<TextBox Name="txtDetails" Grid.Row="2" Grid.Column="2" Background="#cccccc" FontSize="11" Margin="90,10.333,19,15.333" Grid.ColumnSpan="3"/>

<TextBlock Text="Project Names" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="22" FontWeight="Bold" Margin="80,45.667,75,23" Grid.Column="1" Grid.Row="1" Grid.ColumnSpan="2" Height="44" Width="165"/>

<TextBox Name="txtProjectFileNames" Grid.Row="1" Grid.Column="2" Background="#cccccc" FontSize="13" Margin="90,10.333,19,15.333" Grid.ColumnSpan="3"/>

<Button Name="btnCreateNewProject" Grid.Column="2" Grid.Row="3" Content="CREATE NEW" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="90,18.667,62,19" Click="btnCreateNewProject\_Click" Grid.ColumnSpan="2"/>

<Button Name="btnOpenProject" Grid.Column="3" Grid.Row="3" Content="OPEN PROJECT" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="133,19,19,19" Click="btnOpenProject\_Click" Grid.ColumnSpan="2"/>

<Button Name="btnOpenDetails" Grid.Row="3" Content="OPEN DETAILS" FontWeight="Bold" Background="Transparent" Foreground="black" FontSize="23" Margin="49,18.667,98,19" Grid.Column="1" Grid.ColumnSpan="2" Click="btnOpenDetails\_Click"/>

<Button Name="btnSeeProjects" Grid.Row="3" Content="SEE PROJECTS" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="10,18.667,134,60" Grid.ColumnSpan="2" Click="btnSeeProjects\_Click"/>

<Button Name="btnDeleteProject" Grid.Row="3" Content="DELETE PROJECT" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="21" Margin="10,57.667,134,10" Grid.ColumnSpan="2" Click="btnDeleteProject\_Click"/>

</Grid>

</Page>

## HomePage.xaml.cs

This page shows the user their created projects and details about a specific project. The details are put into the textboxes by using text-files and StreamWriter.

using System;

using System.Windows;

using System.Windows.Controls;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Collections;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for HomePage.xaml

/// </summary>

public partial class HomePage : Page

{

public Hashtable Logins;

public HomePage()

{

InitializeComponent();

txtEmail.Text = (Application.Current.MainWindow as MainWindow).email;

Logins = (Application.Current.MainWindow as MainWindow).Logins;

}

private void ShareHashtable()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.Logins = Logins;

}

private void DisplayEmail()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.email = txtEmail.Text;

}

private void DisplayProjectName()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.projectname = txtProjectName.Text;

}

private void btnSettings\_Click(object sender, RoutedEventArgs e)

{

DisplayEmail();

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.Settings)

{

mainWindow.CurrentPage = ApplicationPage.Settings;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private void btnHome\_Click(object sender, RoutedEventArgs e)

{

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.HomePage)

{

mainWindow.CurrentPage = ApplicationPage.HomePage;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private void btnHowTo\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.HowTo)

{

mainWindow.CurrentPage = ApplicationPage.HowTo;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private void btnLogOut\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.LoginScreen)

{

mainWindow.CurrentPage = ApplicationPage.LoginScreen;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private void btnCreateNewProject\_Click(object sender, RoutedEventArgs e)

{

DisplayEmail();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.CreateNewProject)

{

mainWindow.CurrentPage = ApplicationPage.CreateNewProject;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private void btnOpenProject\_Click(object sender, RoutedEventArgs e)

{

DisplayEmail();

DisplayProjectName();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.CurrentProjectSettings)

{

mainWindow.CurrentPage = ApplicationPage.CurrentProjectSettings;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

// Shows details of a given project.

private void btnOpenDetails\_Click(object sender, RoutedEventArgs e)

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (txtEmail.Text == "")

{

MessageBox.Show("Please enter your email.");

}

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

String queryadmin = "SELECT Admin FROM tblUser WHERE Email=@Email";

SqlCommand sqlCmd4 = new SqlCommand(queryadmin, sqlCon);

sqlCmd4.CommandType = CommandType.Text;

sqlCmd4.Parameters.AddWithValue("@Email", txtEmail.Text);

string admin = Convert.ToString(sqlCmd4.ExecuteScalar());

if (admin == "Yes")

{

try

{

String query = "SELECT COUNT(1) FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd = new SqlCommand(query, sqlCon);

sqlCmd.CommandType = CommandType.Text;

sqlCmd.Parameters.AddWithValue("@ProjectName", txtProjectName.Text.Trim());

int count = Convert.ToInt32(sqlCmd.ExecuteScalar());

if (count == 1)

{

ShowDetails();

}

else

{

MessageBox.Show("Please enter the name of an already created project. To create a new project, click 'Create New'.");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

sqlCon.Close();

}

}

else

{

try

{

String query = "SELECT COUNT(1) FROM tblProject WHERE ProjectName=@ProjectName AND Email=@Email";

SqlCommand sqlCmd = new SqlCommand(query, sqlCon);

sqlCmd.CommandType = CommandType.Text;

sqlCmd.Parameters.AddWithValue("@ProjectName", txtProjectName.Text.Trim());

sqlCmd.Parameters.AddWithValue("@Email", txtEmail.Text.Trim());

int count = Convert.ToInt32(sqlCmd.ExecuteScalar());

if (count == 1)

{

ShowDetails();

}

else

{

MessageBox.Show("Please enter the name of a project that you created. To create a new project, click 'Create New'.");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

sqlCon.Close();

}

}

}

// Collects data to be shown of project from the database.

void ShowDetails()

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

String queryLastDate = "SELECT LastEditDate FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmdLastDate = new SqlCommand(queryLastDate, sqlCon);

sqlCmdLastDate.CommandType = CommandType.Text;

sqlCmdLastDate.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string LastDate = Convert.ToString(sqlCmdLastDate.ExecuteScalar());

String querySysRes = "SELECT SystemRes FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmdSysRes = new SqlCommand(querySysRes, sqlCon);

sqlCmdSysRes.CommandType = CommandType.Text;

sqlCmdSysRes.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string SysRes = Convert.ToString(sqlCmdSysRes.ExecuteScalar());

String queryReqAirVolRate = "SELECT ReqAirVolRate FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmdReqAirVolRate = new SqlCommand(queryReqAirVolRate, sqlCon);

sqlCmdReqAirVolRate.CommandType = CommandType.Text;

sqlCmdReqAirVolRate.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string ReqAirVolRate = Convert.ToString(sqlCmdReqAirVolRate.ExecuteScalar());

String queryRoomLength = "SELECT RoomLength FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmdRoomLength = new SqlCommand(queryRoomLength, sqlCon);

sqlCmdRoomLength.CommandType = CommandType.Text;

sqlCmdRoomLength.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string RoomLength = Convert.ToString(sqlCmdRoomLength.ExecuteScalar());

String queryRoomWidth = "SELECT RoomWidth FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmdRoomWidth = new SqlCommand(queryRoomWidth, sqlCon);

sqlCmdRoomWidth.CommandType = CommandType.Text;

sqlCmdRoomWidth.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string RoomWidth = Convert.ToString(sqlCmdRoomWidth.ExecuteScalar());

String queryRoomHeight = "SELECT RoomHeight FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmdRoomHeight = new SqlCommand(queryRoomHeight, sqlCon);

sqlCmdRoomHeight.CommandType = CommandType.Text;

sqlCmdRoomHeight.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string RoomHeight = Convert.ToString(sqlCmdRoomHeight.ExecuteScalar());

String queryFanCode = "SELECT FanCode FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmdFanCode = new SqlCommand(queryFanCode, sqlCon);

sqlCmdFanCode.CommandType = CommandType.Text;

sqlCmdFanCode.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string FanCode = Convert.ToString(sqlCmdFanCode.ExecuteScalar());

String queryTypeOfRoom = "SELECT TypeOfRoom FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmdTypeOfRoom = new SqlCommand(queryTypeOfRoom, sqlCon);

sqlCmdTypeOfRoom.CommandType = CommandType.Text;

sqlCmdTypeOfRoom.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string TypeOfRoom = Convert.ToString(sqlCmdTypeOfRoom.ExecuteScalar());

String queryPeople = "SELECT NumberOfPeople FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmdPeople = new SqlCommand(queryPeople, sqlCon);

sqlCmdPeople.CommandType = CommandType.Text;

sqlCmdPeople.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string People = Convert.ToString(sqlCmdPeople.ExecuteScalar());

String query1 = "SELECT SupplyAttenuatorType FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmd1 = new SqlCommand(query1, sqlCon);

sqlCmd1.CommandType = CommandType.Text;

sqlCmd1.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string SupplyType = Convert.ToString(sqlCmd1.ExecuteScalar());

String query2 = "SELECT ExtractAttenuatorType FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmd2 = new SqlCommand(query2, sqlCon);

sqlCmd2.CommandType = CommandType.Text;

sqlCmd2.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string ExtractType = Convert.ToString(sqlCmd2.ExecuteScalar());

String query3 = "SELECT AcousticBarrierType FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmd3 = new SqlCommand(query3, sqlCon);

sqlCmd3.CommandType = CommandType.Text;

sqlCmd3.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string BarrierType = Convert.ToString(sqlCmd3.ExecuteScalar());

String query4 = "SELECT RoomTerminalSideLength FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmd4 = new SqlCommand(query4, sqlCon);

sqlCmd4.CommandType = CommandType.Text;

sqlCmd4.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string RoomTerminalSideLength = Convert.ToString(sqlCmd4.ExecuteScalar());

String query5 = "SELECT Units FROM tblProject WHERE [tblProject.ProjectName=@ProjectName](mailto:tblProject.ProjectName=@ProjectName)";

SqlCommand sqlCmd5 = new SqlCommand(query5, sqlCon);

sqlCmd5.CommandType = CommandType.Text;

sqlCmd5.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string Units = Convert.ToString(sqlCmd5.ExecuteScalar());

double roomVolume = Convert.ToDouble(RoomLength) \* Convert.ToDouble(RoomWidth) \* Convert.ToDouble(RoomHeight);

string RoomVolume = Convert.ToString(roomVolume);

// Writes a text file showing details and outputs content into a textbox on screen.

try

{

String path = @"C:\Users\peybi\Downloads\" + txtProjectName.Text + "Details.txt";

StreamWriter file = new StreamWriter(path);

file.Write("DETAILS OF PROJECT\r\n");

file.Write(" \r\n");

file.Write("Project Name: " + txtProjectName.Text+ "\r\n");

file.Write("Last edit date: " + LastDate+ "\r\n");

file.Write("System Resistance: " + SysRes+ "\r\n");

file.Write("Required Air Volume Flow Rate: " + ReqAirVolRate+ "\r\n");

file.Write("Room Length: " + RoomLength+ "\r\n");

file.Write("Room Width: " + RoomWidth+ "\r\n");

file.Write("Room Height: " + RoomHeight + "\r\n");

file.Write("Room Volume: " + RoomVolume+ "\r\n");

file.Write("Fan code: " + FanCode+ "\r\n");

file.Write("Type of room: " + TypeOfRoom+ "\r\n");

file.Write("Number of people: " + People+ "\r\n");

file.Write("Supply attenuator type: " + SupplyType+ "\r\n");

file.Write("Extract attenuator type: " + ExtractType+ "\r\n");

file.Write("Acoustic barrier type: " + BarrierType+ "\r\n");

file.Write("Room terminal side length: " + RoomTerminalSideLength+ "\r\n");

file.Write("Units: " + Units+ "\r\n");

file.Close();

using (StreamReader sr = new StreamReader(path))

{

string filetext = sr.ReadToEnd();

txtDetails.Text = filetext;

}

file.Close();

File.Delete(path);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

// Shows the user what projects they have created; if the user is an admin, they see everyone's project, if not then they just see their own.

private void btnSeeProjects\_Click(object sender, RoutedEventArgs e)

{

if(txtEmail.Text == "")

{

MessageBox.Show("Please enter your email.");

}

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

String queryadmin = "SELECT Admin FROM tblUser WHERE Email=@Email";

SqlCommand sqlCmd4 = new SqlCommand(queryadmin, sqlCon);

sqlCmd4.CommandType = CommandType.Text;

sqlCmd4.Parameters.AddWithValue("@Email", txtEmail.Text);

string admin = Convert.ToString(sqlCmd4.ExecuteScalar());

if (admin == "Yes")

{

ArrayList filenames = new ArrayList();

String querynames = "SELECT ProjectName FROM tblProject";

SqlCommand sqlCmd5 = new SqlCommand(querynames, sqlCon);

sqlCmd5.CommandType = CommandType.Text;

SqlDataReader reader = sqlCmd5.ExecuteReader();

while (reader.Read())

{

filenames.Add(reader[0]);

}

int arraylength = filenames.Count;

try

{

String path = @"C:\Users\peybi\Downloads\" + txtProjectName.Text + "FileNames.txt";

StreamWriter file = new StreamWriter(path);

for(int count = 0;count <= arraylength - 1;count++)

{

file.Write(filenames[count] + "\r\n");

}

file.Close();

using (StreamReader sr = new StreamReader(path))

{

string filetext = sr.ReadToEnd();

txtProjectFileNames.Text = filetext;

}

file.Close();

File.Delete(path);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else

{

String querynames = "SELECT ProjectName FROM tblProject WHERE Email=@Email";

SqlCommand sqlCmd5 = new SqlCommand(querynames, sqlCon);

sqlCmd5.CommandType = CommandType.Text;

sqlCmd5.Parameters.AddWithValue("@Email", txtEmail.Text);

string filenames = Convert.ToString(sqlCmd5.ExecuteScalar());

try

{

String path = @"C:\Users\peybi\Downloads\" + txtProjectName.Text + "FileNames.txt";

StreamWriter file = new StreamWriter(path);

file.Write(filenames + "\r\n");

file.Close();

using (StreamReader sr = new StreamReader(path))

{

string filetext = sr.ReadToEnd();

txtProjectFileNames.Text = filetext;

}

file.Close();

File.Delete(path);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

// Deletes project from database.

private void btnDeleteProject\_Click(object sender, RoutedEventArgs e)

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

try

{

string sql = "DELETE FROM tblProject WHERE Email=@Email AND ProjectName=@ProjectName";

using (SqlCommand cmd = new SqlCommand(sql, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@Email", SqlDbType.NVarChar).Value = txtEmail.Text.Trim();

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.ExecuteNonQuery();

}

MessageBox.Show("Your information has been deleted.");

}

catch (Exception ex)

{

MessageBox.Show(Convert.ToString(ex));

}

}

}

}

## HowTo.xaml

This is a page with instructions on to help users find their way around the application.

<Page x:Class="Nuacoustic.HowTo"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="HowTo">

<Grid Background="white">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<Button Name="btnHome" Grid.Column="1" Grid.Row="0" Content="HOME" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19" Click="btnHome\_Click"/>

<Button Name="btnHowTo" Grid.Column="2" Grid.Row="0" Content="HOW TO" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19" Click="btnHowTo\_Click"/>

<Button Name="btnSettings" Grid.Column="3" Grid.Row="0" Content="SETTINGS" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19" Click="btnSettings\_Click"/>

<Button Name="btnLogOut" Grid.Column="4" Grid.Row="0" Content="LOG OUT" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19" Click="btnLogOut\_Click"/>

<TextBlock Text="HOW TO" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="30" FontWeight="Bold" Margin="10" Grid.Row="1" Grid.ColumnSpan="5"/>

<StackPanel Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="5" Grid.RowSpan="4">

<TextBlock Text="To create a new project, click 'Create new project', enter the details of the room and click 'Save'." FontSize="12"/>

<TextBlock Text="On the Home page, to see the names of the projects you have created, enter your email and click 'See Projects'." FontSize="12"/>

<TextBlock Text="On the Home page, to see the details of a project you have created, enter your email and project name and click 'Show details'." FontSize="12"/>

<TextBlock Text="On the Home page, if you click 'Open Project', you will be taken to a page that has blank boxes. To fill in these boxes with your project details, enter " FontSize="12"/>

<TextBlock Text="your email and project name and click 'Show'. Now you can edit the boxes and click 'Update' afterwards to save the changes." FontSize="12"/>

<TextBlock Text="" FontSize="12"/>

<TextBlock Text="After you are happy with your changes, click 'Project Page' which will take you to a screen with a grid. This is a grid to help you model your room with" FontSize="12"/>

<TextBlock Text="the location of the fan, and the location of the listener. When entering the details, you will be asked to enter the number of people there will be in the " FontSize="12"/>

<TextBlock Text="room. If this is more than one, put the location of the listener in the centre of where the people would be." FontSize="12"/>

<TextBlock Text="" FontSize="12"/>

<TextBlock Text="To change the details, click on 'EDIT' which will take you to another page that you can enter the details for." FontSize="12"/>

<TextBlock Text="If you are unsure of the specific fan's fancode, put the fancode of any fan in that range and the system will give you better options, if there are any." FontSize="12"/>

<TextBlock Text="To change the units you work in, go to 'Settings' where you can choose to use metric or Imperial units." FontSize="12"/>

<TextBlock Text="After modelling your room, click 'Summary' which will give you a summary in the blank box with details of the suitability of the fan you have chosen, " FontSize="12"/>

<TextBlock Text="and where necessary, any amendments that need to be made." FontSize="12"/>

</StackPanel>

</Grid>

</Page>

## LoginScreen.xaml

<Page x:Class="Nuacoustic.LoginScreen"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

Title="LoginScreen" Height="450" Width="800">

<Grid Background="white">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<TextBlock Text="LOGIN" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="30" FontWeight="Bold" Margin="10" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3" FontFamily="{StaticResource MontserratRegular}"/>

<!--email section-->

<TextBlock Text="Email" Foreground="black" FontSize="30" Grid.Column="0" Grid.Row="2" Margin=" 210 10 160 10" Grid.ColumnSpan="2" FontFamily="/Fonts/Montserrat/Montserrat-Regular.ttf"/>

<TextBox x:Name="txtEmail" Grid.Column="1" Grid.Row="2" Background ="#cccccc" Margin="150.333,10.333,18,10" Grid.ColumnSpan="2" FontSize="25" FontFamily="/Fonts/Montserrat/Montserrat-Regular.ttf" />

<!--password section-->

<TextBlock Text="Password" Foreground="black" FontSize="30" Grid.Column="0" Grid.Row="3" Margin=" 210 10 160 10" Grid.ColumnSpan="2"/>

<PasswordBox PasswordChar="\*" x:Name="txtPassword" Grid.Column="1" Grid.Row="3" Background ="#cccccc" Margin="150.333,10,18,9.667" Grid.ColumnSpan="2" FontSize="25"/>

<!--buttons-->

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="4" BorderThickness="1" CornerRadius="25" Margin="10"/>

<Button Name="btnSubmit" Click="btnSubmit\_Click" Grid.Column="1" Grid.Row="4" Content="SUBMIT" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="23" Margin="19"/>

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="2" Grid.Row="0" BorderThickness="1" CornerRadius="25" Margin="10"/>

