# School Database

Contents

[School Database 1](bookmark://_Toc5873086#_Toc5873086)

[Analysis 4](bookmark://_Toc5873087#_Toc5873087)

[Background 4](bookmark://_Toc5873088#_Toc5873088)

[Objectives 4](bookmark://_Toc5873089#_Toc5873089)

[End user 5](bookmark://_Toc5873090#_Toc5873090)

[Entity relationship diagram 6](bookmark://_Toc5873091#_Toc5873091)

[Research Notes 7](bookmark://_Toc5873092#_Toc5873092)

[Questionnaire 11](bookmark://_Toc5873093#_Toc5873093)

[Limitations 14](bookmark://_Toc5873094#_Toc5873094)

[Data 15](bookmark://_Toc5873095#_Toc5873095)

[Data Dictionary 17](bookmark://_Toc5873096#_Toc5873096)

[Object Orientation Planning 19](bookmark://_Toc5873097#_Toc5873097)

[User Class (Super class) 19](bookmark://_Toc5873098#_Toc5873098)

[Student Class (Inherits From User) 19](bookmark://_Toc5873099#_Toc5873099)

[Teacher Class (Inherits From User) 19](bookmark://_Toc5873100#_Toc5873100)

[Lesson Class 19](bookmark://_Toc5873101#_Toc5873101)

[Behaviour Point (Parent Class Of Achievement Point Class) 19](bookmark://_Toc5873102#_Toc5873102)

[Achievement Point Class(Inherits from Behaviour Point Class) 19](bookmark://_Toc5873103#_Toc5873103)

[Grade Class 20](bookmark://_Toc5873104#_Toc5873104)

[Documented Design 20](bookmark://_Toc5873105#_Toc5873105)

[Overall System Design 20](bookmark://_Toc5873106#_Toc5873106)

[Navigation Based Design 21](bookmark://_Toc5873107#_Toc5873107)

[Data Dictionary 21](bookmark://_Toc5873108#_Toc5873108)

[Database Design 23](bookmark://_Toc5873109#_Toc5873109)

[SQL Queries 23](bookmark://_Toc5873110#_Toc5873110)

[Identification of storage media 24](bookmark://_Toc5873111#_Toc5873111)

[Main Algorithms 24](bookmark://_Toc5873112#_Toc5873112)

[Class Definitions 27](bookmark://_Toc5873113#_Toc5873113)

[User Interface Design 28](bookmark://_Toc5873114#_Toc5873114)

[Login Page 28](bookmark://_Toc5873115#_Toc5873115)

[Main Menu(s) 28](bookmark://_Toc5873116#_Toc5873116)

[Creating a user 30](bookmark://_Toc5873117#_Toc5873117)

[Creating a Class 32](bookmark://_Toc5873118#_Toc5873118)

[View Classes 33](bookmark://_Toc5873119#_Toc5873119)

[Creating a Subject 33](bookmark://_Toc5873120#_Toc5873120)

[Searching Users 34](bookmark://_Toc5873121#_Toc5873121)

[View Homeworks 35](bookmark://_Toc5873122#_Toc5873122)

[Creating homework 36](bookmark://_Toc5873123#_Toc5873123)

[Creating achievement/behaviour points 37](bookmark://_Toc5873124#_Toc5873124)

[Technical Solution 38](bookmark://_Toc5873125#_Toc5873125)

[Code 38](bookmark://_Toc5873126#_Toc5873126)

[Main.py 38](bookmark://_Toc5873127#_Toc5873127)

[Userclasses.py 111](bookmark://_Toc5873128#_Toc5873128)

[Screenshots 113](bookmark://_Toc5873129#_Toc5873129)

[Login Form 113](bookmark://_Toc5873130#_Toc5873130)

[Main Windows 114](bookmark://_Toc5873131#_Toc5873131)

[Creating and Editing Users 115](bookmark://_Toc5873132#_Toc5873132)

[Behavior and Achievement Points 119](bookmark://_Toc5873133#_Toc5873133)

[Create/Edit Class 119](bookmark://_Toc5873134#_Toc5873134)

[Search Users 120](bookmark://_Toc5873135#_Toc5873135)

[Update Grades 121](bookmark://_Toc5873136#_Toc5873136)

[Homework 121](bookmark://_Toc5873137#_Toc5873137)

[Reset Password 122](bookmark://_Toc5873138#_Toc5873138)

[Database 123](bookmark://_Toc5873139#_Toc5873139)

[Objectives 124](bookmark://_Toc5873140#_Toc5873140)

[Testing 128](bookmark://_Toc5873141#_Toc5873141)

[Trace Tables 128](bookmark://_Toc5873142#_Toc5873142)

[Testing Navigation 136](bookmark://_Toc5873143#_Toc5873143)

[Evaluation 138](bookmark://_Toc5873144#_Toc5873144)

[Evaluation of system objectives 138](bookmark://_Toc5873145#_Toc5873145)

[Discussion of possible improvements 139](bookmark://_Toc5873146#_Toc5873146)

[User feedback and analysis of feedback 140](bookmark://_Toc5873147#_Toc5873147)

[Conclusion 145](bookmark://_Toc5873148#_Toc5873148)

[Appendix A – Questionnaire 147](bookmark://_Toc5873149#_Toc5873149)

[Appendix B – Feedback Form 148](bookmark://_Toc5873150#_Toc5873150)

[Bibliography 149](bookmark://_Toc5873151#_Toc5873151)

# Analysis

## Background

For my project, I am creating a school database. This database will allow school administrators, teachers and students to manage their accounts in one centralized place. Here I will store all the relevant data needed to get this database to work and allow users to sign in and manage their accounts. Each user will have access to different features depending on their role and the experience will be tailored to their accounts.

The school I am tailoring my project to will be Robert Smyth Academy. This is the school I currently attend and therefore I have witnessed how things work from a student perspective, but I am also able to speak to teachers etc. to get a better view which I possibly wouldn’t have at another school. This school is an upper school in Market Harborough for 11-19-year-olds. This therefore means that students will have very different timetables – ie. students at GCSE level will have lessons constantly – whereas Sixth Form students typically have just 27 hours of lessons every two weeks.

I plan to allow all the administrators access to the relevant features and data. Teachers should be able to administrate their classes, update students’ progress etc. and a student should be able to access his/her data and be able to receive messages, homework reminders and see the relevant information of their subjects/classes.

## Objectives

The main objective of this project is to create a program that interacts with working relational databases, that communicate with each other to give each type of user a tailored view of their profile, information and the ability to access the data that may affect them. Instead of having to go through multiple different sites. The plan is for:

* Administrators to be able to add, edit and remove students, teachers and subjects from the database
* Teachers to be able to see all their classes in a timetable, to see the details including a list of students etc. and the ability of adding homework, behavior/achievement points and to add grades for assignments, mini-tests etc.
* For students, the plan is so that they can see their timetable and lesson details, they should have access to homework, and news feeds etc. as well as the details such as how they are performing.

All the users should have a personalized experience and will only be able to access this through logging in. This will all be presented to the user with a clean and user-friendly GUI which will be displayed in a simple and easy to understand way – so that all users are able to understand and use the program – no matter what skill level they have. This should all be achievable in the given time – and I am happy I have and will gain the given skills to create this program.

## End user

My end user is Robert Smyth Academy’s Sixth Form. This sixth form is formed of two-year groups – year 12 and year 13 but I will also be storing staff/teachers' profiles on my database.

At present, this school uses iSams[[1]](#footnote-1) for the administration work, this is a desktop application that only administrators and teachers have access to. This connects to the network and allows read/write access to the database. This however, is also available as an app on iPhone and Android. This suite includes iTeacher, iParent, iStudent and iReport. However, currently – students and parents do not use this application – and from what I’ve seen it is just the desktop application that is used by teachers and other members of staff.

Show My Homework[[2]](#footnote-2) is used for illustrating homework and deadlines. This is a website that allows teachers to set homework for their classes and students to access from anywhere online to get the details of their homework.

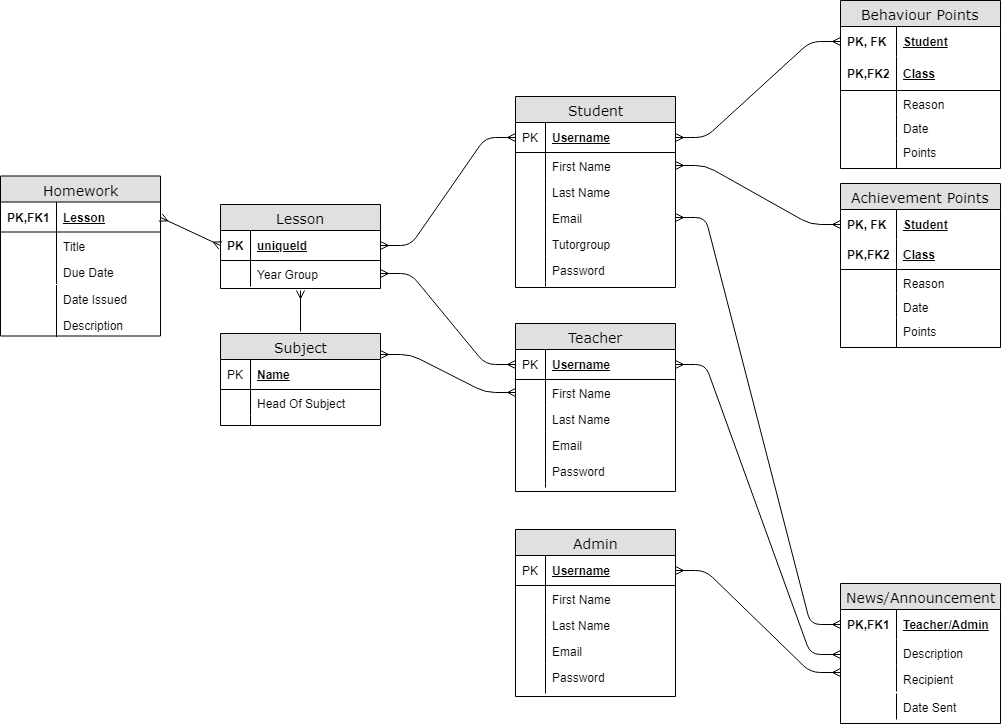
For communication, most big announcements are made to students in assemblies or in tutor times. This often results in users forgetting the information later or anyone absent missing out. There is also the rare occasion where an email may be sent to the students with reminders for extra-curricular meetings. The problem with the current system is there are too many sites storing this information. Administrators, teachers and students could easily miss out on crucial information as they haven’t been on one of the many sites as it takes too long to work through all the sites.

By having one program that everyone can use to access and manage all this information would significantly reduce the amount of admin work needed to be done – as they will only have to set up one system. This will also make it easier for users to use as they will only have one system where all the features and data they need will be.

The users for this system would be the school administrators, teachers and students. This means there will be a large skills gap between users – some that are good with technology and some that are not so good. Therefore, this system needs to appear very user-friendly so that anybody can read and understand what to do.

* Administrators should be able to access features that include adding/removing students/teachers to the database, editing subjects, timetables etc.
* Teachers will need to have the ability to access their timetable and view their timetable. They will also want to be able to set homework, fill in a register, add grades for assignments/tests, and message other staff, read staff announcements and more.
* Students will want to be able to access their timetable, read and send their own messages, view their class information, view their homework and read announcements relevant to them.

## Entity relationship diagram



## Research Notes

To learn object-oriented programming, I went through a course on Future Learn called “Object-oriented Programming in Python: Create Your Own Adventure Game”.[[3]](#footnote-3)

To learn how Python interacts with databases using SQLite, I went through a video tutorial on YouTube called “Python SQLite Tutorial: Complete Overview - Creating a Database, Table, and Running Queries” by Corey Schafer.[[4]](#footnote-4)

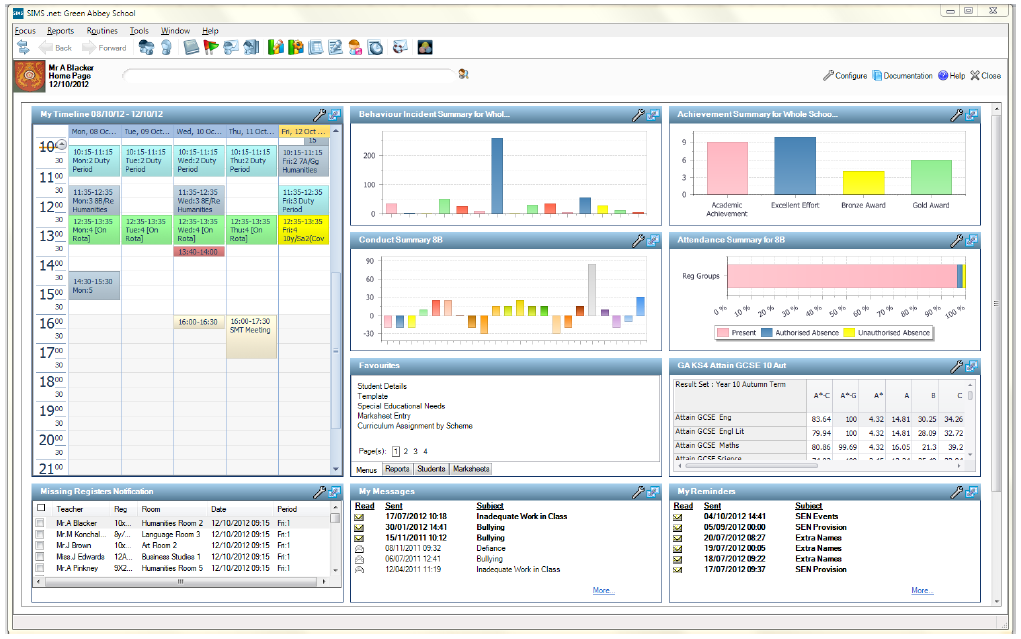
To understand the complexity of what makes a strong password and how to incorporate it into my own programs, I watched a presentation named “zxcvbn: Low-Budget Password Strength Estimation” by Daniel Lowe Wheeler.[[5]](#footnote-5)

To understand how to correctly store these passwords I followed a small tutorial on Hashing named “Hashing Strings with Python” on Python Central.[[6]](#footnote-6)

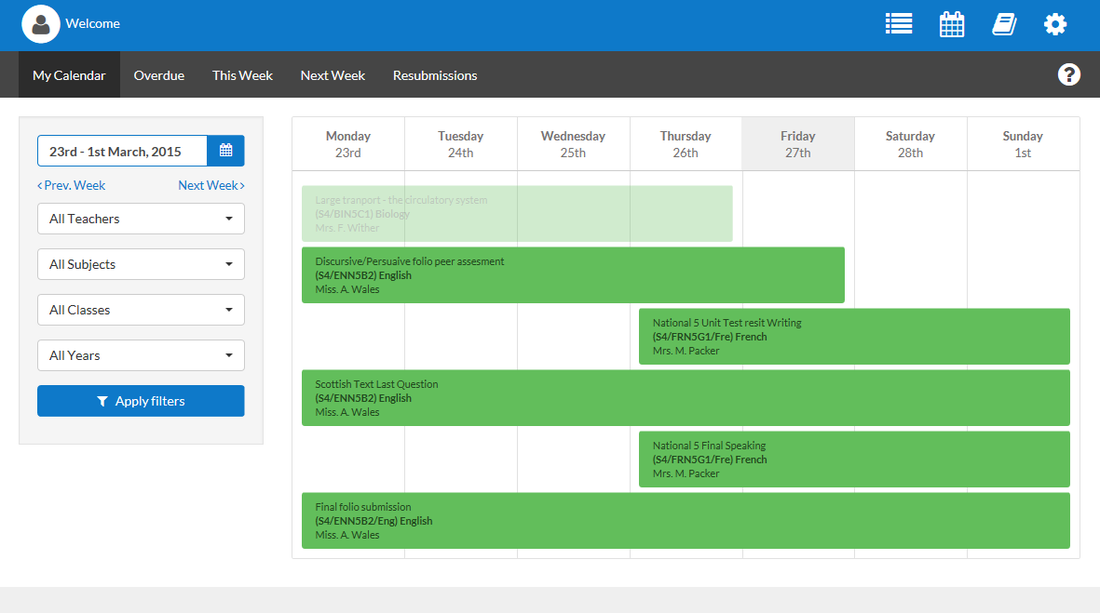
I have also followed YouTube tutorials on PyQt to learn the framework and how to use this for my project. To get a more general idea of the framework I followed a tutorial by Sentdex[[7]](#footnote-7) and for a more practical run-through, I followed a tutorial by Ssj6[[8]](#footnote-8)

By reviewing the Capita Sims[[9]](#footnote-9), iSams and Show My Homework[[10]](#endnote-1)2 programs it allowed me to understand what relevant information must be stored and the functions an ideal database would incorporate and note what works well and what does not.

**Capita Sims**  
  
By reviewing the Capita Sims program – the good features of this are that there is a live timetable tracker, that shows the teachers classes. This also included progress trackers where, from the data inserted – a graph displays the class and student progress, so it is much easier to read and understand. This program also includes reminders and messages that allows teachers to communicate with each other – and allows a teacher to schedule plans. The main purpose of this program, however, is to manage the students. They do this by having a register that allows the teacher to mark students in the lesson, away, late etc. and the program allows the administrators to manage classes/add students etc. The big disadvantage I see in the program is that it is exclusive to teachers and administrators who are connected on the network. I would improve this by allowing students to have access to the program and manage their own profiles, messages etc. This program comes in the form of a desktop application.



Capita Sims



Show My Homework

**Show My Homework**

The key feature of Show My Homework is the fact that it gives students a platform to access their homework remotely at home. This can be accessed either through the internet or through the mobile application. This shows the students upcoming homework – which also allows them to tick off whether they have completed it (a function that is rarely actually used by users but a good idea), it allows the user to also see any past homework. Show My Homework gives each homework a title and the subject – and then if you expand it allows the user to read a full description and access any media that goes with the homework i.e. Worksheets. The site also comes with notices and reminders. The current procedure for adding homework is by just signing on and clicking add homework – selecting the class etc. with all the buttons and instructions clearly labelled. For students to view their homework – they simply log on and their homework is shown to them. The big disadvantage I see in the program is that it is just an application for homework reminders – and nothing else.

**iSams**



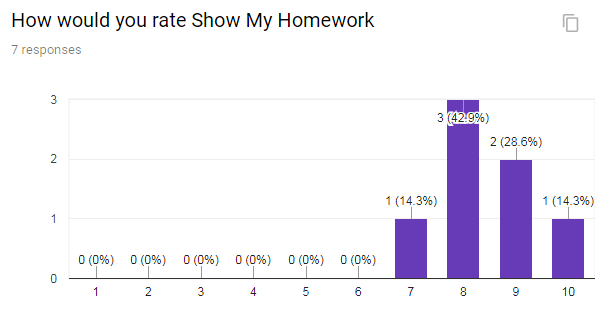
iSams

iSams has only recently been the method the school used for administration work – as previously it was Capita Sims. There are a few obvious advantages to this system and it is clear why Robert Smyth made the change. iSams is available on the web, as a desktop application or as a app on iPhone or Android – and accessible anywhere with an internet connection. This application includes almost everything a school could need – and if it is missing anything it includes third party integration allowing it to be added. The dashboard includes all the information needed such as a daily bulletin, blog, statistics, reports, timetables etc. The apps also include most of these features. On the iParents app, it shows the parents information on their student, for example, attendance records, detention notifications, assessments, timetables etc. and on the students’ app you can do the same. I believe this is a very good program with all the features a school would ever need. However, from reading reviews online there seems to be a few complaints on how complicated it is. One user rated it 3 stars and said “*Everything is a lot more complicated than it should be! Plus, it looks like its from 1980 and has so many glitches which makes everything so much more annoying.*”[[11]](#footnote-10)

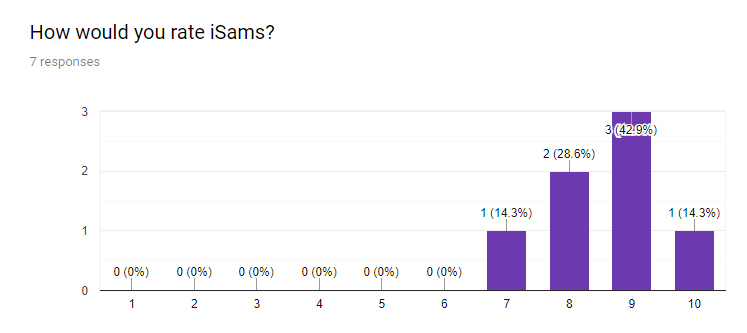
# Questionnaire

To get a further understanding and to get the teachers perspective, I created a Google Form asking a few basic questions and sent them around to teachers. (Reference Appendix A). The results are summarized below.

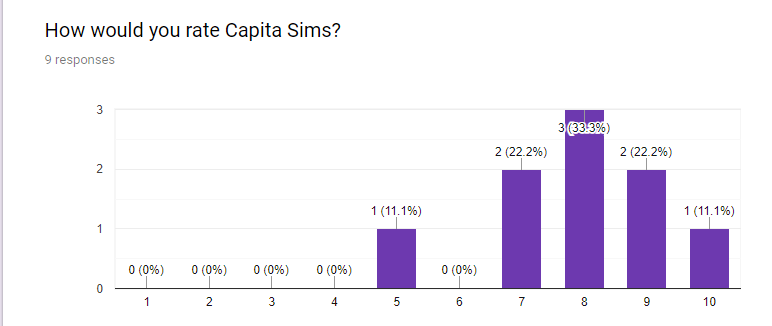
**If you could change how students and lessons are managed - what would you improve?**Almost all users seemed very happy with how it is running. This question received a lot of responses along the lines of “Nothing” or “I am happy with the current system” – however, there were also responses asking for this process to be simplified and made quicker and easier to use. When creating this question – I knew as it was slightly vague there could have been room for all types of complaints/feedback, however, I doubt anyone that was taking this questionnaire took too long to think of the answer.



**What are the drawbacks, improvements and favourite features of Show My Homework?**These results were clearly very positive, and users had very little complains about this program. The only drawback users suggested was that the program sometimes “takes too long to load”. The main reason that this could be is because it is connected through the internet. This allows the program to be portable, however, it can be frustrating to some users when the servers are slow or far away. My suggestion to combat that, is that my program would be ran off the school servers – therefore users at school would connect via LAN and speeds would be faster – however, it would still be accessible from wherever there is an internet connection. The improvements that were suggested included a wide range of suggestions such as “create a desktop application”, “add a functionality to allow students to interact with their homework” and a suggestion to show “how long you have left”. This feedback is very helpful to me as it gives me an idea of what to put in my project that users are missing. I also asked a question asking users what their favorite features were. This will then allow me to not only see what Show My Homework are doing wrong but showing me what they are doing right as well. I received a lot of feedback commenting on the ‘Ease of use’ and the ‘Phone notifications’ seemed also very well received.



**What are the drawbacks, improvements and favourite features of iSams?**



**If you’ve previously used Capita Sims - what are the drawbacks, improvements and favourite features of the program?**This program is also very well received as shown – but when asked why – the reasons were slightly different to the responses of Show My Homework. Instead, most of the responses were along the lines of ‘A lot of functionality’ and ‘the graphs that display the statistics about the school/class’. This is obviously for more advanced users- whereas Show My Homework is designed at appealing to everyone and being as simple as possible. The drawbacks of this program were mostly ‘overcomplicated’ and ‘simplify the program’.

**Conclusion**  
From this questionnaire, it has helped me get a further understanding of what users want and currently enjoy – which will hence help me create a successful program. I feel the questions asked and the responses have given me a better idea on where to take my project next. To improve – I would increase the sample size and give the questionnaire to more experienced users – ie. Long serving teachers who have used a variety of different technologies during their careers.

# Limitations

In a normal school database there are hundreds of students and a lot of information stored for each person. This information can be anything from your address, the city you were born in or even your doctors contact details. A lot of this data would never have any function in my program – it would be there just for show. Hence, to save me a lot of time which I could focus to add other functions to my program, I will use abstraction to only use the data that will have a use in creating my program. If my final product was to be used however, data like that would need to be added – for safeguarding reasons.

Another big limitation for the development of this program – is that I am only using Python. Although this is probably the best language for handling data and databases like this – it is very poor for creating a good user interface. I will be using PyQT 4 to create my user interface – however, as python is not a very portable language it is very hard to create an app/web version of this as well. If I had time- I would learn another language that is more general purpose – for example C# or JS.

One function I wanted to add was to be able to generate personal timetables based on students’ options, teachers timetables etc. However, as I do not have access to a very large dataset containing hundreds of students and the subjects they are doing – it will be very hard to create an algorithm to generate a timetable without knowing approximate numbers and how a typical timetable would be created.

Another limitation I will face is that I will only be using SQLite. If I was more advanced/had more time – I believe the best approach for this program would be the “Big Data” approach – which would require using a NoSQL database – which I currently do not have the skillset to do. This would be good for my program as there are lots of *many to many relationships* in my database – which makes SQL very hard to process that kind of data. However, I believe for the volume of data I am using – SQL would be acceptable.

# Data

In my program I will have 2 different databases – one for user’s data – and one for everything else, i.e. Subjects, homework etc. As the school would not be allowed to provide me real data (due to reasons such as GDPR) – I have decided to create my own users – to show that everything works and that it would be able to handle real users. I will have multiple tables in each database. In the Users database – I will have a table for storing administrator profiles, a table for storing teacher’s profiles and a table for storing students’ profiles.

Level 0 Dataflow Diagram

In my data database, I will be having a table for each subject and lesson and a table for behavior and achievement points. If this was used in a real situation, the user’s database would have hundreds of records – and the data would have thousands. However, because I do not have the time to create hundreds of users, I will create around 40 users with a range of different subjects etc. This program must be able to handle more than one user at a time as it is likely if this was the actual application there would be numerous instances where multiple users are on at the same time.A close up of electronics

Description generated with low confidence

A screen shot of a monitor

Description generated with high confidence

Level 1 Dataflow Diagram

# Data Dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Purpose | Field Type | Field Size | Example Data | Validation |
| First name | Stores the first name of the user | String | N/A | Tom | Only characters/not blank |
| Last name | Stores the last name of the user | String | N/A | Rowbotham | Only characters/not blank |
| Username | Stores a Unique ID for the user | String | N/A | trowbotham | Generates automatically |
| Password | Secures the users account | String | N/A | PassWord123! | Strong Password (Hard to guess) |
| Year Group | Stores the users year group | Integer | 2 Choices Year 12 or 13 | Year 12 | N/A |
| Subject name | Used to recognise a subject | String | N/A | Computer Science | Only characters/not blank |
| Short name | Used to refer to a subject without typing in full | String | N/A | Comp Sci | Only characters/not blank |
| Teacher (username) | Used to link a teacher to a class | String | N/A | tsmith | Checks valid teacher |
| Date Of Birth | Used to give information on a user | Date | Any valid date | 23/07/2001 | Choice |
| Date Created/Sent | Used to show when something was sent | Date Time | Current date/time | 20/10/18 13:42 | Generated automatically |
| Points | Running total – ie. Achievement points | Integer | N/A | 28 | Generated automatically by adding one each time |
| Description | Description of something ie. Homework | String | 512 | “Sample homework do this etc.” | N/A |
| Message | Sample message | String | 512 | “Hi etc.” | N/A |
| Recipient | Who you are sending a message to | String | N/A | Trowbotham | Valid user |
| Class ID | To identify a class | String | N/A | 12CompSciA | Generated automatically |
| Grade | To give a student based on work | String | Multichoice | A+ | Choice |
| Email | Creates an email for the user | String | N/A | Tsmith@rsa.edu | Generates automatically |

# Object Orientation Planning

### User Class (Super class)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Type | Field Name | Field Type | Initial Value | Description |
| Private | First name | String | Input | First name of the user |
| Private | Last name | String | Input | Last name of the user |
| Private | Password | String | Input | Hashed version of user’s password |
| Private | Full name | String | First + last | Generates full name |
| Private | Email | String | None | Email of user |
| Public | SetEmail | String | Method | Sets the email of the user |

### Student Class (Inherits From User)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Type | Field Name | Field Type | Initial Value | Description |
| Private | Year Group | Integer | Input | Year Group Of User |

### Teacher Class (Inherits From User)

No changes from User class  
Admin Class(Inherits From User)  
No changes from User class  
Subject Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Type | Field Name | Field Type | Initial Value | Description |
| Private | Full name | String | Input | Full name of Subject |
| Private | Short name | String | Input | Shortened name of Subject |
| Private | Head | String | Input | Head Of Faculty |

### Lesson Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Type | Field Name | Field Type | Initial Value | Description |
| Private | Year Group | Integer | Input | Year group of class |
| Private | Subject | String | Input | What subject the lesson is |
| Private | Teacher | String | Input | Who the teacher is |

### Behaviour Point (Parent Class Of Achievement Point Class)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Type | Field Name | Field Type | Initial Value | Description |
| Private | Teacher | String | Input | Who gave the point |
| Private | Student | String | Input | Who received the point |
| Private | Points | String | Input | How many points given |
| Private | Reason | String | Input | Reason for points |

### Achievement Point Class(Inherits from Behaviour Point Class)

No change from behaviour points

### Grade Class

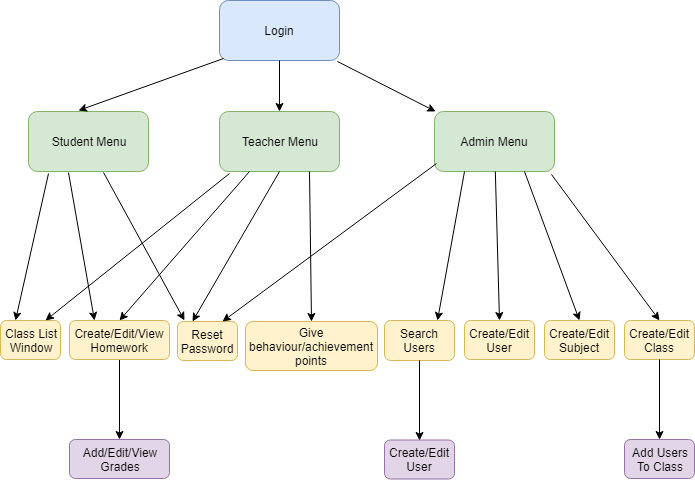
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Type | Field Name | Field Type | Initial Value | Description |
| Private | Grade | String | Input | Grade given to user |
| Private | Task | String | Input | What the grade was given for |
| Private | Student | String | Input | Who the grade is for |
| Private | Lesson | String | Input | What lesson the grade is for |
| Private | Marks given | String | Input | How many marks the user got |
| Private | Total Marks | String | Input | How many total marks there are. |

# Documented Design

## Overall System Design

|  |  |  |  |
| --- | --- | --- | --- |
| **Inputs** | **Processes** | **Storage** | **Outputs** |
| User Details  Subject Details  Class Details  Homework Details  Achievement/Behaviour Point Details  Message Info  Grades  New password | Search Users  Add/retrieve to database  Validation  Generate Random Password  Verify password strength | Userphotos folder (contains all images)  Database tables:  User  Student  StudentClass  Class (Table per class created)  Classes  Subjects  Messages  Points | Display users  Display subjects  Display classes  Display achievement/ behaviour points  Search  Display homework/grades |

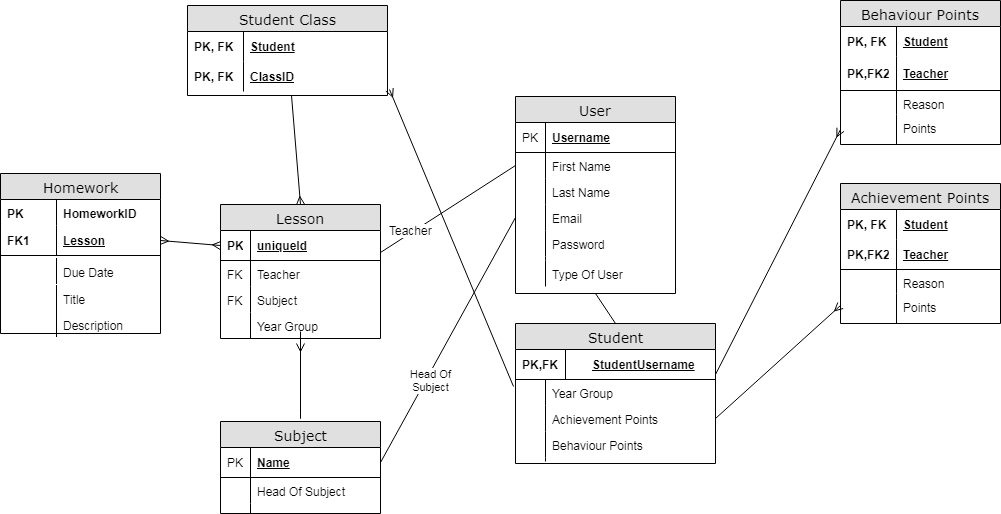
## Navigation Based Design



# Data Dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Purpose | Field Type | Field Size | Example Data | Validation |
| First name | Stores the first name of the user | String | N/A | Tom | Only characters/not blank Regex:  [a-z-A-Z]+ |
| Last name | Stores the last name of the user | String | N/A | Rowbotham | Only characters and – symbol. Not blank  Regex:  [a-z-A-Z-]+ |
| Username | Stores a Unique ID for the user | String | N/A | trowbotham | Generates automatically. First initial of first name plus surname and then a number if there are duplicates.  Regex:  [a-z-A-Z-0-9-]+ |
| Password | Secures the users account | String | N/A | PassWord123! | Strong Password (Hard to guess) Any characters. Use library ZXCVBN which tests strength. |
| Year Group | Stores the users year group | Integer | 2 Choices Year 12 or 13 | Year 12 | 12 or 13 – Multi-choice |
| Subject name | Used to recognise a subject | String | N/A | Computer Science | Only characters/not blank |
| Short Subject name | Used to refer to a subject without typing in full | String | N/A | Comp Sci | Only characters/not blank |
| Date Of Birth | Used to give information on a user | Date | Any valid date before 2003. | 23/07/2001 | Choice using built in widget. |
| Date Created/Sent | Used to show when something was sent | Date Time | Current date/time | 20/10/18 13:42 | Generated automatically |
| Points | Running total – ie. Achievement points | Integer | N/A | 28 | Generated automatically by adding one each time |
| Description | Description of something ie. Homework | String | 512 | “Sample homework do this etc.” | N/A |
| Class ID | To identify a class | String | N/A | CompSciA | Generated automatically |
| Grade | To give a student based on work | String | Multichoice | A\* | Choice from a combo box |
| Email | Put in from the admin/user | String | N/A | Tsmith@rsa.edu | Any valid email. Shown by letters, numbers and valid symbols followed by an @ followed by characters then a . Followed by characters Regex: (^[a-zA-Z0-9\_.+-]+@[a-zA-Z]+**\.**[a-zA-Z]+$) |