<Button Name="btnClickToRegister" Grid.Column="2" Grid.Row="0" Content="First time? Register here" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="15" Margin="18.5" Click="btnClickToRegister\_Click"/>

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="5" BorderThickness="1" CornerRadius="25" Margin="10"/>

<Button Name="btnForgotPassword" Grid.Column="1" Grid.Row="5" Content="Forgot Password" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="21" Margin="18.5" Click="btnForgotPassword\_Click" FontFamily="{StaticResource MontserratRegular}"/>

</Grid>

</Page>

## LoginScreen.xaml.cs

When users enter their email and password, their password is hashed and the key is calculated using their email. If the key is found in the hashtable, then the hashed password is checked against the value stored in the key’s position. If the email and password combination is correct then the user is taken to the Home page.

using System;

using System.Data;

using System.Data.SqlClient;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Security.Cryptography;

using System.Collections;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for LoginScreen.xaml

/// </summary>

public partial class LoginScreen : Page

{

public Hashtable Logins;

public LoginScreen()

{

InitializeComponent();

Logins = (Application.Current.MainWindow as MainWindow).Logins;

}

private void DisplayEmail()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.email = txtEmail.Text;

}

private void ShareHashtable()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.Logins = Logins;

}

// Calculates the hash of entered password.

static string ComputeSha256Hash(string rawData)

{

// Create a SHA256

using (SHA256 sha256Hash = SHA256.Create())

{

// ComputeHash - returns byte array

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

// Convert byte array to a string

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

void Temp()

{

string temp = ComputeSha256Hash("hello");

MessageBox.Show(temp);

}

int GetHashtableTotal()

{

string email = Convert.ToString(txtEmail.Text);

int countemail = 0;

int arrayLength = email.Length;

int[] emailstr = new int[arrayLength];

int total = 0;

foreach (char c in email)

{

emailstr[countemail] = System.Convert.ToInt32(c);

total = emailstr[countemail] + total;

countemail++;

}

return total;

}

// Checks if email and password combination are in the database.

private void btnSubmit\_Click(object sender, RoutedEventArgs e)

{

string plainData = Convert.ToString(txtPassword.Password.Trim());

string hashedData = ComputeSha256Hash(plainData);

int total = GetHashtableTotal();

int hashlocation = total % 997;

if(Logins.ContainsKey(hashlocation) == false)

{

MessageBox.Show("Your email or password is incorrect.");

}

else

{

if ((Logins.ContainsKey(hashlocation) == true && (Convert.ToString(Logins[hashlocation]) == hashedData)))

{

DisplayEmail();

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.CurrentPage = ApplicationPage.HomePage;

}

}

}

private void btnClickToRegister\_Click(object sender, RoutedEventArgs e)

{

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.Register)

{

mainWindow.CurrentPage = ApplicationPage.Register;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private void btnForgotPassword\_Click(object sender, RoutedEventArgs e)

{

DisplayEmail();

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.ForgotPassword)

{

mainWindow.CurrentPage = ApplicationPage.ForgotPassword;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

}

}

## MainPage.xaml

<Page x:Class="Nuacoustic.MainPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="MainPage">

<Grid Background="white">

<StackPanel>

<Label Content="Welcome to Nuacoustic!" FontSize="40" FontWeight="Bold" HorizontalAlignment="Center" Margin="30"/>

<Image Source="/images/nuairelogo1.jpg" Height="178" Margin="262,10,262.333,10" />

<Border Background="#00a3e0" CornerRadius="40" Margin="275,1,275,20">

<Button x:Name="btnClickToLogin" Content="Click To Login" FontFamily="resources\Montserrat-Regular" FontWeight="Bold" Background="#00a3e0" Margin="10,20,10.333,20" Foreground="Black" FontSize="25" Click="btnClickToLogin\_Click"/>

</Border>

</StackPanel>

</Grid>

</Page>

## MainPage.xaml.cs

using System.Windows;

using System.Windows.Controls;

using System.Collections;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for MainPage.xaml

/// </summary>

public partial class MainPage : Page

{

public Hashtable hashtable;

public MainPage()

{

InitializeComponent();

hashtable = (Application.Current.MainWindow as MainWindow).Logins;

}

private void ShareHashtable()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.Logins = hashtable;

}

private void btnClickToLogin\_Click(object sender, RoutedEventArgs e)

{

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.LoginScreen)

{

mainWindow.CurrentPage = ApplicationPage.LoginScreen;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

}

}

## MainWindow.xaml

MainWindow is used as frame for all the pages to be placed on to – this allowed me to link pages so users could move between them.

<Window x:Class="Nuacoustic.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Nuacoustic"

xmlns:localValueConverters="clr-namespace:Nuacoustic.ValueConverters"

mc:Ignorable="d"

Title="Welcome" Height="450" Width="800">

<Grid>

<Frame Name="MainFrame" Content="{Binding CurrentPage, Converter={localValueConverters:ApplicationPageValueConverter}}" HorizontalAlignment="Center" VerticalAlignment="Center" NavigationUIVisibility="Hidden"/>

</Grid>

</Window>

## MainWindow.xaml.cs

using System.ComponentModel;

using PropertyChanged;

using System.Collections;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

[AddINotifyPropertyChangedInterface]

public partial class MainWindow : INotifyPropertyChanged

{

public MainWindow()

{

DataContext = this;

InitializeComponent();

}

public event PropertyChangedEventHandler PropertyChanged = (sender, e) => { };

public ApplicationPage CurrentPage { get; set; }

public string email { get; set; }

public string projectname { get; set; }

public Hashtable Logins { get; set; }

}

}

## Project.xaml

This section is used for the display of the Project page. There is a grid, drawn by a series lines, on which there are 4 pictures that represent the positions of the listener, the fan, the fan supply and the fan extract.

<Page x:Class="Nuacoustic.Project"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Project">

<Grid Background="white">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Height="46" Grid.Row="0" Grid.Column="0"/>

<Button Name="btnHome" Content="HOME" Background="Transparent" Grid.Row="0" Grid.Column="1" FontWeight="bold" FontSize="25" Margin="15" Click="btnHome\_Click"/>

<Button Name="btnEdit" Content="EDIT" Background="Transparent" Grid.Row="1" Grid.Column="0" FontWeight="Bold" FontSize="25" Margin="15" Click="btnEdit\_Click"/>

<TextBlock Text="UNITS" Background="Transparent" Grid.Row="2" Grid.Column="0" FontWeight="Bold" FontSize="23" Margin="15,14.667,117,15"/>

<ComboBox Name="txtUnits" Grid.Column="0" Grid.Row="2" Background="Transparent" FontSize="20" Margin="88,14.667,15,15">

<ComboBoxItem IsSelected="True">Metres</ComboBoxItem>

</ComboBox>

<TextBox Name="txtSummary" Grid.Row="3" Grid.Column="0" FontSize="11" Background="Transparent" Margin="5,5,5,15" Grid.RowSpan="4"/>

<Button Name="btnHowTo" Content="HOW TO" Background="Transparent" Grid.Row="6" Grid.Column="2" FontWeight="Bold" FontSize="25" Margin="15" Click="btnHowTo\_Click"/>

<Button Name="btnGraduations" Content="GRADUATIONS" Background="Transparent" Grid.Row="6" Grid.Column="1" FontWeight="Bold" FontSize="25" Margin="15" Click="btnGraduations\_Click"/>

<TextBlock Text="Project Name:" Grid.Column="2" FontWeight="bold" FontSize="26" HorizontalAlignment="center" VerticalAlignment="Center"/>

<TextBox Name="txtProjectName" Grid.Row="0" Grid.Column="3" FontSize="26" Background="LightGray" Margin="5"/>

<Canvas Name="canvas" MouseLeftButtonDown="CanvasMouseLeftButtonDown"

MouseLeftButtonUp="CanvasMouseLeftButtonUp"

MouseMove="CanvasMouseMove" HorizontalAlignment="Right" VerticalAlignment="Center" Height="320" Width="600" Background="WhiteSmoke" Margin="0,0.333,0,0" Grid.ColumnSpan="3" Grid.RowSpan="5" Grid.Row="1" Grid.Column="1">

<!-- vertical lines -->

<Line Panel.ZIndex="2" X1="0" Y1="00" X2="0" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="40" Y1="00" X2="40" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="80" Y1="00" X2="80" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="120" Y1="00" X2="120" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="160" Y1="00" X2="160" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="200" Y1="00" X2="200" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="240" Y1="00" X2="240" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="280" Y1="00" X2="280" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="320" Y1="00" X2="320" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="360" Y1="00" X2="360" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="400" Y1="00" X2="400" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="440" Y1="00" X2="440" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="480" Y1="00" X2="480" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="520" Y1="00" X2="520" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="560" Y1="00" X2="560" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="600" Y1="00" X2="600" Y2="320" Stroke="Black" StrokeThickness="1"/>

<!-- horizontal lines -->

<Line Panel.ZIndex="2" X1="0" Y1="0" X2="600" Y2="0" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="32" X2="600" Y2="32" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="64" X2="600" Y2="64" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="96" X2="600" Y2="96" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="128" X2="600" Y2="128" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="160" X2="600" Y2="160" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="192" X2="600" Y2="192" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="224" X2="600" Y2="224" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="256" X2="600" Y2="256" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="288" X2="600" Y2="288" Stroke="Black" StrokeThickness="1"/>

<Line Panel.ZIndex="2" X1="0" Y1="320" X2="600" Y2="320" Stroke="Black" StrokeThickness="1"/>

<Image Name="FanSymbol" Panel.ZIndex="40" Source="pack://application:,,,/images/fan-symbol.jpg" Height="32" Canvas.Left="23" Canvas.Top="16" Width="35" />

<Image Name="ListenerSymbol" Panel.ZIndex="40" Source="pack://application:,,,/images/listener-symbol.jpg" Height="32" Canvas.Left="23" Canvas.Top="48" Width="35" />

<Image Name="SupplySymbol" Panel.ZIndex="40" Source="pack://application:,,,/images/supply-symbol.jpg" Height="32" Canvas.Left="23" Canvas.Top="80" Width="35" />

<Image Name="ExtractSymbol" Panel.ZIndex="40" Source="pack://application:,,,/images/extract-symbol.jpg" Height="32" Canvas.Left="23" Canvas.Top="112" Width="35" />

</Canvas>

<Button Name="btnSummary" Content="SUMMARY" Background="Transparent" Grid.Row="6" Grid.Column="3" FontWeight="Bold" FontSize="25" Margin="15" Click="btnSummary\_Click"/>

</Grid>

</Page>

## 

## Project.xaml.cs

The first part of this section identifies where the pictures are on the grid. As soon as the user stops dragging the image, its location is updated in the database. Then the program finds the total length and width of the room and finds out the length in metres or feet one pixel is according to that scale. Then, Pythagorus’ theorem is used to find the length in pixels between the objects. Finally, it finds the real-life distance between the objects represented by the pictures by using the ratio found earlier.

When the user presses ‘Summary’, a series of calculations are performed to find if their proposed plan is suitable. If it is not suitable, then alternative fan models are identified and suggested on the summary data sheet. The data sheet is formed with a text file.

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Data;

using System.Data.SqlClient;

using System.IO;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for Project.xaml

/// </summary>

public partial class Project : Page

{

// Initiating required variables.

public string newFanCode = "0";

public double FanXCoord;

public double FanYCoord;

public double ListenerXCoord;

public double ListenerYCoord;

public double SupplyXCoord;

public double SupplyYCoord;

public double ExtractXCoord;

public double ExtractYCoord;

public Project()

{

InitializeComponent();

txtProjectName.Text = (Application.Current.MainWindow as MainWindow).projectname;

}

public string message;

private void btnHowTo\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.HowTo)

{

mainWindow.CurrentPage = ApplicationPage.HowTo;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

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

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.CurrentProjectSettings)

{

mainWindow.CurrentPage = ApplicationPage.CurrentProjectSettings;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private void btnHome\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.HomePage)

{

mainWindow.CurrentPage = ApplicationPage.HomePage;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

private Image draggedImage;

private Point mousePosition;

// Moves picture with mouse.

private void CanvasMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

canvas.MaxHeight = 320;

canvas.MaxWidth = 600;

var image = e.Source as Image;

canvas.ClipToBounds = true;

if (image != null && canvas.CaptureMouse())

{

mousePosition = e.GetPosition(canvas);

draggedImage = image;

Panel.SetZIndex(draggedImage, 4); // in case of multiple images

}

}

// Reads where picture is located pixel-wise, and puts this into the database.

void CanvasMouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

if (draggedImage != null)

{

if(txtProjectName.Text == "")

{

MessageBox.Show("Please enter a project name.");

}

canvas.ReleaseMouseCapture();

Panel.SetZIndex(draggedImage, 0);

if (draggedImage.Name == "FanSymbol")

{

FanXCoord = Canvas.GetLeft(draggedImage);

FanYCoord = Canvas.GetTop(draggedImage);

var FanXCoordstr = Convert.ToString(FanXCoord);

var FanYCoordstr = Convert.ToString(FanYCoord);

using (sqlCon)

{

string sql = "UPDATE tblProject SET XCoordFan = @XCoordFan WHERE ProjectName = @ProjectName";

using (SqlCommand cmd = new SqlCommand(sql, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@XCoordFan", SqlDbType.NVarChar).Value = FanXCoordstr;

cmd.ExecuteNonQuery();

}

string sql1 = "UPDATE tblProject SET YCoordFan = @YCoordFan WHERE ProjectName = @ProjectName";

using (SqlCommand cmd = new SqlCommand(sql1, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@YCoordFan", SqlDbType.NVarChar).Value = FanYCoordstr;

cmd.ExecuteNonQuery();

}

}

}

else if (draggedImage.Name == "ListenerSymbol")

{

if (txtProjectName.Text == "")

{

MessageBox.Show("Please enter a project name.");

}

ListenerXCoord = Canvas.GetLeft(draggedImage);

ListenerYCoord = Canvas.GetTop(draggedImage);

var ListenerXCoordstr = Convert.ToString(ListenerXCoord);

var ListenerYCoordstr = Convert.ToString(ListenerYCoord);

using (sqlCon)

{

string sql = "UPDATE tblProject SET XCoordListener = @XCoordListener WHERE ProjectName = @ProjectName";

using (SqlCommand cmd = new SqlCommand(sql, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@XCoordListener", SqlDbType.NVarChar).Value = ListenerXCoordstr;

cmd.ExecuteNonQuery();

}

string sql1 = "UPDATE tblProject SET YCoordListener = @YCoordListener WHERE ProjectName = @ProjectName";

using (SqlCommand cmd = new SqlCommand(sql1, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@YCoordListener", SqlDbType.NVarChar).Value = ListenerYCoordstr;

cmd.ExecuteNonQuery();

}

}

}

else if (draggedImage.Name == "SupplySymbol")

{

if (txtProjectName.Text == "")

{

MessageBox.Show("Please enter a project name.");

}

SupplyXCoord = Canvas.GetLeft(draggedImage);

SupplyYCoord = Canvas.GetTop(draggedImage);

var SupplyXCoordstr = Convert.ToString(SupplyXCoord);

var SupplyYCoordstr = Convert.ToString(SupplyYCoord);

using (sqlCon)

{

string sql = "UPDATE tblProject SET XCoordSupply = @XCoordSupply WHERE ProjectName = @ProjectName";

using (SqlCommand cmd = new SqlCommand(sql, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@XCoordSupply", SqlDbType.NVarChar).Value = SupplyXCoordstr;

cmd.ExecuteNonQuery();

}

string sql1 = "UPDATE tblProject SET YCoordSupply = @YCoordSupply WHERE ProjectName = @ProjectName";

using (SqlCommand cmd = new SqlCommand(sql1, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@YCoordSupply", SqlDbType.NVarChar).Value = SupplyYCoordstr;

cmd.ExecuteNonQuery();

}

}

}

else if (draggedImage.Name == "ExtractSymbol")

{

if (txtProjectName.Text == "")

{

MessageBox.Show("Please enter a project name.");

}

ExtractXCoord = Canvas.GetLeft(draggedImage);

ExtractYCoord = Canvas.GetTop(draggedImage);

var ExtractXCoordstr = Convert.ToString(ExtractXCoord);

var ExtractYCoordstr = Convert.ToString(ExtractYCoord);

using (sqlCon)

{

string sql = "UPDATE tblProject SET XCoordExtract = @XCoordExtract WHERE ProjectName = @ProjectName";

using (SqlCommand cmd = new SqlCommand(sql, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@XCoordExtract", SqlDbType.NVarChar).Value = ExtractXCoordstr;

cmd.ExecuteNonQuery();

}

string sql1 = "UPDATE tblProject SET YCoordExtract = @YCoordExtract WHERE ProjectName = @ProjectName";

using (SqlCommand cmd = new SqlCommand(sql1, sqlCon))

{

// Create and set the parameters values

cmd.Parameters.Add("@ProjectName", SqlDbType.NVarChar).Value = txtProjectName.Text.Trim();

cmd.Parameters.Add("@YCoordExtract", SqlDbType.NVarChar).Value = ExtractYCoordstr;

cmd.ExecuteNonQuery();

}

}

}

draggedImage = null;

}

}

// Calculates the graduations of each square on the grid.

string Graduations()

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

String queryRoomLength = "SELECT RoomLength FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd8 = new SqlCommand(queryRoomLength, sqlCon);

sqlCmd8.CommandType = CommandType.Text;

sqlCmd8.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string RoomLengthstr = Convert.ToString(sqlCmd8.ExecuteScalar());

decimal RoomLength = Convert.ToDecimal(RoomLengthstr);

String queryRoomWidth = "SELECT RoomWidth FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd9 = new SqlCommand(queryRoomWidth, sqlCon);

sqlCmd9.CommandType = CommandType.Text;

sqlCmd9.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string RoomWidthstr = Convert.ToString(sqlCmd9.ExecuteScalar());

decimal RoomWidth = Convert.ToDecimal(RoomWidthstr);

decimal horizontalGraduationfull = RoomLength / 15;

decimal horizontalGraduation = Decimal.Round(horizontalGraduationfull, 2);

decimal verticalGraduationfull = RoomWidth / 10;

decimal verticalGraduation = Decimal.Round(verticalGraduationfull, 2);

string message = "Horizontal: starting from 0 on the left, each graduation is " + horizontalGraduation + "m. Vertical: starting from 0 at the bottom, each graduation is " + verticalGraduation + "m.";

if (txtUnits.Text == "Feet")

{

decimal convertToFeet = Convert.ToDecimal(3.28);

horizontalGraduation = Decimal.Round((horizontalGraduationfull \* convertToFeet), 2);

verticalGraduation = Decimal.Round((verticalGraduationfull \* convertToFeet), 2);

message = "Horizontal: starting from 0 on the left, each graduation is " + horizontalGraduation + "ft. Vertical: starting from 0 at the bottom, each graduation is " + verticalGraduation + "ft.";

}

return message;

}

private void CanvasMouseMove(object sender, MouseEventArgs e)