# Database Design



# SQL Queries

**Login Query:**  
SELECT \* FROM users WHERE username=USERNAME AND password= PASSWORD  
If user is a student:  
SELECT yeargroup FROM student WHERE username= USERNAME

**Search Queries:**  
SELECT \* FROM homework WHERE homeworkid = ID  
SELECT \* FROM users WHERE type = 'Teacher' ORDER BY first ASC  
SELECT fullname FROM subjects ORDER BY fullname ASC  
SELECT id FROM classes WHERE id = ID  
SELECT username FROM users WHERE username LIKE USERNAME ORDER BY username ASC"

**Updating Queries:**  
UPDATE users SET password = PASSWORD WHERE username = USERNAME  
UPDATE classes SET teacher = TEACHER AND yeargroup = YEARGROUP WHERE id = ID  
UPDATE subjects SET head = HEADOFSUBJECT WHERE id = ID  
UPDATE student SET behaviourpoints = NUMBER WHERE username = USERNAME

**Insert Queries:**  
INSERT INTO classes VALUES (YEARGROUP,TEACHER,SUBJECT,ID)  
INSERT INTO subjects VALUES (ID,HEADOFSUBJECT)  
INSERT INTO student VALUES (USERNAME,YEARGROUP,0,0)  
INSERT INTO users VALUES (FIRST,LAST,EMAIL,PASSWORD,TYPE)  
INSERT INTO studentclass VALUES (USERNAME,CLASSID)  
INSERT INTO homework VALUES (HOMEWORKID,CLASSID,DUEDATE,TITLE,DESCRIPTION)  
INSERT INTO achievementpoints VALUES (STUDENTID, TEACHERID, REASON, POINTS)

**Delete Queries:**  
DELETE FROM studentclass WHERE username = USERNAME AND classid = CLASSID

**CREATING TABLES:**  
CREATE TABLE users (first text, last text, username text, password text, email text,   
type text)  
CREATE TABLE student (username text, yeargroup integer, behaviourpoints integer, achievementpoints integer)  
CREATE TABLE subjects (id text,head text)  
CREATE TABLE classes (yeargroup integer, teacher text, subject text, id text )  
CREATE TABLE behaviourpoints (teacher text, student text, points integer, reason text)  
CREATE TABLE achievementpoints (teacher text, student text, points integer, reason text)  
CREATE TABLE studentclass (student text, lesson text)  
CREATE TABLE homework(homeworkid text, classid text, duedate date, title text, description text)

# Identification of storage media

In my program – I will ideally be having 2 areas of storage media – the actual program and main server. On the main server I will store my database and also store files such as user profile pictures. This would then communicate with the users program – who will be connected to the server so that the users program can fetch and send data to make changes to the database/read from it. By doing this it will improve the security of the system – so instead of users having access to the entire database – they will only have access to parts they need which will only be fetched if the program allows it ie. They have permission to view it. The users program will be able to be stored anywhere as long as it is on a computer and the user is able to run it. This means they can store it on a USB stick, hard drive, CD ROM etc. Because the users program will be communicating with the server – it will have to be over an internet connection – so if the user was in the school for example – it should connect through LAN using localhost – however – I want the ability for users to access from their homes as well via an IP address. The database size will be very minimal – as it is only storing text values – and the user pictures will likely take up the majority of the memory used by the data server. On the clients end however – as none of this will be stored on their end – all that will be stored is the program which size is very minimal. As for the speeds of the program – as long as there is a strong connection with the data server I imagine speeds will be very fast – as the code will not require anything intensive.

# Main Algorithms

**Login Algorithm**  
My login algorithm will simply fetch the inputted details and compare it to the database. If it is correct then collect the information and proceed to the main menu  
PSEUDOCODE:  
username1 <- INPUT  
password1 <- INPUT  
password1 <- hashed(password1)  
SELECT \* FROM users WHERE username = username1 AND password = password1  
data <- FETCH FROM database  
IF data!= None:  
 user <- data  
 accessgranted()  
ELSE:  
 OUTPUT “Access Denied”  
  
**Creating a user**  
In this algorithm I will be showing the algorithm on how a user will be created. There will be some obstacles such as generating a unique ID for the user and avoiding potential collisions when users have the same name.   
PSEUDOCODE:  
firstname <- INPUT  
last <- INPUT  
username1 <- firstname[0]+last  
length = length(username1)  
SELECT username FROM users WHERE username LIKE username1 ORDERBY username ASC  
data <- FETCH FROM database  
IF username1 IN data:  
 username1 += “1”  
 chosen <- False  
 WHILE NOT chosen:  
 IF username1 IN data:  
 username1 = username1[:length-1] + STRING(INTEGER(username1[length-1:])+1)  
 ELSE:  
 chosen <- True  
email <- INPUT  
password <- None  
type <- INPUT  
INSERT INTO users VALUES (FIRST,LAST,EMAIL,PASSWORD)

**Upload A Users Picture**  
Here I will show I will give the user the ability to upload a profile picture. To do this I must give the user an option of valid inputs and then move that image to a new image.  
PSEUDOCODE:  
originalpic = INPUT(OPENFILEWINDOW(FILTER = “.png”))  
filelocation = “DATABASESERVER\userpics”  
UPDATE users SET pic = filelocation + username1 WHERE username = username1  
  
**Generate Password**  
When a user's account is created or when a user forgets their password – it is important each user has a different password and it is random and doesn’t follow a format – so that others won't be able to know what it is. Because of this, the password will be generated for them using random letters. This will be 9 letters long with a mix of digits and letters. This means that the chances of guessing are practically impossible (1.3537087e+16). This is not meant to be memorable though – it acts just as a temporary password until the user logs on for the first time and chooses a secure *memorable* password.  
PSEUDOCODE:  
letters <- [“A”,”B”,”C”...”Y”,”Z”,”a”,”b”,”c”...”y”,”z”,”0”,”1”,”2”...”8”,”9”]  
password <- “”  
FOR index IN RANGE(9):  
 number <- RANDOMINTEGER(0,62)  
 password += letters[number]  
password1 = HASHED(password)  
UPDATE users SET password = password1 WHERE username = username1

**Search Users**  
In a school there will likely be hundreds of users and hence it is important that the client is able to filter users to help find what they want. I plan to do this by many things – including filtering by username, first name, surname, classes, etc.   
PSEUDOCODE:  
SELECT \* FROM USERS WHERE TYPE = 'Admin' OR TYPE = 'Teacher'  
data1 <- FETCH FROM database  
SELECT \* FROM USERS,STUDENT WHERE USERS.USERNAME = STUDENT.USERNAME  
data2 <- FETCH FROM database  
data <- MERGE data1 + data2

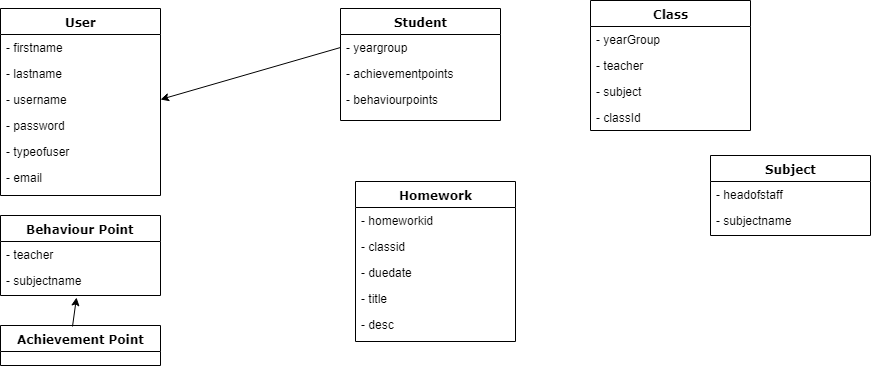
IF FILTERBY fullname:  
 search <- INPUT  
 filtered <- []  
 FOR index IN RANGE(LENGTH(data)):  
 IF search SIMILAR TO data[index][firstname + lastname] OR search IN data[index] [firstname + lastname] THEN  
 filtered.APPEND(data[index])

data <- filtered

IF FILTERBY type:  
 filtered <- []  
 search = INPUT(Admin,Student or Teacher)  
 FOR index IN RANGE(LENGTH(data)):  
 IF data[index][type] = search:  
 filtered.APPEND(data[index])  
 data <- filtered  
IF FILTERBY class:  
 filtered <- []  
 class <- INPUT(All Classes)  
 SELECT teacher FROM classes WHERE id = class  
 teacher <- FETCH FROM database   
 FOR index IN range(length(data)):  
 IF data[index][type] = “Teacher”:  
 IF data[index][username] = teacher:  
 filtered.APPEND(data[index])  
 ELIF data[index][type] = “Student”:  
 SELECT username FROM studentclass WHERE username = data[index][username]  
 AND classid = class   
 temp <- FETCH FROM database  
 IF temp != None:  
 filtered.APPEND(data[index])

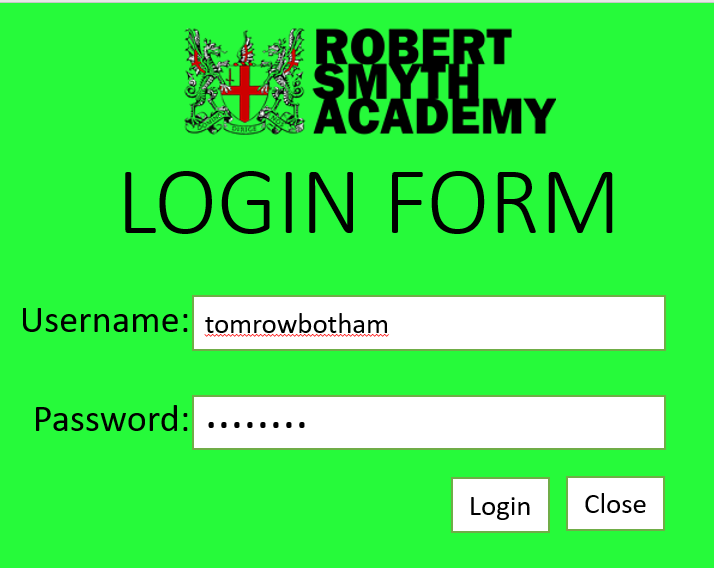
**Change to custom password**  
As mentioned previously, upon creation of a user – they will be assigned a random password. My program will then give the option for a user to change their password to a more memorable one. I will be using an external library for this which determines how easy it is to guess – based on input and common passwords.  
PSEUDOCODE:  
oldpass <- INPUT  
newpass <- INPUT  
confirmpass <- INPUT  
oldpass <- hashed(oldpass)  
IF oldpass = loggedInUser.password:  
 IF newpass = confirmpass:  
 strength <- passwordstrength(loggedInUser)  
 IF strength >= SECURE:  
 newpass <- hashed(newpass)  
 loggedInUser.password <- newpass  
 UPDATE users SET password = loggedInUser.password WHERE username  
 =loggedInUser.username  
 COMMIT TO DATABASE

# Class Definitions



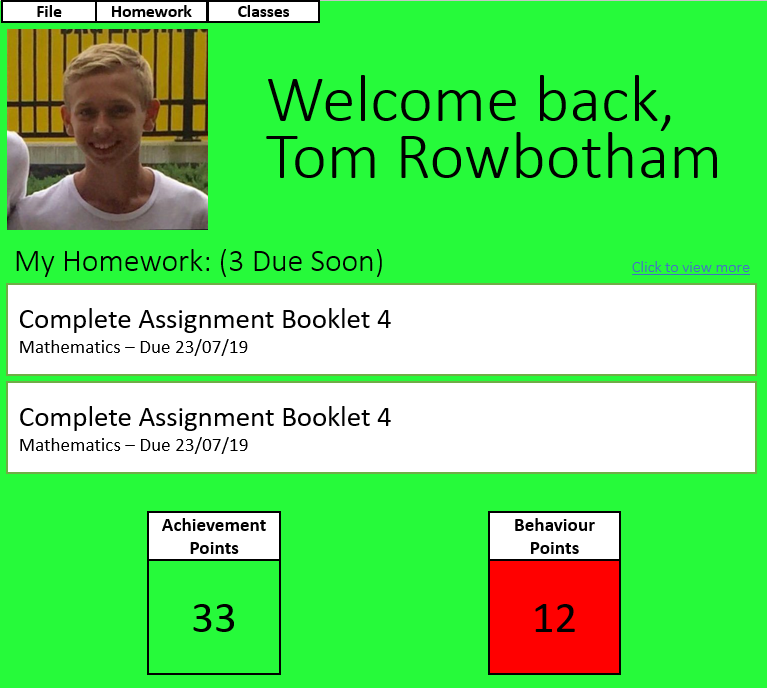
# User Interface Design

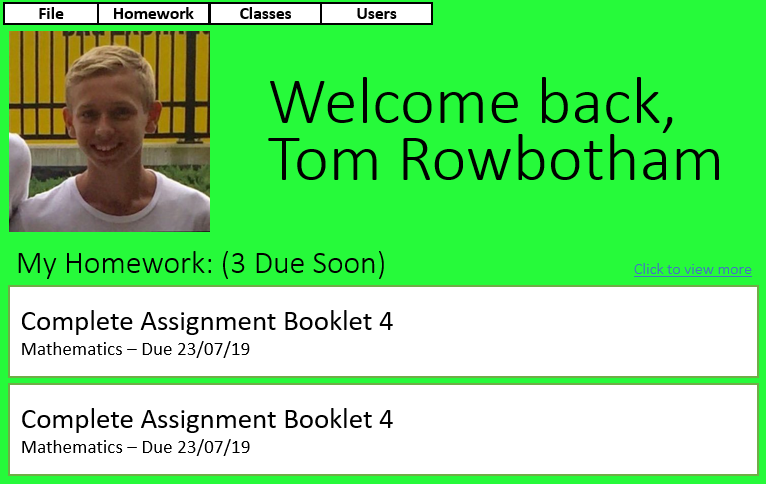
## Login Page



My login page will be the first window any user will encounter and hence it is important they are welcomed by a professional and simple looking window. The school logo – Robert Smyth Academy is centred at the top not just to tell the user they are on a Robert Smyth application – but to give it a more personalised welcoming feel. The choice of green background is because the Trust (the school is in – Tudor Grange) heavily uses green as their style choice – and this is colour is also used throughout other applications i.e. the website. The login page will have two clear boxes where information may be entered – and the text inputted in the password will be hidden. If the username and password are correct then login button is clicked then the user will proceed to the main menu. Else – a error message should appear. If the close button is pressed – then the program should close and end.

## Main Menu(s)

For my main menu – I will have 3 adaptations of the same page. Each one adapting based on the user type as they will all have different methods.  
This first template is the basis for all main menus and shown above is my choice for a student's perspective on the menu. As shown on the top will be drop down menus – which will allow the user to go through homework, classes and file options. On the top left – will be the users profile picture. This would have been uploaded by the admin. The text to the right of that is a simple welcome message with the user's full name. This will give the window a more friendly welcoming design. Under will be the homework title – describing how many homework's are due within the next week. The ”click to view more” tab will be a quick shortcut to the ”View Homework” window – and under are two buttons leading to the two most upcoming homework's details. And finally, a viewable statistic of how many achievement and behaviour points the student has.

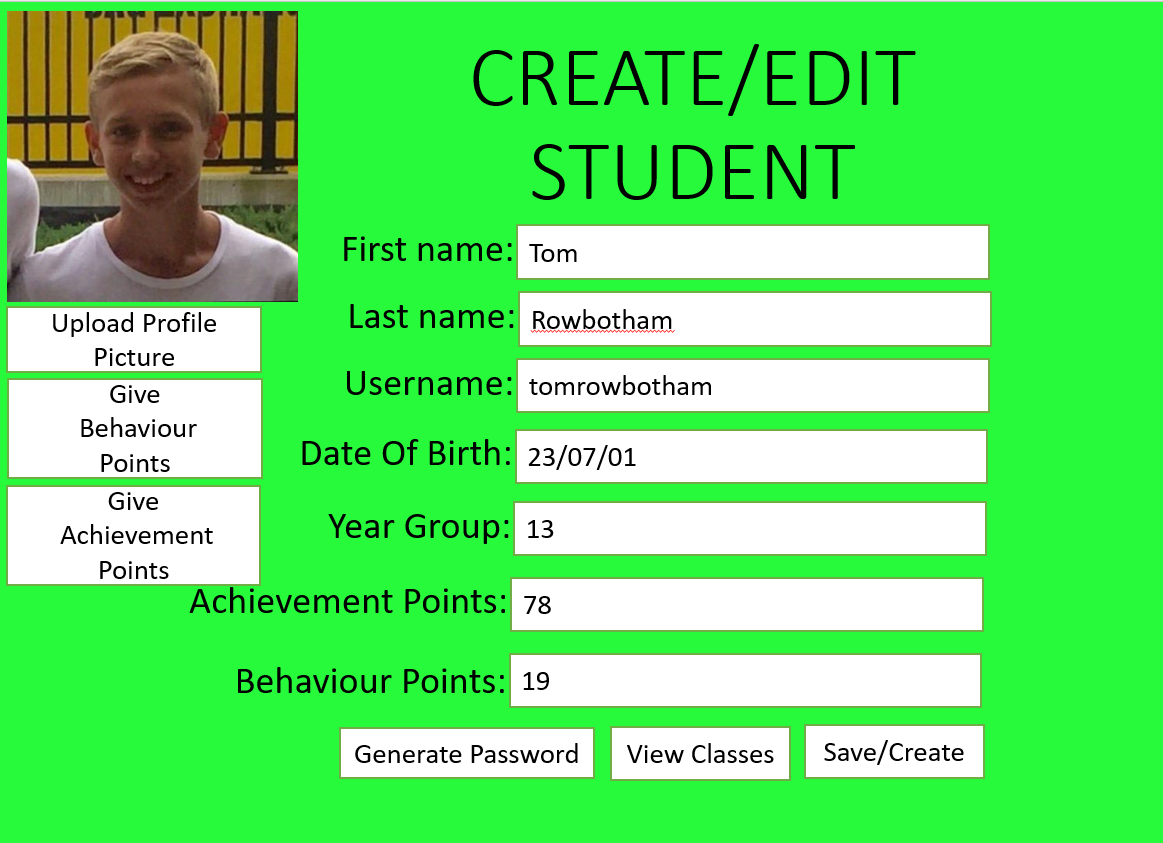
This is my adaption for a teacher view. This is still the same window – however as teachers do not have behaviour and achievement points and do not have anything else relevant enough to be summarised on their home page – I have just made the window smaller.   


Finally – this is the admins homepage. As there is not a lot stored on the database about the administrators – there was not a lot I could put on the homepage except from a welcoming message and the drop-down menus. As shown in the different menus – each have different drop down menus each type of user has different functions they can access – hence, administrators not having the homework tab.

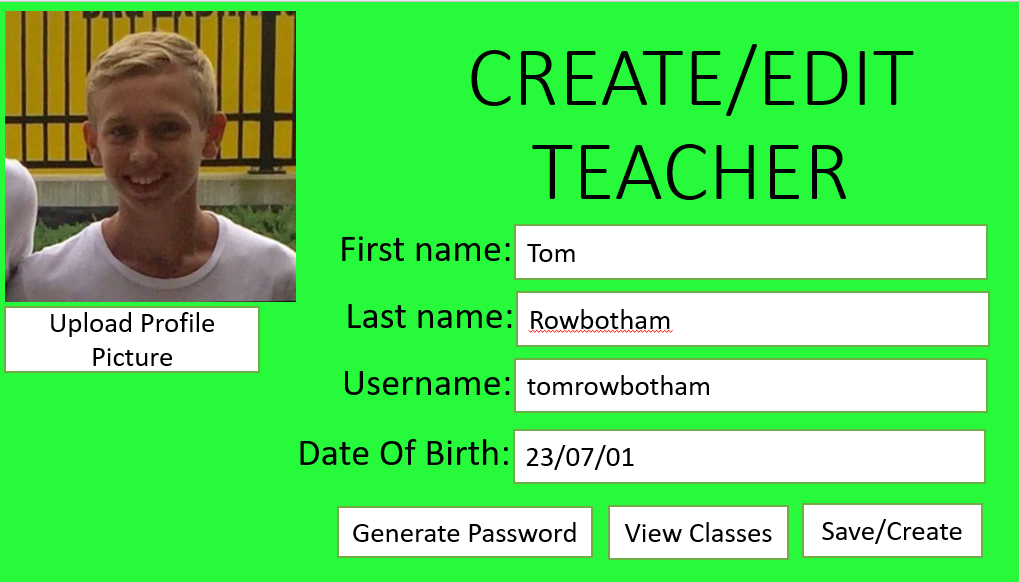
I feel like everything on this homepage is clear and it should be very easy to navigate around – so that all users will be able to understand.

## Creating a user

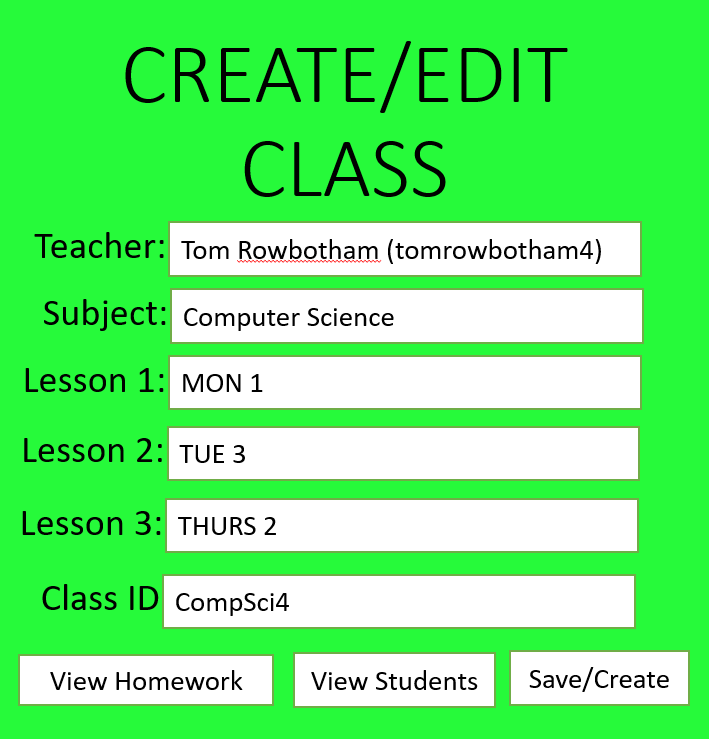
This window will be for creating and editing users. As shown below is a sample of what it will look like – although it will adapt based on the situation the window is used in. This window will not only show all the most relevant details but the buttons will lead



Upon creation the image top left will be a sample placeholder image – however the button below will allow the user to open file explorer and select what image they choose for the picture to be which the user then can preview. The title of the window – will be create student if it is an admin creating the user for the first time – or if somebody is going back to the user I.e to change the picture etc. Then the title will change to edit followed by the type of user being used – which will either be admin, student or teacher. The give behaviour points and achievement points buttons will then open the appropriate window. The labels to the left of the boxes are all clear so that the user knows which box is for what information. The first name and last name will be boxes the user can type in – and the username will be non-editable and will be generated by the program based on the first name and last name. The date of birth will have a calendar widget so that the user can either enter the date of birth by changing the numbers – or open the calendar and select the date. This will have a maximum and minimum date of birth to stop any outliers. For students only they will have a drop-down combo box with two choices – year 12 or year 13. The behaviour points and achievement points will too be un-editable – but will change automatically if any are added. Generate password will generate a new random password that will only be shown once. View classes will only be shown for teachers and students which will take them to a window to show their classes. And save button will append the changes made and create will add the user to the database. The create button will then close and open up the same edit user window.

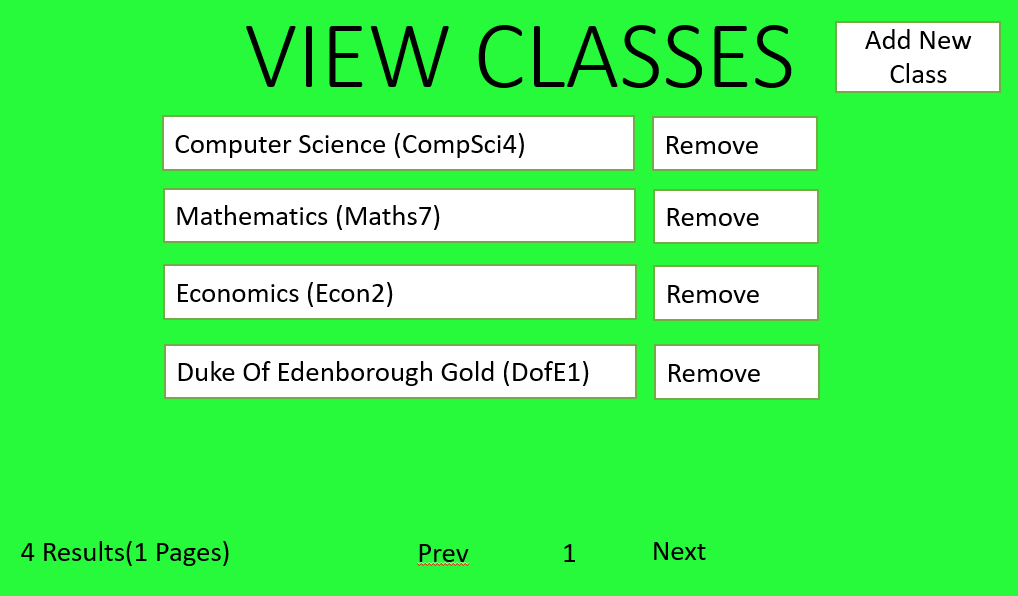


## Creating a Class



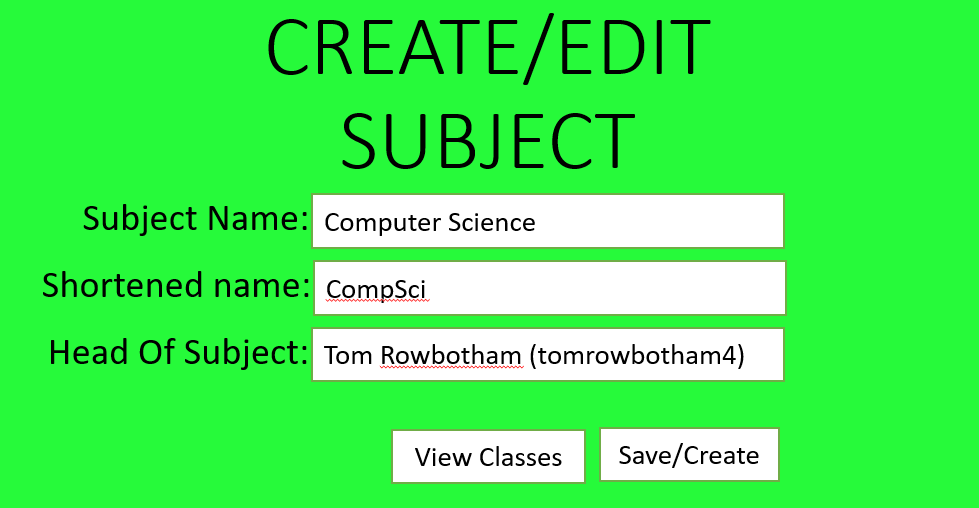
When creating or editing a class this is the window that will appear. The clear title at the top will remind the user this – and if the user clicked create class it will just say “create” however if it is an existing class the title will be “Edit Class”. The labels on the left are to give the user clear instructions of what each box are for. The teacher box will be a drop down box with every teacher stored on the database. This will have the teachers full name with their username following. The subject will also be a drop down box with all the subjects stored on the database. Each lesson will also be a drop down box, where every possible lesson is shown – and also an option for N/A if the class only has 2 lessons a week for example. The class id box will be un-editable and will be generated automatically upon creation. The view homework button and view students will only be generated if the class is edited and already created. They will open the appropriate windows. The create button will add the class to the database, generate a classid and open the edit class window.

## View Classes



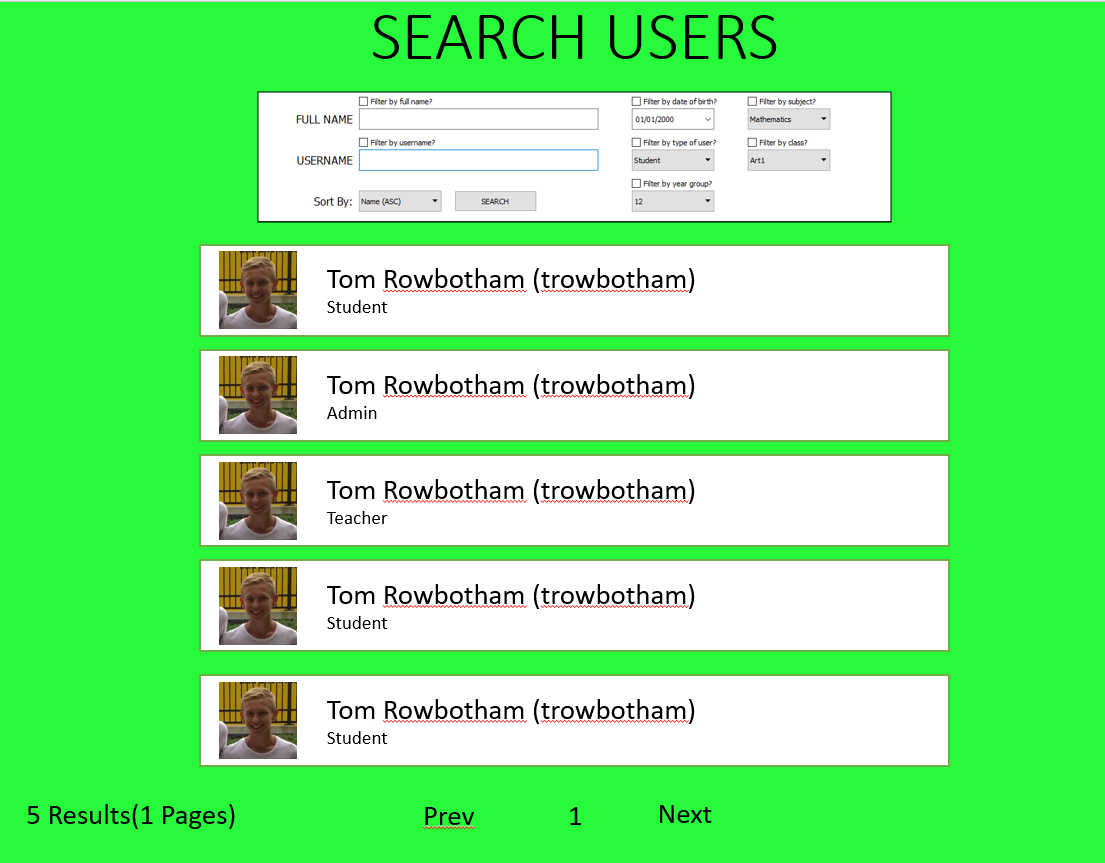
The view classes window will show all the classes a teacher/student is in. The title will be centred and big to make sure the user is aware of what window they are on. Then will be a list of classes the user is in/teaches. Each box will have the subject name and class id in brackets and clicking on it will open the subject page. Clicking remove will remove them from the class. Add new class will take them to a similar page but with every class from every subject which you can then filter by class. On the bottom there will be a little information text, saying how many classes the user is in and how many pages it covers. Then there will be a page number with next and previous buttons. This will be disabled if there is only one page.

## Creating a Subject



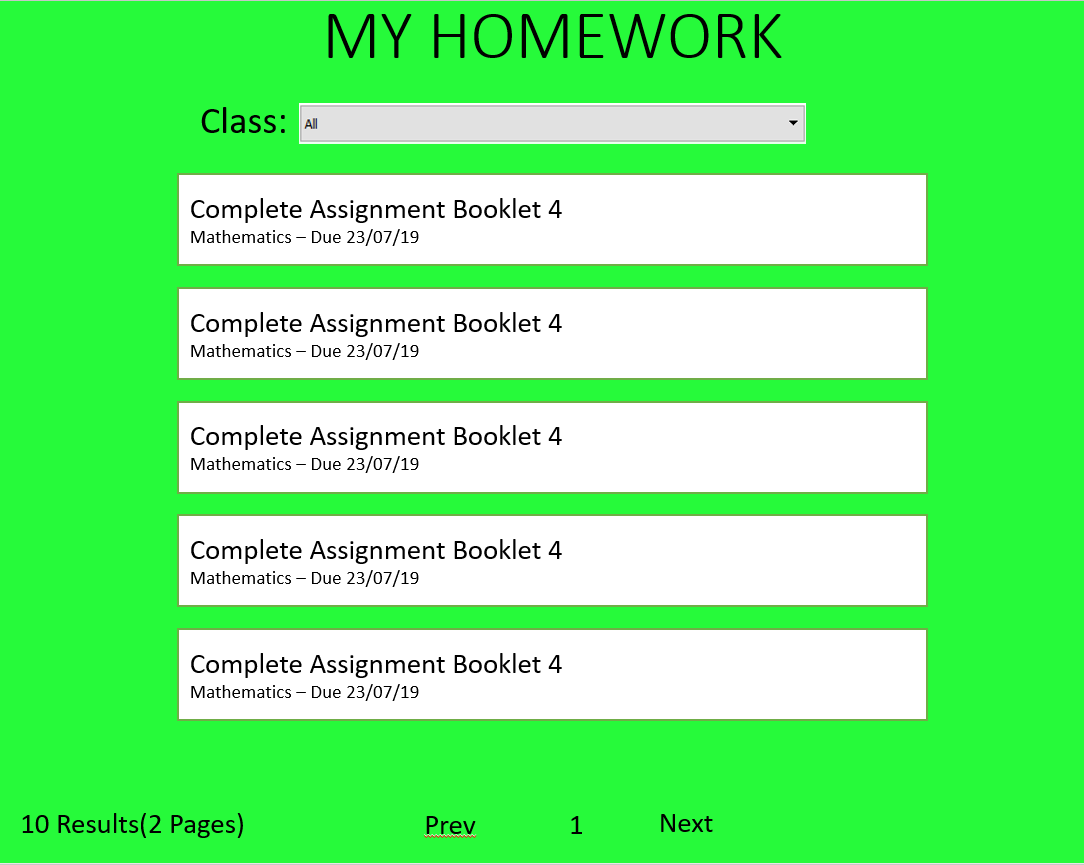
This window will be when a user decides to create/edit an existing subject. There will be a clear title stating whether they are creating or editing the subject. Each label describes what the box does. The subject name and shortened name will both be user inputs. The shortened name will be limited to just 10 letters. The head of subject will be a drop down box – with every teacher stored on the database. The view classes button will only be shown if the subject has been prior created. The create button will add the subject to the database and then open the edit subject window. By clicking the save button it will append the changes to the database. Before changes are made a window will pop up asking if they are sure they want to save/create and after changes are made a window will pop up saying save successful.

## Searching Users



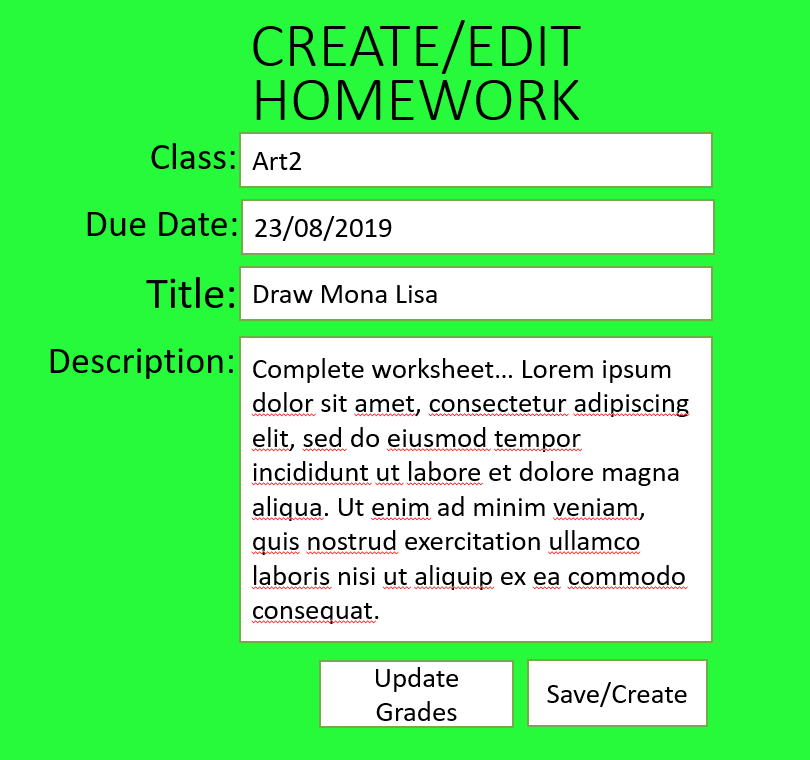
When searching for a user this window will pop up. This will collect every user off the database. There is a clear title on the top. Underneath there will be a filter box. For full name and username – there will be a small but readable check box followed by text saying “Filter by username”. Then to the under will be a label saying what the box needs to contain and then the editable box where the user will enter their query. This will only be editable if the box is checked. Next is the filter for date of birth. This will be a checkbox asking if the user wants to “filter by date of birth” followed by underneath calendar widget where to user can type the date manually or select the date from a calendar. Under will be a check box saying “filter by type of user” and with that will be a combo box containing “Teacher”,”Student”,”Admin”. There will be the same for year group but with “Year 12” and “Year 13” as options. You will also be able to filter by subjects where there will be a checkbox and then a drop down box with every subject. This will select every user based on classes they are in and if any of their classes are in that subject. The same will apply for classes where every class will be listed on a drop down box with a checkbox saying “Filter by class” above. Then bottom left there will be a sortby drop down box. This will default be on Name(Asc) however, the user will be able to filter by Date Of Birth aswell. It will give the option to do both ascending and descending. On the right of that will be the search button. This will fetch all users from database and apply all the filters that are checked. This will then sort the results and present them as shown in the screenshot. Each user will have a box, with their image, fullname and type. Clicking on their button will take the user to their profile. The results label will show how many results and how many pages there are. The page number will say what page the user is on and the previous and next buttons will allow you to navigate through the users. These will be disabled if there is no previous or next page.

## View Homeworks



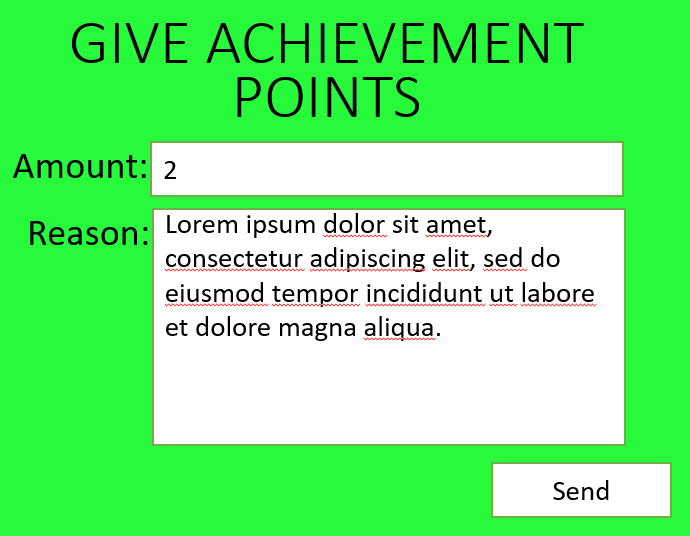
This window will show all the homework for the user as shown by the title. Underneath is a clear label stating that the ComboBox holds the classes and the drop down box will hold all the classes the user is in. When changed it will signal all the boxes below to change too. Each box will hae the title of the homework, the subject and due date. By clicking on it, it will take the user to the homework page which would give the user more details. The results label will show how many results and how many pages there are. The page number will say what page the user is on and the previous and next buttons will allow you to navigate through the users. These will be disabled if there is no previous or next page.

## Creating homework



This is my create/edit homework window. Here, there will be 4 different boxes all clearly labelled by the text on the left. The first one will be a drop down box. This will contain all the users classes they teach and the selected one will the the class the homework is set for. The due date will be a calendar widget. This will be able to set for any time in the future. The title will be editable by the user based on input and will have a maximum length. Then the description will have a longer max length (around 1000 words). This will be so the user can explain further about what the homework entails. The update grades button will take the user to a window that will let the user update each users grades. The create button will add the homework to the database and then open the edit window. The save button will commit the changes.

## Creating achievement/behaviour points



This is a much simpler window for achievement and behaviour points. The title will be changed depending on what instance it is opened in. The amount will be editable by number (max 9) and will also have arrows on the right to increase and decrease by 1 point. The reason will make the achievement point more identifiable in the database. The send button will then close the window and add the achievement points to the user.

# Technical Solution

# Code

\*Note – program I used to paste code added some indentations that were not actually in the code.   
Full code can be found <https://github.com/tomrowbo/School-Database>

## Main.py

1. Variable formatting
2. #Camelcasing for all variables.
3. #Use underscores for all methods


7. #These are all the libraries that are being used in my program.
8. **from** PyQt4 **import** QtCore, QtGui #PyQt4 is the library I use to create the entire GUI and interactions with it.
9. #import MySQLdb
10. **import** sqlite3 #SQLite is the SQL interface for python where this allows me to interact with my database.
11. **import** hashlib #Hashlib is the default python library for hashing. This is useful for me as
12. **import** sys #Used to interact with the system. Examples where this is used is sys.exit which allows me to close the program.
13. #from adminmain import Ui\_AdminWindow
14. **from** userclasses **import** \* #This is my own library. This imports the user classes.
15. **import** random #Used for generating random items. Ie. Random password
16. **import** string #Has a few useful functions that help dealing with strings
17. **from** shutil **import** copyfile #Used for copying a file from a location on the users
18. #PC onto the database server
19. **import** os #Used for a couple useful functions. ie. Find program file path and isFile.
20. **from** difflib **import** SequenceMatcher #Good for searches and finding similar results rather exact search results
21. **from** zxcvbn **import** zxcvbn #Password strength tester
22. #These two are used for interacting with dates.
23. **import** time
24. **import** datetime
26. #Finding the file path of this program.
27. path = os.getcwd()
29. #This is used if you are connecting to a database server
30. ##host = "localhost"
31. ##username = "default"
32. ##password = "tomrowbotham"
33. #conn = MySQLdb.connect(host,username,password,db = "users")
35. #Connecting to the database
36. conn = sqlite3.connect("school.db")
37. c = conn.cursor()
39. #Needed for all PyQt programs to ensure everything is working.
40. **try**:
41. \_fromUtf8 = QtCore.QString.fromUtf8
42. **except** AttributeError:
43. **def** \_fromUtf8(s):
44. **return** s
46. **try**:
47. \_encoding = QtGui.QApplication.UnicodeUTF8
48. **def** \_translate(context, text, disambig):
49. **return** QtGui.QApplication.translate(context, text, disambig, \_encoding)
50. **except** AttributeError:
51. **def** \_translate(context, text, disambig):
52. **return** QtGui.QApplication.translate(context, text, disambig)
54. #Global Background for all windows
55. css = \_fromUtf8("QMainWindow {\n"
56. "background-color: qlineargradient(spread:pad, x1:0.494364, y1:0.806, x2:0.471, y2:0.142045, stop:0 rgba(17, 255, 56, 255), stop:1 rgba(255, 255, 255, 255));}")

59. #These are all the global fonts. Instead of redefining what font I want to use everytime I create an object with text
60. #It will be more efficient to have a long list of all the fonts used. Every font follows a similar style.
61. #The font names are pretty self-explanatory on where they will be used.
62. labelfont = QtGui.QFont()
63. labelfont.setPointSize(12)
65. labelfont2 = QtGui.QFont()
66. labelfont2.setFamily(\_fromUtf8("Gadugi"))
67. labelfont2.setPointSize(16)
68. labelfont2.setBold(False)
69. labelfont2.setWeight(50)
71. labelfont3 = QtGui.QFont()
72. labelfont3.setFamily(\_fromUtf8("Gadugi"))
73. labelfont3.setBold(False)
74. labelfont3.setWeight(50)
75. labelfont3.setPointSize(12)
77. labelfont4 = QtGui.QFont()
78. labelfont4.setFamily(\_fromUtf8("Gadugi"))
79. labelfont4.setBold(False)
80. labelfont4.setWeight(50)
81. labelfont4.setPointSize(10)


85. titlefont = QtGui.QFont()
86. titlefont.setFamily(\_fromUtf8("Microsoft New Tai Lue"))
87. titlefont.setPointSize(36)
88. titlefont.setBold(True)
89. titlefont.setWeight(75)
91. titlefont2 = QtGui.QFont()
92. titlefont2.setFamily(\_fromUtf8("Arial Black"))
93. titlefont2.setPointSize(24)
94. titlefont2.setBold(True)
95. titlefont2.setWeight(75)
97. titlefont3 = QtGui.QFont()
98. titlefont3.setPointSize(36)
100. titlefont4 = QtGui.QFont()
101. titlefont4.setFamily(\_fromUtf8("Quicksand Light"))
102. titlefont4.setPointSize(18)
104. headerfont = QtGui.QFont()
105. headerfont.setFamily(\_fromUtf8("Gadugi"))
106. headerfont.setPointSize(12)
107. headerfont.setBold(True)
108. headerfont.setWeight(75)
110. normalfont = QtGui.QFont()
111. normalfont.setFamily(\_fromUtf8("Gadugi"))
112. normalfont.setPointSize(12)
113. normalfont.setBold(False)
114. normalfont.setWeight(50)
116. smalltitlefont = QtGui.QFont()
117. smalltitlefont.setFamily(\_fromUtf8("Gadugi"))
118. smalltitlefont.setPointSize(15)
119. smalltitlefont.setBold(True)
120. smalltitlefont.setWeight(75)
122. typefont = QtGui.QFont()
123. typefont.setFamily(\_fromUtf8("MS Shell Dlg 2"))
124. typefont.setPointSize(12)
126. numberfont = QtGui.QFont()
127. numberfont.setFamily(\_fromUtf8("OCR A Std"))
128. numberfont.setPointSize(26)
130. #These are the regex equations. This is used for validating inputs and only allowing
131. #A user to enter valid keys.
133. #Allows a-z and A-Z and -
134. regex=QtCore.QRegExp("[a-z-A-Z]+")
135. lettersvalidator = QtGui.QRegExpValidator(regex)
137. #Allows A-Z a-z and space bar and -
138. regex=QtCore.QRegExp("[a-z-A-Z ]+")
139. lettersandspacevalidator = QtGui.QRegExpValidator(regex)
141. #Allows A-Z a-z and 0-9 and -
142. regex=QtCore.QRegExp("[a-z-A-Z-0-9]+")
143. lettersandnumbersvalidator = QtGui.QRegExpValidator(regex)
145. #Only allows valid email addresses.
146. regex=QtCore.QRegExp(r"(^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)")
147. emailvalidator = QtGui.QRegExpValidator(regex)

1

1. #This class is the original window that opens when the program is run.
2. #Here the user will sign in.
3. **class** Ui\_MainWindow(object):
4. #SetupUi is for creating all the objects in the window. This will include all the labels,
5. #images, etc. This will basically be used the same way a \_\_init\_\_ function is used and will
6. #be used when creating the window for the first time.
7. **def** setupUi(self, MainWindow):
8. #Creating Main Window
9. MainWindow.setObjectName(\_fromUtf8("MainWindow"))
10. MainWindow.resize(800, 550)
11. MainWindow.setAutoFillBackground(False)
13. #Style Sheet (CSS)
14. MainWindow.setStyleSheet(css)
16. #Central Widget - Contains all buttons and labels
17. self.centralwidget = QtGui.QWidget(MainWindow)
18. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
20. #Username Label
21. self.usernameLabel = QtGui.QLabel(self.centralwidget)
22. self.usernameLabel.setGeometry(QtCore.QRect(190, 190, 131, 51))
23. self.usernameLabel.setFont(labelfont)
24. self.usernameLabel.setObjectName(\_fromUtf8("usernamelabel"))
26. #Username Line Edit(Where you enter the username)
27. self.usernameEdit = QtGui.QLineEdit(self.centralwidget)
28. self.usernameEdit.setGeometry(QtCore.QRect(300, 200, 351, 31))
29. self.usernameEdit.setFont(labelfont)
30. self.usernameEdit.setObjectName(\_fromUtf8("usernameEdit"))
31. self.usernameEdit.setValidator(lettersandnumbersvalidator)
33. #Password Label
34. self.passwordLabel = QtGui.QLabel(self.centralwidget)
35. self.passwordLabel.setGeometry(QtCore.QRect(190, 260, 131, 51))
36. self.passwordLabel.setFont(labelfont)
37. self.passwordLabel.setObjectName(\_fromUtf8("passwordlabel"))
39. #Password Line Edit
40. self.passwordEdit = QtGui.QLineEdit(self.centralwidget)
41. self.passwordEdit.setEchoMode(QtGui.QLineEdit.Password)
42. self.passwordEdit.setGeometry(QtCore.QRect(300, 270, 351, 31))
43. self.passwordEdit.setFont(labelfont)
44. self.passwordEdit.setObjectName(\_fromUtf8("passwordEdit"))
46. #Incorrect Label
47. self.incorrectLabel = QtGui.QLabel(self.centralwidget)
48. self.incorrectLabel.setGeometry(QtCore.QRect(150, 150, 600, 51))
49. self.incorrectLabel.setFont(labelfont)
50. self.incorrectLabel.setObjectName(\_fromUtf8("incorrectLabel"))
51. self.incorrectLabel.setStyleSheet("color:red")
52. self.incorrectLabel.hide()
54. #Login Button
55. self.loginBtn = QtGui.QPushButton(self.centralwidget)
56. self.loginBtn.setGeometry(QtCore.QRect(485, 337, 75, 27))
57. self.loginBtn.setFont(labelfont)
58. self.loginBtn.setObjectName(\_fromUtf8("loginBtn"))
59. self.loginBtn.setShortcut("Enter")
60. ########################Login Event###########################
61. self.loginBtn.clicked.connect(self.login\_)
62. ##############################################################
64. #Close Button
65. self.closeBtn = QtGui.QPushButton(self.centralwidget)
66. self.closeBtn.setGeometry(QtCore.QRect(575, 337, 75, 27))
67. self.closeBtn.setFont(labelfont)
68. self.closeBtn.setObjectName(\_fromUtf8("closeBtn"))
69. #######################Close Event###########################
70. self.closeBtn.clicked.connect(self.close\_app)
71. ##############################################################

74. #School Logo
75. self.logo = QtGui.QLabel(self.centralwidget)
76. self.logo.setGeometry(QtCore.QRect(239, 0, 321, 181))
77. self.logo.setText(\_fromUtf8(""))
78. self.logo.setPixmap(QtGui.QPixmap(\_fromUtf8("logo-609437332.png")))
79. self.logo.setScaledContents(False)
80. self.logo.setAlignment(QtCore.Qt.AlignCenter)
81. self.logo.setObjectName(\_fromUtf8("logo"))
83. MainWindow.setCentralWidget(self.centralwidget)
85. #Menu Bar
86. self.menuBar = QtGui.QMenuBar(MainWindow)
87. self.menuBar.setGeometry(QtCore.QRect(0, 0, 800, 21))
88. self.menuBar.setObjectName(\_fromUtf8("menuBar"))
89. MainWindow.setMenuBar(self.menuBar)
91. #Status Bar
92. self.statusBar = QtGui.QStatusBar(MainWindow)
93. self.statusBar.setObjectName(\_fromUtf8("statusBar"))
94. MainWindow.setStatusBar(self.statusBar)
96. self.retranslateUi(MainWindow)
97. QtCore.QMetaObject.connectSlotsByName(MainWindow)
99. #Retranslate Ui Method is used for changing the text in the window by calling this one function instead
100. #Of having to redefine every object again.
101. **def** retranslateUi(self, MainWindow):
102. MainWindow.setWindowTitle(\_translate("MainWindow", "Robert Smyth Login Form", None))
103. self.usernameLabel.setText(\_translate("MainWindow", "USERNAME", None))
104. self.incorrectLabel.setText(\_translate("MainWindow", "Error: User not found. Please verify details are correct and CAPS LOCK is not on.", None))
105. self.passwordLabel.setText(\_translate("MainWindow", "PASSWORD", None))
106. self.loginBtn.setText(\_translate("MainWindow", "Login", None))
107. self.closeBtn.setText(\_translate("MainWindow", "Close", None))
108. MainWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
109. #Sets the entire program style.
110. QtGui.QApplication.setStyle(QtGui.QStyleFactory.create("Windows Vista"))

113. #Close app function. When pressed the program will close and the user will no longer by able to interact with it.
114. **def** close\_app(self):

2

1. #QMessage Box is a small pop up dialogue window - where I am able to display a small message and the option for a
2. #Yes or No or Ok button.
3. choice = QtGui.QMessageBox.question(MainWindow, "Close Application",
4. "Are you sure you would like to quit?",QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)
5. **if** choice == QtGui.QMessageBox.Yes:
6. #Sys.exit closes the program.
7. sys.exit()
8. **else**:
9. **pass**
11. #Login function
12. **def** login\_(self):
13. #Retrieving inputs from the window and storing them as local variables.
14. self.username = self.usernameEdit.text().lower()
15. self.password = self.passwordEdit.text()
16. self.password = hashing(self.password)
18. #Fetching user from database
19. c.execute("SELECT \* FROM users WHERE username=:username AND password=:password ORDER BY username ASC",
20. {"username":self.username,"password":self.password})
21. data = c.fetchone()
23. #None will be returned if there is no user with that username or password.
24. **if** data == None:
25. self.incorrectLabel.show()
27. #If the user is found.
28. **else**:
29. #Current user will be the only variable in my entire program that is global. This is important to be global as it will
30. #be needed in almost every window. Other variables that will need to be passed from one window to another will be passed
31. #though parameters.
32. **global** currentUser
34. #Data[6] is the user type
35. **if** data[6] == "Student":
36. #If the user is a student then they will have more information than the default user. This is the extra data
37. #needs to be stored on a seperate table. This includes the students year group, achievement points and
38. #Behaviour points
39. c.execute("SELECT yeargroup,achievementpoints,behaviourpoints FROM student WHERE username=:username",{"username":self.username})
40. details = c.fetchone()
42. #Setting the user as a student class.
43. currentUser = Student(data[0],data[1],data[2],data[3],data[4],data[5],data[6],details[0],data[7],details[1],details[2])
44. **else**:
45. #If the user is not a student then save as a default User.
46. currentUser = User(data[0],data[1],data[2],data[3],data[4],data[5],data[6],data[7])
48. self.open\_user()
50. #Open user is the function that closes the login page and opens the welcome page
51. **def** open\_user(self):
52. MainWindow.hide()
54. #Creating the welcome page
55. self.window = QtGui.QMainWindow()
56. self.ui = Ui\_WelcomeWindow()
57. self.ui.setupUi(self.window)
58. self.window.show()

3

1. **class** Ui\_WelcomeWindow(object):
2. **def** setupUi(self, MainWindow):
3. self.window = MainWindow
4. #Creating Main Window
5. MainWindow.setObjectName(\_fromUtf8("MainWindow"))
7. #A student has homework and achievement and behaviour points - whereas Teachers don't have behaviour/achievement points
8. # and admins dont have points nor homework. Because of this - each welcome screen will be slightly different. Instead of having
9. #a big empty window - I believe it is better for the user just to have a shorter window. This is why for each type of user
10. #they have different heights.
11. **if** currentUser.type == "Admin":
12. height = 185
13. **elif** currentUser.type == "Student":
14. height = 593
15. **else**:
16. height = 400
17. MainWindow.resize(640, height)
18. MainWindow.setStyleSheet(\_fromUtf8("QMainWindow {\n"
19. "background-color: qlineargradient(spread:pad, x1:0.494364, y1:0.806, x2:0.471, y2:0.142045, stop:0 rgba(17, 255, 56, 255), stop:1 rgba(255, 255, 255, 255));}\n"
20. "QPushButton{background: transparent;"
21. "border: 1px solid black;}"))
22. self.centralwidget = QtGui.QWidget(MainWindow)
23. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
25. #Profile Picture
26. self.profilePic = QtGui.QLabel(self.centralwidget)
27. self.profilePic.setGeometry(QtCore.QRect(80, 10, 126, 126))
28. self.profilePic.setText(\_fromUtf8(""))
30. #If the program can find the users profile picture then set image as profile picture.
31. **if** os.path.isfile(path + currentUser.pic):
32. self.profilePic.setPixmap(QtGui.QPixmap(\_fromUtf8(path + currentUser.pic)))
33. **else**:
34. self.profilePic.setPixmap(QtGui.QPixmap(\_fromUtf8("placeholder.png")))
35. self.profilePic.setScaledContents(True)
36. self.profilePic.setObjectName(\_fromUtf8("profilePic"))

39. #Welcome Label
40. self.welcomeLabel = QtGui.QLabel(self.centralwidget)
41. self.welcomeLabel.setGeometry(QtCore.QRect(220, 0, 341, 141))
42. self.welcomeLabel.setFont(titlefont3)
43. self.welcomeLabel.setAlignment(QtCore.Qt.AlignCenter)
44. self.welcomeLabel.setObjectName(\_fromUtf8("welcomeLabel"))

47. #Homework Title - For Teachers and Admins This Will Be Labelled As Notices
48. self.homeworkTitle = QtGui.QLabel(self.centralwidget)
49. self.homeworkTitle.setGeometry(QtCore.QRect(80, 132, 481, 41))
50. self.homeworkTitle.setFont(titlefont4)
51. self.homeworkTitle.setObjectName(\_fromUtf8("homeworkTitle"))

54. #View Homework/Notices Button
55. self.viewButton = QtGui.QPushButton(self.centralwidget)
56. self.viewButton.setGeometry(QtCore.QRect(460, 330, 111, 23))
57. font = QtGui.QFont()
58. font.setUnderline(True)
59. self.viewButton.setFont(font)
60. self.viewButton.setLayoutDirection(QtCore.Qt.LeftToRight)
61. #This button will be shown differently to default buttons - as it will be blue underlined and have a
62. #transparent background. This is why I have created a seperate CSS for this button.
63. self.viewButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;\n"
64. "color: rgb(0, 0, 255);\n"
65. "text-decoration: underline;\n"
66. "border: 0px;}"))
67. self.viewButton.setObjectName(\_fromUtf8("viewButton"))
68. self.viewButton.clicked.connect(self.view\_homework)

71. ###################### RESULTS #######################################
72. #Here will show the top 3 results of homework. Each result will be made up
73. #of a button, title and description. They will be combined into one class object.
75. ## RESULT ONE
77. #Top button
78. self.topButton = QtGui.QPushButton(self.centralwidget)
79. self.topButton.setGeometry(QtCore.QRect(80, 180, 481, 51))
80. self.topButton.setText(\_fromUtf8(""))
81. self.topButton.setObjectName(\_fromUtf8("topButton"))
83. #Result 1 Title
84. self.topTitle = QtGui.QLabel(self.centralwidget)
85. self.topTitle.setGeometry(QtCore.QRect(90, 187, 461, 21))
86. self.topTitle.setFont(labelfont3)
87. self.topTitle.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
88. self.topTitle.setObjectName(\_fromUtf8("topTitle"))
90. #Result 1 Description
91. self.topDesc = QtGui.QLabel(self.centralwidget)
92. self.topDesc.setGeometry(QtCore.QRect(90, 207, 461, 21))
93. self.topDesc.setFont(labelfont4)
94. self.topDesc.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
95. self.topDesc.setObjectName(\_fromUtf8("topDesc"))
97. #WindowsButtons is another class. This will combine all of the objects.
98. self.top1 = WindowButtons(self.topButton,self.topTitle,self.topDesc,"Homework")

101. ## RESULT TWO
103. self.topTitle\_2 = QtGui.QLabel(self.centralwidget)
104. self.topTitle\_2.setGeometry(QtCore.QRect(90, 237, 461, 21))
105. self.topTitle\_2.setFont(labelfont3)
106. self.topTitle\_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
107. self.topTitle\_2.setObjectName(\_fromUtf8("topTitle\_2"))
109. self.topDesc\_2 = QtGui.QLabel(self.centralwidget)
110. self.topDesc\_2.setGeometry(QtCore.QRect(90, 257, 461, 21))
111. self.topDesc\_2.setFont(labelfont4)
112. self.topDesc\_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
113. self.topDesc\_2.setObjectName(\_fromUtf8("topDesc\_2"))
115. self.topButton\_2 = QtGui.QPushButton(self.centralwidget)
116. self.topButton\_2.setGeometry(QtCore.QRect(80, 230, 481, 51))
117. self.topButton\_2.setText(\_fromUtf8(""))
118. self.topButton\_2.setObjectName(\_fromUtf8("topButton\_2"))
120. self.top2 = WindowButtons(self.topButton\_2,self.topTitle\_2,self.topDesc\_2,"Homework")
122. ##RESULT THREE
123. self.topTitle\_3 = QtGui.QLabel(self.centralwidget)
124. self.topTitle\_3.setGeometry(QtCore.QRect(90, 287, 461, 21))
125. self.topTitle\_3.setFont(labelfont3)
126. self.topTitle\_3.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
127. self.topTitle\_3.setObjectName(\_fromUtf8("topTitle\_3"))
129. self.topDesc\_3 = QtGui.QLabel(self.centralwidget)
130. self.topDesc\_3.setGeometry(QtCore.QRect(90, 307, 461, 21))
131. self.topDesc\_3.setFont(labelfont4)
132. self.topDesc\_3.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
133. self.topDesc\_3.setObjectName(\_fromUtf8("topDesc\_3"))
135. self.topButton\_3 = QtGui.QPushButton(self.centralwidget)
136. self.topButton\_3.setGeometry(QtCore.QRect(80, 280, 481, 51))
137. self.topButton\_3.setText(\_fromUtf8(""))
138. self.topButton\_3.setObjectName(\_fromUtf8("topButton\_3"))
140. self.top3 = WindowButtons(self.topButton\_3,self.topTitle\_3,self.topDesc\_3,"Homework")

143. #Achievement Point Box
145. #Number
146. self.achievementNumber = QtGui.QLabel(self.centralwidget)
147. self.achievementNumber.setGeometry(QtCore.QRect(160, 410, 101, 61))
148. self.achievementNumber.setFont(numberfont)
149. self.achievementNumber.setAutoFillBackground(False)
150. self.achievementNumber.setStyleSheet(\_fromUtf8("QLabel{border: 1px solid black;\n"
151. "background-color: rgb(0, 255, 0);}\n"
152. ""))
153. self.achievementNumber.setAlignment(QtCore.Qt.AlignCenter)
154. self.achievementNumber.setObjectName(\_fromUtf8("achievementNumber"))
156. #Title
157. self.achievementTitle = QtGui.QLabel(self.centralwidget)
158. self.achievementTitle.setGeometry(QtCore.QRect(160, 370, 101, 41))
159. self.achievementTitle.setFont(typefont)
160. self.achievementTitle.setAutoFillBackground(False)
161. self.achievementTitle.setStyleSheet(\_fromUtf8("QLabel{border: 1px solid black;\n"
162. "background-color: rgb(255, 255, 255);}\n"
163. ""))
164. self.achievementTitle.setAlignment(QtCore.Qt.AlignCenter)
165. self.achievementTitle.setObjectName(\_fromUtf8("achievementTitle"))

168. #Behaviour Point Box
169. #Number
170. self.behaviourNumber = QtGui.QLabel(self.centralwidget)
171. self.behaviourNumber.setGeometry(QtCore.QRect(370, 410, 101, 61))
172. self.behaviourNumber.setFont(numberfont)
173. self.behaviourNumber.setAutoFillBackground(False)
174. self.behaviourNumber.setStyleSheet(\_fromUtf8("QLabel{border: 1px solid black;\n"
175. "background-color: rgb(255, 0, 0);}"))
176. self.behaviourNumber.setAlignment(QtCore.Qt.AlignCenter)
177. self.behaviourNumber.setObjectName(\_fromUtf8("behaviourNumber"))
179. #Title
180. self.behaviourTitle = QtGui.QLabel(self.centralwidget)
181. self.behaviourTitle.setGeometry(QtCore.QRect(370, 370, 101, 41))
182. self.behaviourTitle.setFont(typefont)
183. self.behaviourTitle.setAutoFillBackground(False)
184. self.behaviourTitle.setStyleSheet(\_fromUtf8("QLabel{border: 1px solid black;\n"
185. "background-color: rgb(255, 255, 255);}\n"
186. ""))
187. self.behaviourTitle.setAlignment(QtCore.Qt.AlignCenter)
188. self.behaviourTitle.setObjectName(\_fromUtf8("behaviourTitle"))
190. #Sorting out alignment
191. self.topTitle.raise\_()
192. self.topDesc.raise\_()
193. self.topTitle\_2.raise\_()
194. self.topDesc\_2.raise\_()
195. self.topTitle\_3.raise\_()
196. self.topDesc\_3.raise\_()
197. self.topButton.raise\_()
198. self.topButton\_2.raise\_()
199. self.topButton\_3.raise\_()