{

if (draggedImage != null)

{

var position = e.GetPosition(canvas);

var offset = position - mousePosition;

mousePosition = position;

Canvas.SetLeft(draggedImage, Canvas.GetLeft(draggedImage) + offset.X);

Canvas.SetTop(draggedImage, Canvas.GetTop(draggedImage) + offset.Y);

}

}

private void btnSummary\_Click(object sender, RoutedEventArgs e)

{

SqlConnection sqlCon = new SqlConnection(@"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;");

if (sqlCon.State == ConnectionState.Closed)

sqlCon.Open();

// get units

String queryUnits = "SELECT Units FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmdUnits = new SqlCommand(queryUnits, sqlCon);

sqlCmdUnits.CommandType = CommandType.Text;

sqlCmdUnits.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string Units = Convert.ToString(sqlCmdUnits.ExecuteScalar());

// calculating the total air flow of the room

String query = "SELECT NumberOfPeople FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd = new SqlCommand(query, sqlCon);

sqlCmd.CommandType = CommandType.Text;

sqlCmd.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

int NumberOfPeople = Convert.ToInt32(sqlCmd.ExecuteScalar());

// get supply flow rate

double GetSupplyFlowRate()

{

String query1 = "SELECT SupplyFlowRate FROM tblTypeOfRoom INNER JOIN tblProject ON tblProject.TypeOfRoom = tblTypeOfRoom.TypeName WHERE tblProject.ProjectName=@ProjectName";

SqlCommand sqlCmd1 = new SqlCommand(query1, sqlCon);

sqlCmd1.CommandType = CommandType.Text;

sqlCmd1.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string SupplyFlowRatestr = Convert.ToString(sqlCmd1.ExecuteScalar());

double SupplyFlowRate = Convert.ToDouble(SupplyFlowRatestr);

return SupplyFlowRate;

}

// calculating total air flow

double totalAirFlow = NumberOfPeople \* GetSupplyFlowRate();

// calculating constant a in formula y=ax^2

double x = totalAirFlow;

String query2 = "SELECT SystemRes FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd2 = new SqlCommand(query2, sqlCon);

sqlCmd2.CommandType = CommandType.Text;

sqlCmd2.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

double SystemRes = Convert.ToInt32(sqlCmd2.ExecuteScalar());

double y = SystemRes;

double a = y / Math.Pow(x, 2);

// find the types of attenuators and acoustic barrier that were inputted by the user from database

string FindSupplyAttenuationType()

{

String query5 = "SELECT SupplyAttenuatorType FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd5 = new SqlCommand(query5, sqlCon);

sqlCmd5.CommandType = CommandType.Text;

sqlCmd5.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string SupplyAttenuation = Convert.ToString(sqlCmd5.ExecuteScalar());

return SupplyAttenuation;

}

string FindExtractAttenuationType()

{

String query6 = "SELECT ExtractAttenuatorType FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmdextract = new SqlCommand(query6, sqlCon);

sqlCmdextract.CommandType = CommandType.Text;

sqlCmdextract.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string ExtractAttenuation = Convert.ToString(sqlCmdextract.ExecuteScalar());

return ExtractAttenuation;

}

string FindBreakoutBarrierType()

{

String query6 = "SELECT AcousticBarrierType FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmdbreakout = new SqlCommand(query6, sqlCon);

sqlCmdbreakout.CommandType = CommandType.Text;

sqlCmdbreakout.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string BarrierType = Convert.ToString(sqlCmdbreakout.ExecuteScalar());

return BarrierType;

}

// each 'SuitabilityXBC'x'' subroutine calculates the corrected supply, extract and breakout values for each of 8 octave bands.

// These are then sent to another subroutine called 'calculatingTheSuitability'.

// The different 'x' values are the fan codes.

#region findingSuitability

string SuitabilityXBC15()

{

// y = 9677.894325825380000x3 - 6494.877163984340000x2 - 3796.919390602680000x + 900.172738777446000

double A = 9677.894325825380000;

double B = -6494.877163984340000 - a;

double C = -3796.919390602680000;

double D = 900.172738777446000;

double FanFlowRate = SolvingACubic(A, B, C, D);

double DeductionIndB = FindDeductionIndB(FanFlowRate);

// for 63Hz octave band

double correctedSupply63 = 75 - DeductionIndB;

double correctedExtract63 = 69 - DeductionIndB;

double correctedBreakout63 = 61 - DeductionIndB;

// for 125Hz octave band

double correctedSupply125 = 72 - DeductionIndB;

double correctedExtract125 = 59 - DeductionIndB;

double correctedBreakout125 = 57 - DeductionIndB;

// for 250Hz octave band

double correctedSupply250 = 65 - DeductionIndB;

double correctedExtract250 = 55 - DeductionIndB;

double correctedBreakout250 = 42 - DeductionIndB;

// for 500Hz octave band

double correctedSupply500 = 66 - DeductionIndB;

double correctedExtract500 = 55 - DeductionIndB;

double correctedBreakout500 = 43 - DeductionIndB;

// for 1000Hz octave band

double correctedSupply1000 = 68 - DeductionIndB;

double correctedExtract1000 = 61 - DeductionIndB;

double correctedBreakout1000 = 41 - DeductionIndB;

// for 2000Hz octave band

double correctedSupply2000 = 64 - DeductionIndB;

double correctedExtract2000 = 55 - DeductionIndB;

double correctedBreakout2000 = 37 - DeductionIndB;

// for 4000Hz octave band

double correctedSupply4000 = 59 - DeductionIndB;

double correctedExtract4000 = 45 - DeductionIndB;

double correctedBreakout4000 = 34 - DeductionIndB;

// for 8000Hz octave band

double correctedSupply8000 = 57 - DeductionIndB;

double correctedExtract8000 = 41 - DeductionIndB;

double correctedBreakout8000 = 23 - DeductionIndB;

string suitable = calculatingTheSuitability(correctedSupply63, correctedExtract63, correctedBreakout63, correctedSupply125, correctedExtract125, correctedBreakout125,

correctedSupply250, correctedExtract250, correctedBreakout250, correctedSupply500, correctedExtract500, correctedBreakout500,

correctedSupply1000, correctedExtract1000, correctedBreakout1000, correctedSupply2000, correctedExtract2000, correctedBreakout2000,

correctedSupply4000, correctedExtract4000, correctedBreakout4000, correctedSupply8000, correctedExtract8000, correctedBreakout8000);

return suitable;

}

string SuitabilityXBC25()

{

// y = -22,748.652434210700000x3 + 7,641.356938960030000x2 - 1,808.489952946460000x + 965.580779882142000

double A = -22748.652434210700000;

double B = 7641.356938960030000 - a;

double C = -1808.489952946460000;

double D = 965.580779882142000;

double FanFlowRate = SolvingACubic(A, B, C, D);

double DeductionIndB = FindDeductionIndB(FanFlowRate);

// for 63Hz octave band

double correctedSupply63 = 85 - DeductionIndB;

double correctedExtract63 = 79 - DeductionIndB;

double correctedBreakout63 = 72 - DeductionIndB;

// for 125Hz octave band

double correctedSupply125 = 86 - DeductionIndB;

double correctedExtract125 = 73 - DeductionIndB;

double correctedBreakout125 = 71 - DeductionIndB;

// for 250Hz octave band

double correctedSupply250 = 81 - DeductionIndB;

double correctedExtract250 = 71 - DeductionIndB;

double correctedBreakout250 = 58 - DeductionIndB;

// for 500Hz octave band

double correctedSupply500 = 85 - DeductionIndB;

double correctedExtract500 = 74 - DeductionIndB;

double correctedBreakout500 = 61 - DeductionIndB;

// for 1000Hz octave band

double correctedSupply1000 = 75 - DeductionIndB;

double correctedExtract1000 = 68 - DeductionIndB;

double correctedBreakout1000 = 48 - DeductionIndB;

// for 2000Hz octave band

double correctedSupply2000 = 75 - DeductionIndB;

double correctedExtract2000 = 65 - DeductionIndB;

double correctedBreakout2000 = 47 - DeductionIndB;

// for 4000Hz octave band

double correctedSupply4000 = 71 - DeductionIndB;

double correctedExtract4000 = 57 - DeductionIndB;

double correctedBreakout4000 = 47 - DeductionIndB;

// for 8000Hz octave band

double correctedSupply8000 = 73 - DeductionIndB;

double correctedExtract8000 = 57 - DeductionIndB;

double correctedBreakout8000 = 39 - DeductionIndB;

string suitable = calculatingTheSuitability(correctedSupply63, correctedExtract63, correctedBreakout63, correctedSupply125, correctedExtract125, correctedBreakout125,

correctedSupply250, correctedExtract250, correctedBreakout250, correctedSupply500, correctedExtract500, correctedBreakout500,

correctedSupply1000, correctedExtract1000, correctedBreakout1000, correctedSupply2000, correctedExtract2000, correctedBreakout2000,

correctedSupply4000, correctedExtract4000, correctedBreakout4000, correctedSupply8000, correctedExtract8000, correctedBreakout8000);

return suitable;

}

string SuitabilityXBC45()

{

// y = -225.400901870103000x3 - 1,336.437266669990000x2 - 770.006574013649000x + 811.711703606484000

double A = -225.400901870103000;

double B = -1336.437266669990000 - a;

double C = -770.006574013649000;

double D = 811.711703606484000;

double FanFlowRate = SolvingACubic(A, B, C, D);

double DeductionIndB = FindDeductionIndB(FanFlowRate);

// for 63Hz octave band

double correctedSupply63 = 87 - DeductionIndB;

double correctedExtract63 = 84 - DeductionIndB;

double correctedBreakout63 = 74 - DeductionIndB;

// for 125Hz octave band

double correctedSupply125 = 80 - DeductionIndB;

double correctedExtract125 = 75 - DeductionIndB;

double correctedBreakout125 = 65 - DeductionIndB;

// for 250Hz octave band

double correctedSupply250 = 85 - DeductionIndB;

double correctedExtract250 = 76 - DeductionIndB;

double correctedBreakout250 = 62 - DeductionIndB;

// for 500Hz octave band

double correctedSupply500 = 71 - DeductionIndB;

double correctedExtract500 = 63 - DeductionIndB;

double correctedBreakout500 = 47 - DeductionIndB;

// for 1000Hz octave band

double correctedSupply1000 = 72 - DeductionIndB;

double correctedExtract1000 = 64 - DeductionIndB;

double correctedBreakout1000 = 45 - DeductionIndB;

// for 2000Hz octave band

double correctedSupply2000 = 71 - DeductionIndB;

double correctedExtract2000 = 63 - DeductionIndB;

double correctedBreakout2000 = 44 - DeductionIndB;

// for 4000Hz octave band

double correctedSupply4000 = 66 - DeductionIndB;

double correctedExtract4000 = 53 - DeductionIndB;

double correctedBreakout4000 = 40 - DeductionIndB;

// for 8000Hz octave band

double correctedSupply8000 = 62 - DeductionIndB;

double correctedExtract8000 = 44 - DeductionIndB;

double correctedBreakout8000 = 29 - DeductionIndB;

string suitable = calculatingTheSuitability(correctedSupply63, correctedExtract63, correctedBreakout63, correctedSupply125, correctedExtract125, correctedBreakout125,

correctedSupply250, correctedExtract250, correctedBreakout250, correctedSupply500, correctedExtract500, correctedBreakout500,

correctedSupply1000, correctedExtract1000, correctedBreakout1000, correctedSupply2000, correctedExtract2000, correctedBreakout2000,

correctedSupply4000, correctedExtract4000, correctedBreakout4000, correctedSupply8000, correctedExtract8000, correctedBreakout8000);

return suitable;

}

string SuitabilityXBC55()

{

// y = -1,254.877471604040000x3 + 9.914530366717370x2 - 865.732222992418000x + 811.032050316677000

double A = -1254.877471604040000;

double B = 9.914530366717370 - a;

double C = -865.732222992418000;

double D = 811.032050316677000;

double FanFlowRate = SolvingACubic(A, B, C, D);

double DeductionIndB = FindDeductionIndB(FanFlowRate);

// for 63Hz octave band

double correctedSupply63 = 85 - DeductionIndB;

double correctedExtract63 = 82 - DeductionIndB;

double correctedBreakout63 = 72 - DeductionIndB;

// for 125Hz octave band

double correctedSupply125 = 80 - DeductionIndB;

double correctedExtract125 = 75 - DeductionIndB;

double correctedBreakout125 = 65 - DeductionIndB;

// for 250Hz octave band

double correctedSupply250 = 84 - DeductionIndB;

double correctedExtract250 = 75 - DeductionIndB;

double correctedBreakout250 = 61 - DeductionIndB;

// for 500Hz octave band

double correctedSupply500 = 71 - DeductionIndB;

double correctedExtract500 = 63 - DeductionIndB;

double correctedBreakout500 = 47 - DeductionIndB;

// for 1000Hz octave band

double correctedSupply1000 = 72 - DeductionIndB;

double correctedExtract1000 = 64 - DeductionIndB;

double correctedBreakout1000 = 45 - DeductionIndB;

// for 2000Hz octave band

double correctedSupply2000 = 70 - DeductionIndB;

double correctedExtract2000 = 62 - DeductionIndB;

double correctedBreakout2000 = 43 - DeductionIndB;

// for 4000Hz octave band

double correctedSupply4000 = 66 - DeductionIndB;

double correctedExtract4000 = 53 - DeductionIndB;

double correctedBreakout4000 = 40 - DeductionIndB;

// for 8000Hz octave band

double correctedSupply8000 = 61 - DeductionIndB;

double correctedExtract8000 = 43 - DeductionIndB;

double correctedBreakout8000 = 28 - DeductionIndB;

string suitable = calculatingTheSuitability(correctedSupply63, correctedExtract63, correctedBreakout63, correctedSupply125, correctedExtract125, correctedBreakout125,

correctedSupply250, correctedExtract250, correctedBreakout250, correctedSupply500, correctedExtract500, correctedBreakout500,

correctedSupply1000, correctedExtract1000, correctedBreakout1000, correctedSupply2000, correctedExtract2000, correctedBreakout2000,

correctedSupply4000, correctedExtract4000, correctedBreakout4000, correctedSupply8000, correctedExtract8000, correctedBreakout8000);

return suitable;

}

string SuitabilityXBC65()

{

// y = -165.000155283749000x3 - 315.762639557594000x2 - 427.811504641693000x + 616.069251925225000

double A = -165.000155283749000;

double B = -315.762639557594000 - a;

double C = -427.811504641693000;

double D = 616.069251925225000;

double FanFlowRate = SolvingACubic(A, B, C, D);

double DeductionIndB = FindDeductionIndB(FanFlowRate);

// for 63Hz octave band

double correctedSupply63 = 83 - DeductionIndB;

double correctedExtract63 = 81 - DeductionIndB;

double correctedBreakout63 = 71 - DeductionIndB;

// for 125Hz octave band

double correctedSupply125 = 85 - DeductionIndB;

double correctedExtract125 = 79 - DeductionIndB;

double correctedBreakout125 = 69 - DeductionIndB;

// for 250Hz octave band

double correctedSupply250 = 79 - DeductionIndB;

double correctedExtract250 = 70 - DeductionIndB;

double correctedBreakout250 = 56 - DeductionIndB;

// for 500Hz octave band

double correctedSupply500 = 74 - DeductionIndB;

double correctedExtract500 = 67 - DeductionIndB;

double correctedBreakout500 = 51 - DeductionIndB;

// for 1000Hz octave band

double correctedSupply1000 = 72 - DeductionIndB;

double correctedExtract1000 = 64 - DeductionIndB;

double correctedBreakout1000 = 45 - DeductionIndB;

// for 2000Hz octave band

double correctedSupply2000 = 68 - DeductionIndB;

double correctedExtract2000 = 60 - DeductionIndB;

double correctedBreakout2000 = 41 - DeductionIndB;

// for 4000Hz octave band

double correctedSupply4000 = 61 - DeductionIndB;

double correctedExtract4000 = 48 - DeductionIndB;

double correctedBreakout4000 = 35 - DeductionIndB;

// for 8000Hz octave band

double correctedSupply8000 = 54 - DeductionIndB;

double correctedExtract8000 = 35 - DeductionIndB;

double correctedBreakout8000 = 20 - DeductionIndB;

string suitable = calculatingTheSuitability(correctedSupply63, correctedExtract63, correctedBreakout63, correctedSupply125, correctedExtract125, correctedBreakout125,

correctedSupply250, correctedExtract250, correctedBreakout250, correctedSupply500, correctedExtract500, correctedBreakout500,

correctedSupply1000, correctedExtract1000, correctedBreakout1000, correctedSupply2000, correctedExtract2000, correctedBreakout2000,

correctedSupply4000, correctedExtract4000, correctedBreakout4000, correctedSupply8000, correctedExtract8000, correctedBreakout8000);

return suitable;

}

double SolvingACubic(double A, double B, double C, double D)

{

// solving the equation with Cardano's method to solve a cubic equation

double Q = ((3 \* A \* C) - (B \* B)) / (9 \* (A \* A));

double R = ((9 \* A \* B \* C) - (27 \* A \* A \* D) - (2 \* B \* B \* B)) / (54 \* A \* A \* A);

// M is S^3

double M = R + Math.Pow((Q \* Q \* Q) + (R \* R), 1 / 2);

double S = Math.Pow(M, 1 / 3);

// N is T^3

double N = R - Math.Pow((Q \* Q \* Q) + (R \* R), 1 / 2);

double T = Math.Pow(N, 1 / 3);

double FanFlowRate = S + T - (B / (3 \* A));

return FanFlowRate;

}

double FindDeductionIndB(double FanFlowRate)

{

// finding the deduction in decibels as a result of the speed

String query4 = "SELECT ReqAirVolRate FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd4 = new SqlCommand(query4, sqlCon);

sqlCmd4.CommandType = CommandType.Text;

sqlCmd4.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

double ReqAirVolRate = Convert.ToDouble(sqlCmd4.ExecuteScalar());

double speedRatio = ReqAirVolRate / FanFlowRate;

double DeductionIndB = 50 \* Math.Log10(speedRatio);

return DeductionIndB;

}

#endregion

#region calculatingSuitability

double findingSPL(double SWL, double distancevariable, double volumevariable, int octaveband)

{

double SPL = SWL - distancevariable - volumevariable - (3 \* Math.Log10(octaveband)) + 12;

return SPL;

}

double combineToCalcSPL(double Supply, double Extract, double Breakout)

{

double SPL = 10 \* Math.Log10(Math.Pow(10, (Supply / 10)) + Math.Pow(10, (Extract / 10)) + Math.Pow(10, (Breakout / 10)));

return SPL;

}

string calculatingTheSuitability(double correctedSupply63, double correctedExtract63, double correctedBreakout63, double correctedSupply125, double correctedExtract125, double correctedBreakout125,

double correctedSupply250, double correctedExtract250, double correctedBreakout250, double correctedSupply500, double correctedExtract500, double correctedBreakout500,

double correctedSupply1000, double correctedExtract1000, double correctedBreakout1000, double correctedSupply2000, double correctedExtract2000, double correctedBreakout2000,

double correctedSupply4000, double correctedExtract4000, double correctedBreakout4000, double correctedSupply8000, double correctedExtract8000, double correctedBreakout8000)

{

// working out the attenuation

string SupplyAttenuationType = FindSupplyAttenuationType();

string ExtractAttenuationType = FindExtractAttenuationType();

string BreakoutBarrierType = FindBreakoutBarrierType();

// The subroutines used in lines 683 - 690 are called from lines 923 - 1213.