203. ##        #Setting Menu Bar
204. MainWindow.setCentralWidget(self.centralwidget)
205. self.menubar = QtGui.QMenuBar(MainWindow)
206. self.menubar.setGeometry(QtCore.QRect(0, 0, 640, 21))
207. self.menubar.setObjectName(\_fromUtf8("menubar"))
208. self.statusbar = QtGui.QStatusBar(MainWindow)
209. self.statusbar.setObjectName(\_fromUtf8("statusbar"))

212. #Here I am adding everything to the menu bar at the top of the window
213. #Based on the user - will depend on what methods are shown
214. self.menuFile = QtGui.QMenu(self.menubar)
215. self.menuFile.setObjectName(\_fromUtf8("menuFile"))
216. self.menuClasses = QtGui.QMenu(self.menubar)
217. self.menuClasses.setObjectName(\_fromUtf8("menuClasses"))
218. self.menuHomework = QtGui.QMenu(self.menubar)
219. self.menuHomework.setObjectName(\_fromUtf8("menuHomework"))
220. self.menubar.addAction(self.menuFile.menuAction())
222. #Adding actions to the menu bar depending on the user type
223. **if** currentUser.type != "Admin":
224. self.menuHomework = QtGui.QMenu(self.menubar)
225. self.menuHomework.setObjectName(\_fromUtf8("menuHomework"))
226. self.menubar.addAction(self.menuHomework.menuAction())
228. #Creating an action - View Homework - which will be shown on the dropdown menu
229. #from the menu bar
230. self.viewHomework = QtGui.QAction(MainWindow)
231. self.viewHomework.setObjectName(\_fromUtf8("viewHomework"))
232. self.viewHomework.setStatusTip("View my homework")
233. self.menuHomework.addAction(self.viewHomework)
234. self.viewHomework.triggered.connect(self.view\_homework)

237. #Getting 3 most upcoming homeworks
238. #Today is the date today.
239. today = datetime.datetime.today().strftime('%Y-%m-%d')
240. data = []
242. **if** currentUser.type == "Student":
243. #Fetching all classes the student is in.
244. c.execute("SELECT classid FROM studentclass WHERE username = :username",
245. {"username":currentUser.username})
246. **else**:
247. #Fetching all classes the teacher is in.
248. c.execute("SELECT id FROM classes WHERE teacher = :username",
249. {"username":currentUser.username})
250. self.data = list(c.fetchall())
252. #For every class fetched from the database - fetch all future homeworks
253. **for** classid **in** self.data:
254. c.execute("SELECT \* FROM homework WHERE classid = :classid AND duedate >= :date",{"classid":classid[0],"date":today})
255. temp = c.fetchall()
256. **if** temp != None:
257. data.extend(list(temp))
258. #Sorting data by date. Showing the newest date first.
259. data.sort(key=**lambda** x: time.mktime(time.strptime(x[2],"%Y-%m-%d")))
260. **if** data == None:
261. data = []
262. self.homeworks = data

265. #QActions are all the options on the drop down menus
266. #QMenu are all the drop down names.
268. #Depending on the type of user, the choices will be different. Ie. Only an admin should be able to create user
269. #but they would not be able to view homework.
271. #Adding items to the menu bars
272. **if** currentUser.type != "Student":
273. self.menuSubjects = QtGui.QMenu(self.menubar)
274. self.menuSubjects.setObjectName(\_fromUtf8("menuSubjects"))
275. self.menuUsers = QtGui.QMenu(self.menubar)
276. self.menuUsers.setObjectName(\_fromUtf8("menuUsers"))
277. self.menubar.addAction(self.menuUsers.menuAction())
278. #View Users
279. self.viewUsers = QtGui.QAction(MainWindow)
280. self.viewUsers.setObjectName(\_fromUtf8("viewUsers"))
281. self.viewUsers.setStatusTip("View Users")
282. self.viewUsers.triggered.connect(self.view\_users)
284. **if** currentUser.type == "Teacher":
285. self.addHomework = QtGui.QAction(MainWindow)
286. self.addHomework.setObjectName(\_fromUtf8("addHomework"))
287. self.addHomework.setStatusTip("Create a new homework")
288. self.menuHomework.addAction(self.addHomework)
289. self.addHomework.triggered.connect(self.create\_homework)
290. self.menuUsers.addAction(self.viewUsers)

293. **if** currentUser.type == "Admin":
294. #Add Class
295. self.addClass = QtGui.QAction(MainWindow)
296. self.addClass.setObjectName(\_fromUtf8("addClass"))
297. self.addClass.setStatusTip("Create a new class")
298. self.addClass.triggered.connect(self.create\_class)
300. #Add Subject
301. self.addSubject = QtGui.QAction(MainWindow)
302. self.addSubject.setObjectName(\_fromUtf8("addSubject"))
303. self.addSubject.setStatusTip("Create a new subject")
304. self.addSubject.triggered.connect(self.create\_subject)
306. #Add Admin
307. self.addAdmin = QtGui.QAction(MainWindow)
308. self.addAdmin.setObjectName(\_fromUtf8("addAdmin"))
309. self.addAdmin.setStatusTip("Create a new Admin account")
310. self.addAdmin.triggered.connect(self.create\_admin)
312. #Add Teacher
313. self.addTeacher = QtGui.QAction(MainWindow)
314. self.addTeacher.setObjectName(\_fromUtf8("addTeacher"))
315. self.addTeacher.setStatusTip("Create a new Teacher account")
316. self.addTeacher.triggered.connect(self.create\_teacher)
318. #Add Student
319. self.addStudent = QtGui.QAction(MainWindow)
320. self.addStudent.setObjectName(\_fromUtf8("addStudent"))
321. self.addStudent.setStatusTip("Create a new Student account")
322. self.addStudent.triggered.connect(self.create\_student)
324. #View A List Of All Classes
325. self.viewClasses = QtGui.QAction(MainWindow)
326. self.viewClasses.setObjectName(\_fromUtf8("viewClasses"))
327. self.viewClasses.setStatusTip("View all classes")
328. self.viewClasses.triggered.connect(self.view\_classes)
329. self.menuClasses.addAction(self.viewClasses)
331. self.menuSubjects.addAction(self.addSubject)
332. self.menuUsers.addAction(self.addAdmin)
333. self.menuUsers.addAction(self.addTeacher)
334. self.menuUsers.addAction(self.addStudent)
335. self.menuUsers.addAction(self.viewUsers)
336. self.menuClasses.addAction(self.addClass)
337. self.menuClasses.addAction(self.viewClasses)
338. self.menubar.addAction(self.menuSubjects.menuAction())
340. **else**:
341. #View A List Of \*\*THE USERS\*\* Classes.
342. self.viewClasses = QtGui.QAction(MainWindow)
343. self.viewClasses.setObjectName(\_fromUtf8("viewClasses"))
344. self.viewClasses.setStatusTip("View your classes")
345. self.viewClasses.triggered.connect(self.class\_button)
346. self.menuClasses.addAction(self.viewClasses)
347. self.menubar.addAction(self.menuClasses.menuAction())

350. #Adding to the bars
352. #Log Out
353. self.actionLogOut = QtGui.QAction(MainWindow)
354. self.actionLogOut.setObjectName(\_fromUtf8("actionLogOut"))
355. self.actionLogOut.setShortcut("Ctrl+L")
356. self.actionLogOut.setStatusTip("Sign Out Of Your Account")
357. self.actionLogOut.triggered.connect(self.log\_out)
359. #Quit
360. self.actionQuit = QtGui.QAction(MainWindow)
361. self.actionQuit.setObjectName(\_fromUtf8("actionQuit"))
362. self.actionQuit.setShortcut("Ctrl+Q")
363. self.actionQuit.setStatusTip("Leave The App")
364. self.actionQuit.triggered.connect(self.close\_app)
366. #Reset password
367. self.resetPassword = QtGui.QAction(MainWindow)
368. self.resetPassword.setObjectName(\_fromUtf8("resetPassword"))
369. self.resetPassword.setStatusTip("Change your password.")
370. self.resetPassword.triggered.connect(self.reset\_password)

373. #Add actions onto menu
374. self.menuFile.addAction(self.actionLogOut)
375. self.menuFile.addAction(self.actionQuit)
376. self.menuFile.addAction(self.resetPassword)

379. MainWindow.setMenuBar(self.menubar)
380. MainWindow.setStatusBar(self.statusbar)

383. self.retranslateUi(MainWindow)
384. QtCore.QMetaObject.connectSlotsByName(MainWindow)
386. **def** retranslateUi(self, MainWindow):
388. MainWindow.setWindowTitle(\_translate("MainWindow", "Welcome Window", None))
389. self.welcomeLabel.setText(\_translate("MainWindow", "Welcome back,\n"
390. +str(currentUser.first), None))
391. self.menuFile.setTitle(\_translate("MainWindow", "File", None))
392. self.menuClasses.setTitle(\_translate("MainWindow", "Classes", None))
393. self.behaviourTitle.setText(\_translate("MainWindow","Behaviour\nPoints",None))
394. self.achievementTitle.setText(\_translate("MainWindow","Achievement\nPoints",None))
396. **if** currentUser.type != "Admin":
397. self.viewClasses.setText(\_translate("MainWindow", "View Classes", None))
398. self.menuHomework.setTitle(\_translate("MainWindow", "Homework", None))
399. self.viewHomework.setText(\_translate("MainWindow","View Homework",None))
400. self.homeworkTitle.setText(\_translate("MainWindow", "My Homework: ("+str(len(self.homeworks))+" Due Soon)", None))
401. self.viewButton.setText(\_translate("MainWindow", "Click to view more", None))
403. #Updating text on each homework.
404. **if** len(self.homeworks) >= 1:
405. #If the length of the title is greater than 45 characters then it will be shortened and ... will be added onto the
406. #end of the title.
407. **if** len(self.homeworks[0][3]) > 45:
408. self.homeworks[0][3]= self.homeworks[0][3][:42]+"..."
409. #This retranslateUi function - updates the title and description of the window box.
410. self.top1.retranslateUi(self.homeworks[0][3],
411. self.homeworks[0][1] + " - Due "+self.homeworks[0][2],
412. self.homeworks[0][0])
414. **if** len(self.homeworks) >= 2:
415. **if** len(self.homeworks[1][3]) > 45:
416. self.homeworks[1][3]= self.homeworks[1][3][:42]+"..."
417. self.top2.retranslateUi(self.homeworks[1][3],
418. self.homeworks[1][1] + " - Due "+self.homeworks[1][2],
419. self.homeworks[1][0])
421. **if** len(self.homeworks) >= 3:
422. **if** len(self.homeworks[2][3]) > 45:
423. self.homeworks[2][3]= self.homeworks[2][3][:42]+"..."
424. self.top3.retranslateUi(self.homeworks[2][3],
425. self.homeworks[2][1] + " - Due "+self.homeworks[2][2],
426. self.homeworks[2][0])
428. **if** currentUser.type != "Student":
429. **if** currentUser.type == "Admin":
430. self.addClass.setText(\_translate("MainWindow","Add Class",None))
431. self.addSubject.setText(\_translate("MainWindow","Add Subject",None))
432. self.addTeacher.setText(\_translate("MainWindow","Add Teacher",None))
433. self.addStudent.setText(\_translate("MainWindow","Add Student",None))
434. self.addAdmin.setText(\_translate("MainWindow","Add Admin",None))
435. self.viewClasses.setText(\_translate("MainWindow","View Classes",None))
436. **else**:
437. self.addHomework.setText(\_translate("MainWindow","Add Homework",None))
438. self.menuSubjects.setTitle(\_translate("MainWindow", "Subjects", None))
439. self.menuUsers.setTitle(\_translate("MainWindow", "Users", None))
440. self.viewUsers.setText(\_translate("MainWindow","View Users",None))
441. **else**:
442. self.achievementNumber.setText(\_translate("MainWindow",str(currentUser.achievementpoints),None))
443. self.behaviourNumber.setText(\_translate("MainWindow",str(currentUser.behaviourpoints),None))
444. self.actionQuit.setText(\_translate("MainWindow", "Quit Application", None))
445. self.actionLogOut.setText(\_translate("MainWindow", "Log Out", None))
446. self.resetPassword.setText(\_translate("MainWindow","Reset Password",None))
447. MainWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
449. #All these methods will open different windows.
450. **def** view\_users(self):
451. self.viewUserPage = EditWindow()
452. self.viewUserUi = Ui\_SearchUsers()
453. self.viewUserUi.setupUi(self.viewUserPage,"Search",None)
454. self.viewUserPage.show()
456. #View All Classes
457. **def** view\_classes(self):
458. self.showClassesPage = EditWindow()
459. self.showClassesUi = Ui\_ClassListWindow()
460. self.showClassesUi.setupUi(self.showClassesPage,"NULL","Teacher","Search")
461. self.showClassesPage.show()

464. **def** create\_admin(self):
465. self.createAdminPage = EditWindow()
466. self.createAdminUi = Ui\_EditUserWindow()
467. #The User class is default for any user. It is full of null values and all of those will be updated upon further creation.
468. self.createAdminUi.setupUi(self.createAdminPage,"Admin",User("NULL","NULL","NULL","NULL","NULL","NULL","Admin","NULL"))
469. self.createAdminPage.show()
471. **def** create\_teacher(self):
472. self.createTeacherPage = EditWindow()
473. self.createTeacherUi = Ui\_EditUserWindow()
474. self.createTeacherUi.setupUi(self.createTeacherPage,"Teacher",User("NULL","NULL","NULL","NULL","NULL","NULL","Teacher","NULL"))
475. self.createTeacherPage.show()
477. **def** create\_student(self):
478. self.createStudentPage = EditWindow()
479. self.createStudentUi = Ui\_EditUserWindow()
480. self.createStudentUi.setupUi(self.createStudentPage,"Student",Student("NULL","NULL","NULL","NULL","NULL","NULL","Student","NULL","NULL",”NULL”,”NULL”))
481. self.createStudentPage.show()
483. **def** create\_subject(self):
484. self.subjectPage = EditWindow()
485. self.subjectui = Ui\_EditSubjectWindow()
486. self.subjectui.setupUi(self.subjectPage,Subject("NULL","NULL","NULL"))
487. self.subjectPage.show()
489. **def** create\_class(self):
490. self.classPage = EditWindow()
491. self.classui = Ui\_CreateClassWindow()
492. self.classui.setupUi(self.classPage,Class("NULL","NULL","NULL","NULL","NULL","NULL","NULL","NULL"))
493. self.classPage.show()
495. **def** class\_button(self):
496. self.showClassesPage = EditWindow()
497. self.showClassesUi = Ui\_ClassListWindow()
498. self.showClassesUi.setupUi(self.showClassesPage,currentUser.username,currentUser.type,"List")
499. self.showClassesPage.show()

502. **def** create\_homework(self):
503. self.homeworkPage = EditWindow()
504. self.homeworkui = Ui\_HomeWorkWindow()
505. self.homeworkui.setupUi(self.homeworkPage,Homework("NULL","NULL","2001/01/01","NULL","NULL"))
506. self.homeworkPage.show()
508. **def** view\_homework(self):
509. self.viewHomeworkPage = EditWindow()
510. self.viewhomeworkui = Ui\_ViewHomeworkWindow()
511. self.viewhomeworkui.setupUi(self.viewHomeworkPage,"Future")
512. self.viewHomeworkPage.show()
514. #This will close the program
515. **def** close\_app(self):
516. choice = QtGui.QMessageBox.question(self.window,"Close Application","Are you sure you would like to quit?",QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)
517. **if** choice == QtGui.QMessageBox.Yes:
518. sys.exit()
519. **else**:
520. **pass**
522. #This will close the windows and open login page
523. **def** log\_out(self):
524. self.window.close()
525. self.open\_login()


529. **def** open\_login(self):
530. MainWindow.show()
531. ui.usernameEdit.clear()
532. ui.passwordEdit.clear()
534. **def** reset\_password(self):
535. self.passwordPage = EditWindow()
536. self.passwordUi = Ui\_PasswordWindow()
537. self.passwordUi.setupUi(self.passwordPage)
538. self.passwordPage.show()



543. #This class is to make buttons easier to control. Instead of treating each like individual objects
544. #This class allows them to be treated as a whole.

12

1. **class** WindowButtons():
2. **def** \_\_init\_\_(self,button,title,desc,typeOfBox):
3. self.id = "NULL"
4. self.button = button
5. self.title = title
6. self.desc = desc
7. self.typeOfBox = typeOfBox
9. self.button.clicked.connect(self.open\_window)
11. **def** retranslateUi(self,title,desc,id):
12. self.title.setText(\_translate("MainWindow",title,None))
13. self.desc.setText(\_translate("MainWindow",desc,None))
14. self.id = id
16. #Hides button
17. **def** hide\_all(self):
18. self.button.hide()
19. self.title.hide()
20. self.desc.hide()
22. #Shows button
23. **def** show\_all(self):
24. self.button.show()
25. self.title.show()
26. self.desc.show()
28. **def** open\_window(self):
29. **if** self.typeOfBox == "Homework":
30. c.execute("SELECT \* FROM homework WHERE homeworkid = :id",{"id":self.id})
31. lesson = c.fetchone()
32. #Open homework page upon button click.
33. self.homeworkPage = EditWindow()
34. self.homeworkui = Ui\_HomeWorkWindow()
35. self.homeworkui.setupUi(self.homeworkPage,Homework(lesson[0],lesson[1],lesson[2],lesson[3],lesson[4]))
36. self.homeworkPage.show()

39. **class** Ui\_PasswordWindow(object):

13

1. **def** setupUi(self, MainWindow):
2. self.window = MainWindow
3. #Creating Window
4. MainWindow.setObjectName(\_fromUtf8("MainWindow"))
5. MainWindow.resize(640, 480)
6. MainWindow.setStyleSheet(css)
7. self.centralwidget = QtGui.QWidget(MainWindow)
8. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
10. #Title
11. self.resetPasswordLabel = QtGui.QLabel(self.centralwidget)
12. self.resetPasswordLabel.setGeometry(QtCore.QRect(110, 60, 421, 91))
13. self.resetPasswordLabel.setFont(titlefont)
14. self.resetPasswordLabel.setTextFormat(QtCore.Qt.PlainText)
15. self.resetPasswordLabel.setAlignment(QtCore.Qt.AlignCenter)
16. self.resetPasswordLabel.setObjectName(\_fromUtf8("resetPasswordLabel"))
18. #Old Password Edit
19. self.oldEdit = QtGui.QLineEdit(self.centralwidget)
20. self.oldEdit.setGeometry(QtCore.QRect(210, 150, 351, 31))
21. self.oldEdit.setObjectName(\_fromUtf8("oldEdit"))
22. self.oldEdit.setEchoMode(QtGui.QLineEdit.Password)
24. #Old Password Label
25. self.oldLabel = QtGui.QLabel(self.centralwidget)
26. self.oldLabel.setGeometry(QtCore.QRect(60, 140, 141, 51))
27. self.oldLabel.setFont(labelfont)
28. self.oldLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
29. self.oldLabel.setObjectName(\_fromUtf8("oldLabel"))
31. #New Password Edit
32. self.newEdit = QtGui.QLineEdit(self.centralwidget)
33. self.newEdit.setGeometry(QtCore.QRect(210, 200, 351, 31))
34. self.newEdit.setObjectName(\_fromUtf8("newEdit"))
35. self.newEdit.setEchoMode(QtGui.QLineEdit.Password)
37. #New Password Label
38. self.newLabel = QtGui.QLabel(self.centralwidget)
39. self.newLabel.setGeometry(QtCore.QRect(60, 190, 141, 51))
40. self.newLabel.setFont(labelfont)
41. self.newLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
42. self.newLabel.setObjectName(\_fromUtf8("newLabel"))
44. #Confirm Password Edit
45. self.confirmEdit = QtGui.QLineEdit(self.centralwidget)
46. self.confirmEdit.setGeometry(QtCore.QRect(210, 250, 351, 31))
47. self.confirmEdit.setObjectName(\_fromUtf8("confirmEdit"))
48. self.confirmEdit.setEchoMode(QtGui.QLineEdit.Password)
50. #Confirm Password Label
51. self.confirmLabel = QtGui.QLabel(self.centralwidget)
52. self.confirmLabel.setGeometry(QtCore.QRect(30, 240, 171, 51))
53. self.confirmLabel.setFont(labelfont)
54. self.confirmLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
55. self.confirmLabel.setObjectName(\_fromUtf8("confirmLabel"))
57. #Save Button
58. self.saveBtn = QtGui.QPushButton(self.centralwidget)
59. self.saveBtn.setGeometry(QtCore.QRect(486, 300, 75, 27))
60. self.saveBtn.setFont(labelfont)
61. self.saveBtn.setObjectName(\_fromUtf8("saveBtn"))
62. self.saveBtn.clicked.connect(self.change\_pass)




68. MainWindow.setCentralWidget(self.centralwidget)
69. self.menubar = QtGui.QMenuBar(MainWindow)
70. self.menubar.setGeometry(QtCore.QRect(0, 0, 640, 21))
71. self.menubar.setObjectName(\_fromUtf8("menubar"))
72. MainWindow.setMenuBar(self.menubar)
73. self.statusbar = QtGui.QStatusBar(MainWindow)
74. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
75. MainWindow.setStatusBar(self.statusbar)
77. self.retranslateUi(MainWindow)
78. QtCore.QMetaObject.connectSlotsByName(MainWindow)
80. **def** retranslateUi(self, MainWindow):
81. MainWindow.setWindowTitle(\_translate("MainWindow", "Reset Password", None))
82. MainWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
83. self.resetPasswordLabel.setText(\_translate("MainWindow", "RESET PASSWORD", None))
84. self.oldLabel.setText(\_translate("MainWindow", "OLD PASSWORD:", None))
85. self.newLabel.setText(\_translate("MainWindow", "NEW PASSWORD:", None))
86. self.confirmLabel.setText(\_translate("MainWindow", "CONFIRM PASSWORD:", None))
87. self.saveBtn.setText(\_translate("MainWindow", "Save", None))

90. #This function tests the strength of the password. If the password is determined to be strong enough
91. #Then the password will be updated. If not - the user will be given a tip on how to improve.
92. **def** change\_pass(self):
93. old = self.oldEdit.text()
94. old = hashing(old)
95. **if** currentUser.password == old:
96. new = self.newEdit.text()
97. **if** new == self.confirmEdit.text():
98. strength = zxcvbn(new,[currentUser.first,currentUser.last,currentUser.dob])["score"]
99. **if** strength > 2:
100. currentUser.password = hashing(new)
101. c.execute("UPDATE users SET password = :password WHERE username =:username",
102. {"password":currentUser.password,"username":currentUser.username})
103. conn.commit()
104. QtGui.QMessageBox.question(self.window,"Saved","Save Successful",
105. QtGui.QMessageBox.Ok)
106. self.window.hide()
107. **return**
108. QtGui.QMessageBox.question(self.window,"Error","Error: Your password is too weak and could be guessed easy. Please try use a more complex password.",
109. QtGui.QMessageBox.Ok)
110. **return**
111. QtGui.QMessageBox.question(self.window,"Error","Error: Passwords did not match.",
112. QtGui.QMessageBox.Ok)
113. **return**
114. QtGui.QMessageBox.question(self.window,"Error","Error: Incorrect password.",
115. QtGui.QMessageBox.Ok)

8

2. **class** Ui\_CreateClassWindow(object):
3. **def** setupUi(self, CreateClassWindow,currentClass):
5. #Creating Main Window
6. CreateClassWindow.setObjectName(\_fromUtf8("CreateClassWindow"))
7. CreateClassWindow.resize(640, 480)
9. #Style Sheet (CSS)
10. CreateClassWindow.setStyleSheet(css)
12. #Central Widget
13. self.centralwidget = QtGui.QWidget(CreateClassWindow)
14. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
15. self.currentClass = currentClass
16. self.window = CreateClassWindow
18. #Label for subject name
19. self.subjectLabel = QtGui.QLabel(self.centralwidget)
20. self.subjectLabel.setGeometry(QtCore.QRect(100, 105, 71, 51))
21. self.subjectLabel.setFont(labelfont)
22. self.subjectLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
23. self.subjectLabel.setObjectName(\_fromUtf8("subjectLabel"))
25. #Label for teacher name
26. self.teacherLabel = QtGui.QLabel(self.centralwidget)
27. self.teacherLabel.setGeometry(QtCore.QRect(100, 145, 71, 51))
28. self.teacherLabel.setFont(labelfont)
29. self.teacherLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
30. self.teacherLabel.setObjectName(\_fromUtf8("teacherLabel"))
32. #Create Subject Label
33. self.createSubjectLabel = QtGui.QLabel(self.centralwidget)
34. self.createSubjectLabel.setGeometry(QtCore.QRect(130, 10, 371,91))
35. self.createSubjectLabel.setFont(titlefont)
36. self.createSubjectLabel.setTextFormat(QtCore.Qt.PlainText)
37. self.createSubjectLabel.setAlignment(QtCore.Qt.AlignCenter)
38. self.createSubjectLabel.setObjectName(\_fromUtf8("createSubjectLabel"))
40. #Button to create
41. self.createBtn = QtGui.QPushButton(self.centralwidget)
42. self.createBtn.setGeometry(QtCore.QRect(530, 430, 75, 27))
43. self.createBtn.setFont(labelfont)
44. self.createBtn.setObjectName(\_fromUtf8("createBtn"))
46. #Muli Choice Lesson 1 Timetable
47. self.lesson1Box = QtGui.QComboBox(self.centralwidget)
48. self.lesson1Box.setGeometry(QtCore.QRect(180, 200, 351, 22))
49. self.lesson1Box.setObjectName(\_fromUtf8("lesson1Box"))
51. #Muli Choice Lesson 2 Timetable
52. self.lesson2Box = QtGui.QComboBox(self.centralwidget)
53. self.lesson2Box.setGeometry(QtCore.QRect(180, 240, 351, 22))
54. self.lesson2Box.setObjectName(\_fromUtf8("lesson2Box"))
55. #Lesson 3
56. self.lesson3Box = QtGui.QComboBox(self.centralwidget)
57. self.lesson3Box.setGeometry(QtCore.QRect(180, 280, 351, 22))
58. self.lesson3Box.setObjectName(\_fromUtf8("lesson3Box"))
59. #Lesson 4
60. self.lesson4Box = QtGui.QComboBox(self.centralwidget)
61. self.lesson4Box.setGeometry(QtCore.QRect(180, 320, 351, 22))
62. self.lesson4Box.setObjectName(\_fromUtf8("lesson4Box"))
63. #Year Group
64. self.yearCombo = QtGui.QComboBox(self.centralwidget)
65. self.yearCombo.setGeometry(QtCore.QRect(180, 360, 351, 22))
66. self.yearCombo.setObjectName(\_fromUtf8("yearCombo"))
67. self.yearCombo.addItem("Year 12")
68. self.yearCombo.addItem("Year 13")
70. #Setting the default display option to be what the classes year group is.
71. **if** self.currentClass.yearGroup == "13":
72. self.yearCombo.setCurrentIndex(1)

75. #Adding Lessons to boxes
76. self.lesson1Box.addItem("NULL")
77. self.lesson2Box.addItem("NULL")
78. self.lesson3Box.addItem("NULL")
79. self.lesson4Box.addItem("NULL")
80. week = ["MON","TUE","WED","THURS","FRI"]
81. lessons = []
82. **for** day **in** week:
83. **for** lesson **in** range (1,6):
84. self.lesson1Box.addItem(day + " " + str(lesson))
85. self.lesson2Box.addItem(day + " " + str(lesson))
86. self.lesson3Box.addItem(day + " " + str(lesson))
87. self.lesson4Box.addItem(day + " " + str(lesson))


91. #Lesson 1 Label
92. self.lesson1Label = QtGui.QLabel(self.centralwidget)
93. self.lesson1Label.setGeometry(QtCore.QRect(100, 185, 71, 51))
94. self.lesson1Label.setFont(labelfont)
95. self.lesson1Label.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
96. self.lesson1Label.setObjectName(\_fromUtf8("lesson1Label"))
98. #Lesson 2 Label
99. self.lesson2Label = QtGui.QLabel(self.centralwidget)
100. self.lesson2Label.setGeometry(QtCore.QRect(100, 225, 71, 51))
101. self.lesson2Label.setFont(labelfont)
102. self.lesson2Label.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
103. self.lesson2Label.setObjectName(\_fromUtf8("lesson2Label"))
105. #Lesson 3 Label
106. self.lesson3Label = QtGui.QLabel(self.centralwidget)
107. self.lesson3Label.setGeometry(QtCore.QRect(100, 265, 71, 51))
108. self.lesson3Label.setFont(labelfont)
109. self.lesson3Label.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
110. self.lesson3Label.setObjectName(\_fromUtf8("lesson3Label"))
112. #Lesson 4 Label
113. self.lesson4Label = QtGui.QLabel(self.centralwidget)
114. self.lesson4Label.setGeometry(QtCore.QRect(90, 305, 81, 51))
115. self.lesson4Label.setFont(labelfont)
116. self.lesson4Label.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
117. self.lesson4Label.setObjectName(\_fromUtf8("lesson4Label"))
119. #Year Group Label
120. self.yearLabel = QtGui.QLabel(self.centralwidget)
121. self.yearLabel.setGeometry(QtCore.QRect(50, 345, 121, 51))
122. self.yearLabel.setFont(labelfont)
123. self.yearLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
124. self.yearLabel.setObjectName(\_fromUtf8("yearLabel"))
126. #Teacher Combo Box
127. self.teacherCombo = QtGui.QComboBox(self.centralwidget)
128. self.teacherCombo.setGeometry(QtCore.QRect(180, 160, 351, 22))
129. self.teacherCombo.setObjectName(\_fromUtf8("teacherCombo"))
130. #Fetches all teachers and adds them to the ComboBox.
131. c.execute("""SELECT first, last, username
132. FROM users
133. WHERE type = 'Teacher'
134. ORDER BY first ASC""")
135. data = c.fetchall()
136. **for** i **in** range(len(data)):
137. self.teacherCombo.addItem("{} {} ({})".format(data[i][0],data[i][1],data[i][2] ))
138. **if** data[i][2] == self.currentClass.teacher:
139. self.teacherCombo.setCurrentIndex(i)
141. #Subject Combo Box
142. self.subjectCombo = QtGui.QComboBox(self.centralwidget)
143. self.subjectCombo.setGeometry(QtCore.QRect(180, 120, 351, 22))
144. self.subjectCombo.setObjectName(\_fromUtf8("subjectCombo"))
145. #Fetches all subjects from database then adds it to the combo box.
146. c.execute("""SELECT fullname
147. FROM subjects
148. ORDER BY fullname ASC""")
149. data = c.fetchall()
150. **for** i **in** range(len(data)):
151. self.subjectCombo.addItem("{}".format(data[i][0]))
152. **if** data[i][0] == self.currentClass.subject:
153. self.subjectCombo.setCurrentIndex(i)
155. **if** self.currentClass.id == "NULL":
156. self.createBtn.clicked.connect(self.create\_)
158. **else**:
159. self.createBtn.clicked.connect(self.save\_changes)
160. self.subjectCombo.setEnabled(False)
161. #Class ID Line Edit
162. self.idEdit = QtGui.QLineEdit(self.centralwidget)
163. self.idEdit.setGeometry(QtCore.QRect(180, 400, 351, 21))
164. self.idEdit.setObjectName(\_fromUtf8("idEdit"))
165. self.idEdit.setEnabled(False)
167. #Class ID Label
168. self.idLabel = QtGui.QLabel(self.centralwidget)
169. self.idLabel.setGeometry(QtCore.QRect(90, 385, 81, 51))
170. self.idLabel.setFont(labelfont)
171. self.idLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
172. self.idLabel.setObjectName(\_fromUtf8("idLabel"))
173. self.idLabel.setEnabled(False)
175. **if** self.currentClass.id == "NULL":
176. self.createBtn.clicked.connect(self.create\_)
178. **else**:
179. self.createBtn.clicked.connect(self.save\_changes)
180. self.subjectCombo.setEnabled(False)
182. #If the current user not an admin then all capabilities of changing the class are taken away - the details will just be
183. #there to view.
184. **if** currentUser.type == "Student" **or** currentUser.type == "Teacher":
185. self.teacherCombo.setEnabled(False)
186. self.yearCombo.setEnabled(False)
187. self.lesson1Box.setEnabled(False)
188. self.lesson2Box.setEnabled(False)
189. self.lesson3Box.setEnabled(False)
190. self.lesson4Box.setEnabled(False)
191. self.subjectCombo.setEnabled(False)
192. self.createBtn.hide()