// They each calculated the new 'attenuated' value, i.e. the new sound level after attenuators are added.

// Attenuators are devices that absorb sound.

double finalSupply63 = FindAttenuatedSupply63(SupplyAttenuationType, correctedSupply63);

double finalSupply125 = FindAttenuatedSupply125(SupplyAttenuationType, correctedSupply125);

double finalSupply250 = FindAttenuatedSupply250(SupplyAttenuationType, correctedSupply250);

double finalSupply500 = FindAttenuatedSupply500(SupplyAttenuationType, correctedSupply500);

double finalSupply1000 = FindAttenuatedSupply1000(SupplyAttenuationType, correctedSupply1000);

double finalSupply2000 = FindAttenuatedSupply2000(SupplyAttenuationType, correctedSupply2000);

double finalSupply4000 = FindAttenuatedSupply4000(SupplyAttenuationType, correctedSupply4000);

double finalSupply8000 = FindAttenuatedSupply8000(SupplyAttenuationType, correctedSupply8000);

double finalExtract63 = FindAttenuatedExtract63(ExtractAttenuationType, correctedExtract63);

double finalExtract125 = FindAttenuatedExtract125(ExtractAttenuationType, correctedExtract125);

double finalExtract250 = FindAttenuatedExtract250(ExtractAttenuationType, correctedExtract250);

double finalExtract500 = FindAttenuatedExtract500(ExtractAttenuationType, correctedExtract500);

double finalExtract1000 = FindAttenuatedExtract1000(ExtractAttenuationType, correctedExtract1000);

double finalExtract2000 = FindAttenuatedExtract2000(ExtractAttenuationType, correctedExtract2000);

double finalExtract4000 = FindAttenuatedExtract4000(ExtractAttenuationType, correctedExtract4000);

double finalExtract8000 = FindAttenuatedExtract8000(ExtractAttenuationType, correctedExtract8000);

double finalBreakout63 = FindAttenuatedBreakout63(BreakoutBarrierType, correctedBreakout63);

double finalBreakout125 = FindAttenuatedBreakout125(BreakoutBarrierType, correctedBreakout125);

double finalBreakout250 = FindAttenuatedBreakout250(BreakoutBarrierType, correctedBreakout250);

double finalBreakout500 = FindAttenuatedBreakout500(BreakoutBarrierType, correctedBreakout500);

double finalBreakout1000 = FindAttenuatedBreakout1000(BreakoutBarrierType, correctedBreakout1000);

double finalBreakout2000 = FindAttenuatedBreakout2000(BreakoutBarrierType, correctedBreakout2000);

double finalBreakout4000 = FindAttenuatedBreakout4000(BreakoutBarrierType, correctedBreakout4000);

double finalBreakout8000 = FindAttenuatedBreakout8000(BreakoutBarrierType, correctedBreakout8000);

// using Schultz's formula to calculate the sound pressure level from the sound power level at each octave band

double RoomVolume = FindRoomVolume();

double volumevariable = 5 \* Math.Log10(RoomVolume);

double FanDistanceToListener = FindDistancefromFantoListener();

double fandistancevariable = 10 \* Math.Log10(FanDistanceToListener);

double SupplyDistanceToListener = FindDistancefromSupplytoListener();

double supplydistancevariable = 10 \* Math.Log10(SupplyDistanceToListener);

double ExtractDistanceToListener = FindDistancefromExtracttoListener();

double extractdistancevariable = 10 \* Math.Log10(ExtractDistanceToListener);

// the subroutine findingSPL is on line 657. It uses Schultz's formula to determine the SPL for each of the labeled values.

double Supply63 = findingSPL(finalSupply63, supplydistancevariable, volumevariable, 63);

double Supply125 = findingSPL(finalSupply125, supplydistancevariable, volumevariable, 125);

double Supply250 = findingSPL(finalSupply250, supplydistancevariable, volumevariable, 250);

double Supply500 = findingSPL(finalSupply500, supplydistancevariable, volumevariable, 500);

double Supply1000 = findingSPL(finalSupply1000, supplydistancevariable, volumevariable, 1000);

double Supply2000 = findingSPL(finalSupply2000, supplydistancevariable, volumevariable, 2000);

double Supply4000 = findingSPL(finalSupply4000, supplydistancevariable, volumevariable, 4000);

double Supply8000 = findingSPL(finalSupply8000, supplydistancevariable, volumevariable, 8000);

double Extract63 = findingSPL(finalExtract63, extractdistancevariable, volumevariable, 63);

double Extract125 = findingSPL(finalExtract125, extractdistancevariable, volumevariable, 125);

double Extract250 = findingSPL(finalExtract250, extractdistancevariable, volumevariable, 250);

double Extract500 = findingSPL(finalExtract500, extractdistancevariable, volumevariable, 500);

double Extract1000 = findingSPL(finalExtract1000, extractdistancevariable, volumevariable, 1000);

double Extract2000 = findingSPL(finalExtract2000, extractdistancevariable, volumevariable, 2000);

double Extract4000 = findingSPL(finalExtract4000, extractdistancevariable, volumevariable, 4000);

double Extract8000 = findingSPL(finalExtract8000, extractdistancevariable, volumevariable, 8000);

double Breakout63 = findingSPL(finalBreakout63, fandistancevariable, volumevariable, 63);

double Breakout125 = findingSPL(finalBreakout125, fandistancevariable, volumevariable, 125);

double Breakout250 = findingSPL(finalBreakout250, fandistancevariable, volumevariable, 250);

double Breakout500 = findingSPL(finalBreakout500, fandistancevariable, volumevariable, 500);

double Breakout1000 = findingSPL(finalBreakout1000, fandistancevariable, volumevariable, 1000);

double Breakout2000 = findingSPL(finalBreakout2000, fandistancevariable, volumevariable, 2000);

double Breakout4000 = findingSPL(finalBreakout4000, fandistancevariable, volumevariable, 4000);

double Breakout8000 = findingSPL(finalBreakout8000, fandistancevariable, volumevariable, 8000);

// combining to give SPL values

// the subroutine combineToCalcSPL is on line 663. This uses logarithmic addition to combine the 3 parameters.

double SPL63 = combineToCalcSPL(Supply63, Extract63, Breakout63);

double SPL125 = combineToCalcSPL(Supply125, Extract125, Breakout125);

double SPL250 = combineToCalcSPL(Supply250, Extract250, Breakout250);

double SPL500 = combineToCalcSPL(Supply500, Extract500, Breakout500);

double SPL1000 = combineToCalcSPL(Supply1000, Extract1000, Breakout1000);

double SPL2000 = combineToCalcSPL(Supply2000, Extract2000, Breakout2000);

double SPL4000 = combineToCalcSPL(Supply4000, Extract4000, Breakout4000);

double SPL8000 = combineToCalcSPL(Supply8000, Extract8000, Breakout8000);

string overallsuitability;

// comparing the SPL values to the NR values

double NRVal = FindingNRValueOfRoom(); // this subroutine finds the NR value stored in the database.

if (NRVal == 25)

{

double SWL63 = 55;

double SWL125 = 44;

double SWL250 = 35;

double SWL500 = 29;

double SWL1000 = 25;

double SWL2000 = 22;

double SWL4000 = 20;

double SWL8000 = 18;

bool suitable63 = FindIfSuitable(SPL63, SWL63);

bool suitable125 = FindIfSuitable(SPL125, SWL125);

bool suitable250 = FindIfSuitable(SPL250, SWL250);

bool suitable500 = FindIfSuitable(SPL500, SWL500);

bool suitable1000 = FindIfSuitable(SPL1000, SWL1000);

bool suitable2000 = FindIfSuitable(SPL2000, SWL2000);

bool suitable4000 = FindIfSuitable(SPL4000, SWL4000);

bool suitable8000 = FindIfSuitable(SPL8000, SWL8000);

overallsuitability = FindOverallSuitability(suitable63, suitable125, suitable250, suitable500, suitable1000, suitable2000, suitable4000, suitable8000);

return overallsuitability;

}

else if(NRVal == 30)

{

double SWL63 = 59;

double SWL125 = 48;

double SWL250 = 40;

double SWL500 = 34;

double SWL1000 = 30;

double SWL2000 = 27;

double SWL4000 = 25;

double SWL8000 = 23;

bool suitable63 = FindIfSuitable(SPL63, SWL63);

bool suitable125 = FindIfSuitable(SPL125, SWL125);

bool suitable250 = FindIfSuitable(SPL250, SWL250);

bool suitable500 = FindIfSuitable(SPL500, SWL500);

bool suitable1000 = FindIfSuitable(SPL1000, SWL1000);

bool suitable2000 = FindIfSuitable(SPL2000, SWL2000);

bool suitable4000 = FindIfSuitable(SPL4000, SWL4000);

bool suitable8000 = FindIfSuitable(SPL8000, SWL8000);

overallsuitability = FindOverallSuitability(suitable63, suitable125, suitable250, suitable500, suitable1000, suitable2000, suitable4000, suitable8000);

return overallsuitability;

}

else if(NRVal == 35)

{

double SWL63 = 63;

double SWL125 = 52;

double SWL250 = 45;

double SWL500 = 39;

double SWL1000 = 35;

double SWL2000 = 32;

double SWL4000 = 30;

double SWL8000 = 28;

bool suitable63 = FindIfSuitable(SPL63, SWL63);

bool suitable125 = FindIfSuitable(SPL125, SWL125);

bool suitable250 = FindIfSuitable(SPL250, SWL250);

bool suitable500 = FindIfSuitable(SPL500, SWL500);

bool suitable1000 = FindIfSuitable(SPL1000, SWL1000);

bool suitable2000 = FindIfSuitable(SPL2000, SWL2000);

bool suitable4000 = FindIfSuitable(SPL4000, SWL4000);

bool suitable8000 = FindIfSuitable(SPL8000, SWL8000);

overallsuitability = FindOverallSuitability(suitable63, suitable125, suitable250, suitable500, suitable1000, suitable2000, suitable4000, suitable8000);

return overallsuitability;

}

else if (NRVal == 40)

{

double SWL63 = 67;

double SWL125 = 57;

double SWL250 = 49;

double SWL500 = 44;

double SWL1000 = 40;

double SWL2000 = 37;

double SWL4000 = 35;

double SWL8000 = 33;

bool suitable63 = FindIfSuitable(SPL63, SWL63);

bool suitable125 = FindIfSuitable(SPL125, SWL125);

bool suitable250 = FindIfSuitable(SPL250, SWL250);

bool suitable500 = FindIfSuitable(SPL500, SWL500);

bool suitable1000 = FindIfSuitable(SPL1000, SWL1000);

bool suitable2000 = FindIfSuitable(SPL2000, SWL2000);

bool suitable4000 = FindIfSuitable(SPL4000, SWL4000);

bool suitable8000 = FindIfSuitable(SPL8000, SWL8000);

overallsuitability = FindOverallSuitability(suitable63, suitable125, suitable250, suitable500, suitable1000, suitable2000, suitable4000, suitable8000);

return overallsuitability;

}

else if (NRVal == 45)

{

double SWL63 = 71;

double SWL125 = 61;

double SWL250 = 54;

double SWL500 = 48;

double SWL1000 = 45;

double SWL2000 = 42;

double SWL4000 = 40;

double SWL8000 = 38;

bool suitable63 = FindIfSuitable(SPL63, SWL63);

bool suitable125 = FindIfSuitable(SPL125, SWL125);

bool suitable250 = FindIfSuitable(SPL250, SWL250);

bool suitable500 = FindIfSuitable(SPL500, SWL500);

bool suitable1000 = FindIfSuitable(SPL1000, SWL1000);

bool suitable2000 = FindIfSuitable(SPL2000, SWL2000);

bool suitable4000 = FindIfSuitable(SPL4000, SWL4000);

bool suitable8000 = FindIfSuitable(SPL8000, SWL8000);

overallsuitability = FindOverallSuitability(suitable63, suitable125, suitable250, suitable500, suitable1000, suitable2000, suitable4000, suitable8000);

return overallsuitability;

}

else

{

overallsuitability = "No";

return overallsuitability;

}

}

// Compares the SPL and SWL to see if the fan is suitable at each octave band.

bool FindIfSuitable(double SPLCalc, double NRSPL)

{

if (SPLCalc >= NRSPL)

{

bool suitable = true;

return suitable;

}

else

{

bool suitable = false;

return suitable;

}

}

// If the fan is suitable at all the octave bands, then the fan is suitable overall.

// Otherwise, it is not suitable.

string FindOverallSuitability(bool suitable63, bool suitable125, bool suitable250, bool suitable500, bool suitable1000, bool suitable2000, bool suitable4000, bool suitable8000)

{

if ((suitable63 == true) && (suitable125 == true) && (suitable250 == true) && (suitable500 == true) && (suitable1000 == true) && (suitable2000 == true) && (suitable4000 == true) && (suitable8000 == true))

{

string overallsuitability = "Yes";

return overallsuitability;

}

else

{

string overallsuitability = "No";

return overallsuitability;

}

}

#endregion

#region finalSupply

// find the corrected values for the attenuators and acoustic barrier

double FindAttenuatedSupply63(string SupplyAttenuationType, double correctedSupply63)

{

if(SupplyAttenuationType == "Short")

{

double attenuatedSupply63 = correctedSupply63 - 5;

double hydraulicArea = FindHydraulicArea();

double endReflection63 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 63)), 1.88)));

double finalSupply63 = FindFinal63(attenuatedSupply63, endReflection63);

return finalSupply63;

}

else if (SupplyAttenuationType == "Medium")

{

double attenuatedSupply63 = correctedSupply63 - 7;

double hydraulicArea = FindHydraulicArea();

double endReflection63 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 63)), 1.88)));

double finalSupply63 = FindFinal63(attenuatedSupply63, endReflection63);

return finalSupply63;

}

else if (SupplyAttenuationType == "Long")

{

double attenuatedSupply63 = correctedSupply63 - 9;

double hydraulicArea = FindHydraulicArea();

double endReflection63 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 63)), 1.88)));

double finalSupply63 = FindFinal63(attenuatedSupply63, endReflection63);

return finalSupply63;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedSupply125(string SupplyAttenuationType, double correctedSupply125)

{

if (SupplyAttenuationType == "Short")

{

double attenuatedSupply125 = correctedSupply125 - 8;

double hydraulicArea = FindHydraulicArea();

double endReflection125 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 125)), 1.88)));

double finalSupply125 = FindFinal125(attenuatedSupply125, endReflection125);

return finalSupply125;

}

else if (SupplyAttenuationType == "Medium")

{

double attenuatedSupply125 = correctedSupply125 - 10;

double hydraulicArea = FindHydraulicArea();

double endReflection125 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 125)), 1.88)));

double finalSupply125 = FindFinal125(attenuatedSupply125, endReflection125);

return finalSupply125;

}

else if (SupplyAttenuationType == "Long")

{

double attenuatedSupply125 = correctedSupply125 - 13;

double hydraulicArea = FindHydraulicArea();

double endReflection125 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 125)), 1.88)));

double finalSupply125 = FindFinal125(attenuatedSupply125, endReflection125);

return finalSupply125;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedSupply250(string SupplyAttenuationType, double correctedSupply250)

{

if (SupplyAttenuationType == "Short")

{

double attenuatedSupply250 = correctedSupply250 - 15;

double hydraulicArea = FindHydraulicArea();

double endReflection250 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 250)), 1.88)));

double finalSupply250 = FindFinal250(attenuatedSupply250, endReflection250);

return finalSupply250;

}

else if (SupplyAttenuationType == "Medium")

{

double attenuatedSupply250 = correctedSupply250 - 18;

double hydraulicArea = FindHydraulicArea();

double endReflection250 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 250)), 1.88)));

double finalSupply250 = FindFinal250(attenuatedSupply250, endReflection250);

return finalSupply250;

}

else if (SupplyAttenuationType == "Long")

{

double attenuatedSupply250 = correctedSupply250 - 23;

double hydraulicArea = FindHydraulicArea();

double endReflection250 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 250)), 1.88)));

double finalSupply250 = FindFinal250(attenuatedSupply250, endReflection250);

return finalSupply250;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedSupply500(string SupplyAttenuationType, double correctedSupply500)

{

if (SupplyAttenuationType == "Short")

{

double attenuatedSupply500 = correctedSupply500 - 30;

double hydraulicArea = FindHydraulicArea();

double endReflection500 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 500)), 1.88)));

double finalSupply500 = FindFinal500(attenuatedSupply500, endReflection500);

return finalSupply500;

}

else if (SupplyAttenuationType == "Medium")

{

double attenuatedSupply500 = correctedSupply500 - 36;

double hydraulicArea = FindHydraulicArea();

double endReflection500 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 500)), 1.88)));

double finalSupply500 = FindFinal500(attenuatedSupply500, endReflection500);

return finalSupply500;

}

else if (SupplyAttenuationType == "Long")

{

double attenuatedSupply500 = correctedSupply500 - 42;

double hydraulicArea = FindHydraulicArea();

double endReflection500 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 500)), 1.88)));

double finalSupply500 = FindFinal500(attenuatedSupply500, endReflection500);

return finalSupply500;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedSupply1000(string SupplyAttenuationType, double correctedSupply1000)

{

if (SupplyAttenuationType == "Short")

{

double attenuatedSupply1000 = correctedSupply1000 - 41;

double hydraulicArea = FindHydraulicArea();

double endReflection1000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 1000)), 1.88)));

double finalSupply1000 = FindFinal1000(attenuatedSupply1000, endReflection1000);

return finalSupply1000;

}

else if (SupplyAttenuationType == "Medium")

{

double attenuatedSupply1000 = correctedSupply1000 - 51;

double hydraulicArea = FindHydraulicArea();

double endReflection1000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 1000)), 1.88)));

double finalSupply1000 = FindFinal1000(attenuatedSupply1000, endReflection1000);

return finalSupply1000;

}

else if (SupplyAttenuationType == "Long")

{

double attenuatedSupply1000 = correctedSupply1000 - 55;

double hydraulicArea = FindHydraulicArea();

double endReflection1000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 1000)), 1.88)));

double finalSupply1000 = FindFinal1000(attenuatedSupply1000, endReflection1000);

return finalSupply1000;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedSupply2000(string SupplyAttenuationType, double correctedSupply2000)

{

if (SupplyAttenuationType == "Short")

{

double attenuatedSupply2000 = correctedSupply2000 - 31;

double hydraulicArea = FindHydraulicArea();

double endReflection2000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 2000)), 1.88)));

double finalSupply2000 = FindFinal2000(attenuatedSupply2000, endReflection2000);

return finalSupply2000;

}

else if (SupplyAttenuationType == "Medium")

{

double attenuatedSupply2000 = correctedSupply2000 - 39;

double hydraulicArea = FindHydraulicArea();

double endReflection2000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 2000)), 1.88)));

double finalSupply2000 = FindFinal2000(attenuatedSupply2000, endReflection2000);

return finalSupply2000;

}

else if (SupplyAttenuationType == "Long")

{

double attenuatedSupply2000 = correctedSupply2000 - 49;

double hydraulicArea = FindHydraulicArea();

double endReflection2000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 2000)), 1.88)));

double finalSupply2000 = FindFinal2000(attenuatedSupply2000, endReflection2000);

return finalSupply2000;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedSupply4000(string SupplyAttenuationType, double correctedSupply4000)

{

if (SupplyAttenuationType == "Short")

{

double attenuatedSupply4000 = correctedSupply4000 - 21;

double hydraulicArea = FindHydraulicArea();

double endReflection4000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 4000)), 1.88)));

double finalSupply4000 = FindFinal4000(attenuatedSupply4000, endReflection4000);

return finalSupply4000;

}

else if (SupplyAttenuationType == "Medium")

{

double attenuatedSupply4000 = correctedSupply4000 - 26;

double hydraulicArea = FindHydraulicArea();

double endReflection4000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 4000)), 1.88)));

double finalSupply4000 = FindFinal4000(attenuatedSupply4000, endReflection4000);

return finalSupply4000;

}

else if (SupplyAttenuationType == "Long")

{

double attenuatedSupply4000 = correctedSupply4000 - 32;

double hydraulicArea = FindHydraulicArea();

double endReflection4000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 4000)), 1.88)));

double finalSupply4000 = FindFinal4000(attenuatedSupply4000, endReflection4000);

return finalSupply4000;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedSupply8000(string SupplyAttenuationType, double correctedSupply8000)

{

if (SupplyAttenuationType == "Short")

{

double attenuatedSupply8000 = correctedSupply8000 - 16;

double hydraulicArea = FindHydraulicArea(); // called from line 1723.