196. CreateClassWindow.setCentralWidget(self.centralwidget)
198. #Creating Menu Bar
199. self.menubar = QtGui.QMenuBar(CreateClassWindow)
200. self.menubar.setGeometry(QtCore.QRect(0, 0, 640, 21))
202. CreateClassWindow.setMenuBar(self.menubar)
203. self.statusbar = QtGui.QStatusBar(CreateClassWindow)
204. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
205. CreateClassWindow.setStatusBar(self.statusbar)
206. self.retranslateUi(CreateClassWindow)
207. QtCore.QMetaObject.connectSlotsByName(CreateClassWindow)
209. **def** retranslateUi(self, CreateClassWindow):
210. **if** self.currentClass.id == "NULL":
211. CreateClassWindow.setWindowTitle(\_translate("CreateClassWindow", "Create Class Window", None))
212. self.createSubjectLabel.setText(\_translate("CreateClassWindow", "CREATE CLASS", None))
213. self.createBtn.setText(\_translate("CreateClassWindow", "Create", None))
214. **else**:
215. CreateClassWindow.setWindowTitle(\_translate("CreateClassWindow", "Edit Class Window", None))
216. self.createSubjectLabel.setText(\_translate("CreateClassWindow", "EDIT CLASS", None))
217. self.createBtn.setText(\_translate("CreateClassWindow", "Save", None))
218. self.idEdit.setText(\_translate("EditUserWindow",self.currentClass.id,None))
219. **if** currentUser.type == "Student" **or** currentUser.type == "Teacher":
220. self.createSubjectLabel.setText(\_translate("CreateClassWindow", "VIEW CLASS", None))
221. self.subjectLabel.setText(\_translate("CreateClassWindow", "SUBJECT", None))
222. self.teacherLabel.setText(\_translate("CreateClassWindow", "TEACHER", None))
223. self.lesson1Label.setText(\_translate("CreateClassWindow", "LESSON 1", None))
224. self.lesson2Label.setText(\_translate("CreateClassWindow", "LESSON 2", None))
225. self.lesson3Label.setText(\_translate("CreateClassWindow", "LESSON 3", None))
226. self.lesson4Label.setText(\_translate("CreateClassWindow", "LESSON 4", None))
227. self.yearLabel.setText(\_translate("CreateClassWindow", "YEAR GROUP", None))
228. self.idLabel.setText(\_translate("CreateClassWindow", "CLASS ID", None))
229. CreateClassWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))

232. **def** create\_(self):
234. self.get\_details()
235. c.execute("SELECT shortname FROM subjects WHERE fullname = :fullname",
236. {"fullname":self.currentClass.subject})
237. self.currentClass.id = c.fetchone()[0]
238. c.execute("SELECT id FROM classes WHERE id LIKE :id",
239. {"id":self.currentClass.id + "%"})
240. data = c.fetchall()
241. classes = []
242. self.currentClass.id += "1"
243. #Generating a valid id for the class. To do this, the default id will be subjectname1 and if that is taken
244. #Then that one will become a 2, then 3 etc.
245. **for** item **in** data:
246. classes.append(item[0])
247. chosen = False
248. **while** **not** chosen:
249. **if** self.currentClass.id **in** classes:
250. self.currentClass.id = self.currentClass.id[:-1] + str(int(self.currentClass.id[-1])+1)
251. **else**:
252. chosen = True
253. c.execute("INSERT INTO classes VALUES (:yeargroup,:teacher,:subject,:id,:lesson1,:lesson2,:lesson3,:lesson4)",
254. {"yeargroup":self.currentClass.yearGroup,"teacher":self.currentClass.teacher,"subject":self.currentClass.subject,
255. "id":self.currentClass.id,"lesson1":self.currentClass.lesson1,"lesson2":self.currentClass.lesson2,"lesson3":self.currentClass.lesson3
256. ,"lesson4":self.currentClass.lesson4})
257. c.execute("CREATE TABLE "+self.currentClass.id+
258. "(Student text)")
260. conn.commit()
262. self.window.hide()
263. self.newWindow = EditWindow()
264. self.newPage = Ui\_CreateClassWindow()
265. self.newPage.setupUi(self.newWindow,self.currentClass)
266. self.newWindow.show()
268. #Getting all inputs.
269. **def** get\_details(self):
270. self.currentClass.subject = self.subjectCombo.currentText()
271. self.currentClass.teacher = self.teacherCombo.currentText()
272. self.currentClass.teacher = self.currentClass.teacher[self.currentClass.teacher.find("(")+1:self.currentClass.teacher.find(")")]
273. self.currentClass.yearGroup = self.yearCombo.currentText()[-2:]
274. self.currentClass.lesson1 = self.lesson1Box.currentText()
275. self.currentClass.lesson2 = self.lesson2Box.currentText()
276. self.currentClass.lesson3 = self.lesson3Box.currentText()
277. self.currentClass.lesson4 = self.lesson4Box.currentText()
279. #Pop Up Window
280. **def** save\_changes(self):
281. choice = QtGui.QMessageBox.question(self.window,"Save Changes","Are you sure you would like save the changes?\n"
282. "Any previous data will be lost.",QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)
283. **if** choice == QtGui.QMessageBox.Yes:
284. self.save\_()
286. #Save functions
287. **def** save\_(self):
288. self.get\_details()
289. c.execute("UPDATE classes SET teacher = :teacher,yeargroup = :yearGroup, lesson1 = :lesson1, lesson2 =:lesson2,"
290. "lesson3 =:lesson3, lesson4 = :lesson4 WHERE id = :id",
291. {"teacher":self.currentClass.teacher,"yearGroup":self.currentClass.yearGroup,"id":self.currentClass.id,
292. "lesson1":self.currentClass.lesson1,"lesson2":self.currentClass.lesson2,"lesson3":self.currentClass.lesson3,
293. "lesson4":self.currentClass.lesson4})
294. conn.commit()
295. self.saved\_window()
297. #Pop Up Window
298. **def** saved\_window(self):
299. QtGui.QMessageBox.question(self.window,"Saved","Save Successful",
300. QtGui.QMessageBox.Ok)

7

2. **class** Ui\_EditSubjectWindow(object):
3. **def** setupUi(self, EditSubjectWindow,subject):
4. self.subject = subject
5. #Creating Main Window
6. EditSubjectWindow.setObjectName(\_fromUtf8("EditSubjectWindow"))
7. EditSubjectWindow.resize(640, 480)
8. #Style Sheet
9. EditSubjectWindow.setStyleSheet(css)
10. self.window = EditSubjectWindow
11. #Central Widget
12. self.centralwidget = QtGui.QWidget(EditSubjectWindow)
13. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
15. #Edit Subject Label
16. self.editSubjectLabel = QtGui.QLabel(self.centralwidget)
17. self.editSubjectLabel.setGeometry(QtCore.QRect(130, 40, 391, 91))
18. self.editSubjectLabel.setFont(titlefont)
19. self.editSubjectLabel.setTextFormat(QtCore.Qt.PlainText)
20. self.editSubjectLabel.setAlignment(QtCore.Qt.AlignCenter)
21. self.editSubjectLabel.setObjectName(\_fromUtf8("editSubjectLabel"))
23. #Full name label
24. self.fullNameLabel = QtGui.QLabel(self.centralwidget)
25. self.fullNameLabel.setGeometry(QtCore.QRect(70, 150, 141, 51))
26. self.fullNameLabel.setFont(labelfont)
27. self.fullNameLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
28. self.fullNameLabel.setObjectName(\_fromUtf8("fullNameLabel"))
30. #Full name text
31. self.fullNameEdit = QtGui.QLineEdit(self.centralwidget)
32. self.fullNameEdit.setGeometry(QtCore.QRect(220, 160, 351, 31))
33. self.fullNameEdit.setFont(labelfont)
34. self.fullNameEdit.setObjectName(\_fromUtf8("fullNameEdit"))
35. self.fullNameEdit.setValidator(lettersandspacevalidator)
37. #Short name edit
38. self.shortNameEdit = QtGui.QLineEdit(self.centralwidget)
39. self.shortNameEdit.setGeometry(QtCore.QRect(220, 210, 351, 31))
40. self.shortNameEdit.setFont(labelfont)
41. self.shortNameEdit.setObjectName(\_fromUtf8("shortNameEdit"))
42. self.shortNameEdit.setValidator(lettersvalidator)
43. self.shortNameEdit.setMaxLength(8)
45. #Short name label
46. self.shortNameLabel = QtGui.QLabel(self.centralwidget)
47. self.shortNameLabel.setGeometry(QtCore.QRect(70, 200, 141, 51))
48. self.shortNameLabel.setFont(labelfont)
49. self.shortNameLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
50. self.shortNameLabel.setObjectName(\_fromUtf8("shortNameLabel"))
52. #Head Of Subject Label
53. self.headOfSubjectLabel = QtGui.QLabel(self.centralwidget)
54. self.headOfSubjectLabel.setGeometry(QtCore.QRect(70, 250, 141, 51))
55. self.headOfSubjectLabel.setFont(labelfont)
56. self.headOfSubjectLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
57. self.headOfSubjectLabel.setObjectName(\_fromUtf8("headOfSubjectLabel"))
59. #Save Button
60. self.saveBtn = QtGui.QPushButton(self.centralwidget)
61. self.saveBtn.setGeometry(QtCore.QRect(500, 300, 75, 27))
62. self.saveBtn.setFont(labelfont)
63. self.saveBtn.setObjectName(\_fromUtf8("saveBtn"))
64. #######################Button Event###########################
65. **if** self.subject.fullname == "NULL":
66. self.saveBtn.clicked.connect(self.create\_)
67. **else**:
68. self.saveBtn.clicked.connect(self.save\_changes)
70. #Head Of Subject combo box
71. self.headCombo = QtGui.QComboBox(self.centralwidget)
72. self.headCombo.setGeometry(QtCore.QRect(220, 260, 351, 31))
73. self.headCombo.setObjectName(\_fromUtf8("headCombo"))
74. #Fetching all teachers
75. c.execute("""SELECT first, last, username
76. FROM users
77. WHERE type = 'Teacher'
78. ORDER BY first ASC""")
79. data = c.fetchall()
80. **for** i **in** range(len(data)):
81. self.headCombo.addItem("{} {} ({})".format(data[i][0],data[i][1],data[i][2] ))
82. **if** data[i][2] == self.subject.head:
83. #Sets current index as the current head of subject
84. self.headCombo.setCurrentIndex(i)

87. #Menu and status bars
88. EditSubjectWindow.setCentralWidget(self.centralwidget)
89. self.menubar = QtGui.QMenuBar(EditSubjectWindow)
90. self.menubar.setGeometry(QtCore.QRect(0, 0, 640, 21))
91. self.menubar.setObjectName(\_fromUtf8("menubar"))
92. EditSubjectWindow.setMenuBar(self.menubar)
93. self.statusbar = QtGui.QStatusBar(EditSubjectWindow)
94. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
95. EditSubjectWindow.setStatusBar(self.statusbar)
97. self.retranslateUi(EditSubjectWindow)
98. QtCore.QMetaObject.connectSlotsByName(EditSubjectWindow)
99. self.statusbar = QtGui.QStatusBar(EditSubjectWindow)
100. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
101. EditSubjectWindow.setStatusBar(self.statusbar)
102. self.retranslateUi(EditSubjectWindow)
103. QtCore.QMetaObject.connectSlotsByName(EditSubjectWindow)

106. **def** retranslateUi(self, EditSubjectWindow):
107. EditSubjectWindow.setWindowTitle(\_translate("EditSubjectWindow", "Edit Subject", None))
108. **if** self.subject.fullname == "NULL":
109. self.editSubjectLabel.setText(\_translate("EditSubjectWindow", "CREATE SUBJECT", None))
110. self.saveBtn.setText(\_translate("EditSubjectWindow", "Create", None))
111. **else**:
112. self.fullNameEdit.setText(\_translate("EditSubjectWindow", self.subject.fullname, None))
113. #Setting enabled as false keeps it visible however the user cannot edit it.
114. self.fullNameEdit.setEnabled(False)
115. self.shortNameEdit.setEnabled(False)
116. self.shortNameEdit.setText(\_translate("EditSubjectWindow", self.subject.shortname, None))
117. self.saveBtn.setText(\_translate("EditSubjectWindow", "Save", None))
118. self.editSubjectLabel.setText(\_translate("EditSubjectWindow", "EDIT SUBJECT", None))
119. self.fullNameLabel.setText(\_translate("EditSubjectWindow", "FULL NAME", None))
120. self.shortNameLabel.setText(\_translate("EditSubjectWindow", "SHORT NAME", None))
121. self.headOfSubjectLabel.setText(\_translate("EditSubjectWindow", "HEAD OF SUBJECT", None))
122. EditSubjectWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))

125. **def** create\_(self):
127. self.get\_details()
128. c.execute("INSERT INTO subjects VALUES (:full,:short,:head)",
129. {"full":self.subject.full,"short":self.subject.short,"head":self.subject.head})
131. self.subject = Subject(self.subject.full,self.subject.short,self.subject.head)
133. conn.commit()
135. #Once created the window will disappear and a new window with "Edit" instead of "Create"
136. self.window.hide()
137. self.newWindow = EditWindow()
138. self.newPage = Ui\_EditSubjectWindow()
139. self.newPage.setupUi(self.newWindow,self.subject)
140. self.newWindow.show()

143. **def** get\_details(self):
144. self.subject.full = self.fullNameEdit.text()
145. self.subject.short = self.shortNameEdit.text()
146. self.subject.head = self.headCombo.currentText()
147. self.subject.head = self.subject.head[self.subject.head.find("(")+1:self.subject.head.find(")")]
148. **return** self.subject.full,self.subject.short,self.subject.head
150. **def** save\_changes(self):
151. choice = QtGui.QMessageBox.question(self.window,"Save Changes","Are you sure you would like save the changes?\n"
152. "Any previous data will be lost.",QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)
153. **if** choice == QtGui.QMessageBox.Yes:
154. self.save\_()
156. **def** save\_(self):
157. self.get\_details()
158. c.execute("UPDATE subjects SET head = :head WHERE fullname = :full",
159. {"head":self.subject.head,"full":self.subject.full})
160. conn.commit()
161. self.saved\_window()
163. **def** saved\_window(self):
164. QtGui.QMessageBox.question(self.window,"Saved","Save Successful",
165. QtGui.QMessageBox.Ok)

4

1. **class** Ui\_EditUserWindow(object):
2. **def** setupUi(self, EditUserWindow,typeOfUser,user):
3. self.user = user
4. #Creating Main Window
5. EditUserWindow.setObjectName(\_fromUtf8("EditUserWindow"))
6. EditUserWindow.resize(640, 480)
7. EditUserWindow.setAutoFillBackground(False)
8. self.window = EditUserWindow
9. self.type = typeOfUser
10. moveY = 0
11. self.centralwidget = QtGui.QWidget(EditUserWindow)
12. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))

15. #CSS
16. EditUserWindow.setStyleSheet(css)
17. #Creating widget
18. self.centralwidget = QtGui.QWidget(EditUserWindow)
19. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
21. **if** self.type == "Student":
22. #Year Group chooser
23. self.yearGroup = QtGui.QComboBox(self.centralwidget)
24. self.yearGroup.setGeometry(QtCore.QRect(190, 340, 351, 31))
25. self.yearGroup.setObjectName(\_fromUtf8("yearGroup"))
26. self.yearGroup.addItem("12")
27. self.yearGroup.addItem("13")
29. #Year group label
30. self.yearGroupLabel = QtGui.QLabel(self.centralwidget)
31. self.yearGroupLabel.setGeometry(QtCore.QRect(40, 330, 141, 51))
32. self.yearGroupLabel.setFont(labelfont)
33. self.yearGroupLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
34. self.yearGroupLabel.setObjectName(\_fromUtf8("yearGroupLabel"))
36. #MoveY is used so that if it is a student certain objects will be placed lower down on the window
37. moveY += 30
39. self.achievementButton = QtGui.QPushButton(self.centralwidget)
40. self.achievementButton.setGeometry(QtCore.QRect(15, 380, 186, 27))
41. self.achievementButton.setFont(labelfont)
42. self.achievementButton.setObjectName(\_fromUtf8("achievementButton"))
43. self.achievementButton.clicked.connect(self.achievement\_window)
45. self.behaviourButton = QtGui.QPushButton(self.centralwidget)
46. self.behaviourButton.setGeometry(QtCore.QRect(15, 410, 186, 27))
47. self.behaviourButton.setFont(labelfont)
48. self.behaviourButton.setObjectName(\_fromUtf8("behaviourButton"))
49. self.behaviourButton.clicked.connect(self.behaviour\_window)

52. #Profile Picture
53. self.profilePic = QtGui.QLabel(self.centralwidget)
54. self.profilePic.setGeometry(QtCore.QRect(30, 10, 126, 126))
55. self.profilePic.setText(\_fromUtf8(""))
56. #If user profile pic exists then display
57. **if** os.path.isfile(path + self.user.pic):
58. self.profilePic.setPixmap(QtGui.QPixmap(\_fromUtf8(path + self.user.pic)))
59. **else**:
60. self.profilePic.setPixmap(QtGui.QPixmap(\_fromUtf8("placeholder.png")))
61. self.profilePic.setScaledContents(True)
62. self.profilePic.setObjectName(\_fromUtf8("profilePic"))

65. #Title label
66. self.titleLabel = QtGui.QLabel(self.centralwidget)
67. self.titleLabel.setGeometry(QtCore.QRect(140, 30, 411, 91))
68. self.titleLabel.setFont(titlefont)
69. self.titleLabel.setTextFormat(QtCore.Qt.PlainText)
70. self.titleLabel.setAlignment(QtCore.Qt.AlignCenter)
71. self.titleLabel.setObjectName(\_fromUtf8("titleLabel"))
73. #Date of birth editor
74. self.dateEdit = QtGui.QDateEdit(self.centralwidget)
75. self.dateEdit.setGeometry(QtCore.QRect(190, 300, 351, 31))
76. self.dateEdit.setObjectName(\_fromUtf8("dateEdit"))
77. self.dateEdit.setCalendarPopup(True)
78. self.dateEdit.calendarWidget().installEventFilter(EditUserWindow)
79. self.dateEdit.setMaximumDate(QtCore.QDate.currentDate())
81. #Date of birth label
82. self.dateLabel = QtGui.QLabel(self.centralwidget)
83. self.dateLabel.setGeometry(QtCore.QRect(40, 290, 141, 51))
84. self.dateLabel.setFont(labelfont)
85. self.dateLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
86. self.dateLabel.setObjectName(\_fromUtf8("dateLabel"))
88. #Last name label
89. self.lastLabel = QtGui.QLabel(self.centralwidget)
90. self.lastLabel.setGeometry(QtCore.QRect(40, 170, 141, 51))
91. self.lastLabel.setFont(labelfont)
92. self.lastLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
93. self.lastLabel.setObjectName(\_fromUtf8("lastLabel"))
95. #Lastname editor
96. self.lastEdit = QtGui.QLineEdit(self.centralwidget)
97. self.lastEdit.setGeometry(QtCore.QRect(190, 180, 351, 31))
98. self.lastEdit.setFont(labelfont)
99. self.lastEdit.setObjectName(\_fromUtf8("lastEdit"))
100. self.lastEdit.setValidator(lettersvalidator)

103. #Username editor
104. self.usernameEdit = QtGui.QLineEdit(self.centralwidget)
105. self.usernameEdit.setGeometry(QtCore.QRect(190, 220, 351, 31))
106. self.usernameEdit.setFont(labelfont)
107. self.usernameEdit.setObjectName(\_fromUtf8("usernameEdit"))
108. self.usernameEdit.setEnabled(False)
110. #Username label
111. self.usernameLabel = QtGui.QLabel(self.centralwidget)
112. self.usernameLabel.setGeometry(QtCore.QRect(40, 210, 141, 51))
113. self.usernameLabel.setFont(labelfont)
114. self.usernameLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
115. self.usernameLabel.setObjectName(\_fromUtf8("usernameLabel"))
116. self.usernameLabel.setEnabled(False)
118. #Email Label
119. self.emailLabel = QtGui.QLabel(self.centralwidget)
120. self.emailLabel.setGeometry(QtCore.QRect(40, 250, 141, 51))
121. self.emailLabel.setFont(labelfont)
122. self.emailLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
123. self.emailLabel.setObjectName(\_fromUtf8("emailLabel"))
125. #Email Editor
126. self.emailEdit = QtGui.QLineEdit(self.centralwidget)
127. self.emailEdit.setGeometry(QtCore.QRect(190, 260, 351, 31))
128. self.emailEdit.setFont(labelfont)
129. self.emailEdit.setText(\_fromUtf8(""))
130. self.emailEdit.setObjectName(\_fromUtf8("emailEdit"))
131. self.emailEdit.setValidator(emailvalidator)
133. #First name editor
134. self.firstEdit = QtGui.QLineEdit(self.centralwidget)
135. self.firstEdit.setGeometry(QtCore.QRect(190, 140, 351, 31))
136. self.firstEdit.setFont(labelfont)
137. self.firstEdit.setObjectName(\_fromUtf8("firstEdit"))
138. self.firstEdit.setValidator(lettersvalidator)
140. #Firstname label
141. self.firstLabel = QtGui.QLabel(self.centralwidget)
142. self.firstLabel.setGeometry(QtCore.QRect(40, 130, 141, 51))
143. self.firstLabel.setFont(labelfont)
144. self.firstLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
145. self.firstLabel.setObjectName(\_fromUtf8("firstLabel"))
147. #Edit button
148. self.editBtn = QtGui.QPushButton(self.centralwidget)
149. self.editBtn.setGeometry(QtCore.QRect(470, 350+moveY, 75, 27))
150. self.editBtn.setFont(labelfont)
151. self.editBtn.setObjectName(\_fromUtf8("editBtn"))
152. #######################Button Event###########################
153. **if** self.user.first == "NULL":
154. self.editBtn.clicked.connect(self.create\_)
155. **else**:
156. self.editBtn.clicked.connect(self.save\_changes)
157. #Upload pic button
158. self.uploadPic = QtGui.QPushButton(self.centralwidget)
159. self.uploadPic.setGeometry(QtCore.QRect(340, 380+moveY, 206, 27))
160. self.uploadPic.setFont(labelfont)
161. self.uploadPic.setObjectName(\_fromUtf8("uploadPic"))
162. self.uploadPic.clicked.connect(self.upload\_picture)
163. #Reset password button
164. self.resetPassword = QtGui.QPushButton(self.centralwidget)
165. self.resetPassword.setGeometry(QtCore.QRect(340, 350+moveY, 121, 27))
166. self.resetPassword.setFont(labelfont)
167. self.resetPassword.setObjectName(\_fromUtf8("resetPassword"))
168. self.resetPassword.clicked.connect(self.generate\_password)
170. **if** self.user.type != "Admin":
171. #Show class button if user is not an admin
172. self.selectClasses = QtGui.QPushButton(self.centralwidget)
173. self.selectClasses.setGeometry(QtCore.QRect(210, 350+moveY, 121, 57))
174. self.selectClasses.setFont(labelfont)
175. self.selectClasses.setObjectName(\_fromUtf8("selectClasses"))
176. self.selectClasses.clicked.connect(self.select\_classes)
178. ##############################################################
180. **if** currentUser.type == "Teacher":
181. self.firstEdit.setEnabled(False)
182. self.lastEdit.setEnabled(False)
183. self.emailEdit.setEnabled(False)
184. self.dateEdit.setEnabled(False)
185. **if** self.user.type == "Student":
186. self.yearGroup.setEnabled(False)
187. **if** self.user.type != "Admin":
188. self.selectClasses.hide()
189. self.resetPassword.hide()
190. self.editBtn.hide()
191. self.uploadPic.hide()

194. #Menu and status bars
195. EditUserWindow.setCentralWidget(self.centralwidget)
196. self.menubar = QtGui.QMenuBar(EditUserWindow)
197. self.menubar.setGeometry(QtCore.QRect(0, 0, 640, 21))
198. self.menubar.setObjectName(\_fromUtf8("menubar"))
199. EditUserWindow.setMenuBar(self.menubar)
200. self.statusbar = QtGui.QStatusBar(EditUserWindow)
201. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
202. EditUserWindow.setStatusBar(self.statusbar)
203. self.retranslateUi(EditUserWindow)
204. QtCore.QMetaObject.connectSlotsByName(EditUserWindow)

207. **def** retranslateUi(self, EditUserWindow):
208. **if** self.user.first == "NULL":
209. self.editBtn.setText(\_translate("EditUserWindow", "Create", None))
210. **if** self.type == "Student":
211. EditUserWindow.setWindowTitle(\_translate("EditUserWindow", "Create Student Page", None))
212. self.yearGroupLabel.setText(\_translate("EditUserWindow", "YEAR GROUP", None))
213. self.titleLabel.setText(\_translate("EditUserWindow", "CREATE STUDENT", None))
214. **elif** self.type == "Admin":
215. self.titleLabel.setText(\_translate("EditUserWindow", "CREATE ADMIN", None))
216. EditUserWindow.setWindowTitle(\_translate("EditUserWindow", "Create Admin Page", None))
217. **else**:
218. self.titleLabel.setText(\_translate("EditUserWindow", "CREATE TEACHER", None))
219. EditUserWindow.setWindowTitle(\_translate("EditUserWindow", "Create Teacher Page", None))
221. **else**:
222. date = QtCore.QDate(int(self.user.dobyear),int(self.user.dobmonth),int(self.user.dobday))
223. self.dateEdit.setDate(date)
224. self.resetPassword.setText(\_translate("EditUserWindow", "Reset Password", None))
225. self.uploadPic.setText(\_translate("EditUserWindow", "Upload Profile Picture", None))
226. self.editBtn.setText(\_translate("EditUserWindow", "Save", None))
227. self.firstEdit.setText(\_translate("EditUserWindow",self.user.first,None))
228. self.lastEdit.setText(\_translate("EditUserWindow",self.user.last,None))
229. **if** self.user.email != "NULL":
230. self.emailEdit.setText(\_translate("EditUserWindow",self.user.email,None))
231. self.usernameEdit.setText(\_translate("EditUserWindow",self.user.username,None))
232. **if** self.type == "Student":
233. EditUserWindow.setWindowTitle(\_translate("EditUserWindow", "Edit Student Page", None))
234. self.yearGroupLabel.setText(\_translate("EditUserWindow", "YEAR GROUP", None))
235. self.titleLabel.setText(\_translate("EditUserWindow", "EDIT STUDENT", None))
236. self.achievementButton.setText(\_translate("EditUserWindow", "Add Achievements", None))
237. self.behaviourButton.setText(\_translate("EditUserWindow", "Add Behaviours", None))
238. **if** self.user.yeargroup == "13":
239. self.yearGroup.setCurrentIndex(1)
241. self.selectClasses.setText(\_translate("EditUserWindow", "Select\nClasses", None))
242. **elif** self.type == "Admin":
243. self.titleLabel.setText(\_translate("EditUserWindow", "EDIT ADMIN", None))
244. EditUserWindow.setWindowTitle(\_translate("EditUserWindow", "Edit Admin Page", None))
245. **else**:
246. self.titleLabel.setText(\_translate("EditUserWindow", "EDIT TEACHER", None))
247. EditUserWindow.setWindowTitle(\_translate("EditUserWindow", "Edit Teacher Page", None))
248. self.selectClasses.setText(\_translate("EditUserWindow", "Select\nClasses", None))

251. self.dateLabel.setText(\_translate("EditUserWindow", "DATE OF BIRTH", None))
252. EditUserWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
253. self.lastLabel.setText(\_translate("EditUserWindow", "LAST NAME", None))
254. self.usernameLabel.setText(\_translate("EditUserWindow", "USERNAME", None))
255. self.emailLabel.setText(\_translate("EditUserWindow", "EMAIL", None))
256. self.firstLabel.setText(\_translate("EditUserWindow", "FIRST NAME", None))
258. **def** create\_(self):
259. self.user.first = self.firstEdit.text()
260. self.user.last = self.lastEdit.text()
261. self.user.username = (self.user.first[0]+self.user.last).lower()
262. #Creating unique username. Surname+firstname[0]1 and if taken 1 becomes 2,3 etc.
263. length = len(self.user.username)
264. c.execute("SELECT username FROM users WHERE username LIKE :username ORDER BY username ASC",
265. {"username":str(self.user.username)+"%"})
266. data = c.fetchall()
267. usernames = []
268. **for** item **in** data:
269. usernames.append(item[0])
270. **if** self.user.username **in** usernames:
271. self.user.username += "1"
272. chosen = False
273. **while** **not** chosen:
274. **if** self.user.username **in** usernames:
275. self.user.username = self.user.username[:length-1] + str(int(self.user.username[length-1:])+1)
276. **else**:
277. chosen = True
278. #Converting the Qt Date to a date I will be interacting with.
279. self.user.dob = str(self.dateEdit.date().toPyDate())
280. self.user.dob = self.user.dob[8:]+"-"+self.user.dob[5:7]+"-"+self.user.dob[:4]
281. self.user.email = self.emailEdit.text()
282. #Password and pic are intially NULL and have to be changed after creation.
283. self.user.password = "NULL"
284. self.user.pic = "NULL"
285. c.execute("INSERT INTO users VALUES (:first,:last,:username,:password, :dob,:email,:type,:pic)",
286. {"first":self.user.first,"last":self.user.last,"username":self.user.username,"password":self.user.password,"dob":self.user.dob,"email":self.user.email,"type":self.user.type,"pic":self.user.pic})
287. **if** self.user.type == "Student":
288. self.user.yeargroup = self.yearGroup.currentText()
289. c.execute("INSERT INTO student VALUES (:username,:yeargroup,0,0)",{"username":self.user.username,"yeargroup":self.user.yeargroup})
290. conn.commit()
291. self.window.hide()
292. self.newWindow = EditWindow()
293. self.newPage = Ui\_EditUserWindow()
294. self.newPage.setupUi(self.newWindow,self.type,self.user)
295. self.newWindow.show()
296. self.saved\_window()


300. **def** save\_(self):
301. self.first = self.firstEdit.text()
302. self.last = self.lastEdit.text()
303. self.dob = str(self.dateEdit.date().toPyDate())
304. self.dob = self.dob[8:]+"-"+self.dob[5:7]+"-"+self.dob[:4]
305. self.email = self.emailEdit.text()
306. **try**:
307. #Move file to a folder within the program.
308. copyfile(self.picture,path + self.user.pic)
309. **except**:
310. **pass**
311. c.execute("UPDATE users SET first = :first, last = :last,dob = :dob, email = :email, pic = :pic WHERE username = :username",
312. {"first":self.first,"last":self.last,"dob":self.dob,"email":self.email,"pic": self.user.pic,"username":self.user.username})
313. **if** self.user.type == "Student":
314. self.user.yeargroup = self.yearGroup.currentText()
315. c.execute("UPDATE student SET yeargroup = :yeargroup WHERE username = :username",{"yeargroup":self.user.yeargroup,"username":self.user.username})
316. conn.commit()
317. self.saved\_window()

320. #Upload picture method
321. **def** upload\_picture(self):
322. #Open file explorer showing only folders and png files
323. self.picture = QtGui.QFileDialog.getOpenFileName(None,"Select Profile Picture","","Images (\*png)")
325. self.user.pic = "\\userphotos\\"+self.user.username + ".png"
327. **try**:
328. self.profilePic.setPixmap(QtGui.QPixmap(\_fromUtf8(path + self.user.pic)))
329. **except** FileNotFoundError:
330. QtGui.QMessageBox.question(self.window,"Error","Error: File not found",
331. QtGui.QMessageBox.Ok)


335. #Generating a random password
336. **def** generate\_password(self):
337. #Creating a empty string
338. self.password = ""
339. **for** i **in** range(9):
340. #Random choice of lowercase,uppercase and digits.
341. self.password += random.choice(string.ascii\_lowercase + string.ascii\_uppercase + string.digits)
342. choice = QtGui.QMessageBox.question(self.window,"Generate Password?","Are you sure you would like to create a new password?"
343. "\nAny previous password will be lost.",QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)
344. **if** choice == QtGui.QMessageBox.Yes:
345. self.password\_popup()
347. **def** save\_changes(self):
348. choice = QtGui.QMessageBox.question(self.window,"Save Changes","Are you sure you would like save the changes?\n"
349. "Any previous data will be lost.",QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)
350. **if** choice == QtGui.QMessageBox.Yes:
351. self.save\_()

354. **def** password\_popup(self):
355. QtGui.QMessageBox.question(self.window,"Password","The generated password is: "+ self.password +
356. "\nMake a note of this password as you will not be able\nto access this again.",
357. QtGui.QMessageBox.Ok)
358. self.password = hashing(self.password)
359. c.execute("UPDATE users SET password = :password WHERE username = :username",{"password":self.password,"username":self.user.username})
360. conn.commit()
362. **def** saved\_window(self):
363. QtGui.QMessageBox.question(self.window,"Saved","Save Successful",
364. QtGui.QMessageBox.Ok)
366. **def** select\_classes(self):
367. #Opens classes window
368. self.selectClass = EditWindow()
369. self.selectClassUi = Ui\_ClassListWindow()
370. **if** self.type == "Student":
371. self.selectClassUi.setupUi(self.selectClass,self.user.username,self.user.type,"List")
372. **else**:
373. self.selectClassUi.setupUi(self.selectClass,self.user.username,self.user.type,"List")
374. self.selectClass.show()
376. **def** behaviour\_window(self):
377. self.behaviourWindow = EditWindow()
378. self.behaviourUi = Ui\_PointsWindow()
379. self.behaviourUi.setupUi(self.behaviourWindow,"behaviour",self.user)
380. self.behaviourWindow.show()
382. **def** achievement\_window(self):
383. self.achievementWindow = EditWindow()
384. self.achievementUi = Ui\_PointsWindow()
385. self.achievementUi.setupUi(self.achievementWindow,"achievement",self.user)
386. self.achievementWindow.show()

9

1. **class** Ui\_SearchUsers(object):
2. **def** setupUi(self, SearchUsers,typeOfWindow,homework):
4. self.typeOfWindow = typeOfWindow
5. self.homework = homework
6. self.data = []
7. self.page = 1
8. self.window = SearchUsers
10. #Setting the window up
11. SearchUsers.setObjectName(\_fromUtf8("SearchUsers"))
12. SearchUsers.resize(973, 860)
13. SearchUsers.setStyleSheet(\_fromUtf8("QMainWindow {\n"
14. "background-color: qlineargradient(spread:pad, x1:0.494364, y1:0.806, x2:0.471, y2:0.142045, stop:0 rgba(17, 255, 56, 255), stop:1 rgba(255, 255, 255, 255));}"))
15. self.centralwidget = QtGui.QWidget(SearchUsers)
16. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
18. #Title label
19. self.titleLabel = QtGui.QLabel(self.centralwidget)
20. self.titleLabel.setGeometry(QtCore.QRect(280, -10, 411, 91))
21. self.titleLabel.setFont(titlefont)
22. self.titleLabel.setTextFormat(QtCore.Qt.PlainText)
23. self.titleLabel.setAlignment(QtCore.Qt.AlignCenter)
24. self.titleLabel.setObjectName(\_fromUtf8("titleLabel"))
26. #########################################   FILTERS   #######################################################
27. #Rectangle that stores all of the filter options
28. self.filterRect = QtGui.QLabel(self.centralwidget)
29. self.filterRect.setGeometry(QtCore.QRect(20, 65, 931, 191))
30. self.filterRect.setAutoFillBackground(False)
31. self.filterRect.setStyleSheet(\_fromUtf8("border: 1px solid black;\n"
32. "background-color: rgb(255, 255, 255);"))
33. self.filterRect.setText(\_fromUtf8(""))
34. self.filterRect.setObjectName(\_fromUtf8("filterRect"))
36. ## FULL NAME FILTER
37. self.nameLabel = QtGui.QLabel(self.centralwidget)
38. self.nameLabel.setGeometry(QtCore.QRect(20, 80, 141, 51))
39. self.nameLabel.setFont(labelfont)
40. self.nameLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
41. self.nameLabel.setObjectName(\_fromUtf8("nameLabel"))

44. self.nameEdit = QtGui.QLineEdit(self.centralwidget)
45. self.nameEdit.setGeometry(QtCore.QRect(170, 90, 351, 31))
46. self.nameEdit.setFont(labelfont)
47. self.nameEdit.setText(\_fromUtf8(""))
48. self.nameEdit.setObjectName(\_fromUtf8("nameEdit"))
50. self.fullNameCheck = QtGui.QCheckBox(self.centralwidget)
51. self.fullNameCheck.setGeometry(QtCore.QRect(170, 70, 131, 20))
52. self.fullNameCheck.setObjectName(\_fromUtf8("fullNameCheck"))

55. ## DATE OF BIRTH FILTER
56. self.dobCheck = QtGui.QCheckBox(self.centralwidget)
57. self.dobCheck.setGeometry(QtCore.QRect(570, 70, 131, 20))
58. self.dobCheck.setObjectName(\_fromUtf8("dobCheck"))
60. self.dateEdit = QtGui.QDateEdit(self.centralwidget)
61. self.dateEdit.setGeometry(QtCore.QRect(570, 90, 121, 31))
62. self.dateEdit.setObjectName(\_fromUtf8("dateEdit"))
63. self.dateEdit.setCalendarPopup(True)
64. self.dateEdit.calendarWidget().installEventFilter(SearchUsers)
65. self.dateEdit.setMaximumDate(QtCore.QDate.currentDate())


69. ## USERNAME FILTER
70. self.usernameLabel = QtGui.QLabel(self.centralwidget)
71. self.usernameLabel.setGeometry(QtCore.QRect(20, 140, 141, 51))
72. self.usernameLabel.setFont(labelfont)
73. self.usernameLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
74. self.usernameLabel.setObjectName(\_fromUtf8("usernameLabel"))
76. self.usernameEdit = QtGui.QLineEdit(self.centralwidget)
77. self.usernameEdit.setGeometry(QtCore.QRect(170, 150, 351, 31))
78. self.usernameEdit.setFont(labelfont)
79. self.usernameEdit.setText(\_fromUtf8(""))
80. self.usernameEdit.setObjectName(\_fromUtf8("usernameEdit"))
82. self.usernameCheck = QtGui.QCheckBox(self.centralwidget)
83. self.usernameCheck.setGeometry(QtCore.QRect(170, 130, 131, 20))
84. self.usernameCheck.setObjectName(\_fromUtf8("usernameCheck"))

87. ## TYPE OF USER FILTER
88. self.typeCombo = QtGui.QComboBox(self.centralwidget)
89. self.typeCombo.setGeometry(QtCore.QRect(570, 150, 121, 31))
90. self.typeCombo.setObjectName(\_fromUtf8("typeCombo"))
91. self.typeCombo.addItem("Student")
92. self.typeCombo.addItem("Teacher")
93. self.typeCombo.addItem("Admin")
95. self.typeCheck = QtGui.QCheckBox(self.centralwidget)
96. self.typeCheck.setGeometry(QtCore.QRect(570, 130, 131, 20))
97. self.typeCheck.setObjectName(\_fromUtf8("typeCheck"))

100. ## SUBJECT FILTER
101. self.subjectCheck = QtGui.QCheckBox(self.centralwidget)
102. self.subjectCheck.setGeometry(QtCore.QRect(740, 70, 131, 20))
103. self.subjectCheck.setObjectName(\_fromUtf8("subjectCheck"))
105. self.subjectCombo = QtGui.QComboBox(self.centralwidget)
106. self.subjectCombo.setGeometry(QtCore.QRect(740, 90, 121, 31))
107. self.subjectCombo.setObjectName(\_fromUtf8("subjectCombo"))
109. **if** self.typeOfWindow == "Homework":
110. #All the possible grades
111. self.subjectCombo.addItem("Not Completed")
112. self.subjectCombo.addItem("Completed")
113. self.subjectCombo.addItem("A\*")
114. self.subjectCombo.addItem("A")
115. self.subjectCombo.addItem("B")
116. self.subjectCombo.addItem("C")
117. self.subjectCombo.addItem("D")
118. self.subjectCombo.addItem("E")
119. self.subjectCombo.addItem("F")
120. self.subjectCombo.addItem("U")
121. **else**:
122. c.execute("SELECT fullname FROM subjects")
123. data = c.fetchall()
124. **for** i **in** range(len(data)):
125. self.subjectCombo.addItem(data[i][0])


129. ## CLASS FILTER
130. self.classCheck = QtGui.QCheckBox(self.centralwidget)
131. self.classCheck.setGeometry(QtCore.QRect(740, 130, 131, 20))
132. self.classCheck.setObjectName(\_fromUtf8("classCheck"))
134. self.classCombo = QtGui.QComboBox(self.centralwidget)
135. self.classCombo.setGeometry(QtCore.QRect(740, 150, 121, 31))
136. self.classCombo.setObjectName(\_fromUtf8("classCombo"))
138. c.execute("SELECT id FROM classes")
139. data = c.fetchall()
140. **for** i **in** range(len(data)):
141. self.classCombo.addItem(data[i][0])

144. ## SORTING RESULTS
145. self.sortCombo = QtGui.QComboBox(self.centralwidget)
146. self.sortCombo.setGeometry(QtCore.QRect(170, 210, 121, 31))
147. self.sortCombo.setObjectName(\_fromUtf8("sortCombo"))
148. self.sortCombo.addItem("Name (ASC)")
149. self.sortCombo.addItem("Name (DESC)")
150. self.sortCombo.addItem("Date Of Birth(ASC)")
151. self.sortCombo.addItem("Date Of Birth(DESC)")
153. self.sortLabel = QtGui.QLabel(self.centralwidget)
154. self.sortLabel.setGeometry(QtCore.QRect(20, 200, 141, 51))
155. self.sortLabel.setFont(labelfont)
156. self.sortLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
157. self.sortLabel.setObjectName(\_fromUtf8("sortLabel"))

160. ## YEAR GROUP FILTER
161. self.yearCheck = QtGui.QCheckBox(self.centralwidget)
162. self.yearCheck.setGeometry(QtCore.QRect(570, 190, 131, 20))
163. self.yearCheck.setObjectName(\_fromUtf8("yearCheck"))
165. self.yearCombo = QtGui.QComboBox(self.centralwidget)
166. self.yearCombo.setGeometry(QtCore.QRect(570, 210, 121, 31))
167. self.yearCombo.setObjectName(\_fromUtf8("yearCombo"))
168. self.yearCombo.addItem("12")
169. self.yearCombo.addItem("13")
171. ## SEARCH BUTTON
172. self.searchButton = QtGui.QPushButton(self.centralwidget)
173. self.searchButton.setGeometry(QtCore.QRect(310, 210, 121, 31))
174. font = QtGui.QFont()
175. font.setPointSize(8)
176. self.searchButton.setFont(font)
177. self.searchButton.setObjectName(\_fromUtf8("searchButton"))
178. self.searchButton.clicked.connect(self.search)
180. **if** self.typeOfWindow == "Homework":
181. #If used for the grade window hide alot of the filters
182. self.classCheck.hide()
183. self.classCombo.hide()
184. self.yearCombo.hide()
185. self.yearCheck.hide()
186. self.typeCheck.hide()
187. self.typeCombo.hide()
189. ##############################################################################################
190. #Search results (Contains alot)
192. ## RESULT ONE
194. #Pic
195. self.profilePic = QtGui.QLabel(self.centralwidget)
196. self.profilePic.setGeometry(QtCore.QRect(60, 270, 63, 63))
197. self.profilePic.setText(\_fromUtf8(""))
198. self.profilePic.setPixmap(QtGui.QPixmap(\_fromUtf8(path+"/placeholder.png")))
199. self.profilePic.setScaledContents(True)
200. self.profilePic.setObjectName(\_fromUtf8("profilePic"))
202. #Name Label
203. self.resultTitle = QtGui.QLabel(self.centralwidget)
204. self.resultTitle.setGeometry(QtCore.QRect(140, 280, 551, 30))
205. self.resultTitle.setFont(labelfont2)
206. self.resultTitle.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
207. self.resultTitle.setObjectName(\_fromUtf8("resultTitle"))
209. #Button that links to user page
210. self.resultButton = QtGui.QPushButton(self.centralwidget)
211. self.resultButton.setGeometry(QtCore.QRect(20, 270, 931, 71))
212. self.resultButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
213. self.resultButton.setText(\_fromUtf8(""))
214. self.resultButton.setObjectName(\_fromUtf8("resultButton"))
216. #White Rectangle for the result
217. self.resultRectangle = QtGui.QLabel(self.centralwidget)
218. self.resultRectangle.setGeometry(QtCore.QRect(20, 267, 931, 71))
219. self.resultRectangle.setAutoFillBackground(False)
220. self.resultRectangle.setStyleSheet(\_fromUtf8("border: 1px solid black;\n"
221. "background-color: rgb(255, 255, 255);"))
222. self.resultRectangle.setText(\_fromUtf8(""))
223. self.resultRectangle.setObjectName(\_fromUtf8("resultRectangle"))
225. #Label describes the type of user.
226. self.resultType = QtGui.QLabel(self.centralwidget)
227. self.resultType.setGeometry(QtCore.QRect(140, 305, 91, 21))
228. self.resultType.setFont(labelfont3)
229. self.resultType.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
230. self.resultType.setObjectName(\_fromUtf8("resultType"))
232. self.result1 = SearchBox(self.resultRectangle,self.resultType,self.profilePic,self.resultButton,self.resultTitle,self.typeOfWindow,self.homework)
234. ## RESULT 2
236. self.resultTitle\_2 = QtGui.QLabel(self.centralwidget)
237. self.resultTitle\_2.setGeometry(QtCore.QRect(140, 355, 551, 30))
238. self.resultTitle\_2.setFont(labelfont2)
239. self.resultTitle\_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
240. self.resultTitle\_2.setObjectName(\_fromUtf8("resultTitle\_2"))
242. self.profilePic\_2 = QtGui.QLabel(self.centralwidget)
243. self.profilePic\_2.setGeometry(QtCore.QRect(60, 345, 63, 63))
244. self.profilePic\_2.setText(\_fromUtf8(""))
245. self.profilePic\_2.setPixmap(QtGui.QPixmap(\_fromUtf8(path+"/placeholder.png")))
246. self.profilePic\_2.setScaledContents(True)
247. self.profilePic\_2.setObjectName(\_fromUtf8("profilePic\_2"))
249. self.resultButton\_2 = QtGui.QPushButton(self.centralwidget)
250. self.resultButton\_2.setGeometry(QtCore.QRect(20, 345, 931, 71))
251. self.resultButton\_2.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
252. self.resultButton\_2.setText(\_fromUtf8(""))
253. self.resultButton\_2.setObjectName(\_fromUtf8("resultButton\_2"))
255. self.resultType\_2 = QtGui.QLabel(self.centralwidget)
256. self.resultType\_2.setGeometry(QtCore.QRect(140, 380, 91, 21))
257. self.resultType\_2.setFont(labelfont3)
258. self.resultType\_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
259. self.resultType\_2.setObjectName(\_fromUtf8("resultType\_2"))
261. self.resultRectangle\_2 = QtGui.QLabel(self.centralwidget)
262. self.resultRectangle\_2.setGeometry(QtCore.QRect(20, 342, 931, 71))
263. self.resultRectangle\_2.setAutoFillBackground(False)
264. self.resultRectangle\_2.setStyleSheet(\_fromUtf8("border: 1px solid black;\n"
265. "background-color: rgb(255, 255, 255);"))
266. self.resultRectangle\_2.setText(\_fromUtf8(""))
267. self.resultRectangle\_2.setObjectName(\_fromUtf8("resultRectangle\_2"))
269. self.result2 = SearchBox(self.resultRectangle\_2,self.resultType\_2,self.profilePic\_2,self.resultButton\_2,self.resultTitle\_2,self.typeOfWindow,self.homework)

272. ## RESULT 3
274. self.resultTitle\_3 = QtGui.QLabel(self.centralwidget)
275. self.resultTitle\_3.setGeometry(QtCore.QRect(140, 430, 551, 30))
276. self.resultTitle\_3.setFont(labelfont2)
277. self.resultTitle\_3.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
278. self.resultTitle\_3.setObjectName(\_fromUtf8("resultTitle\_3"))
280. self.resultType\_3 = QtGui.QLabel(self.centralwidget)
281. self.resultType\_3.setGeometry(QtCore.QRect(140, 455, 91, 21))
282. self.resultType\_3.setFont(labelfont3)
283. self.resultType\_3.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
284. self.resultType\_3.setObjectName(\_fromUtf8("resultType\_3"))
286. self.resultRectangle\_3 = QtGui.QLabel(self.centralwidget)
287. self.resultRectangle\_3.setGeometry(QtCore.QRect(20, 417, 931, 71))
288. self.resultRectangle\_3.setAutoFillBackground(False)
289. self.resultRectangle\_3.setStyleSheet(\_fromUtf8("border: 1px solid black;\n"
290. "background-color: rgb(255, 255, 255);"))
291. self.resultRectangle\_3.setText(\_fromUtf8(""))
292. self.resultRectangle\_3.setObjectName(\_fromUtf8("resultRectangle\_3"))
294. self.resultButton\_3 = QtGui.QPushButton(self.centralwidget)
295. self.resultButton\_3.setGeometry(QtCore.QRect(20, 420, 931, 71))
296. self.resultButton\_3.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
297. self.resultButton\_3.setText(\_fromUtf8(""))
298. self.resultButton\_3.setObjectName(\_fromUtf8("resultButton\_3"))
300. self.profilePic\_3 = QtGui.QLabel(self.centralwidget)
301. self.profilePic\_3.setGeometry(QtCore.QRect(60, 420, 63, 63))
302. self.profilePic\_3.setText(\_fromUtf8(""))
303. self.profilePic\_3.setPixmap(QtGui.QPixmap(\_fromUtf8(path+"/placeholder.png")))
304. self.profilePic\_3.setScaledContents(True)
305. self.profilePic\_3.setObjectName(\_fromUtf8("profilePic\_3"))
307. self.result3 = SearchBox(self.resultRectangle\_3,self.resultType\_3,self.profilePic\_3,self.resultButton\_3,self.resultTitle\_3,self.typeOfWindow,self.homework)
309. ## RESULT 4
310. self.resultTitle\_4 = QtGui.QLabel(self.centralwidget)
311. self.resultTitle\_4.setGeometry(QtCore.QRect(140, 505, 551, 30))
312. self.resultTitle\_4.setFont(labelfont2)
313. self.resultTitle\_4.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
314. self.resultTitle\_4.setObjectName(\_fromUtf8("resultTitle\_4"))
316. self.resultRectangle\_4 = QtGui.QLabel(self.centralwidget)
317. self.resultRectangle\_4.setGeometry(QtCore.QRect(20, 492, 931, 71))
318. self.resultRectangle\_4.setAutoFillBackground(False)
319. self.resultRectangle\_4.setStyleSheet(\_fromUtf8("border: 1px solid black;\n"
320. "background-color: rgb(255, 255, 255);"))
321. self.resultRectangle\_4.setText(\_fromUtf8(""))
322. self.resultRectangle\_4.setObjectName(\_fromUtf8("resultRectangle\_4"))
324. self.profilePic\_4 = QtGui.QLabel(self.centralwidget)
325. self.profilePic\_4.setGeometry(QtCore.QRect(60, 495, 63, 63))
326. self.profilePic\_4.setText(\_fromUtf8(""))
327. self.profilePic\_4.setPixmap(QtGui.QPixmap(\_fromUtf8(path+"/placeholder.png")))
328. self.profilePic\_4.setScaledContents(True)
329. self.profilePic\_4.setObjectName(\_fromUtf8("profilePic\_4"))
331. self.resultButton\_4 = QtGui.QPushButton(self.centralwidget)
332. self.resultButton\_4.setGeometry(QtCore.QRect(20, 495, 931, 71))
333. self.resultButton\_4.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
334. self.resultButton\_4.setText(\_fromUtf8(""))
335. self.resultButton\_4.setObjectName(\_fromUtf8("resultButton\_4"))
337. self.resultType\_4 = QtGui.QLabel(self.centralwidget)
338. self.resultType\_4.setGeometry(QtCore.QRect(140, 530, 91, 21))
339. self.resultType\_4.setFont(labelfont3)
340. self.resultType\_4.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
341. self.resultType\_4.setObjectName(\_fromUtf8("resultType\_4"))
343. self.result4 = SearchBox(self.resultRectangle\_4,self.resultType\_4,self.profilePic\_4,self.resultButton\_4,self.resultTitle\_4,self.typeOfWindow,self.homework)
345. ## RESULT 5
346. self.resultRectangle\_5 = QtGui.QLabel(self.centralwidget)
347. self.resultRectangle\_5.setGeometry(QtCore.QRect(20, 567, 931, 71))
348. self.resultRectangle\_5.setAutoFillBackground(False)
349. self.resultRectangle\_5.setStyleSheet(\_fromUtf8("border: 1px solid black;\n"
350. "background-color: rgb(255, 255, 255);"))
351. self.resultRectangle\_5.setText(\_fromUtf8(""))
352. self.resultRectangle\_5.setObjectName(\_fromUtf8("resultRectangle\_5"))
354. self.resultTitle\_5 = QtGui.QLabel(self.centralwidget)
355. self.resultTitle\_5.setGeometry(QtCore.QRect(140, 580, 551, 30))
356. self.resultTitle\_5.setFont(labelfont2)
357. self.resultTitle\_5.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
358. self.resultTitle\_5.setObjectName(\_fromUtf8("resultTitle\_5"))
360. self.profilePic\_5 = QtGui.QLabel(self.centralwidget)
361. self.profilePic\_5.setGeometry(QtCore.QRect(60, 570, 63, 63))
362. self.profilePic\_5.setText(\_fromUtf8(""))
363. self.profilePic\_5.setPixmap(QtGui.QPixmap(\_fromUtf8(path+"/placeholder.png")))
364. self.profilePic\_5.setScaledContents(True)
365. self.profilePic\_5.setObjectName(\_fromUtf8("profilePic\_5"))
367. self.resultType\_5 = QtGui.QLabel(self.centralwidget)
368. self.resultType\_5.setGeometry(QtCore.QRect(140, 605, 91, 21))
369. self.resultType\_5.setFont(labelfont3)
370. self.resultType\_5.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
371. self.resultType\_5.setObjectName(\_fromUtf8("resultType\_5"))
373. self.resultButton\_5 = QtGui.QPushButton(self.centralwidget)
374. self.resultButton\_5.setGeometry(QtCore.QRect(20, 570, 931, 71))
375. self.resultButton\_5.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
376. self.resultButton\_5.setText(\_fromUtf8(""))
377. self.resultButton\_5.setObjectName(\_fromUtf8("resultButton\_5"))
379. self.result5 = SearchBox(self.resultRectangle\_5,self.resultType\_5,self.profilePic\_5,self.resultButton\_5,self.resultTitle\_5,self.typeOfWindow,self.homework)
381. ## RESULT 6
382. self.resultButton\_6 = QtGui.QPushButton(self.centralwidget)
383. self.resultButton\_6.setGeometry(QtCore.QRect(20, 645, 931, 71))
384. self.resultButton\_6.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
385. self.resultButton\_6.setText(\_fromUtf8(""))
386. self.resultButton\_6.setObjectName(\_fromUtf8("resultButton\_6"))
388. self.resultRectangle\_6 = QtGui.QLabel(self.centralwidget)
389. self.resultRectangle\_6.setGeometry(QtCore.QRect(20, 642, 931, 71))
390. self.resultRectangle\_6.setAutoFillBackground(False)
391. self.resultRectangle\_6.setStyleSheet(\_fromUtf8("border: 1px solid black;\n"
392. "background-color: rgb(255, 255, 255);"))
393. self.resultRectangle\_6.setText(\_fromUtf8(""))
394. self.resultRectangle\_6.setObjectName(\_fromUtf8("resultRectangle\_6"))
396. self.resultTitle\_6 = QtGui.QLabel(self.centralwidget)
397. self.resultTitle\_6.setGeometry(QtCore.QRect(140, 655, 551, 30))
398. self.resultTitle\_6.setFont(labelfont2)
399. self.resultTitle\_6.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
400. self.resultTitle\_6.setObjectName(\_fromUtf8("resultTitle\_6"))
402. self.profilePic\_6 = QtGui.QLabel(self.centralwidget)
403. self.profilePic\_6.setGeometry(QtCore.QRect(60, 645, 63, 63))
404. self.profilePic\_6.setText(\_fromUtf8(""))
405. self.profilePic\_6.setPixmap(QtGui.QPixmap(\_fromUtf8(path+"/placeholder.png")))
406. self.profilePic\_6.setScaledContents(True)
407. self.profilePic\_6.setObjectName(\_fromUtf8("profilePic\_6"))
409. self.resultType\_6 = QtGui.QLabel(self.centralwidget)
410. self.resultType\_6.setGeometry(QtCore.QRect(140, 680, 91, 21))
411. self.resultType\_6.setFont(labelfont3)
412. self.resultType\_6.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
413. self.resultType\_6.setObjectName(\_fromUtf8("resultType\_6"))
415. self.result6 = SearchBox(self.resultRectangle\_6,self.resultType\_6,self.profilePic\_6,self.resultButton\_6,self.resultTitle\_6,self.typeOfWindow,self.homework)
417. ## RESULT 7
418. self.resultRectangle\_7 = QtGui.QLabel(self.centralwidget)
419. self.resultRectangle\_7.setGeometry(QtCore.QRect(20, 717, 931, 71))
420. self.resultRectangle\_7.setAutoFillBackground(False)
421. self.resultRectangle\_7.setStyleSheet(\_fromUtf8("border: 1px solid black;\n"
422. "background-color: rgb(255, 255, 255);"))
423. self.resultRectangle\_7.setText(\_fromUtf8(""))
424. self.resultRectangle\_7.setObjectName(\_fromUtf8("resultRectangle\_7"))
426. self.resultType\_7 = QtGui.QLabel(self.centralwidget)
427. self.resultType\_7.setGeometry(QtCore.QRect(140, 755, 91, 21))
428. self.resultType\_7.setFont(labelfont3)
429. self.resultType\_7.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
430. self.resultType\_7.setObjectName(\_fromUtf8("resultType\_7"))
432. self.profilePic\_7 = QtGui.QLabel(self.centralwidget)
433. self.profilePic\_7.setGeometry(QtCore.QRect(60, 720, 63, 63))
434. self.profilePic\_7.setText(\_fromUtf8(""))
435. self.profilePic\_7.setPixmap(QtGui.QPixmap(\_fromUtf8(path+"/placeholder.png")))
436. self.profilePic\_7.setScaledContents(True)
437. self.profilePic\_7.setObjectName(\_fromUtf8("profilePic\_7"))
439. self.resultButton\_7 = QtGui.QPushButton(self.centralwidget)
440. self.resultButton\_7.setGeometry(QtCore.QRect(20, 720, 931, 71))
441. self.resultButton\_7.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
442. self.resultButton\_7.setText(\_fromUtf8(""))
443. self.resultButton\_7.setObjectName(\_fromUtf8("resultButton\_7"))
445. self.resultTitle\_7 = QtGui.QLabel(self.centralwidget)
446. self.resultTitle\_7.setGeometry(QtCore.QRect(140, 730, 551, 30))
447. self.resultTitle\_7.setFont(labelfont2)
448. self.resultTitle\_7.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
449. self.resultTitle\_7.setObjectName(\_fromUtf8("resultTitle\_7"))
451. self.result7 = SearchBox(self.resultRectangle\_7,self.resultType\_7,self.profilePic\_7,self.resultButton\_7,self.resultTitle\_7,self.typeOfWindow,self.homework)
453. ################## BOTTOM OF PAGE INFO ####################################
455. #Label includes amount of results and how many pages worth of users there are.
456. self.amountLabel = QtGui.QLabel(self.centralwidget)
457. self.amountLabel.setGeometry(QtCore.QRect(30, 810, 151, 21))
458. self.amountLabel.setFont(labelfont3)
459. self.amountLabel.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
460. self.amountLabel.setObjectName(\_fromUtf8("amountLabel"))
462. #Page number
463. self.pageNumber = QtGui.QLabel(self.centralwidget)
464. self.pageNumber.setGeometry(QtCore.QRect(460, 810, 51, 21))
465. self.pageNumber.setFont(labelfont3)
466. self.pageNumber.setAlignment(QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
467. self.pageNumber.setObjectName(\_fromUtf8("pageNumber"))
469. #Previous button
470. self.previousButton = QtGui.QPushButton(self.centralwidget)
471. self.previousButton.setGeometry(QtCore.QRect(380, 810, 81, 21))
472. self.previousButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
473. self.previousButton.setText(\_fromUtf8(""))
474. self.previousButton.setObjectName(\_fromUtf8("previousButton"))
475. self.previousButton.clicked.connect(self.previous\_page)
477. #Next button
478. self.nextButton = QtGui.QPushButton(self.centralwidget)
479. self.nextButton.setGeometry(QtCore.QRect(510, 810, 81, 21))
480. self.nextButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
481. self.nextButton.setText(\_fromUtf8(""))
482. self.nextButton.setObjectName(\_fromUtf8("nextButton"))
483. self.nextButton.clicked.connect(self.next\_page)

486. #################################################################################################

489. #Arranging objects
490. #Though this may seem long and pointless - this is just to ensure that every object
491. #is visible and not covered by another object.
492. self.resultRectangle.raise\_()
493. self.filterRect.raise\_()
494. self.titleLabel.raise\_()
495. self.nameLabel.raise\_()
496. self.nameEdit.raise\_()
497. self.dateEdit.raise\_()
498. self.dobCheck.raise\_()
499. self.fullNameCheck.raise\_()
500. self.usernameLabel.raise\_()
501. self.usernameEdit.raise\_()
502. self.usernameCheck.raise\_()
503. self.typeCombo.raise\_()
504. self.typeCheck.raise\_()
505. self.subjectCheck.raise\_()
506. self.subjectCombo.raise\_()
507. self.classCheck.raise\_()
508. self.classCombo.raise\_()
509. self.sortCombo.raise\_()
510. self.sortLabel.raise\_()
511. self.profilePic.raise\_()
512. self.resultTitle.raise\_()
513. self.resultType.raise\_()
514. self.searchButton.raise\_()
515. self.yearCheck.raise\_()
516. self.yearCombo.raise\_()
517. self.resultRectangle\_2.raise\_()
518. self.profilePic\_2.raise\_()
519. self.resultType\_2.raise\_()
520. self.resultTitle\_2.raise\_()
521. self.resultRectangle\_3.raise\_()
522. self.profilePic\_3.raise\_()
523. self.resultType\_3.raise\_()
524. self.resultTitle\_3.raise\_()
525. self.resultRectangle\_4.raise\_()
526. self.profilePic\_4.raise\_()
527. self.resultType\_4.raise\_()
528. self.resultTitle\_4.raise\_()
529. self.resultRectangle\_5.raise\_()
530. self.resultTitle\_5.raise\_()
531. self.profilePic\_5.raise\_()
532. self.resultType\_5.raise\_()
533. self.resultRectangle\_6.raise\_()
534. self.resultTitle\_6.raise\_()
535. self.profilePic\_6.raise\_()
536. self.resultType\_6.raise\_()
537. self.resultButton\_6.raise\_()
538. self.resultButton.raise\_()
539. self.resultButton\_2.raise\_()
540. self.resultButton\_3.raise\_()
541. self.resultButton\_4.raise\_()
542. self.resultButton\_5.raise\_()
543. self.resultRectangle\_7.raise\_()
544. self.resultType\_7.raise\_()
545. self.profilePic\_7.raise\_()
546. self.resultButton\_7.raise\_()
547. self.resultTitle\_7.raise\_()


551. SearchUsers.setCentralWidget(self.centralwidget)
552. self.menubar = QtGui.QMenuBar(SearchUsers)
553. self.menubar.setGeometry(QtCore.QRect(0, 0, 973, 21))
554. self.menubar.setObjectName(\_fromUtf8("menubar"))
555. SearchUsers.setMenuBar(self.menubar)
556. self.statusbar = QtGui.QStatusBar(SearchUsers)
557. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
558. SearchUsers.setStatusBar(self.statusbar)
560. self.retranslateUi()
561. QtCore.QMetaObject.connectSlotsByName(SearchUsers)