// calculating end reflection - calculated with set formula.

double endReflection8000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 8000)), 1.88)));

double finalSupply8000 = FindFinal8000(attenuatedSupply8000, endReflection8000);

return finalSupply8000;

}

else if (SupplyAttenuationType == "Medium")

{

double attenuatedSupply8000 = correctedSupply8000 - 20;

double hydraulicArea = FindHydraulicArea();

// calculating end reflection

double endReflection8000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 8000)), 1.88)));

double finalSupply8000 = FindFinal8000(attenuatedSupply8000, endReflection8000);

return finalSupply8000;

}

else if (SupplyAttenuationType == "Long")

{

double attenuatedSupply8000 = correctedSupply8000 - 25;

double hydraulicArea = FindHydraulicArea();

double endReflection8000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 8000)), 1.88)));

double finalSupply8000 = FindFinal8000(attenuatedSupply8000, endReflection8000);

return finalSupply8000;

}

else

{

double number = 1;

return number;

}

}

#endregion

#region finalExtract

double FindAttenuatedExtract63(string ExtractAttenuationType, double correctedExtract63)

{

if (ExtractAttenuationType == "Short")

{

double attenuatedExtract63 = correctedExtract63 - 4;

double hydraulicArea = FindHydraulicArea();

double endReflection63 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 63)), 1.88)));

double finalExtract63 = FindFinal63(attenuatedExtract63, endReflection63);

return finalExtract63;

}

else if (ExtractAttenuationType == "Medium")

{

double attenuatedExtract63 = correctedExtract63 - 5;

double hydraulicArea = FindHydraulicArea();

double endReflection63 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 63)), 1.88)));

double finalExtract63 = FindFinal63(attenuatedExtract63, endReflection63);

return finalExtract63;

}

else if (ExtractAttenuationType == "Long")

{

double attenuatedExtract63 = correctedExtract63 - 6;

double hydraulicArea = FindHydraulicArea();

double endReflection63 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 63)), 1.88)));

double finalExtract63 = FindFinal63(attenuatedExtract63, endReflection63);

return finalExtract63;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedExtract125(string ExtractAttenuationType, double correctedExtract125)

{

if (ExtractAttenuationType == "Short")

{

double attenuatedExtract125 = correctedExtract125 - 4;

double hydraulicArea = FindHydraulicArea();

double endReflection125 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 125)), 1.88)));

double finalExtract125 = FindFinal125(attenuatedExtract125, endReflection125);

return finalExtract125;

}

else if (ExtractAttenuationType == "Medium")

{

double attenuatedExtract125 = correctedExtract125 - 6;

double hydraulicArea = FindHydraulicArea();

double endReflection125 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 125)), 1.88)));

double finalExtract125 = FindFinal125(attenuatedExtract125, endReflection125);

return finalExtract125;

}

else if (ExtractAttenuationType == "Long")

{

double attenuatedExtract125 = correctedExtract125 - 8;

double hydraulicArea = FindHydraulicArea();

double endReflection125 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 125)), 1.88)));

double finalExtract125 = FindFinal125(attenuatedExtract125, endReflection125);

return finalExtract125;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedExtract250(string ExtractAttenuationType, double correctedExtract250)

{

if (ExtractAttenuationType == "Short")

{

double attenuatedExtract250 = correctedExtract250 - 10;

double hydraulicArea = FindHydraulicArea();

double endReflection250 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 250)), 1.88)));

double finalExtract250 = FindFinal250(attenuatedExtract250, endReflection250);

return finalExtract250;

}

else if (ExtractAttenuationType == "Medium")

{

double attenuatedExtract250 = correctedExtract250 - 12;

double hydraulicArea = FindHydraulicArea();

double endReflection250 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 250)), 1.88)));

double finalExtract250 = FindFinal250(attenuatedExtract250, endReflection250);

return finalExtract250;

}

else if (ExtractAttenuationType == "Long")

{

double attenuatedExtract250 = correctedExtract250 - 15;

double hydraulicArea = FindHydraulicArea();

double endReflection250 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 250)), 1.88)));

double finalExtract250 = FindFinal250(attenuatedExtract250, endReflection250);

return finalExtract250;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedExtract500(string ExtractAttenuationType, double correctedExtract500)

{

if (ExtractAttenuationType == "Short")

{

double attenuatedExtract500 = correctedExtract500 - 22;

double hydraulicArea = FindHydraulicArea();

double endReflection500 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 500)), 1.88)));

double finalExtract500 = FindFinal500(attenuatedExtract500, endReflection500);

return finalExtract500;

}

else if (ExtractAttenuationType == "Medium")

{

double attenuatedExtract500 = correctedExtract500 - 27;

double hydraulicArea = FindHydraulicArea();

double endReflection500 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 500)), 1.88)));

double finalExtract500 = FindFinal500(attenuatedExtract500, endReflection500);

return finalExtract500;

}

else if (ExtractAttenuationType == "Long")

{

double attenuatedExtract500 = correctedExtract500 - 33;

double hydraulicArea = FindHydraulicArea();

double endReflection500 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 500)), 1.88)));

double finalExtract500 = FindFinal500(attenuatedExtract500, endReflection500);

return finalExtract500;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedExtract1000(string ExtractAttenuationType, double correctedExtract1000)

{

if (ExtractAttenuationType == "Short")

{

double attenuatedExtract1000 = correctedExtract1000 - 26;

double hydraulicArea = FindHydraulicArea();

double endReflection1000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 1000)), 1.88)));

double finalExtract1000 = FindFinal1000(attenuatedExtract1000, endReflection1000);

return finalExtract1000;

}

else if (ExtractAttenuationType == "Medium")

{

double attenuatedExtract1000 = correctedExtract1000 - 34;

double hydraulicArea = FindHydraulicArea();

double endReflection1000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 1000)), 1.88)));

double finalExtract1000 = FindFinal1000(attenuatedExtract1000, endReflection1000);

return finalExtract1000;

}

else if (ExtractAttenuationType == "Long")

{

double attenuatedExtract1000 = correctedExtract1000 - 43;

double hydraulicArea = FindHydraulicArea();

double endReflection1000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 1000)), 1.88)));

double finalExtract1000 = FindFinal1000(attenuatedExtract1000, endReflection1000);

return finalExtract1000;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedExtract2000(string ExtractAttenuationType, double correctedExtract2000)

{

if (ExtractAttenuationType == "Short")

{

double attenuatedExtract2000 = correctedExtract2000 - 15;

double hydraulicArea = FindHydraulicArea();

double endReflection2000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 2000)), 1.88)));

double finalExtract2000 = FindFinal2000(attenuatedExtract2000, endReflection2000);

return finalExtract2000;

}

else if (ExtractAttenuationType == "Medium")

{

double attenuatedExtract2000 = correctedExtract2000 - 20;

double hydraulicArea = FindHydraulicArea();

double endReflection2000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 2000)), 1.88)));

double finalExtract2000 = FindFinal2000(attenuatedExtract2000, endReflection2000);

return finalExtract2000;

}

else if (ExtractAttenuationType == "Long")

{

double attenuatedExtract2000 = correctedExtract2000 - 25;

double hydraulicArea = FindHydraulicArea();

double endReflection2000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 2000)), 1.88)));

double finalExtract2000 = FindFinal2000(attenuatedExtract2000, endReflection2000);

return finalExtract2000;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedExtract4000(string ExtractAttenuationType, double correctedExtract4000)

{

if (ExtractAttenuationType == "Short")

{

double attenuatedExtract4000 = correctedExtract4000 - 10;

double hydraulicArea = FindHydraulicArea();

double endReflection4000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 4000)), 1.88)));

double finalExtract4000 = FindFinal4000(attenuatedExtract4000, endReflection4000);

return finalExtract4000;

}

else if (ExtractAttenuationType == "Medium")

{

double attenuatedExtract4000 = correctedExtract4000 - 13;

double hydraulicArea = FindHydraulicArea();

double endReflection4000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 4000)), 1.88)));

double finalExtract4000 = FindFinal4000(attenuatedExtract4000, endReflection4000);

return finalExtract4000;

}

else if (ExtractAttenuationType == "Long")

{

double attenuatedExtract4000 = correctedExtract4000 - 15;

double hydraulicArea = FindHydraulicArea();

double endReflection4000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 4000)), 1.88)));

double finalExtract4000 = FindFinal4000(attenuatedExtract4000, endReflection4000);

return finalExtract4000;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedExtract8000(string ExtractAttenuationType, double correctedExtract8000)

{

if (ExtractAttenuationType == "Short")

{

double attenuatedExtract8000 = correctedExtract8000 - 8;

double hydraulicArea = FindHydraulicArea();

// calculating end reflection

double endReflection8000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 8000)), 1.88)));

double finalExtract8000 = FindFinal8000(attenuatedExtract8000, endReflection8000);

return finalExtract8000;

}

else if (ExtractAttenuationType == "Medium")

{

double attenuatedExtract8000 = correctedExtract8000 - 9;

double hydraulicArea = FindHydraulicArea();

// calculating end reflection

double endReflection8000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 8000)), 1.88)));

double finalExtract8000 = FindFinal8000(attenuatedExtract8000, endReflection8000);

return finalExtract8000;

}

else if (ExtractAttenuationType == "Long")

{

double attenuatedExtract8000 = correctedExtract8000 - 11;

double hydraulicArea = FindHydraulicArea();

// calculating end reflection

double endReflection8000 = 10 \* (Math.Log10(1 + Math.Pow(((0.8 \* 344) / (3.142 \* hydraulicArea \* 8000)), 1.88)));

double finalExtract8000 = FindFinal8000(attenuatedExtract8000, endReflection8000);

return finalExtract8000;

}

else

{

double number = 1;

return number;

}

}

#endregion

#region finalBreakout

double FindAttenuatedBreakout63(string BreakoutBarrierType, double correctedBreakout63)

{

if (BreakoutBarrierType == "Basic")

{

double finalBreakout63 = correctedBreakout63 - 2;

return finalBreakout63;

}

else if (BreakoutBarrierType == "Acoustic Grade")

{

double finalBreakout63 = correctedBreakout63 - 7;

return finalBreakout63;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedBreakout125(string BreakoutBarrierType, double correctedBreakout125)

{

if (BreakoutBarrierType == "Basic")

{

double finalBreakout125 = correctedBreakout125 - 3;

return finalBreakout125;

}

else if (BreakoutBarrierType == "Acoustic Grade")

{

double finalBreakout125 = correctedBreakout125 - 9;

return finalBreakout125;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedBreakout250(string BreakoutBarrierType, double correctedBreakout250)

{

if (BreakoutBarrierType == "Basic")

{

double finalBreakout250 = correctedBreakout250 - 3;

return finalBreakout250;

}

else if (BreakoutBarrierType == "Acoustic Grade")

{

double finalBreakout250 = correctedBreakout250 - 10;

return finalBreakout250;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedBreakout500(string BreakoutBarrierType, double correctedBreakout500)

{

if (BreakoutBarrierType == "Basic")

{

double finalBreakout500 = correctedBreakout500 - 5;

return finalBreakout500;

}

else if (BreakoutBarrierType == "Acoustic Grade")

{

double finalBreakout500 = correctedBreakout500 - 15;

return finalBreakout500;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedBreakout1000(string BreakoutBarrierType, double correctedBreakout1000)

{

if (BreakoutBarrierType == "Basic")

{

double finalBreakout1000 = correctedBreakout1000 - 5;

return finalBreakout1000;

}

else if (BreakoutBarrierType == "Acoustic Grade")

{

double finalBreakout1000 = correctedBreakout1000 - 20;

return finalBreakout1000;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedBreakout2000(string BreakoutBarrierType, double correctedBreakout2000)

{

if (BreakoutBarrierType == "Basic")

{

double finalBreakout2000 = correctedBreakout2000 - 8;

return finalBreakout2000;

}

else if (BreakoutBarrierType == "Acoustic Grade")

{

double finalBreakout2000 = correctedBreakout2000 - 24;

return finalBreakout2000;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedBreakout4000(string BreakoutBarrierType, double correctedBreakout4000)

{

if (BreakoutBarrierType == "Basic")

{

double finalBreakout4000 = correctedBreakout4000 - 9;

return finalBreakout4000;

}

else if (BreakoutBarrierType == "Acoustic Grade")

{

double finalBreakout4000 = correctedBreakout4000 - 27;

return finalBreakout4000;

}

else

{

double number = 1;

return number;

}

}

double FindAttenuatedBreakout8000(string BreakoutBarrierType, double correctedBreakout8000)

{

if (BreakoutBarrierType == "Basic")

{

double finalBreakout8000 = correctedBreakout8000 - 8;

return finalBreakout8000;

}

else if (BreakoutBarrierType == "Acoustic Grade")

{

double finalBreakout8000 = correctedBreakout8000 - 24;

return finalBreakout8000;

}

else

{

double number = 1;

return number;

}

}

#endregion

#region findFinal

double FindFinal63(double attenuatedExtract63, double endReflection63)

{

double finalExtract63 = attenuatedExtract63 - endReflection63;

return finalExtract63;

}

double FindFinal125(double attenuatedExtract63, double endReflection63)

{

double finalExtract63 = attenuatedExtract63 - endReflection63;

return finalExtract63;

}

double FindFinal250(double attenuatedExtract63, double endReflection63)

{

double finalExtract63 = attenuatedExtract63 - endReflection63;

return finalExtract63;

}

double FindFinal500(double attenuatedExtract63, double endReflection63)

{

double finalExtract63 = attenuatedExtract63 - endReflection63;

return finalExtract63;

}

double FindFinal1000(double attenuatedExtract63, double endReflection63)

{

double finalExtract63 = attenuatedExtract63 - endReflection63;

return finalExtract63;

}

double FindFinal2000(double attenuatedExtract63, double endReflection63)

{

double finalExtract63 = attenuatedExtract63 - endReflection63;

return finalExtract63;

}

double FindFinal4000(double attenuatedExtract63, double endReflection63)

{

double finalExtract63 = attenuatedExtract63 - endReflection63;

return finalExtract63;

}

double FindFinal8000(double attenuatedExtract63, double endReflection63)

{

double finalExtract63 = attenuatedExtract63 - endReflection63;

return finalExtract63;

}

#endregion

// finding the 'flush' or end reflection from the room terminal side length

double FindHydraulicArea()

{

// finding a side length of the room terminal

String query4 = "SELECT RoomTerminalSideLength FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd4 = new SqlCommand(query4, sqlCon);

sqlCmd4.CommandType = CommandType.Text;

sqlCmd4.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

double RoomTerminalSideLength = Convert.ToDouble(sqlCmd4.ExecuteScalar());

if(Units == "Feet")

{

RoomTerminalSideLength = RoomTerminalSideLength \* 3.28;

}

// calculating the hydraulic area of the terminal

double hydraulicArea = Math.Pow(((4 \* RoomTerminalSideLength \* RoomTerminalSideLength) / 3.142), 1 / 2);

return hydraulicArea;

}

#region findingDistances

// finding commonly used distances from database

String queryXCoordListener = "SELECT XCoordListener FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd6 = new SqlCommand(queryXCoordListener, sqlCon);

sqlCmd6.CommandType = CommandType.Text;

sqlCmd6.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string XCoordListenerstr = Convert.ToString(sqlCmd6.ExecuteScalar());

double XCoordListener = Convert.ToDouble(XCoordListenerstr);

String queryYCoordListener = "SELECT YCoordListener FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd7 = new SqlCommand(queryYCoordListener, sqlCon);

sqlCmd7.CommandType = CommandType.Text;

sqlCmd7.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string YCoordListenerstr = Convert.ToString(sqlCmd7.ExecuteScalar());

double YCoordListener = Convert.ToDouble(YCoordListenerstr);

String queryRoomLength = "SELECT RoomLength FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd8 = new SqlCommand(queryRoomLength, sqlCon);

sqlCmd8.CommandType = CommandType.Text;

sqlCmd8.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

double RoomLength = Convert.ToDouble(sqlCmd8.ExecuteScalar());

String queryRoomWidth = "SELECT RoomWidth FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd9 = new SqlCommand(queryRoomWidth, sqlCon);

sqlCmd9.CommandType = CommandType.Text;

sqlCmd9.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

double RoomWidth = Convert.ToDouble(sqlCmd9.ExecuteScalar());

double FindRoomVolume()

{

String queryRoomHeight = "SELECT RoomHeight FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd10 = new SqlCommand(queryRoomHeight, sqlCon);

sqlCmd10.CommandType = CommandType.Text;

sqlCmd10.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

double RoomHeight = Convert.ToDouble(sqlCmd10.ExecuteScalar());

// converting units if specified by user

if (Units == "Feet")

{

RoomLength = RoomLength \* 3.28;