566. **def** retranslateUi(self):
568. #Going through data. If there is not enough results to show all results then
569. #result box will be hidden. Else, the objects will be changed so the correct details appear.
570. **if** len(self.data) >= 1+((self.page-1)\*7):
571. self.result1.retranslateUi(self.data[0+(self.page-1)\*7][0],self.data[0+(self.page-1)\*7][1],self.data[0+(self.page-1)\*7][2],self.data[0+(self.page-1)\*7][6],self.data[0+(self.page-1)\*7][7])
572. self.result1.show\_all()
574. **else**:
575. self.result1.hide\_all()
577. **if** len(self.data) >= 2+((self.page-1)\*7):
578. self.result2.retranslateUi(self.data[1+(self.page-1)\*7][0],self.data[1+(self.page-1)\*7][1],self.data[1+(self.page-1)\*7][2],self.data[1+(self.page-1)\*7][6],self.data[1+(self.page-1)\*7][7])
579. self.result2.show\_all()
580. **else**:
581. self.result2.hide\_all()
583. **if** len(self.data) >= 3+(self.page-1)\*7:
584. self.result3.retranslateUi(self.data[2+(self.page-1)\*7][0],self.data[2+(self.page-1)\*7][1],self.data[2+(self.page-1)\*7][2],self.data[2+(self.page-1)\*7][6],self.data[2+(self.page-1)\*7][7])
585. self.result3.show\_all()
586. **else**:
587. self.result3.hide\_all()
589. **if** len(self.data) >= 4+(self.page-1)\*7:
590. self.result4.retranslateUi(self.data[3+(self.page-1)\*7][0],self.data[3+(self.page-1)\*7][1],self.data[3+(self.page-1)\*7][2],self.data[3+(self.page-1)\*7][6],self.data[3+(self.page-1)\*7][7])
591. self.result4.show\_all()
592. **else**:
593. self.result4.hide\_all()
595. **if** len(self.data) >= 5+(self.page-1)\*7:
596. self.result5.retranslateUi(self.data[4+(self.page-1)\*7][0],self.data[4+(self.page-1)\*7][1],self.data[4+(self.page-1)\*7][2],self.data[4+(self.page-1)\*7][6],self.data[4+(self.page-1)\*7][7])
597. self.result5.show\_all()
598. **else**:
599. self.result5.hide\_all()
601. **if** len(self.data) >= 6+(self.page-1)\*7:
602. self.result6.retranslateUi(self.data[5+(self.page-1)\*7][0],self.data[5+(self.page-1)\*7][1],self.data[5+(self.page-1)\*7][2],self.data[5+(self.page-1)\*7][6],self.data[5+(self.page-1)\*7][7])
603. self.result6.show\_all()
604. **else**:
605. self.result6.hide\_all()
607. **if** len(self.data) >= 7+(self.page-1)\*7:
608. self.result7.retranslateUi(self.data[6+(self.page-1)\*7][0],self.data[6+(self.page-1)\*7][1],self.data[6+(self.page-1)\*7][2],self.data[6+(self.page-1)\*7][6],self.data[6+(self.page-1)\*7][7])
609. self.result7.show\_all()
610. **else**:
611. self.result7.hide\_all()

614. self.window.setWindowTitle(\_translate("SearchUsers", "Search User Window", None))
615. self.titleLabel.setText(\_translate("SearchUsers", "SEARCH USERS", None))
616. self.nameLabel.setText(\_translate("SearchUsers", "FULL NAME", None))
617. self.dobCheck.setText(\_translate("SearchUsers", "Filter by date of birth?", None))
618. self.fullNameCheck.setText(\_translate("SearchUsers", "Filter by full name?", None))
619. self.usernameLabel.setText(\_translate("SearchUsers", "USERNAME", None))
620. self.usernameCheck.setText(\_translate("SearchUsers", "Filter by username?", None))
621. self.typeCheck.setText(\_translate("SearchUsers", "Filter by type of user?", None))
622. **if** self.typeOfWindow == "Homework":
623. self.subjectCheck.setText(\_translate("SearchUsers", "Filter by grade?", None))
624. **else**:
625. self.subjectCheck.setText(\_translate("SearchUsers", "Filter by subject?", None))
626. self.classCheck.setText(\_translate("SearchUsers", "Filter by class?", None))
627. self.sortLabel.setText(\_translate("SearchUsers", "Sort By:", None))
628. self.searchButton.setText(\_translate("SearchUsers", "SEARCH", None))
629. self.amountLabel.setText(\_translate("SearchUsers", str(len(self.data)) + " Results (" + str((len(self.data))//7+1 )+ " Pages)", None))
630. self.pageNumber.setText(\_translate("SearchUsers", str(self.page), None))
631. self.previousButton.setText(\_translate("SearchUsers", "Previous", None))
632. self.window.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
633. **if** self.page == 1:
634. self.previousButton.setEnabled(False)
635. **else**:
636. self.previousButton.setEnabled(True)
637. self.nextButton.setText(\_translate("SearchUsers", "Next", None))
639. **if** self.page < (len(self.data)+1)//7+1:
640. self.nextButton.setEnabled(True)
641. **else**:
642. self.nextButton.setEnabled(False)
643. self.yearCheck.setText(\_translate("SearchUsers", "Filter by year group?", None))
645. **def** next\_page(self):
646. self.page += 1
647. self.retranslateUi()
649. **def** previous\_page(self):
650. self.page -= 1
651. self.retranslateUi()
653. **def** search(self):
655. #If searching for all users
656. **if** self.typeOfWindow == "Search":
657. c.execute("SELECT \* FROM USERS WHERE TYPE = 'Admin' OR TYPE = 'Teacher'")
658. data1 = c.fetchall()
659. c.execute("SELECT \* FROM USERS,STUDENT WHERE USERS.USERNAME = STUDENT.USERNAME")
660. data2 = c.fetchall()
662. self.data = []
663. **for** i **in** range(len(data1)):
664. #Converting the tuples to a list
665. self.data.append(list(data1[i]))
667. **for** i **in** range(len(data2)):
668. #Converting the tuples to a list
669. self.data.append(list(data2[i]))
671. #If a teacher is searching through homework only users from the class. This instance will be used for updating grades.
672. **elif** self.typeOfWindow == "Homework":
673. c.execute("SELECT \* FROM USERS,"+self.homework.classId + " WHERE USERS.USERNAME = "+self.homework.classId+".Student")
674. data = c.fetchall()
675. self.data = []
676. **for** i **in** range(len(data)):
677. self.data.append(list(data[i]))



682. #Filter by full name
683. **if** self.fullNameCheck.isChecked():
684. search = self.nameEdit.text()
685. filtered = []
686. **for** i **in** range(len(self.data)):
687. **if** SequenceMatcher(None, self.data[i][0] + " " + self.data[i][1],search).ratio() > 0.75 **or** search.lower() **in** self.data[i][0].lower() + " " + self.data[i][1].lower():
688. filtered.append(self.data[i])
689. self.data = filtered

692. #Filter by username
693. **if** self.usernameCheck.isChecked():
694. search = self.usernameEdit.text()
695. filtered = []
696. **for** i **in** range(len(self.data)):
697. **if** SequenceMatcher(None, self.data[i][2],search).ratio() > 0.75 **or** search.lower() **in** self.data[i][2].lower():
698. filtered.append(self.data[i])
699. self.data = filtered
701. #Filter by date of birth
702. **if** self.dobCheck.isChecked():
703. filtered = []
704. dob = str(self.dateEdit.date().toPyDate())
705. **for** i **in** range(len(self.data)):
706. **if** self.data[i][4] == dob:
707. filtered.append(self.data[i])
708. self.data = filtered
710. #Filter by type of user
711. **if** self.typeCheck.isChecked():
712. filtered = []
713. typeOfUser = self.typeCombo.currentText()
714. **for** i **in** range(len(self.data)):
715. **if** self.data[i][6] == typeOfUser:
716. filtered.append(self.data[i])
717. self.data = filtered
719. #Filter by year group - only if the type of user is student.
720. **if** self.yearCheck.isChecked():
721. filtered = []
722. yearGroup = self.yearCombo.currentText()
723. **if** self.typeCheck.isChecked():
724. **if** typeOfUser != "Student":
725. QtGui.QMessageBox.question(self.window,"Error has occurred","Search Unsuccessful: " + typeOfUser + "does not have attribute 'Year Group'",
726. QtGui.QMessageBox.Ok)
727. **return**
729. **for** i **in** range(len(self.data)):
730. **if** self.data[i][6] == "Student":
731. **if** self.data[i][9] == int(yearGroup):
732. filtered.append(self.data[i])
733. self.data = filtered
735. #Filter by those studying/teaching a subject
736. **if** self.subjectCheck.isChecked():
737. filtered = []
738. **if** self.typeOfWindow == "Search":
739. **if** self.typeCheck.isChecked():
740. **if** typeOfUser == "Admin":
741. QtGui.QMessageBox.question(self.window,"Error has occurred","Search Unsuccessful: " + typeOfUser + "does not have attribute 'Subject'",
742. QtGui.QMessageBox.Ok)
743. **return**
745. c.execute("SELECT id,teacher FROM classes WHERE subject = :subject",{"subject":self.subjectCombo.currentText()})
746. classes = c.fetchall()
747. users = []
748. **for** i **in** range(len(classes)):
749. c.execute("SELECT student FROM "+classes[i][0])
750. temp = c.fetchall()
751. students = []
752. **if** temp != None:
753. **for** i **in** range(len(temp)):
754. students.append(temp[i][0])
755. users.extend(students)
756. **for** i **in** range(len(self.data)):
757. **if** self.data[i][2] **in** users:
758. filtered.append(self.data[i][2])
759. teachers = []
760. **for** i **in** range(len(classes)):
761. teachers.append(classes[i][2])
762. **for** i **in** range(len(self.data)):
763. **if** self.data[i][2] **in** teachers:
764. filtered.append(self.data)
765. #If being used for the context of homework - this will be a filter by grade option.
766. **else**:
767. grade = self.subjectCombo.currentText()
769. **if** grade == "Not Completed":
770. #c.execute("SELECT Student FROM "+self.homework.classId+" WHERE :homeworkId is Null",
771. #{"homeworkId":self.homework.homeworkId})
772. c.execute("SELECT Student,"+ self.homework.homeworkId+" FROM "+self.homework.classId)
773. data = c.fetchall()
774. notcompleted = []
775. **for** i **in** range(len(data)):
776. **if** data[i][1] == None:
777. notcompleted.append(data[i][0])
778. **for** i **in** range(len(self.data)):
779. **if** self.data[i][2] **in** notcompleted:
780. filtered.append(self.data[i])
781. **else**:
782. c.execute("SELECT Student FROM "+self.homework.classId+" WHERE :homeworkId = :grade",
783. {"homeworkId":self.homework.homeworkId,"grade":grade})
784. data = c.fetchall()
785. students = []
786. **for** i **in** range(len(data)):
787. students.append(data[i][0])
788. **for** i **in** range(len(self.data)):
789. **if** self.data[i][2] **in** students:
790. filtered.append(self.data[i])
792. self.data = filtered
794. #Filter by class
795. **if** self.classCheck.isChecked():
796. filtered = []
797. **if** self.typeCheck.isChecked():
798. **if** typeOfUser == "Admin":
799. QtGui.QMessageBox.question(self.window,"Error has occurred","Search Unsuccessful: " + typeOfUser + "does not have attribute 'Class'",
800. QtGui.QMessageBox.Ok)
801. **return**
802. c.execute("SELECT student FROM "+classes[i][0])
803. temp = c.fetchall()
804. students = []
805. **if** temp != None:
806. **for** i **in** range(len(temp)):
807. students.append(temp[i][0])
808. **for** i **in** range(len(self.data)):
809. **if** self.data[i][2] **in** students:
810. filtered.append(self.data)
811. c.execute("SELECT teacher FROM classes WHERE id = :id",{"id":self.classCombo.currentText()})
812. data = c.fetchall()
813. **for** i **in** range(len(self.data)):
814. **if** self.data[i][2] == data[0][0]:
815. filtered.append(self.data[i])
817. self.data = filtered
819. #Sorting the results.
820. **if** "Name" **in** self.sortCombo.currentText():
821. self.data = sorted(self.data, key=**lambda** x : x[0])
822. **if** "DESC" **in** self.sortCombo.currentText():
823. self.data.reverse()
824. **elif** self.sortCombo.currentText() == "Date Of Birth":
825. self.data = sorted(self.data, key=**lambda** x : x[4])
826. **if** "DESC" **in** self.sortCombo.currentText():
827. self.data = self.data.reverse()
829. self.retranslateUi()
831. #Search box is a class so I can combine all searchbox objects into one.
832. **class** SearchBox():
834. **def** \_\_init\_\_(self,rectangle,resultType,pic,button,name,typeOfSearch,homework):
835. self.rectangle = rectangle

9

1. self.type = resultType
2. self.pic = pic
3. self.button = button
4. self.name = name
5. self.typeOfSearch = typeOfSearch
6. self.homework = homework
8. self.button.clicked.connect(self.open\_window)


12. **def** retranslateUi(self,firstName,lastName,username,resultType,pic):
13. self.username = username
14. self.first = firstName
15. self.last = lastName
16. self.type.setText(\_translate("SearchUsers", resultType, None))
17. self.name.setText(\_translate("SearchUsers", firstName + " " + lastName + "(" + username
18. + ")", None))
19. **if** os.path.isfile(path + pic):
20. self.pic.setPixmap(QtGui.QPixmap(\_fromUtf8(path + pic)))
22. #Hides box
23. **def** hide\_all(self):
24. self.rectangle.hide()
25. self.type.hide()
26. self.pic.hide()
27. self.button.hide()
28. self.name.hide()
29. #Show box
30. **def** show\_all(self):
31. self.rectangle.show()
32. self.type.show()
33. self.pic.show()
34. self.button.show()
35. self.name.show()
37. **def** open\_window(self):
38. #Giving the button different function depending on what context it is used in.
39. **if** self.typeOfSearch == "Homework":
40. self.edit\_grade()
41. **return**
43. c.execute("SELECT \* FROM users WHERE username=:username ORDER BY username ASC",
44. {"username":self.username})
46. data = c.fetchone()
48. #Edit users
49. **if** data[6] == "Student":
50. c.execute("SELECT yeargroup,achievementpoints,behaviourpoints FROM student WHERE username=:username",{"username":self.username})
51. details = c.fetchone()
53. self.user = Student(data[0],data[1],data[2],data[3],data[4],data[5],data[6],details[0],data[7],details[1],details[2])
54. self.edit\_student()
55. **else**:
56. self.user = User(data[0],data[1],data[2],data[3],data[4],data[5],data[6],data[7])
57. **if** self.user.type == "Admin":
58. self.edit\_admin()
59. **if** self.user.type == "Teacher":
60. self.edit\_teacher()
62. #Edit Grade Window
63. **def** edit\_grade(self):
64. self.editGradePage = EditWindow()
65. self.editGradeUi = Ui\_GradeWindow()
66. self.editGradeUi.setupUi(self.editGradePage,self.homework,self.username,self.first,self.last)
67. self.editGradePage.show()
69. #Edit Admin Page
70. **def** edit\_admin(self):
71. self.createAdminPage = EditWindow()
72. self.createAdminUi = Ui\_EditUserWindow()
73. self.createAdminUi.setupUi(self.createAdminPage,"Admin",self.user)
74. self.createAdminPage.show()
76. #Edit Teacher Page
77. **def** edit\_teacher(self):
78. self.createTeacherPage = EditWindow()
79. self.createTeacherUi = Ui\_EditUserWindow()
80. self.createTeacherUi.setupUi(self.createTeacherPage,"Teacher",self.user)
81. self.createTeacherPage.show()
83. #Edit Student Page
84. **def** edit\_student(self):
85. self.createStudentPage = EditWindow()
86. self.createStudentUi = Ui\_EditUserWindow()
87. self.createStudentUi.setupUi(self.createStudentPage,"Student",self.user)
88. self.createStudentPage.show()

5

1. **class** Ui\_ClassListWindow(object):
3. **def** setupUi(self, ClassListWindow,username,typeOfUser,typeOfSearch):
5. #Year Group can be left as NULL for instances not needed.
6. #and TypeOfUser = Search to show classes window
8. self.data = []
9. self.page = 1
10. self.username = username
11. self.type = typeOfUser
12. self.typeOfUser = typeOfUser
13. self.typeOfSearch = typeOfSearch
14. self.window = ClassListWindow
15. self.window.setStyleSheet(css)
17. #Creating main window
18. ClassListWindow.setObjectName(\_fromUtf8("ClassListWindow"))
19. ClassListWindow.resize(642, 571)
20. self.centralwidget = QtGui.QWidget(ClassListWindow)
21. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
23. #Title Label
24. self.titleLabel = QtGui.QLabel(self.centralwidget)
25. self.titleLabel.setFont(titlefont)
26. self.titleLabel.setTextFormat(QtCore.Qt.PlainText)
27. self.titleLabel.setAlignment(QtCore.Qt.AlignCenter)
28. self.titleLabel.setObjectName(\_fromUtf8("titleLabel"))
30. #Find all of the users classes
31. **if** self.typeOfUser == "Student":
32. c.execute("SELECT classid FROM studentclass WHERE username = :username",{"username":self.username})
33. self.allClasses = list(c.fetchall())
34. **else**:
35. c.execute("SELECT id FROM classes WHERE teacher = :username",{"username":self.username})
36. self.allClasses = list(c.fetchall())
38. #If listing every possible class - there will be an option to filter by subject.
39. **if** self.typeOfSearch == "Search":
40. self.titleLabel.setGeometry(QtCore.QRect(0, 0, 641, 91))
42. self.subjectLabel = QtGui.QLabel(self.centralwidget)
43. self.subjectLabel.setFont(labelfont)
44. self.subjectLabel.setTextFormat(QtCore.Qt.PlainText)
45. self.subjectLabel.setGeometry(QtCore.QRect(20,90,141,51))
46. self.subjectLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
47. self.subjectLabel.setObjectName(\_fromUtf8("subjectLabel"))
49. self.subjectCombo = QtGui.QComboBox(self.centralwidget)
50. self.subjectCombo.setGeometry(QtCore.QRect(170, 100, 421, 31))
51. self.subjectCombo.setObjectName(\_fromUtf8("subjectCombo"))
53. #Selecting all subjects
54. c.execute("SELECT fullname FROM subjects ORDER BY fullname ASC")
55. data = c.fetchall()
56. **for** i **in** range(len(data)):
57. self.subjectCombo.addItem(data[i][0])
58. self.subject = self.subjectCombo.currentText()
59. self.subjectCombo.activated[str].connect(self.show\_classes)
61. **else**:
62. self.titleLabel.setGeometry(QtCore.QRect(140, 40, 411, 91))
63. #Profile Pic
64. self.profilePic = QtGui.QLabel(self.centralwidget)
65. self.profilePic.setGeometry(QtCore.QRect(30, 20, 126, 126))
66. self.profilePic.setText(\_fromUtf8(""))
67. #Fetching picture of user and setting it as the profile pic
68. c.execute("SELECT pic FROM users WHERE username = :username",{"username":self.username})
69. pic = c.fetchone()
70. **if** os.path.isfile(path + pic[0]):
71. self.profilePic.setPixmap(QtGui.QPixmap(\_fromUtf8(path + pic[0])))
72. **else**:
73. self.profilePic.setPixmap(QtGui.QPixmap(\_fromUtf8("placeholder.png")))
74. self.profilePic.setScaledContents(True)
75. self.profilePic.setObjectName(\_fromUtf8("profilePic"))
77. #Refresh button
78. self.refresh = QtGui.QPushButton(self.centralwidget)
79. self.refresh.setGeometry(QtCore.QRect(561, 0, 81,31))
80. self.refresh.setObjectName(\_fromUtf8("refresh"))
81. self.refresh.clicked.connect(self.refresh\_page)
83. ## Class 1
85. #Class Button
86. self.class\_1 = QtGui.QPushButton(self.centralwidget)
87. self.class\_1.setGeometry(QtCore.QRect(90, 150, 411, 31))
88. self.class\_1.setObjectName(\_fromUtf8("class\_1"))
90. #Remove Button
91. self.remove\_1 = QtGui.QPushButton(self.centralwidget)
92. self.remove\_1.setGeometry(QtCore.QRect(510, 150, 81, 31))
93. self.remove\_1.setObjectName(\_fromUtf8("remove\_1"))
95. self.class1 = UsersClass(self.class\_1,self.remove\_1,self.typeOfSearch,self.allClasses,self.typeOfUser,self.username,self.window)

98. ## Class 2
100. self.remove\_2 = QtGui.QPushButton(self.centralwidget)
101. self.remove\_2.setGeometry(QtCore.QRect(510, 190, 81, 31))
102. self.remove\_2.setObjectName(\_fromUtf8("remove\_2"))
104. self.class\_2 = QtGui.QPushButton(self.centralwidget)
105. self.class\_2.setGeometry(QtCore.QRect(90, 190, 411, 31))
106. self.class\_2.setObjectName(\_fromUtf8("class\_2"))
108. self.class2 = UsersClass(self.class\_2,self.remove\_2,self.typeOfSearch,self.allClasses,self.typeOfUser,self.username,self.window)
110. ## Class 3
112. self.remove\_3 = QtGui.QPushButton(self.centralwidget)
113. self.remove\_3.setGeometry(QtCore.QRect(510, 230, 81, 31))
114. self.remove\_3.setObjectName(\_fromUtf8("remove\_3"))
116. self.class\_3 = QtGui.QPushButton(self.centralwidget)
117. self.class\_3.setGeometry(QtCore.QRect(90, 230, 411, 31))
118. self.class\_3.setObjectName(\_fromUtf8("class\_3"))
120. self.class3 = UsersClass(self.class\_3,self.remove\_3,self.typeOfSearch,self.allClasses,self.typeOfUser,self.username,self.window)
122. ## Class 4
124. self.remove\_4 = QtGui.QPushButton(self.centralwidget)
125. self.remove\_4.setGeometry(QtCore.QRect(510, 270, 81, 31))
126. self.remove\_4.setObjectName(\_fromUtf8("remove\_4"))
128. self.class\_4 = QtGui.QPushButton(self.centralwidget)
129. self.class\_4.setGeometry(QtCore.QRect(90, 270, 411, 31))
130. self.class\_4.setObjectName(\_fromUtf8("class\_4"))
132. self.class4 = UsersClass(self.class\_4,self.remove\_4,self.typeOfSearch,self.allClasses,self.typeOfUser,self.username,self.window)
134. ##Class 5
136. self.remove\_5 = QtGui.QPushButton(self.centralwidget)
137. self.remove\_5.setGeometry(QtCore.QRect(510, 310, 81, 31))
138. self.remove\_5.setObjectName(\_fromUtf8("remove\_5"))
140. self.class\_5 = QtGui.QPushButton(self.centralwidget)
141. self.class\_5.setGeometry(QtCore.QRect(90, 310, 411, 31))
142. self.class\_5.setObjectName(\_fromUtf8("class\_5"))
144. self.class5 = UsersClass(self.class\_5,self.remove\_5,self.typeOfSearch,self.allClasses,self.typeOfUser,self.username,self.window)
146. ## Class 6
148. self.remove\_6 = QtGui.QPushButton(self.centralwidget)
149. self.remove\_6.setGeometry(QtCore.QRect(510, 350, 81, 31))
150. self.remove\_6.setObjectName(\_fromUtf8("remove\_6"))
152. self.class\_6 = QtGui.QPushButton(self.centralwidget)
153. self.class\_6.setGeometry(QtCore.QRect(90, 350, 411, 31))
154. self.class\_6.setObjectName(\_fromUtf8("class\_6"))
156. self.class6 = UsersClass(self.class\_6,self.remove\_6,self.typeOfSearch,self.allClasses,self.typeOfUser,self.username,self.window)
158. ## Class 7
160. self.class\_7 = QtGui.QPushButton(self.centralwidget)
161. self.class\_7.setGeometry(QtCore.QRect(90, 390, 411, 31))
162. self.class\_7.setObjectName(\_fromUtf8("class\_7"))
164. self.remove\_7 = QtGui.QPushButton(self.centralwidget)
165. self.remove\_7.setGeometry(QtCore.QRect(510, 390, 81, 31))
166. self.remove\_7.setObjectName(\_fromUtf8("remove\_7"))
168. self.class7 = UsersClass(self.class\_7,self.remove\_7,self.typeOfSearch,self.allClasses,self.typeOfUser,self.username,self.window)
170. ##Class 8
172. self.remove\_8 = QtGui.QPushButton(self.centralwidget)
173. self.remove\_8.setGeometry(QtCore.QRect(510, 430, 81, 31))
174. self.remove\_8.setObjectName(\_fromUtf8("remove\_8"))
176. self.class\_8 = QtGui.QPushButton(self.centralwidget)
177. self.class\_8.setGeometry(QtCore.QRect(90, 430, 411, 31))
178. self.class\_8.setObjectName(\_fromUtf8("class\_8"))
180. self.class8 = UsersClass(self.class\_8,self.remove\_8,self.typeOfSearch,self.allClasses,self.typeOfUser,self.username,self.window)
182. #Previous Button
183. self.previousButton = QtGui.QPushButton(self.centralwidget)
184. self.previousButton.setGeometry(QtCore.QRect(220, 510, 81, 21))
185. self.previousButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
186. self.previousButton.setText(\_fromUtf8(""))
187. self.previousButton.setObjectName(\_fromUtf8("previousButton"))
188. self.previousButton.clicked.connect(self.previous\_page)
190. #Next Button
191. self.nextButton = QtGui.QPushButton(self.centralwidget)
192. self.nextButton.setGeometry(QtCore.QRect(350, 510, 81, 21))
193. self.nextButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
194. self.nextButton.setText(\_fromUtf8(""))
195. self.nextButton.setObjectName(\_fromUtf8("nextButton"))
196. self.nextButton.clicked.connect(self.next\_page)
198. #Page Number
199. self.pageNumber = QtGui.QLabel(self.centralwidget)
200. self.pageNumber.setGeometry(QtCore.QRect(300, 510, 51, 21))
201. self.pageNumber.setFont(labelfont3)
202. self.pageNumber.setAlignment(QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
203. self.pageNumber.setObjectName(\_fromUtf8("pageNumber"))
205. #Amount Of Results
206. self.amountLabel = QtGui.QLabel(self.centralwidget)
207. self.amountLabel.setGeometry(QtCore.QRect(20, 510, 151, 21))
208. self.amountLabel.setFont(labelfont3)
209. self.amountLabel.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
210. self.amountLabel.setObjectName(\_fromUtf8("amountLabel"))
212. #New Class Button
213. self.newClassButton = QtGui.QPushButton(self.centralwidget)
214. self.newClassButton.setGeometry(QtCore.QRect(510, 510, 111, 21))
215. self.newClassButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;}"))
216. self.newClassButton.setText(\_fromUtf8(""))
217. self.newClassButton.setObjectName(\_fromUtf8("newClassButton"))
218. **if** self.typeOfSearch == "Search":
219. self.newClassButton.clicked.connect(self.new\_class)
220. **else**:
221. self.newClassButton.clicked.connect(self.class\_button)
223. **if** currentUser.type == "Student" **or** currentUser.type == "Teacher":
224. self.newClassButton.hide()
226. #Menu and Status Bars
227. ClassListWindow.setCentralWidget(self.centralwidget)
228. self.menubar = QtGui.QMenuBar(ClassListWindow)
229. self.menubar.setGeometry(QtCore.QRect(0, 0, 642, 21))
230. self.menubar.setObjectName(\_fromUtf8("menubar"))
231. ClassListWindow.setMenuBar(self.menubar)
232. self.statusbar = QtGui.QStatusBar(ClassListWindow)
233. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
234. ClassListWindow.setStatusBar(self.statusbar)
236. self.retranslateUi(ClassListWindow)
237. QtCore.QMetaObject.connectSlotsByName(ClassListWindow)
239. **def** retranslateUi(self, ClassListWindow):
240. self.search()
241. ClassListWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
243. #Similar to the search window - going through the data and adding the data to result boxes. If
244. # not enough data then hide the box
245. **if** len(self.data) >= 1+((self.page-1)\*8):
246. self.class1.retranslateUi(self.data[0+(self.page-1)\*8][3],self.data[0+(self.page-1)\*8][2])
247. self.class1.show\_all()
248. **else**:
249. self.class1.hide\_all()
251. **if** len(self.data) >= 2+((self.page-1)\*8):
252. self.class2.retranslateUi(self.data[1+(self.page-1)\*8][3],self.data[1+(self.page-1)\*8][2])
253. self.class2.show\_all()
254. **else**:
255. self.class2.hide\_all()
257. **if** len(self.data) >= 3+((self.page-1)\*8):
258. self.class3.retranslateUi(self.data[2+(self.page-1)\*8][3],self.data[2+(self.page-1)\*8][2])
259. self.class3.show\_all()
260. **else**:
261. self.class3.hide\_all()
263. **if** len(self.data) >= 4+((self.page-1)\*8):
264. self.class4.retranslateUi(self.data[3+(self.page-1)\*8][3],self.data[3+(self.page-1)\*8][2])
265. self.class4.show\_all()
266. **else**:
267. self.class4.hide\_all()
269. **if** len(self.data) >= 5+((self.page-1)\*8):
270. self.class5.retranslateUi(self.data[4+(self.page-1)\*8][3],self.data[4+(self.page-1)\*8][2])
271. self.class5.show\_all()
272. **else**:
273. self.class5.hide\_all()
275. **if** len(self.data) >= 6+((self.page-1)\*8):
276. self.class6.retranslateUi(self.data[5+(self.page-1)\*8][3],self.data[5+(self.page-1)\*8][2])
277. self.class6.show\_all()
278. **else**:
279. self.class6.hide\_all()
281. **if** len(self.data) >= 7+((self.page-1)\*8):
282. self.class7.retranslateUi(self.data[6+(self.page-1)\*8][3],self.data[6+(self.page-1)\*8][2])
283. self.class7.show\_all()
284. **else**:
285. self.class7.hide\_all()
287. **if** len(self.data) >= 8+((self.page-1)\*8):
288. self.class8.retranslateUi(self.data[7+(self.page-1)\*8][3],self.data[7+(self.page-1)\*8][2])
289. self.class8.show\_all()
290. **else**:
291. self.class8.hide\_all()
293. ClassListWindow.setWindowTitle(\_translate("ClassListWindow", "Classes Window", None))
294. **if** self.typeOfSearch == "Search":
295. self.titleLabel.setText(\_translate("ClassListWindow", "CLASSES", None))
296. self.subjectLabel.setText(\_translate("ClassListWindow", "Subject:", None))
297. **else**:
298. self.titleLabel.setText(\_translate("ClassListWindow", "USER CLASSES", None))
299. self.previousButton.setText(\_translate("ClassListWindow", "Previous", None))
300. **if** self.page == 1:
301. self.previousButton.setEnabled(False)
302. **else**:
303. self.previousButton.setEnabled(True)
305. **if** self.page < (len(self.data)+1)//7:
306. self.nextButton.setEnabled(True)
307. **else**:
308. self.nextButton.setEnabled(False)
310. self.nextButton.setText(\_translate("ClassListWindow", "Next", None))
311. self.refresh.setText(\_translate("ClassListWindow","Refresh",None))
312. self.pageNumber.setText(\_translate("ClassListWindow", str(self.page), None))
313. self.amountLabel.setText(\_translate("ClassListWindow", str(len(self.data))+ " Results ("+str(len(self.data)//7+1)+" Pages)", None))
314. **if** self.typeOfSearch == "Search":
315. self.newClassButton.setText(\_translate("ClassListWindow", "Create New Class", None))
316. **else**:
317. self.newClassButton.setText(\_translate("ClassListWindow", "Add New Class", None))

320. **def** next\_page(self):
321. self.page += 1
322. self.retranlsateUi()
324. **def** previous\_page(self):
325. self.page += 1
326. self.retranlsateUi(self.window)
328. **def** search(self):
329. self.data = []
330. #Depending on the context used in - searches will be different. Will either show a students classes
331. #or all classes.
332. **if** self.typeOfSearch == "Search":
333. c.execute("SELECT \* FROM classes WHERE subject = :subject",{"subject":self.subject})
334. data = list(c.fetchall())
335. **for** i **in** range(len(data)):
336. self.data.append(list(data[i]))
337. **elif** self.type == "Student":
339. c.execute("SELECT classid FROM studentclass WHERE username = :username",{"username":self.username})
340. data = list(c.fetchall())
341. **for** i **in** range(len(data)):
342. c.execute("SELECT \* FROM classes WHERE id = :id",{"id":data[i][0]})
343. temp = c.fetchone()
344. **if** temp != None:
345. self.data.append(list(temp))

348. **else**:
349. c.execute("SELECT \* FROM classes WHERE teacher = :username",{"username":self.username})
350. self.data = c.fetchall()

353. **def** class\_button(self):
354. self.showClassesPage = EditWindow()
355. self.showClassesUi = Ui\_ClassListWindow()
356. self.showClassesUi.setupUi(self.showClassesPage,self.username,self.typeOfUser,"Search")
357. self.showClassesPage.show()
359. **def** show\_classes(self,text):
360. self.subject = text
361. self.retranslateUi(self.window)
363. **def** new\_class(self):
364. self.classPage = EditWindow()
365. self.classui = Ui\_CreateClassWindow()
366. self.classui.setupUi(self.classPage,Class("NULL",self.username,"NULL","NULL","NULL","NULL","NULL","NULL"))
367. self.classPage.show()
369. **def** refresh\_page(self):
370. self.retranslateUi(self.window)

373. #UsersClass is another class for a box combining all objects into one making it easier
374. #to communicate with.