RoomWidth = RoomWidth \* 3.28;

}

double RoomVolume = RoomLength \* RoomWidth \* RoomHeight;

return RoomVolume;

}

double FindDistancefromFantoListener()

{

// finding necessary variables from the database

String queryXCoordFan = "SELECT XCoordFan FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd4 = new SqlCommand(queryXCoordFan, sqlCon);

sqlCmd4.CommandType = CommandType.Text;

sqlCmd4.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string XCoordFanstr = Convert.ToString(sqlCmd4.ExecuteScalar());

double XCoordFan = Convert.ToDouble(XCoordFanstr);

String queryYCoordFan = "SELECT YCoordFan FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd5 = new SqlCommand(queryYCoordFan, sqlCon);

sqlCmd5.CommandType = CommandType.Text;

sqlCmd5.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string YCoordFanstr = Convert.ToString(sqlCmd5.ExecuteScalar());

double YCoordFan = Convert.ToDouble(YCoordFanstr);

if (Units == "Feet")

{

RoomLength = RoomLength \* 3.28;

RoomWidth = RoomWidth \* 3.28;

}

double FanToListener = calculatingDistancestoListener(XCoordFan, YCoordFan);

return FanToListener;

}

double FindDistancefromSupplytoListener()

{

// finding necessary variables from the database

String queryXCoordS = "SELECT XCoordSupply FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd4 = new SqlCommand(queryXCoordS, sqlCon);

sqlCmd4.CommandType = CommandType.Text;

sqlCmd4.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string XCoordSstr = Convert.ToString(sqlCmd4.ExecuteScalar());

double XCoordS = Convert.ToDouble(XCoordSstr);

String queryYCoordS = "SELECT YCoordSupply FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd5 = new SqlCommand(queryYCoordS, sqlCon);

sqlCmd5.CommandType = CommandType.Text;

sqlCmd5.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string YCoordSstr = Convert.ToString(sqlCmd5.ExecuteScalar());

double YCoordS = Convert.ToDouble(YCoordSstr);

if (Units == "Feet")

{

RoomLength = RoomLength \* 3.28;

RoomWidth = RoomWidth \* 3.28;

}

double SupplyToListener = calculatingDistancestoListener(XCoordS, YCoordS);

return SupplyToListener;

}

double FindDistancefromExtracttoListener()

{

// finding necessary variables from the database

String queryXCoordE = "SELECT XCoordExtract FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd4 = new SqlCommand(queryXCoordE, sqlCon);

sqlCmd4.CommandType = CommandType.Text;

sqlCmd4.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string XCoordEstr = Convert.ToString(sqlCmd4.ExecuteScalar());

double XCoordE = Convert.ToDouble(XCoordEstr);

String queryYCoordE = "SELECT YCoordExtract FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd5 = new SqlCommand(queryYCoordE, sqlCon);

sqlCmd5.CommandType = CommandType.Text;

sqlCmd5.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string YCoordEstr = Convert.ToString(sqlCmd5.ExecuteScalar());

double YCoordE = Convert.ToDouble(YCoordEstr);

if (Units == "Feet")

{

RoomLength = RoomLength \* 3.28;

RoomWidth = RoomWidth \* 3.28;

}

double ExtractToListener = calculatingDistancestoListener(XCoordE, YCoordE);

return ExtractToListener;

}

double calculatingDistancestoListener(double XCoord, double YCoord)

{

// calculating difference between x and y coordinates of the fan and the listener

double Xdifference = XCoord - XCoordListener;

double AbsXdifference = Math.Abs(Xdifference);

double Ydifference = YCoord - YCoordListener;

double AbsYdifference = Math.Abs(Ydifference);

// calculating distance in metres represented by a pixel on the canvas

double Xpixel = RoomLength / 600;

double Ypixel = RoomWidth / 320;

// calculating horizontal and vertical distances in metres

double horizontalDistance = Xpixel \* AbsXdifference;

double verticalDistance = Ypixel \* AbsYdifference;

// using Pythagorus' theorem to calculate diagonal distance between fan and listener

double totalDistancesquared = (horizontalDistance \* horizontalDistance) + (verticalDistance \* verticalDistance);

double totalDistance = Math.Pow(totalDistancesquared, 0.5);

return totalDistance;

}

#endregion

double FindingNRValueOfRoom()

{

String queryNRVal = "SELECT NRVal FROM tblTypeOfRoom INNER JOIN tblProject ON tblProject.TypeOfRoom=tblTypeOfRoom.TypeName WHERE tblProject.ProjectName=@ProjectName";

SqlCommand sqlCmd10 = new SqlCommand(queryNRVal, sqlCon);

sqlCmd10.CommandType = CommandType.Text;

sqlCmd10.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

double NRVal = Convert.ToDouble(sqlCmd10.ExecuteScalar());

return NRVal;

}

#region findingNewFan

String query3 = "SELECT FanCode FROM tblProject WHERE ProjectName=@ProjectName";

SqlCommand sqlCmd3 = new SqlCommand(query3, sqlCon);

sqlCmd3.CommandType = CommandType.Text;

sqlCmd3.Parameters.AddWithValue("@ProjectName", txtProjectName.Text);

string FanCode = Convert.ToString(sqlCmd3.ExecuteScalar());

// This determines whether or not each individual fan is suitable for the given project.

// This is regardless of which fan the user intends to use.

string suitable15 = SuitabilityXBC15();

string suitable25 = SuitabilityXBC25();

string suitable45 = SuitabilityXBC45();

string suitable55 = SuitabilityXBC55();

string suitable65 = SuitabilityXBC65();

string actualSuitability = actualsuitability();

string actualnewfancode = actualnewFanCode();

// This finds the suitability of the fan that the user originally chose.

string actualsuitability()

{

string suitability;

if ((FanCode == "XBC15") && (suitable15 == "Yes"))

{

suitability = "Yes";

return suitability;

}

else if ((FanCode == "XBC25") && (suitable25 == "Yes"))

{

suitability = "Yes";

return suitability;

}

else if ((FanCode == "XBC45") && (suitable45 == "Yes"))

{

suitability = "Yes";

return suitability;

}

else if ((FanCode == "XBC55") && (suitable55 == "Yes"))

{

suitability = "Yes";

return suitability;

}

else if ((FanCode == "XBC65") && (suitable65 == "Yes"))

{

suitability = "Yes";

return suitability;

}

else

{

suitability = "No";

return suitability;

}

}

// If the fan the user chose is not suitable, then find the 'new fan code/codes' which would be suitabile.

string actualnewFanCode()

{

if (suitable15 == "No" && suitable25 == "No" && suitable45 == "No" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "There is no fan in the XBC range that is suitable for your room.";

}

else if ((FanCode == "XBC15") && (actualSuitability == "No"))

{

newFanCode = FindNewFanCode15();

}

else if ((FanCode == "XBC25") && (actualSuitability == "No"))

{

newFanCode = FindNewFanCode25();

}

else if ((FanCode == "XBC45") && (actualSuitability == "No"))

{

newFanCode = FindNewFanCode45();

}

else if ((FanCode == "XBC55") && (actualSuitability == "No"))

{

newFanCode = findNewFanCode55();

}

else if ((FanCode == "XBC65") && (actualSuitability == "No"))

{

newFanCode = findNewFanCode65();

}

else

{

newFanCode = "N/A";

}

return newFanCode;

}

string FindNewFanCode15()

{

if (suitable25 == "Yes" && suitable45 == "Yes" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC25, XBC45, XBC55, XBC65";

}

else if (suitable25 == "No" && suitable45 == "Yes" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC45, XBC55, XBC65";

}

else if (suitable25 == "Yes" && suitable45 == "No" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC25, XBC55, XBC65";

}

else if (suitable25 == "Yes" && suitable45 == "Yes" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC25, XBC45, XBC65";

}

else if (suitable25 == "Yes" && suitable45 == "Yes" && suitable55 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC25, XBC45, XBC55";

}

else if (suitable25 == "Yes" && suitable45 == "Yes" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "XBC25, XBC45";

}

else if (suitable25 == "Yes" && suitable45 == "No" && suitable55 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC25, XBC55";

}

else if (suitable25 == "Yes" && suitable45 == "No" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC25, XBC65";

}

else if (suitable25 == "No" && suitable45 == "Yes" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC45, XBC65";

}

else if (suitable25 == "No" && suitable45 == "No" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC55, XBC65";

}

else if (suitable25 == "Yes" && suitable45 == "No" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "XBC25";

}

else if (suitable25 == "No" && suitable45 == "Yes" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "XBC45";

}

else if (suitable25 == "No" && suitable45 == "No" && suitable55 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC55";

}

else if (suitable25 == "No" && suitable45 == "No" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC65";

}

return newFanCode;

}

string FindNewFanCode25()

{

if (suitable15 == "Yes" && suitable45 == "Yes" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC45, XBC55, XBC65";

}

else if (suitable15 == "No" && suitable45 == "Yes" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC45, XBC55, XBC65";

}

else if (suitable15 == "Yes" && suitable45 == "No" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC55, XBC65";

}

else if (suitable15 == "Yes" && suitable45 == "Yes" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC45, XBC65";

}

else if (suitable15 == "Yes" && suitable45 == "Yes" && suitable55 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC15, XBC45, XBC55";

}

else if (suitable15 == "Yes" && suitable45 == "Yes" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "XBC15, XBC45";

}

else if (suitable15 == "Yes" && suitable45 == "No" && suitable55 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC15, XBC55";

}

else if (suitable15 == "Yes" && suitable45 == "No" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC65";

}

else if (suitable15 == "No" && suitable45 == "Yes" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC45, XBC65";

}

else if (suitable15 == "No" && suitable45 == "No" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC55, XBC65";

}

else if (suitable15 == "Yes" && suitable45 == "No" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "XBC15";

}

else if (suitable15 == "No" && suitable45 == "Yes" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "XBC45";

}

else if (suitable15 == "No" && suitable45 == "No" && suitable55 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC55";

}

else if (suitable15 == "No" && suitable45 == "No" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC65";

}

return newFanCode;

}

string FindNewFanCode45()

{

if (suitable15 == "Yes" && suitable25 == "Yes" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC25, XBC55, XBC65";

}

else if (suitable15 == "No" && suitable25 == "Yes" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC25, XBC55, XBC65";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC55, XBC65";

}

else if (suitable15 == "Yes" && suitable25 == "Yes" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC25, XBC65";

}

else if (suitable15 == "Yes" && suitable25 == "Yes" && suitable55 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC15, XBC25, XBC55";

}

else if (suitable15 == "Yes" && suitable25 == "Yes" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "XBC15, XBC25";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable55 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC15, XBC55";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC65";

}

else if (suitable15 == "No" && suitable25 == "Yes" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC25, XBC65";

}

else if (suitable15 == "No" && suitable25 == "No" && suitable55 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC55, XBC65";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "XBC15";

}

else if (suitable15 == "No" && suitable25 == "Yes" && suitable55 == "No" && suitable65 == "No")

{

newFanCode = "XBC25";

}

else if (suitable15 == "No" && suitable25 == "No" && suitable55 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC55";

}

else if (suitable15 == "No" && suitable25 == "No" && suitable55 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC65";

}

return newFanCode;

}

string findNewFanCode55()

{

if (suitable15 == "Yes" && suitable25 == "Yes" && suitable45 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC25, XBC45, XBC65";

}

else if (suitable15 == "No" && suitable25 == "Yes" && suitable45 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC25, XBC45, XBC65";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable45 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC45, XBC65";

}

else if (suitable15 == "Yes" && suitable25 == "Yes" && suitable45 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC25, XBC65";

}

else if (suitable15 == "Yes" && suitable25 == "Yes" && suitable45 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC15, XBC25, XBC45";

}

else if (suitable15 == "Yes" && suitable25 == "Yes" && suitable45 == "No" && suitable65 == "No")

{

newFanCode = "XBC15, XBC25";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable45 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC15, XBC45";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable45 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC15, XBC65";

}

else if (suitable15 == "No" && suitable25 == "Yes" && suitable45 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC25, XBC65";

}

else if (suitable15 == "No" && suitable25 == "No" && suitable45 == "Yes" && suitable65 == "Yes")

{

newFanCode = "XBC45, XBC65";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable45 == "No" && suitable65 == "No")

{

newFanCode = "XBC15";

}

else if (suitable15 == "No" && suitable25 == "Yes" && suitable45 == "No" && suitable65 == "No")

{

newFanCode = "XBC25";

}

else if (suitable15 == "No" && suitable25 == "No" && suitable45 == "Yes" && suitable65 == "No")

{

newFanCode = "XBC45";

}

else if (suitable15 == "No" && suitable25 == "No" && suitable45 == "No" && suitable65 == "Yes")

{

newFanCode = "XBC65";

}

return newFanCode;

}

string findNewFanCode65()

{

if (suitable15 == "Yes" && suitable25 == "Yes" && suitable45 == "Yes" && suitable55 == "Yes")

{

newFanCode = "XBC15, XBC25, XBC45, XBC55";

}

else if (suitable15 == "No" && suitable25 == "Yes" && suitable45 == "Yes" && suitable55 == "Yes")

{

newFanCode = "XBC25, XBC45, XBC55";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable45 == "Yes" && suitable55 == "Yes")

{

newFanCode = "XBC15, XBC45, XBC55";

}

else if (suitable15 == "Yes" && suitable25 == "Yes" && suitable45 == "No" && suitable55 == "Yes")

{

newFanCode = "XBC15, XBC25, XBC55";

}

else if (suitable15 == "Yes" && suitable25 == "Yes" && suitable45 == "Yes" && suitable55 == "No")

{

newFanCode = "XBC15, XBC25, XBC45";

}

else if (suitable15 == "Yes" && suitable25 == "Yes" && suitable45 == "No" && suitable55 == "No")

{

newFanCode = "XBC15, XBC25";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable45 == "Yes" && suitable55 == "No")

{

newFanCode = "XBC15, XBC45";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable45 == "No" && suitable55 == "Yes")

{

newFanCode = "XBC15, XBC55";

}

else if (suitable15 == "No" && suitable25 == "Yes" && suitable45 == "No" && suitable55 == "Yes")

{

newFanCode = "XBC25, XBC55";

}

else if (suitable15 == "No" && suitable25 == "No" && suitable45 == "Yes" && suitable55 == "Yes")

{

newFanCode = "XBC45, XBC55";

}

else if (suitable15 == "Yes" && suitable25 == "No" && suitable45 == "No" && suitable55 == "No")

{

newFanCode = "XBC15";

}

else if (suitable15 == "No" && suitable25 == "Yes" && suitable45 == "No" && suitable55 == "No")

{

newFanCode = "XBC25";

}

else if (suitable15 == "No" && suitable25 == "No" && suitable45 == "Yes" && suitable55 == "No")

{

newFanCode = "XBC45";

}

else if (suitable15 == "No" && suitable25 == "No" && suitable45 == "No" && suitable55 == "Yes")

{

newFanCode = "XBC55";

}

return newFanCode;

}

#endregion

// Finds possible new attenuation for the project.

#region findingNewAttenuation

string newSupply = "0";

string newExtract = "0";

string newBarrier = "0";

if (actualSuitability == "No")

{

newSupply = findNewSupply();

newExtract = findNewExtract();

newBarrier = findNewBarrier();

}

string findNewSupply()

{

string newsupply;

string supply = FindSupplyAttenuationType();

if (supply == "Short")

{

newsupply = "Medium, Long";

}

else if (supply == "Medium")

{

newsupply = "Long";

}

else

{

newsupply = "There is no supply attenuator type suitable.";

}

return newsupply;

}

string findNewExtract()

{

string newextract;

string extract = FindExtractAttenuationType();

if (extract == "Short")

{

newextract = "Medium, Long";

}

else if (extract == "Medium")

{

newextract = "Long";

}

else

{

newextract = "There is no supply attenuator type suitable.";

}

return newextract;

}

string findNewBarrier()

{

string newbarrier;

string barrier = FindBreakoutBarrierType();

if (barrier == "Basic")

{

newbarrier = "Acoustic";

}

else

{

newbarrier = "There is no supply attenuator type suitable.";

}

return newbarrier;

}

#endregion

#region textfile

// creating, writing to and reading from a summary text file of the model

if (actualSuitability == "No")

{

try

{

String path = @"C:\Users\peybi\Downloads\" + txtProjectName.Text + "Summary.txt";

StreamWriter file = new StreamWriter(path);

file.Write("SUMMARY FAN DATA SHEET\r\n");

file.Write(" \r\n");

file.Write("Technical Data\r\n");

file.Write("Fan code: " + FanCode + "\r\n");

file.Write("Suitable? No" + "\r\n");

file.Write(" \r\n");

file.Write("Changes:" + "\r\n");

file.Write("Suggested new fan code: " + actualnewfancode + "\r\n");

file.Write("Supply attenuator type: " + newSupply + "\r\n");

file.Write("Extract attenuator type: " + newExtract + "\r\n");

file.Write("Acoustic barrier type: " + newBarrier + "\r\n");

file.Close();

using (StreamReader sr = new StreamReader(path))

{

string filetext = sr.ReadToEnd();

txtSummary.Text = filetext;

}

file.Close();

File.Delete(path);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

else if (actualSuitability == "Yes")

{

try

{

String path = @"C:\Users\peybi\Downloads\" + txtProjectName.Text + "Summary.txt";

StreamWriter file = new StreamWriter(path);

file.Write("SUMMARY FAN DATA SHEET\r\n");

file.Write(" \r\n");

file.Write("Technical Data\r\n");

file.Write("Fan code: " + FanCode + "\r\n");

file.Write("Suitable? Yes" + "\r\n");

file.Write(" \r\n");

file.Write("Changes: N/A" + "\r\n");

file.Write("Suggested new fan code: N/A" + "\r\n");

file.Write("Supply attenuator type: N/A" + "\r\n");

file.Write("Extract attenuator type: N/A" + "\r\n");

file.Write("Acoustic barrier type: N/A" + "\r\n");

file.Close();

using (StreamReader sr = new StreamReader(path))

{

string filetext = sr.ReadToEnd();

txtSummary.Text = filetext;

}

file.Close();

File.Delete(path);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

#endregion

}

private void btnGraduations\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show(Graduations());

}

}

}

## Register.xaml

The user enters their email and password. A verification code is sent to their email. After they have entered the correct code, a key is calculated from the email string and the password is hashed with SHA256. The hashed password is stored in the calculated key’s position.

<Page x:Class="Nuacoustic.Register"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Register">

<Grid Background="white">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<TextBlock Text="REGISTER" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="28" FontWeight="Bold" Margin="10" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3"/>

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="2" Grid.Row="0" BorderThickness="1" CornerRadius="17" Margin="1"/>

<Button Name="btnClickToLogin" Grid.Column="2" Grid.Row="0" Content="Not first time? Login here" FontWeight="Bold" Background="#00a3e0" Foreground="Black" FontSize="15" Margin="8" Click="btnClickToLogin\_Click"/>

<!-- first name section-->

<TextBlock Text="First Name\*" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="2" Margin="206,10,213.333,10.333" Grid.ColumnSpan="2"/>

<TextBox x:Name="txtFirstName" Grid.Column="1" Grid.Row="2" Background ="#cccccc" Margin="150.333,10,10,10.333" Grid.ColumnSpan="2" FontSize="28" />

<!--email section-->

<TextBlock Text="Email\*" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="3" Margin="249,9.667,213.333,0" Grid.ColumnSpan="2"/>

<TextBox x:Name="txtEmail" Grid.Column="1" Grid.Row="3" Background ="#cccccc" Margin="150.333,9.667,10,10" Grid.ColumnSpan="2" FontSize="20" />

<!--password section-->

<TextBlock Text="Password\*" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="4" Margin="220,10,213.333,10.333" Grid.ColumnSpan="2"/>

<PasswordBox PasswordChar="\*" x:Name="txtPassword" Grid.Column="1" Grid.Row="4" Background ="#cccccc" Margin="150.333,10,10,10.333" Grid.ColumnSpan="2" FontSize="20"/>

<!-- confirm password section-->

<TextBlock Text="Confirm Password\*" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="5" Margin="154,9.667,213.333,10" Grid.ColumnSpan="2"/>

<PasswordBox PasswordChar="\*" x:Name="txtConfirmPass" Grid.Column="1" Grid.Row="5" Background ="#cccccc" Margin="150.333,9.667,10,10" Grid.ColumnSpan="2" FontSize="20" />

<!-- admin section-->

<TextBlock Text="Admin" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="6" Margin="249,10,213.333,40.333" Grid.ColumnSpan="2" RenderTransformOrigin="0.531,-0.31" Grid.RowSpan="2"/>

<ComboBox Name="txtAdmin" Grid.Column="1" Grid.Row="6" Background="#cccccc" Grid.ColumnSpan="2" FontSize="20" Margin="150.333,10,10,9.667">

<ComboBoxItem IsSelected="True">Unknown</ComboBoxItem>

<ComboBoxItem >Yes</ComboBoxItem>

<ComboBoxItem>No</ComboBoxItem>

</ComboBox>

<!--buttons-->

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="7" BorderThickness="1" CornerRadius="17" Margin="1"/>

<Button Name="btnSubmit" Grid.Column="1" Grid.Row="7" Content="SUBMIT" FontWeight="Bold" Background="#00a3e0" Foreground="Black" FontSize="18" Margin="7" Click="btnSubmit\_Click"/>

<!-- verify -->

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="9" BorderThickness="1" CornerRadius="17" Margin="1"/>

<Button Name="btnVerifyEmail" Grid.Column="1" Grid.Row="9" Content="VERIFY" FontWeight="Bold" Background="#00a3e0" Foreground="Black" FontSize="18" Margin="7" Click="btnVerifyEmail\_Click"/>

<TextBlock Text="Verification code" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="8" Margin="167,9.667,213.333,10" Grid.ColumnSpan="2"/>

<TextBox x:Name="txtVerifyEmail" Grid.Column="1" Grid.Row="8" Background ="#cccccc" Margin="150.333,9.667,10,10" Grid.ColumnSpan="2" FontSize="28" />

</Grid>

</Page>

## Register.xaml.cs

using System;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Data.SqlClient;

using System.Data;

using System.Net;

using System.Net.Mail;

using System.Security.Cryptography;

using System.Collections;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for Register.xaml

/// </summary>

public partial class Register : Page

{

public Hashtable Logins;

string randomCode;

public static string to;

string connectionString = @"Data Source=LAPTOP-G9NKDOKB\SQLEXPRESS; Initial Catalog=NuacousticDB; Integrated Security=True;";

public Register()

{

InitializeComponent();

Logins = (Application.Current.MainWindow as MainWindow).Logins;

}

// Converts entered password to hashed password with SHA256 hash.

static string ComputeSha256Hash(string rawData)

{

// Create a SHA256

using (SHA256 sha256Hash = SHA256.Create())

{

// ComputeHash - returns byte array

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

// Convert byte array to a string

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

private void ShareHashtable()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.Logins = Logins;

}

private void btnClickToLogin\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.LoginScreen)

{

mainWindow.CurrentPage = ApplicationPage.LoginScreen;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

// Checks if email has already been used for an account.

private void btnSubmit\_Click(object sender, RoutedEventArgs e)

{

if (txtEmail.Text == "" || txtPassword.Password == "" || txtFirstName.Text == "" || txtConfirmPass.Password == "")

{

MessageBox.Show("Please fill in mandatory fields.");

}

else if (txtPassword.Password != txtConfirmPass.Password)

{

MessageBox.Show("Passwords do not match.");

}

else

{

// using hashtable to see if email is already registered

int total = GetHashtableTotal();

int hashlocation = total % 1000;

if (Logins.ContainsKey(hashlocation) == true)

{

MessageBox.Show("This email has already been registered.");

}

else

{

MessageBox.Show("Registration is successful. A code will be sent to your email, please enter it in the box and click 'Verify.'");

// send email

string from, pass, messageBody;

Random rand = new Random();

randomCode = (rand.Next(999999)).ToString();

MailMessage message = new MailMessage();

to = (txtEmail.Text).ToString();

from = "nuacoustic.nuaire@gmail.com";

pass = "!testPass123!";

messageBody = "Your verification code is " + randomCode + ".";

message.To.Add(to);

message.From = new MailAddress(from);

message.Body = messageBody;

message.Subject = "Email Verification";

SmtpClient smtp = new SmtpClient("smtp.gmail.com");

smtp.EnableSsl = true;

smtp.Port = 587;

smtp.DeliveryMethod = SmtpDeliveryMethod.Network;

smtp.Credentials = new NetworkCredential(from, pass);

smtp.Send(message);

}

}

}

int GetHashtableTotal()

{

// calculates hashtable index

string email = Convert.ToString(txtEmail.Text);

int countemail = 0;

int arrayLength = email.Length;

int[] emailstr = new int[arrayLength];

int total = 0;

foreach (char c in email)

{

emailstr[countemail] = System.Convert.ToInt32(c);

total = emailstr[countemail] + total;

countemail++;

}

return total;

}

void Clear()

{

txtFirstName.Text = txtEmail.Text = txtPassword.Password = txtAdmin.Text = "";

}

// Checks if the verify code is correct.

private void btnVerifyEmail\_Click(object sender, RoutedEventArgs e)

{

int total = GetHashtableTotal();

int hashlocation = total % 1000;

string plainData = Convert.ToString(txtPassword.Password.Trim());

string hashedData = ComputeSha256Hash(plainData);

// If verification code is correct, the details of the new user are added to the database.

if (randomCode == (txtVerifyEmail.Text).ToString())

{

Logins.Add(hashlocation, hashedData);

if (mainWindow.CurrentPage != ApplicationPage.MainPage)

{

mainWindow.CurrentPage = ApplicationPage.MainPage;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

else

{

MessageBox.Show("Wrong code");

}

}

}

}

## ResetPassword.xaml

<Page x:Class="Nuacoustic.ResetPassword"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ResetPassword">

<Grid Background="white">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<TextBlock Text="RESET PASSWORD" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="30" FontWeight="Bold" Margin="10" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3"/>

<!--email section-->

<TextBlock Text="New Password" Foreground="black" FontSize="30" Grid.Column="0" Grid.Row="2" Margin="169,10.333,160.333,10" Grid.ColumnSpan="2"/>

<PasswordBox x:Name="txtNewPassword" Grid.Column="1" Grid.Row="2" Background ="#cccccc" Margin="150 10 150 10" Grid.ColumnSpan="2" FontSize="28" />

<!--password section-->

<TextBlock Text="Confirm Password" Foreground="black" FontSize="30" Grid.Column="0" Grid.Row="3" Margin="124,9,160.333,10.667" Grid.ColumnSpan="2"/>

<PasswordBox x:Name="txtConfirmNewPass" Grid.Column="1" Grid.Row="3" Background ="#cccccc" Margin="150 10 150 10" Grid.ColumnSpan="2" FontSize="28"/>

<!--buttons-->

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="4" BorderThickness="1" CornerRadius="25" Margin="10"/>

<Button Name="btnReset" Grid.Column="1" Grid.Row="4" Content="RESET" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="21" Margin="18.5" Click="btnReset\_Click"/>

</Grid>

</Page>

## 

## ResetPassword.xaml.cs

Here the user chooses their new password and this is updated in the hashtable.

using System;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Data.SqlClient;

using System.Security.Cryptography;

using System.Collections;

using System.Data;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for ResetPassword.xaml

/// </summary>

public partial class ResetPassword : Page

{

public string email = ForgotPassword.to;

public Hashtable Logins;

public ResetPassword()

{

InitializeComponent();

Logins = (Application.Current.MainWindow as MainWindow).Logins;

}

private void ShareHashtable()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.Logins = Logins;

}

// Calculates hashed password.

static string ComputeSha256Hash(string rawData)

{

// Create a SHA256

using (SHA256 sha256Hash = SHA256.Create())

{

// ComputeHash - returns byte array

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

// Convert byte array to a string

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

int GetHashtableTotal()

{

int countemail = 0;

int arrayLength = email.Length;

int[] emailstr = new int[arrayLength];

int total = 0;

foreach (char c in email)

{

emailstr[countemail] = System.Convert.ToInt32(c);

total = emailstr[countemail] + total;

countemail++;

}

return total;

}

private void btnReset\_Click(object sender, RoutedEventArgs e)

{

string plainData = Convert.ToString(txtNewPassword.Password.Trim());

string hashedData = ComputeSha256Hash(plainData);

int total = GetHashtableTotal();

int hashlocation = total % 1000;

// Updates user's password.

if (txtNewPassword.Password == txtConfirmNewPass.Password)

{

Logins[hashlocation] = hashedData;

MessageBox.Show("Password has been reset successfully.");

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.HomePage)

{

mainWindow.CurrentPage = ApplicationPage.HomePage;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

else

{

MessageBox.Show("The two passwords do not match. Please try again.");

}

}

}

}

## Settings.xaml

<Page x:Class="Nuacoustic.Settings"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:Nuacoustic"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Settings">

<Grid Background="white">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Source="/images/nuairelogo1.jpg" Grid.Row="0" Grid.Column="0"/>

<TextBlock Text="SETTINGS" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="30" FontWeight="Bold" Margin="10" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="3"/>

<!-- change password -->

<TextBlock Text="CHANGE PASSWORD" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="25" FontWeight="Bold" Margin="10" Grid.Column="0" Grid.Row="3" Grid.ColumnSpan="3"/>

<TextBlock Text="Email" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="4" Margin=" 210 10 160 10" Grid.ColumnSpan="2"/>

<TextBox x:Name="txtEmail" Grid.Column="1" Grid.Row="4" Background ="#cccccc" Margin="150 10 150 10" Grid.ColumnSpan="2" FontSize="28"/>

<TextBlock Text="Old Password" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="5" Margin=" 210 10 160 10" Grid.ColumnSpan="2"/>

<PasswordBox x:Name="txtOldPassword" Grid.Column="1" Grid.Row="5" Background ="#cccccc" Margin="150 10 150 10" Grid.ColumnSpan="2" FontSize="28"/>

<TextBlock Text="New Password" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="6" Margin=" 210 10 160 10" Grid.ColumnSpan="2"/>

<PasswordBox x:Name="txtNewPassword" Grid.Column="1" Grid.Row="6" Background ="#cccccc" Margin="150 10 150 10" Grid.ColumnSpan="2" FontSize="28"/>

<TextBlock Text="Confirm Password" Foreground="black" FontSize="20" Grid.Column="0" Grid.Row="7" Margin=" 210 10 160 10" Grid.ColumnSpan="2"/>

<PasswordBox x:Name="txtConfirmNewPass" Grid.Column="1" Grid.Row="7" Background ="#cccccc" Margin="150 10 150 10" Grid.ColumnSpan="2" FontSize="28"/>

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="8" BorderThickness="1" CornerRadius="25" Margin="0"/>

<Button Name="btnSaveNewPass" Grid.Column="1" Grid.Row="8" Content="SAVE" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="21" Margin="8" Click="btnSaveNewPass\_Click"/>

<!-- delete account -->

<Border BorderBrush="#00a3e0" Background="#00a3e0" Grid.Column="1" Grid.Row="10" BorderThickness="1" CornerRadius="25" Margin="0"/>

<Button Name="btnDeleteAccount" Grid.Column="1" Grid.Row="10" Content="DELETE ACCOUNT" FontWeight="Bold" Background="Transparent" Foreground="Black" FontSize="21" Margin="8" Click="btnDeleteAccount\_Click"/>

</Grid>

</Page>

## Settings.xaml.cs

using System;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Data;

using System.Data.SqlClient;

using System.Security.Cryptography;

using System.Collections;

namespace Nuacoustic

{

/// <summary>

/// Interaction logic for Settings.xaml

/// </summary>

public partial class Settings : Page

{

public Hashtable Logins;

public Settings()

{

InitializeComponent();

txtEmail.Text = (Application.Current.MainWindow as MainWindow).email;

Logins = (Application.Current.MainWindow as MainWindow).Logins;

}

private void ShareHashtable()

{

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

mainWindow.Logins = Logins;

}

private void btnDeleteAccount\_Click(object sender, RoutedEventArgs e)

{

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.DeleteAccount)

{

mainWindow.CurrentPage = ApplicationPage.DeleteAccount;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

int GetHashtableTotal()

{

string email = Convert.ToString(txtEmail.Text);

int countemail = 0;

int arrayLength = email.Length;

int[] emailstr = new int[arrayLength];

int total = 0;

foreach (char c in email)

{

emailstr[countemail] = System.Convert.ToInt32(c);

total = emailstr[countemail] + total;

countemail++;

}

return total;

}

private void btnSaveNewPass\_Click(object sender, RoutedEventArgs e)

{

string oldHash = ComputeSha256Hash(Convert.ToString(txtOldPassword.Password));

string hashedData = ComputeSha256Hash(Convert.ToString(txtConfirmNewPass.Password));

int total = GetHashtableTotal();

int hashlocation = total % 1000;

if (txtOldPassword.Password == "" || txtConfirmNewPass.Password == "" || txtEmail.Text == "" || txtNewPassword.Password == "")

{

MessageBox.Show("Please enter values into all the boxes.");

}

else if (txtNewPassword.Password != txtConfirmNewPass.Password)

{

MessageBox.Show("Passwords do not match.");

}

else if (Logins.ContainsKey(hashlocation) == false || Logins.ContainsValue(oldHash) == false)

{

MessageBox.Show("Old email or password is incorrect.");

}

else

{

Logins[hashlocation] = hashedData;

MessageBox.Show("Your password has been changed.");

ShareHashtable();

MainWindow mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow.CurrentPage != ApplicationPage.LoginScreen)

{

mainWindow.CurrentPage = ApplicationPage.LoginScreen;

}

else

{

mainWindow.MainFrame.NavigationService.GoForward();

}

}

}

// Hashes entered password with SHA256 hash.

static string ComputeSha256Hash(string rawData)

{

// Create a SHA256

using (SHA256 sha256Hash = SHA256.Create())

{

// ComputeHash - returns byte array

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

// Convert byte array to a string

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

}

}

# Testing

## Test table

| Test Number | Description | Test Data | Expected outcome | Actual outcome |
| --- | --- | --- | --- | --- |
| 1.1:  Login | Testing correct login data | Email: [nuacoustic.nuaire@gmail.com](mailto:nuacoustic.nuaire@gmail.com)  Password: hello | Pass, go to home page with email in email box. | As expected. |
| 1.2:  Login | Testing incorrect login data | Email: [nuacoustic.nuaire@gmail.com](mailto:nuacoustic.nuaire@gmail.com)  Password: incorrect | Show message box saying incorrect. | As expected. |
| 2.1: Register | Entering data | Name: test Email: [test.test145@yahoo.com](mailto:test.test145@yahoo.com)  Password: test1  Confirm password: test1  Admin: No | Sends verification code to email. | As expected. |
| 2.2: Register | Entering verification code | Verification code: 169578 | Takes to start page. | As expected. |
| 2.3: Register | Using previously registered password | Name: test Email: [test.test145@yahoo.com](mailto:test.test145@yahoo.com)  Password: test1  Confirm password: test1  Admin: No | Show message box saying email has previously been used. | Sent a new code to email. Retest in 2.3a. |
| 2.3a: Register | Using previously registered password | Name: test Email: [test.test145@yahoo.com](mailto:test.test145@yahoo.com)  Password: test2  Confirm password: test2  Admin: No |  | As expected. |
| 2.4: Register | Entering passwords that aren’t the same | Name: test Email: [test.test145@yahoo.com](mailto:test.test145@yahoo.com)  Password: test4  Confirm password: test5  Admin: No | Shows message box saying passwords don’t match. | As expected. |
| 2.5: Register | Not entering all required information. | Name: test Email: null  Password: test5 confirm password: test5  Admin: no | Shows message box saying to fill in all required fields. | As expected. |
| 3.1: Home page | See projects as an admin (shows all projects in the system) | Email: nuacoustic.nuaire@gmail.com | Shows Project1 and Project2. | As expected. |
| 3.2: Home page | See projects as someone not an admin (shows all projects in the system) | Email: thestudyhacks@yahoo.com | Shows Project2. | As expected. |
| 3.3: Home page | Open details of a user’s own project as any user. | Email: [nuacoustic.nuaire@gmail.com](mailto:nuacoustic.nuaire@gmail.com)  Project name: Project1 | Shows details of Project1. | As expected. |
| 3.4: Home page | Open project chosen by user | Email: [nuacoustic.nuaire@gmail.com](mailto:nuacoustic.nuaire@gmail.com)  Project name: Project1 | Opens new page with email and project name labelled as given. | As expected. |
| 3.5: Home page | Deleting a project | Email: [nuacoustic.nuaire@gmail.com](mailto:nuacoustic.nuaire@gmail.com)  Project name: Project3 | Deletes project from database. | As expected. |
| 3.6: Home page | Open details of another user’s project as admin. | Email: [nuacoustic.nuaire@gmail.com](mailto:nuacoustic.nuaire@gmail.com)  Project name: Project2 (Created by another user ‘thestudyhacks@yahoo.com’ | Shows details of Project2. | As expected. |
| 3.7: Home page | Open details of another user’s project as someone not an admin. | Email: [thestudyhacks@yahoo.com](mailto:thestudyhacks@yahoo.com)  Project name: Project1  (Created by ‘nuacoustc.nuaire@gmail.com’) | Should not open the details. | As expected. |
| 4.1: Create new project | Creates new project (field) in database | Values relevant to the project. | Creates new project in database. | As expected. |
| 5.1: Current project page | Updating information | Email: [nuacoustic.nuaire@gmail.com](mailto:nuacoustic.nuaire@gmail.com)  Project name: Project1 | Updates details in database. | As expected. |
| 6.1: Project page | Update location of fans | Position of fans on screen | Updates pixel location of each object in database. | As expected. |
| 6.2: Project page | Outputting summary for fan that is suitable. | Fan: XBC15 | Shows it is suitable but does not show any changes. | As expected. |
| 6.3: Project page | Outputting summary for fan that is not suitable. | Fan: XBC45 | Shows it is suitable and shows suitable changes. | As expected. |
| 7.3: Settings page | Changing password | Email: [nuacoustic.nuaire@gmail.com](mailto:nuacoustic.nuaire@gmail.com)  Old password: hello  New & confirm new password: changedpassword | Changes password and takes to login page. | Said old password or email is incorrect. Retest in 7.3a. |
| 7.3a: Settings page | Changing password | Email: test.test145@yahoo.com  Old password: test2  New & confirm new password: changedpassword | Changes password and takes to login page. | As expected. |
| 7.4: Settings page | Delete account | Email: [test.test145@yahoo.com](mailto:test.test145@yahoo.com) | Deletes account and projects made from account (called TestProject) from database. Takes to exit page. | As expected. |

## Test evidence screenshots

1.1

When the user enters the correct email and password combination, they are taken to the home page, shown on the right.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

1.2

When the user enters an incorrect email and password combination, a message box appears stating that this is the case.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

2.1

After a user registers, they are sent an email containing a verification code, I have shown this on the right by using my email.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

2.2

After entering the code from the email, the user is taken back to the first page where they can login.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

2.3

This was a failed test, I input an email that had already been registered, and the application created a second account for it, which is not what is supposed to happen.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

2.3a

The second time was successful, as shown, a message box comes up saying that the email has already been used meaning that the user has to either log in with the email or enter a new one to register.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

2.4

If the text in the ‘Password’ and ‘Confirm Password’ boxes do not match, the user is notified and they are required to type them again.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

2.5

If there is no text in any of the starred fields, then the user is requested to fill them in.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

3.1

The email used here is an admin email. This means pressing ‘See Projects’ allows this user to see all the projects in the system, regardless of who created them.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

3.2

The email here is not an admin email. This means pressing ‘See Projects’ only allows this user to see the projects they have created themselves.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

3.3

When an email and a project name is entered, pressing the ‘Open Details’ button shows the details for that particular project.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

3.4

When an email and a project name is entered, pressing the ‘Open Project’ button opens a new page that allows you to edit the details of the project.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

3.5

After deleting a project, it no longer shows up when pressing ‘See Projects’.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

3.6

Being an admin also allows you to see the details of projects created by other users. Project2 was created by ‘thestudyhacks@yahoo.com’, which is a non-admin email. However, as ‘nuacoustic.nuaire@gmail.com’ is an admin email, it can see the project details of Project2.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

3.7

This is a non-admin email trying to see the project details of a project they did not create. As expected, this was prevented as only admin emails can do this.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

4.1

This is an account creating a new project. As you can see this is Project4, and after the user has created it, it now shows up in the database.

|  |
| --- |
| Input |
|  |
| Output |
|  |

5.1

This is a user updating a project’s information. This is done by entering new values or choosing options from the drop-down lists, then pressing ‘Update’.

|  |
| --- |
| Input |
|  |
| Output |
|  |

6.1

The values in the database screenshots are the pixel coordinates of each of the 4 pictures in the grid. These values are used to calculate the distance from the fan to the listener. The user drags the images around the grid, and then the values update in the databases. The picture of the ear represents the listener. The ‘S’ represents the Supply inlet for the fan and ‘E’ represents the Extract outlet for the fan, and the positions of these impact the output of the application.

|  |  |
| --- | --- |
| Input | Output |
|  |  |

6.2

This is an example of a project which is successful, meaning that the fan is not too loud. In the ‘summary fan data sheet’, there are no alternatives suggested as it is successful.

|  |
| --- |
| Input |
|  |
| Output |
|  |

6.3

This is an example of a project which is unsuccessful, meaning that the fan is too loud. In the ‘summary fan data sheet’, there are suitable alternatives suggested, which have been calculated and checked in the program.

|  |
| --- |
| Input |
|  |
| Output |
|  |

7.3

This is an account attempting to change their password. The user has entered the correct email and password combination, however a message box has popped up saying the opposite. This was due to me not hashing the password they had input, so it was as if they had entered the wrong password as the hash was different.

|  |  |
| --- | --- |
| Input | Output |
|  | Message box said old password or email is incorrect. |

7.3a

After changing my error, this feature works. The hash on the left is the hash of the old password, while on the right is it the hash of the new longer password.

|  |  |
| --- | --- |
| Input | Output |
| Hashed password as stored before: | Hashed password as stored after (proof that it has changed): |

7.4

It was a requirement that if an account is deleted, their projects were deleted as well. TestProject in the left screenshot was created by ‘test.test145@yahoo.com’. After this email was deleted, this project was no longer in the database.

|  |  |
| --- | --- |
| Input | Output |
| Project in database: | Project is no longer in database: |

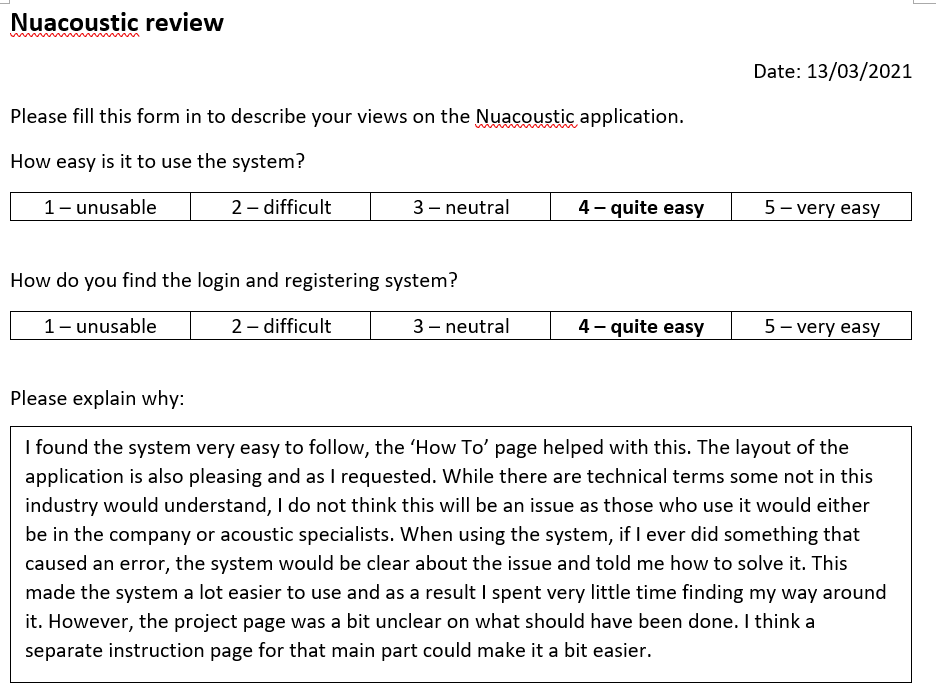
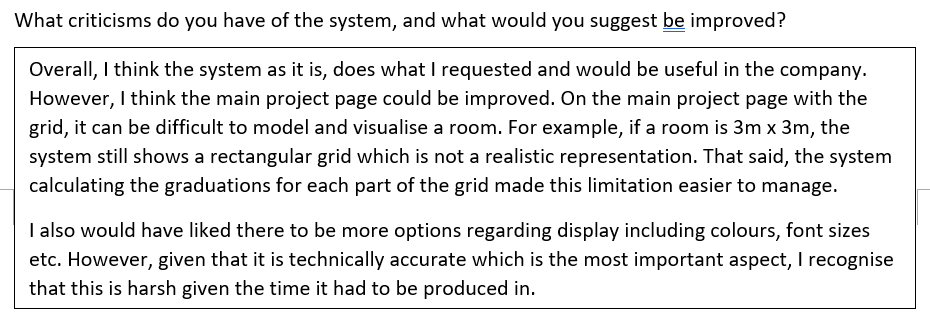
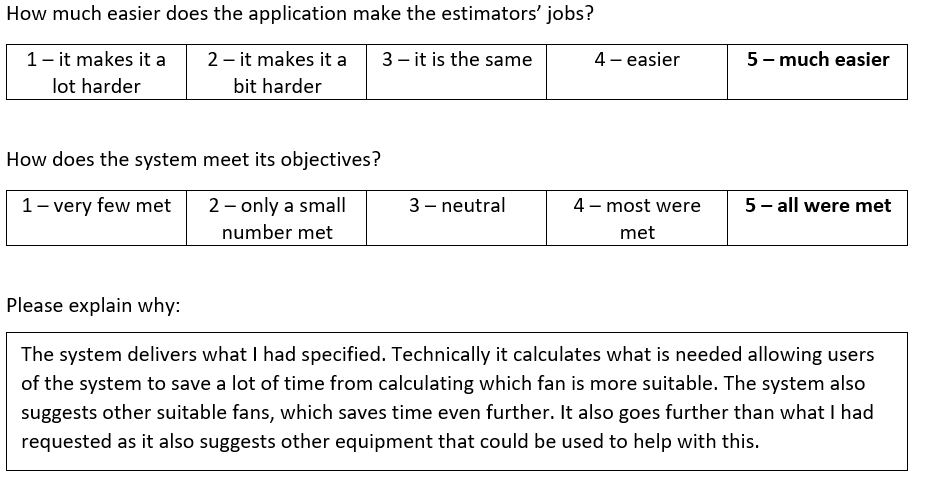
# Evaluation

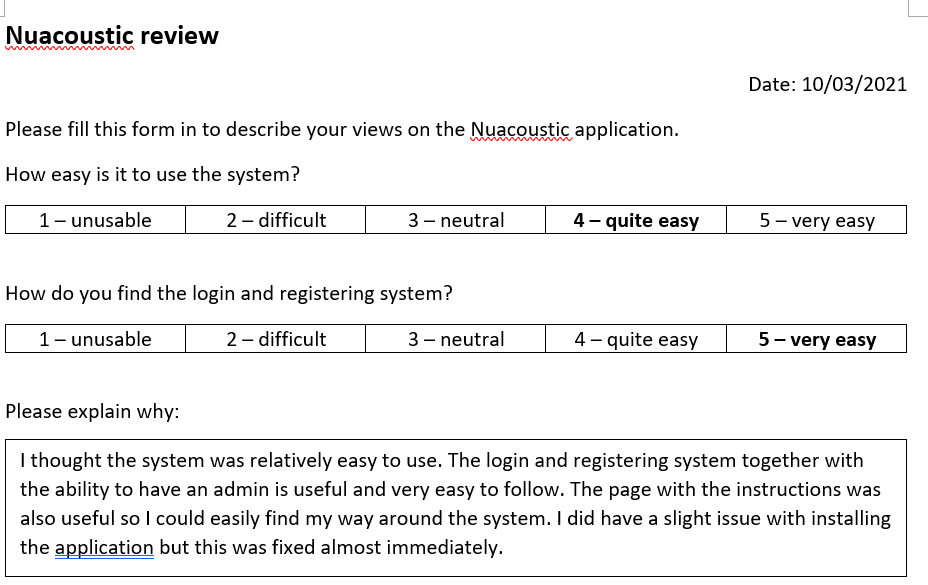
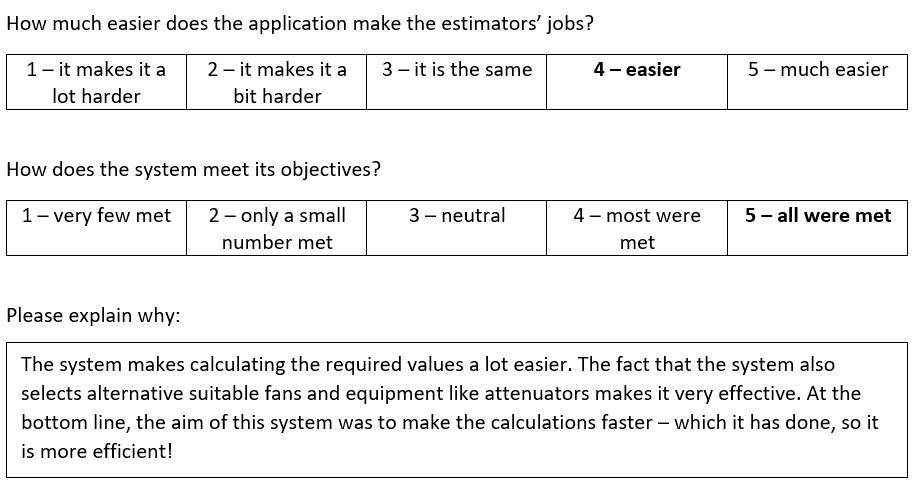
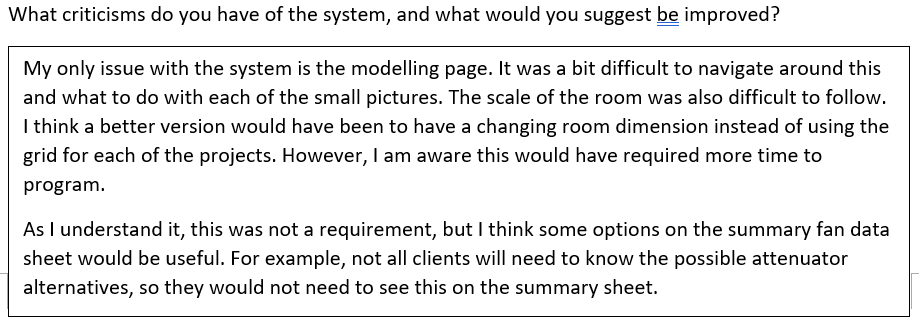
## Objective analysis

| Objective number | Objectives | Has the objective been met? | Comments |
| --- | --- | --- | --- |
| 1.1 | There will be a table where all the users’ projects are stored. | Yes | The user creates a new record for this when they create a new project, and they can edit the values as well. |
| 1.2 | Each type of room would have its NR value stored in a table. | Yes |  |
| 2.1 | It must start up within 5 seconds. | Yes | The application starts up within 5 seconds. |
| 2.2 | The web application will have the colour scheme and font that my client requested and will have to look similar to the company’s current website. | Yes | The website has a white background with blue buttons and black writing, similar to the application created. |
| 2.3 | Must be easy to use. | Yes | My client has used the application and could follow it well without me guiding them. The ‘How to’ page also helps with this. |
| 2.4.1 | Each user will have a password. | Yes | Chosen by the user on registration. |
| 2.4.2 | There will be a register link for people who are using the website for the first time. | Yes | There is a button on the login page that links to the register page and vice versa. |
| 2.4.3 | The login page must have a blank form and simple form asking for email and password. | Yes |  |
| 2.4.4 | The register page must ask for the user’s name, email address, let them choose their password and enter their status. | Yes |  |
| 2.4.5 | There must also be an option to change their password. | Yes | This can be done on either forgot password or change password. |
| 2.4.6 | A user’s account has to be verified by sending them an email. | Yes | The same process is used if the user forgets their password. |
| 2.4.7 | The logins are stored in a hashtable. | Yes | The passwords are hashed and put into a hashtable. The key is calculated from the email string. |
| 2.5.1 | On the home page, the user would see all their saved projects. | Yes | Click ‘see projects’. |
| 2.5.2 | The projects have to have their last edited dates displayed on the home page as well as the project name. | Yes | Click ‘see details’, this shows all details of the project including the last edit date. |
| 2.5.3 | There must be a button to create a new project. | Yes | Click ‘create project’. |
| 2.6.1 | The user must be able to save their projects, so that they are re-editable. | Yes | Click ‘save project’ for a newly created project and ‘update project’ for a previously created project (this is made clear on the application). |
| 2.6.2 | There must be a button that provides an output of all the relevant information. | Yes | Click ‘Summary’ on the project page. |
| 2.7.1 | The ‘How to’ tab will have an explanation of how to use the application. | Yes |  |
| 2.8.1 | Users must be able to choose the units they use (imperial or metric). | Yes |  |
| Ex1 | User can be an admin | Yes | Chosen upon registration. |
| Ex1.1 | Being an admin allows you to see everyone’s projects and their details. | Yes | The details can be seen on the home page if you enter a valid project name. |
| Ex1.2 | An admin cannot edit other people’s projects. | Yes | Admins can see the details of everyone’s projects, but cannot edit them themselves, which is what my client requested. |
| Ex2 | The user can personalise what is in the output summary, they might choose to not have certain sections. | No | This turned out to not be an important need as most people would pick to have all the features anyway. |
| Ex3 | The resulting application that I create will be put as a link on to Nuaire’s company website. | No | This turned out not to be needed as the company decided to distribute it another way. |

## User feedback

To get feedback on the application, I sent a review form to my client and one of the estimators that hypothetically would be using my application. I also sent a summary of the objectives I had been set my client and ones that I had created myself, e.g. the start up time being less than 5 seconds. I asked them to grade the application on the ease of use and whether or not the objectives had been met. I also asked them for any criticisms they had of the system and how they suggested the application could be improved.

Client’s form:  

Estimator’s form:  

## Analysis of user feedback

This chart summarises the scores given by my client and the estimator.

**Desirable changes/additions**

|  |  |
| --- | --- |
| Client | Estimator |
| * Display options, e.g. font size * Scale of the project page grid | * Scale of the project page grid * Options on what the summary fan data sheet displays |

The feedback from my client and the estimator is positive, averaging an overall score of 4.5/5, so I can conclude that my system does what was required. It is reassuring that any added features I put in, including the ‘How to’ page, were useful.

However, both the client and the estimator thought that the scale of the project page grid was difficult to follow, so updating this for it to change for each project would be the next step in improving the application. I have outlined how I would go about this in the ‘Possible extensions’ section below.

The estimator also thought it would be useful for customers if there was a choice in what was displayed on the summary fan data sheet. I would implement this by asking the user to select the possible sections on a checklist after selecting the ‘Summary’ button.

The client thought it would be good to have to display options, and I would put these on the ‘Settings’ page of the system. I would store each user’s choice in the database under a new table called ‘DisplayOptions’.

## Possible extensions

Overall, I am pleased with the outcome of the project, however, if I were to do it again, there are some things that I would make sure to change. My client emphasised that the modelling of the room needed to be improved, so this would be my priority to change. Currently, my system has a pre-drawn grid, and the user drags the fan sources around this. I had previously thought that calculating the graduations for the grid would make it easier – which the client said did help – but I see how it would be difficult to visualise. To fix this, I would possibly redraw the grid for each project. I would make sure that when drawing the grid, its length and width have the same ratio as the user inputted values. The graduations would still be calculated in the same way; however, I would also make sure they were done in such a way so that they were easier to deal with, e.g. 0.5m per square instead of 0.3333456m.

I would also put in more customisable options for the display as the client stated. This could include being able to choose what information is put the summary. This was in my objectives for something extra to do, however as my client said it was not the most important feature, I decided I did not have significant time to do it to a good enough quality. I would also try and put options to change the display, for example, the colour of the background or font size.