5

1. **class** UsersClass():
3. **def** \_\_init\_\_(self,main,remove,typeOfWindow,allClasses,typeOfUser,username,window):
4. self.typeOfUser = typeOfUser
5. self.allClasses =allClasses
6. self.main = main
7. self.remove = remove
8. self.type = typeOfWindow
9. self.username = username
10. self.window = window
11. self.main.clicked.connect(self.open\_window)
12. self.remove.clicked.connect(self.remove\_button)
13. self.signal = ""
15. **def** retranslateUi(self,id, subject):
16. self.id = id
17. self.subject = subject
18. self.main.setText(\_translate("ClassListWindow",id + " - " + subject,None))
19. #Depending on the context - the buttons with do different things.
20. **if** self.typeOfUser == "Teacher":
21. self.remove.setText(\_translate("ClassListWindow","Edit",None))
22. self.signal = "Edit"
24. **elif** self.type == "List":
25. self.signal = "Remove"
26. self.remove.setText(\_translate("ClassListWindow","Remove",None))
28. **else**:
29. self.signal = "Add"
30. self.remove.setText(\_translate("ClassListWindow","Add",None))
32. **for** i **in** range(len(self.allClasses)):
33. **if** self.id == self.allClasses[i][0]:
34. self.remove.setText(\_translate("ClassListWindow","Remove",None))
35. self.signal = "Remove"
36. **break**

39. **def** remove\_button(self):
40. **if** self.signal == "Edit":
41. self.open\_window()
42. **elif** self.signal == "Remove":
43. self.remove\_student()
44. **elif** self.signal == "Add":
45. self.add\_student()

48. **def** hide\_all(self):
49. self.main.hide()
50. self.remove.hide()
52. **def** show\_all(self):
53. self.main.show()
54. **if** currentUser.type == "Teacher" **or** currentUser.type == "Student":
55. self.remove.hide()
56. **else**:
57. self.remove.show()
59. **def** open\_window(self):
60. self.classPage = EditWindow()
61. self.classui = Ui\_CreateClassWindow()
62. c.execute("SELECT \* FROM classes WHERE id = :id",{"id":self.id})
63. data = c.fetchone()
64. self.classui.setupUi(self.classPage,Class(data[0],data[1],data[2],data[3],data[4],data[5],data[6],data[7]))
65. self.classPage.show()
67. **def** add\_student(self):
68. choice = QtGui.QMessageBox.question(MainWindow, "Add Student?",
69. "Are you sure you would add the student to this class?",QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)
70. **if** choice == QtGui.QMessageBox.Yes:
71. c.execute("INSERT INTO studentclass VALUES (:username,:classid)",{"username":self.username,"classid":self.id})
72. c.execute("INSERT INTO " + self.id + " (student) VALUES (:username)",{"username":self.username})
73. conn.commit()
74. self.allClasses.append([self.id])
75. self.retranslateUi(self.id,self.subject)
76. self.saved\_window()


80. **def** saved\_window(self):
81. QtGui.QMessageBox.question(self.window,"Saved","Save Successful",
82. QtGui.QMessageBox.Ok)

85. **def** remove\_student(self):
86. choice = QtGui.QMessageBox.question(MainWindow, "Remove Student?",
87. "Are you sure you would remove the student from this class?",QtGui.QMessageBox.Yes | QtGui.QMessageBox.No)
88. **if** choice == QtGui.QMessageBox.Yes:
89. c.execute("DELETE FROM studentclass WHERE username = :username AND classid = :classid",{"username":self.username,"classid":self.id})
90. c.execute("DELETE FROM "+self.id+ " WHERE student = :username",{"username":self.username})
91. conn.commit()
92. QtGui.QMessageBox.question(self.window,"Saved","Save Successful",
93. QtGui.QMessageBox.Ok)
94. **try**:
95. self.allClasses.remove(self.id)
96. **except**:
97. **pass**
98. self.retranslateUi(self.id,self.subject)
99. **if** self.type == "List":
100. self.hide\_all()


104. **class** Ui\_HomeWorkWindow(object):

11

1. **def** setupUi(self, MainWindow,homework):

4. self.window = MainWindow
5. self.homework = homework
7. #Creating Main Window
8. MainWindow.setObjectName(\_fromUtf8("MainWindow"))
9. MainWindow.resize(640, 480)
10. MainWindow.setStyleSheet(css)
11. self.centralwidget = QtGui.QWidget(MainWindow)
12. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
14. #Save Button
15. self.saveBtn = QtGui.QPushButton(self.centralwidget)
16. self.saveBtn.setGeometry(QtCore.QRect(550, 390, 75, 27))
17. self.saveBtn.setFont(labelfont)
18. self.saveBtn.setObjectName(\_fromUtf8("saveBtn"))

21. #Full Name Button
22. self.classLabel = QtGui.QLabel(self.centralwidget)
23. self.classLabel.setGeometry(QtCore.QRect(30, 80, 141, 51))
24. self.classLabel.setFont(labelfont)
25. self.classLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
26. self.classLabel.setObjectName(\_fromUtf8("classLabel"))
28. #Class Combo Box
29. self.classCombo = QtGui.QComboBox(self.centralwidget)
30. self.classCombo.setGeometry(QtCore.QRect(180, 90, 351, 31))
31. self.classCombo.setObjectName(\_fromUtf8("classCombo"))
32. **if** currentUser.type == "Teacher":
33. c.execute("SELECT id FROM classes WHERE teacher == :teacher",{"teacher":currentUser.username})
34. data = list(c.fetchall())
35. **for** classid **in** data:
36. self.classCombo.addItem(classid[0])
37. **else**:
38. c.execute("SELECT classid FROM studentclass WHERE username == :username",{"username":currentUser.username})
39. data = list(c.fetchall())
40. **for** classid **in** data:
41. self.classCombo.addItem(classid[0])
42. **if** self.homework.homeworkId != "NULL":
43. **for** i **in** range(len(data)):
44. **if** self.homework.classId == data[i][0]:
45. self.classCombo.setCurrentIndex(i)
47. #Edit Homework Label
48. self.editHomeworkLabel = QtGui.QLabel(self.centralwidget)
49. self.editHomeworkLabel.setGeometry(QtCore.QRect(65, 0, 511, 91))
50. self.editHomeworkLabel.setFont(titlefont)
51. self.editHomeworkLabel.setTextFormat(QtCore.Qt.PlainText)
52. self.editHomeworkLabel.setAlignment(QtCore.Qt.AlignCenter)
53. self.editHomeworkLabel.setObjectName(\_fromUtf8("editHomeworkLabel"))
55. #Short Name Label
56. self.dueLabel = QtGui.QLabel(self.centralwidget)
57. self.dueLabel.setGeometry(QtCore.QRect(30, 130, 141, 51))
58. self.dueLabel.setFont(labelfont)
59. self.dueLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
60. self.dueLabel.setObjectName(\_fromUtf8("dueLabel"))
62. #Date Edit
63. self.dateEdit = QtGui.QDateEdit(self.centralwidget)
64. self.dateEdit.setGeometry(QtCore.QRect(180, 140, 351, 31))
65. self.dateEdit.setObjectName(\_fromUtf8("dateEdit"))
66. self.dateEdit.setMinimumDate(QtCore.QDate.currentDate())
67. self.dateEdit.setCalendarPopup(True)
68. self.dateEdit.calendarWidget().installEventFilter(MainWindow)
70. date = QtCore.QDate(int(self.homework.dueyear),int(self.homework.duemonth),int(self.homework.dueday))
71. self.dateEdit.setDate(date)

74. #Description Edit
75. self.descriptionEdit = QtGui.QPlainTextEdit(self.centralwidget)
76. self.descriptionEdit.setGeometry(QtCore.QRect(180, 240, 351, 181))
77. self.descriptionEdit.setObjectName(\_fromUtf8("descriptionEdit"))
78. **if** self.homework.desc != "NULL":
79. self.descriptionEdit.setPlainText(\_translate("EditHomeWorkWindow",self.homework.desc,None))
81. #Description Label
82. self.descriptionLabel = QtGui.QLabel(self.centralwidget)
83. self.descriptionLabel.setGeometry(QtCore.QRect(30, 230, 141, 51))
84. self.descriptionLabel.setFont(labelfont)
85. self.descriptionLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
86. self.descriptionLabel.setObjectName(\_fromUtf8("descriptionLabel"))
88. #Title Label
89. self.titleLabel = QtGui.QLabel(self.centralwidget)
90. self.titleLabel.setGeometry(QtCore.QRect(30, 180, 141, 51))
91. self.titleLabel.setFont(labelfont)
92. self.titleLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
93. self.titleLabel.setObjectName(\_fromUtf8("titleLabel"))
95. #Title Edit
96. self.titleEdit = QtGui.QLineEdit(self.centralwidget)
97. self.titleEdit.setGeometry(QtCore.QRect(180, 190, 351, 31))
98. self.titleEdit.setObjectName(\_fromUtf8("titleEdit"))
99. **if** self.homework.title != "NULL":
100. self.titleEdit.setText(\_translate("EditHomeWorkWindow",self.homework.title,None))
102. self.gradeTitle = QtGui.QLabel(self.centralwidget)
104. **if** currentUser.type == "Student":
105. self.gradeTitle = QtGui.QLabel(self.centralwidget)
106. self.gradeTitle.setGeometry(QtCore.QRect(60, 280, 101, 41))
107. self.gradeTitle.setFont(typefont)
108. self.gradeTitle.setAutoFillBackground(False)
109. self.gradeTitle.setStyleSheet(\_fromUtf8("QLabel{border: 1px solid black;\n"
110. "background-color: rgb(255, 255, 255);}\n"
111. ""))
112. self.gradeTitle.setAlignment(QtCore.Qt.AlignCenter)
113. self.gradeTitle.setObjectName(\_fromUtf8("gradeTitle"))
115. self.grade = QtGui.QLabel(self.centralwidget)
116. self.grade.setGeometry(QtCore.QRect(60, 320, 101, 61))
117. self.grade.setFont(numberfont)
118. self.grade.setAutoFillBackground(False)
119. self.grade.setAlignment(QtCore.Qt.AlignCenter)
120. self.grade.setObjectName(\_fromUtf8("grade"))
122. **if** self.homework.homeworkId == "NULL":
123. self.saveBtn.clicked.connect(self.create)
124. **else**:
125. self.saveBtn.clicked.connect(self.edit)
126. self.classCombo.setEnabled(False)
127. #Grades Button
128. self.gradesBtn = QtGui.QPushButton(self.centralwidget)
129. self.gradesBtn.setGeometry(QtCore.QRect(550, 330, 75, 51))
130. self.gradesBtn.setFont(labelfont)
131. self.gradesBtn.setObjectName(\_fromUtf8("gradesBtn"))
132. self.gradesBtn.clicked.connect(self.grades\_window)
134. MainWindow.setCentralWidget(self.centralwidget)
135. self.menubar = QtGui.QMenuBar(MainWindow)
136. self.menubar.setGeometry(QtCore.QRect(0, 0, 640, 21))
137. self.menubar.setObjectName(\_fromUtf8("menubar"))
138. MainWindow.setMenuBar(self.menubar)
139. self.statusbar = QtGui.QStatusBar(MainWindow)
140. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
141. MainWindow.setStatusBar(self.statusbar)
143. self.retranslateUi(MainWindow)
144. QtCore.QMetaObject.connectSlotsByName(MainWindow)
146. **def** retranslateUi(self, MainWindow):
147. **if** self.homework.homeworkId == "NULL":
148. self.saveBtn.setText(\_translate("MainWindow", "Create", None))
149. MainWindow.setWindowTitle(\_translate("MainWindow", "Create Homework Window", None))
150. self.editHomeworkLabel.setText(\_translate("MainWindow", "CREATE HOMEWORK", None))
151. **else**:
152. self.saveBtn.setText(\_translate("MainWindow", "Save", None))
153. MainWindow.setWindowTitle(\_translate("MainWindow", "Edit Homework Window", None))
154. self.editHomeworkLabel.setText(\_translate("MainWindow", "EDIT HOMEWORK", None))
155. self.gradesBtn.setText(\_translate("MainWindow", "Update\nGrades", None))
156. self.classLabel.setText(\_translate("MainWindow", "CLASS:", None))
157. **if** currentUser.type == "Student":
158. self.editHomeworkLabel.setText(\_translate("MainWindow", "VIEW HOMEWORK", None))
160. self.dueLabel.setText(\_translate("MainWindow", "DUE DATE:", None))
161. self.descriptionLabel.setText(\_translate("MainWindow", "DESCRIPTION:", None))
162. self.titleLabel.setText(\_translate("MainWindow", "TITLE:", None))
163. MainWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
165. **if** currentUser.type == "Student":
166. self.classCombo.setEnabled(False)
167. self.dateEdit.setEnabled(False)
168. self.titleEdit.setEnabled(False)
169. self.descriptionEdit.setEnabled(False)
170. self.saveBtn.hide()
171. self.gradesBtn.hide()
172. self.gradeTitle.setText(\_translate("MainWindow", "Homework\n Status", None))
173. c.execute("SELECT "+self.homework.homeworkId+" FROM "+self.homework.classId+" WHERE Student = :username",
174. {"username":currentUser.username})
175. data = c.fetchall()
176. grade = data[0][0]
178. #Colourcoding grade based on how good it is.
179. **if** grade != None **and** grade != "Completed":
180. self.grade.setText(\_translate("MainWindow", grade, None))
181. **if** grade == "A\*" **or** grade == "A":
182. self.grade.setStyleSheet(\_fromUtf8("QLabel{border: 1px solid black;\n"
183. "background-color: rgb(0, 255, 0);}\n"
184. ""))
185. **if** grade == "B" **or** grade == "C":
186. self.grade.setStyleSheet(\_fromUtf8("QLabel{border: 1px solid black;\n"
187. "background-color: rgb(255, 165, 0);}\n"
188. ""))
189. **else**:
190. self.grade.setStyleSheet(\_fromUtf8("QLabel{border: 1px solid black;\n"
191. "background-color: rgb(255, 0, 0);}\n"
192. ""))
193. **else**:
194. self.grade.setText(\_translate("MainWindow", "N/A", None))
195. self.grade.setStyleSheet(\_fromUtf8("QLabel{border: 1px solid black;\n"
196. "background-color: rgb(255, 255, 255);}\n"
197. ""))

200. **def** create(self):
201. classid = self.classCombo.currentText()
202. duedate = str(self.dateEdit.date().toPyDate())
203. duedate = duedate[:4]+ "-"+duedate[5:7]+"-"+duedate[8:]
204. title = self.titleEdit.text()
205. desc = self.descriptionEdit.toPlainText()
206. homeworkId = classid + "\_"
207. length = len(homeworkId)
209. c.execute("SELECT homeworkid FROM homework WHERE homeworkid LIKE :homeworkid ORDER BY homeworkid ASC",
210. {"homeworkid":homeworkId+ "%"})
211. homeworkId += "1"
212. data = list(c.fetchall())
213. newdata = []
214. **for** i **in** range(len(data)):
215. newdata.append(data[i][0])
216. #Generating homeworkid
217. **if** homeworkId **in** newdata:
218. chosen = False
219. **while** **not** chosen:
220. **if** homeworkId **in** newdata:
221. homeworkId = homeworkId[:length]+str(int(homeworkId[length:])+1)
222. **else**:
223. chosen = True
225. c.execute("INSERT INTO homework VALUES (:homeworkid,:classid,:duedate,:title,:description)",
226. {"homeworkid":homeworkId,"classid":classid,"duedate":duedate,"title":title,"description":desc})
228. #Adding a column to the classId for the homework. All students grades will be stored here.
229. c.execute("ALTER TABLE "+classid+" ADD "+ homeworkId+ " text;")
231. conn.commit()
232. self.window.hide()
233. self.newWindow = EditWindow()
234. self.newPage = Ui\_HomeWorkWindow()
235. self.newPage.setupUi(self.newWindow,Homework(homeworkId,classid,duedate,title,desc))
236. self.newWindow.show()


240. **def** edit(self):
241. #Editting and saving to the database
242. duedate = str(self.dateEdit.date().toPyDate())
243. duedate = duedate[:4]+ "-"+duedate[5:7]+"-"+duedate[8:]
244. title = self.titleEdit.text()
245. desc = self.descriptionEdit.toPlainText()
246. c.execute("UPDATE homework SET duedate = :duedate, title = :title,description = :description WHERE homeworkid = :homeworkid",
247. {"duedate":duedate,"title":title,"desc":desc,"homeworkid":self.homework.homeworkId})
248. conn.commit()
250. **def** grades\_window(self):
251. #Creating the grades window
252. self.searchPage = EditWindow()
253. self.SearchUi = Ui\_SearchUsers()
254. self.SearchUi.setupUi(self.searchPage,"Homework",self.homework)
255. self.searchPage.show()

258. **class** Ui\_ViewHomeworkWindow(object):
259. **def** setupUi(self, ViewHomeworkWindow,typeOfWindow):

12

1. self.window = ViewHomeworkWindow
2. self.page = 1
3. self.typeOfWindow = typeOfWindow

6. #Setting Window Up
7. ViewHomeworkWindow.setObjectName(\_fromUtf8("ViewHomeworkWindow"))
8. ViewHomeworkWindow.resize(640, 551)
9. self.centralwidget = QtGui.QWidget(ViewHomeworkWindow)
10. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
11. self.window.setStyleSheet(\_fromUtf8("QMainWindow {\n"
12. "background-color: qlineargradient(spread:pad, x1:0.494364, y1:0.806, x2:0.471, y2:0.142045, stop:0 rgba(17, 255, 56, 255), stop:1 rgba(255, 255, 255, 255));}\n"
13. "QPushButton{background: transparent;"
14. "border: 1px solid black;}"))
16. #Type Button
17. self.typeBtn = QtGui.QPushButton(self.centralwidget)
18. self.typeBtn.setGeometry(QtCore.QRect(0, 0, 75, 51))
19. self.typeBtn.setFont(labelfont)
20. self.typeBtn.setObjectName(\_fromUtf8("typeBtn"))
21. self.typeBtn.clicked.connect(self.change\_type)

24. #Homework Label
25. self.myHomeworkLabel = QtGui.QLabel(self.centralwidget)
26. self.myHomeworkLabel.setGeometry(QtCore.QRect(0, 0, 641, 91))
27. self.myHomeworkLabel.setFont(titlefont)
28. self.myHomeworkLabel.setTextFormat(QtCore.Qt.PlainText)
29. self.myHomeworkLabel.setAlignment(QtCore.Qt.AlignCenter)
30. self.myHomeworkLabel.setObjectName(\_fromUtf8("myHomeworkLabel"))
32. #Class Label
33. self.classLabel = QtGui.QLabel(self.centralwidget)
34. self.classLabel.setGeometry(QtCore.QRect(-10, 70, 141, 51))
35. self.classLabel.setFont(labelfont)
36. self.classLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
37. self.classLabel.setObjectName(\_fromUtf8("classLabel"))


41. #Class Combo
42. self.classCombo = QtGui.QComboBox(self.centralwidget)
43. self.classCombo.setGeometry(QtCore.QRect(140, 80, 421, 31))
44. self.classCombo.setObjectName(\_fromUtf8("classCombo"))
46. #Getting all the users classes then adding it to the combo box
47. **if** currentUser.type == "Student":
48. c.execute("SELECT classid FROM studentclass WHERE username = :username",
49. {"username":currentUser.username})
50. **else**:
51. c.execute("SELECT id FROM classes WHERE teacher = :username",
52. {"username":currentUser.username})
54. self.data = list(c.fetchall())
55. self.classCombo.addItem("All")
56. **for** classid **in** self.data:
57. self.classCombo.addItem(classid[0])
59. self.classid = self.classCombo.currentText()
60. self.classCombo.activated[str].connect(self.show\_homework)
62. ##################RESULTS##############################
64. ##1st Result
66. #Button
67. self.topButton = QtGui.QPushButton(self.centralwidget)
68. self.topButton.setGeometry(QtCore.QRect(80, 130, 481, 51))
69. self.topButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;\n"
70. "border: 1px solid black;}"))
71. self.topButton.setText(\_fromUtf8(""))
72. self.topButton.setObjectName(\_fromUtf8("topButton"))
74. #Description
75. self.topDesc = QtGui.QLabel(self.centralwidget)
76. self.topDesc.setGeometry(QtCore.QRect(90, 157, 461, 21))
77. self.topDesc.setFont(labelfont4)
78. self.topDesc.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
79. self.topDesc.setObjectName(\_fromUtf8("topDesc"))
81. #Title
82. self.topTitle = QtGui.QLabel(self.centralwidget)
83. self.topTitle.setGeometry(QtCore.QRect(90, 137, 461, 21))
84. self.topTitle.setFont(labelfont3)
85. self.topTitle.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
86. self.topTitle.setObjectName(\_fromUtf8("topTitle"))
88. self.top1 =  WindowButtons(self.topButton,self.topTitle,self.topDesc,"Homework")
90. ##2nd Result
91. self.topTitle\_2 = QtGui.QLabel(self.centralwidget)
92. self.topTitle\_2.setGeometry(QtCore.QRect(90, 187, 461, 21))
93. self.topTitle\_2.setFont(normalfont)
94. self.topTitle\_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
95. self.topTitle\_2.setObjectName(\_fromUtf8("topTitle\_2"))
97. self.topDesc\_2 = QtGui.QLabel(self.centralwidget)
98. self.topDesc\_2.setGeometry(QtCore.QRect(90, 207, 461, 21))
99. self.topDesc\_2.setFont(labelfont4)
100. self.topDesc\_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
101. self.topDesc\_2.setObjectName(\_fromUtf8("topDesc\_2"))
103. self.topButton\_2 = QtGui.QPushButton(self.centralwidget)
104. self.topButton\_2.setGeometry(QtCore.QRect(80, 180, 481, 51))
105. self.topButton\_2.setObjectName(\_fromUtf8("topButton\_2"))
107. self.top2 = WindowButtons(self.topButton\_2,self.topTitle\_2,self.topDesc\_2,"Homework")
109. ##3rd Result
110. self.topButton\_3 = QtGui.QPushButton(self.centralwidget)
111. self.topButton\_3.setGeometry(QtCore.QRect(80, 230, 481, 51))
112. self.topButton\_3.setObjectName(\_fromUtf8("topButton\_3"))
114. self.topDesc\_3 = QtGui.QLabel(self.centralwidget)
115. self.topDesc\_3.setGeometry(QtCore.QRect(90, 257, 461, 21))
116. self.topDesc\_3.setFont(labelfont4)
117. self.topDesc\_3.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
118. self.topDesc\_3.setObjectName(\_fromUtf8("topDesc\_3"))
120. self.topTitle\_3 = QtGui.QLabel(self.centralwidget)
121. self.topTitle\_3.setGeometry(QtCore.QRect(90, 237, 461, 21))
122. self.topTitle\_3.setFont(labelfont3)
123. self.topTitle\_3.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
124. self.topTitle\_3.setObjectName(\_fromUtf8("topTitle\_3"))
126. self.top3 = WindowButtons(self.topButton\_3,self.topTitle\_3,self.topDesc\_3,"Homework")
128. ##4th Result
129. self.topDesc\_4 = QtGui.QLabel(self.centralwidget)
130. self.topDesc\_4.setGeometry(QtCore.QRect(90, 307, 461, 21))
131. self.topDesc\_4.setFont(labelfont4)
132. self.topDesc\_4.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
133. self.topDesc\_4.setObjectName(\_fromUtf8("topDesc\_4"))
135. self.topTitle\_4 = QtGui.QLabel(self.centralwidget)
136. self.topTitle\_4.setGeometry(QtCore.QRect(90, 287, 461, 21))
137. self.topTitle\_4.setFont(labelfont3)
138. self.topTitle\_4.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
139. self.topTitle\_4.setObjectName(\_fromUtf8("topTitle\_4"))
141. self.topButton\_4 = QtGui.QPushButton(self.centralwidget)
142. self.topButton\_4.setGeometry(QtCore.QRect(80, 280, 481, 51))
143. self.topButton\_4.setObjectName(\_fromUtf8("topButton\_4"))
145. self.top4 = WindowButtons(self.topButton\_4,self.topTitle\_4,self.topDesc\_4,"Homework")
147. ##5th Result
148. self.topButton\_5 = QtGui.QPushButton(self.centralwidget)
149. self.topButton\_5.setGeometry(QtCore.QRect(80, 330, 481, 51))
150. self.topButton\_5.setObjectName(\_fromUtf8("topButton\_5"))
152. self.topTitle\_5 = QtGui.QLabel(self.centralwidget)
153. self.topTitle\_5.setGeometry(QtCore.QRect(90, 337, 461, 21))
154. self.topTitle\_5.setFont(labelfont3)
155. self.topTitle\_5.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
156. self.topTitle\_5.setObjectName(\_fromUtf8("topTitle\_5"))
158. self.topDesc\_5 = QtGui.QLabel(self.centralwidget)
159. self.topDesc\_5.setGeometry(QtCore.QRect(90, 357, 461, 21))
160. self.topDesc\_5.setFont(labelfont4)
161. self.topDesc\_5.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
162. self.topDesc\_5.setObjectName(\_fromUtf8("topDesc\_5"))
164. self.top5 = WindowButtons(self.topButton\_5,self.topTitle\_5,self.topDesc\_5,"Homework")
166. ##6th Result
167. self.topDesc\_6 = QtGui.QLabel(self.centralwidget)
168. self.topDesc\_6.setGeometry(QtCore.QRect(90, 407, 461, 21))
169. self.topDesc\_6.setFont(labelfont4)
170. self.topDesc\_6.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
171. self.topDesc\_6.setObjectName(\_fromUtf8("topDesc\_6"))
173. self.topButton\_6 = QtGui.QPushButton(self.centralwidget)
174. self.topButton\_6.setGeometry(QtCore.QRect(80, 380, 481, 51))
175. self.topButton\_6.setObjectName(\_fromUtf8("topButton\_6"))
177. self.topTitle\_6 = QtGui.QLabel(self.centralwidget)
178. self.topTitle\_6.setGeometry(QtCore.QRect(90, 387, 461, 21))
179. self.topTitle\_6.setFont(labelfont3)
180. self.topTitle\_6.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
181. self.topTitle\_6.setObjectName(\_fromUtf8("topTitle\_6"))
183. self.top6 = WindowButtons(self.topButton\_6,self.topTitle\_6,self.topDesc\_6,"Homework")
185. ##7th Result
186. self.topTitle\_7 = QtGui.QLabel(self.centralwidget)
187. self.topTitle\_7.setGeometry(QtCore.QRect(90, 437, 461, 21))
188. self.topTitle\_7.setFont(labelfont3)
189. self.topTitle\_7.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
190. self.topTitle\_7.setObjectName(\_fromUtf8("topTitle\_7"))
192. self.topDesc\_7 = QtGui.QLabel(self.centralwidget)
193. self.topDesc\_7.setGeometry(QtCore.QRect(90, 457, 461, 21))
194. self.topDesc\_7.setFont(labelfont4)
195. self.topDesc\_7.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
196. self.topDesc\_7.setObjectName(\_fromUtf8("topDesc\_7"))
198. self.topButton\_7 = QtGui.QPushButton(self.centralwidget)
199. self.topButton\_7.setGeometry(QtCore.QRect(80, 430, 481, 51))
200. self.topButton\_7.setObjectName(\_fromUtf8("topButton\_7"))
202. self.top7 = WindowButtons(self.topButton\_7,self.topTitle\_7,self.topDesc\_7,"Homework")
204. #######################################################################
206. #New Homework Button
207. **if** currentUser.type == "Teacher":
208. self.newHomeworkButton = QtGui.QPushButton(self.centralwidget)
209. self.newHomeworkButton.setGeometry(QtCore.QRect(470, 510, 151, 21))
210. self.newHomeworkButton.setFont(labelfont3)
211. self.newHomeworkButton.setObjectName(\_fromUtf8("newHomeworkButton"))
212. self.newHomeworkButton.clicked.connect(self.new\_homework)
213. self.newHomeworkButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;border: 0px transparent;}"))
214. self.newHomeworkButton.clicked.connect(self.new\_homework)

217. #Previous Button
218. self.previousButton = QtGui.QPushButton(self.centralwidget)
219. self.previousButton.setGeometry(QtCore.QRect(220, 510, 81, 21))
220. self.previousButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;border: 0px transparent;}"))
221. self.previousButton.setObjectName(\_fromUtf8("previousButton"))
222. self.previousButton.clicked.connect(self.previous\_page)
224. #Next Button
225. self.nextButton = QtGui.QPushButton(self.centralwidget)
226. self.nextButton.setGeometry(QtCore.QRect(350, 510, 81, 21))
227. self.nextButton.setStyleSheet(\_fromUtf8("QPushButton{background: transparent;border: 0px transparent;}"))
228. self.nextButton.setObjectName(\_fromUtf8("nextButton"))
229. self.nextButton.clicked.connect(self.next\_page)
231. #Page Number
232. self.pageNumber = QtGui.QLabel(self.centralwidget)
233. self.pageNumber.setGeometry(QtCore.QRect(300, 510, 51, 21))
234. self.pageNumber.setFont(labelfont3)
235. self.pageNumber.setAlignment(QtCore.Qt.AlignHCenter|QtCore.Qt.AlignTop)
236. self.pageNumber.setObjectName(\_fromUtf8("pageNumber"))
238. #Amount Of Results
239. self.amountLabel = QtGui.QLabel(self.centralwidget)
240. self.amountLabel.setGeometry(QtCore.QRect(20, 510, 151, 21))
241. self.amountLabel.setFont(labelfont3)
242. self.amountLabel.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
243. self.amountLabel.setObjectName(\_fromUtf8("amountLabel"))
245. self.topTitle\_2.raise\_()
246. self.topDesc.raise\_()
247. self.topTitle.raise\_()
248. self.topDesc\_3.raise\_()
249. self.topDesc\_2.raise\_()
250. self.topTitle\_3.raise\_()
251. self.myHomeworkLabel.raise\_()
252. self.classLabel.raise\_()
253. self.classCombo.raise\_()
254. self.topDesc\_6.raise\_()
255. self.topTitle\_5.raise\_()
256. self.topTitle\_6.raise\_()
257. self.topDesc\_4.raise\_()
258. self.topDesc\_5.raise\_()
259. self.topTitle\_4.raise\_()
260. self.topTitle\_7.raise\_()
261. self.topDesc\_7.raise\_()
262. self.topButton\_3.raise\_()
263. self.topButton\_2.raise\_()
264. self.topButton\_5.raise\_()
265. self.topButton\_7.raise\_()
266. self.topButton\_4.raise\_()
267. self.topButton\_6.raise\_()
268. self.topButton.raise\_()
269. self.typeBtn.raise\_()
271. ViewHomeworkWindow.setCentralWidget(self.centralwidget)
272. self.menubar = QtGui.QMenuBar(ViewHomeworkWindow)
273. self.menubar.setGeometry(QtCore.QRect(0, 0, 640, 21))
274. self.menubar.setObjectName(\_fromUtf8("menubar"))
275. ViewHomeworkWindow.setMenuBar(self.menubar)
276. self.statusbar = QtGui.QStatusBar(ViewHomeworkWindow)
277. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
278. ViewHomeworkWindow.setStatusBar(self.statusbar)
279. self.search()
280. self.retranslateUi(ViewHomeworkWindow)
281. QtCore.QMetaObject.connectSlotsByName(ViewHomeworkWindow)

284. **def** retranslateUi(self, ViewHomeworkWindow):
285. #Adding homeworks where appropriate
286. **if** len(self.homeworks) >= 1+((self.page-1)\*7):
287. **if** len(self.homeworks[0+(self.page-1)\*7][3]) > 45:
288. self.homeworks[0+(self.page-1)\*7][3]= self.homeworks[0+(self.page-1)\*7][3][:42]+"..."
289. self.top1.retranslateUi(self.homeworks[0+(self.page-1)\*7][3],
290. self.homeworks[0+(self.page-1)\*7][1] + " - Due "+self.homeworks[0+(self.page-1)\*7][2],
291. self.homeworks[0+(self.page-1)\*7][0])
292. self.top1.show\_all()
293. **else**:
294. self.top1.hide\_all()

297. **if** len(self.homeworks) >= 2+((self.page-1)\*7):
298. **if** len(self.homeworks[1+(self.page-1)\*7][3]) > 45:
299. self.homeworks[1+(self.page-1)\*7][3]= self.homeworks[1+(self.page-1)\*7][3][:42]+"..."
300. self.top2.retranslateUi(self.homeworks[1+(self.page-1)\*7][3],
301. self.homeworks[1+(self.page-1)\*7][1] + " - Due "+self.homeworks[1+(self.page-1)\*7][2],
302. self.homeworks[1+(self.page-1)\*7][0])
303. self.top2.show\_all()
304. **else**:
305. self.top2.hide\_all()
307. **if** len(self.homeworks) >= 3+((self.page-1)\*7):
308. **if** len(self.homeworks[2+(self.page-1)\*7][3]) > 45:
309. self.homeworks[2+(self.page-1)\*7][3]= self.homeworks[2+(self.page-1)\*7][3][:42]+"..."
310. self.top3.retranslateUi(self.homeworks[2+(self.page-1)\*7][3],
311. self.homeworks[2+(self.page-1)\*7][1] + " - Due "+self.homeworks[2+(self.page-1)\*7][2],
312. self.homeworks[2+(self.page-1)\*7][0])
313. self.top3.show\_all()
314. **else**:
315. self.top3.hide\_all()

318. **if** len(self.homeworks) >= 4+((self.page-1)\*7):
319. **if** len(self.homeworks[3+(self.page-1)\*7][3]) > 45:
320. self.homeworks[3+(self.page-1)\*7][3]= self.homeworks[3+(self.page-1)\*7][3][:42]+"..."
321. self.top4.retranslateUi(self.homeworks[3+(self.page-1)\*7][3],
322. self.homeworks[3+(self.page-1)\*7][1] + " - Due "+self.homeworks[3+(self.page-1)\*7][2],
323. self.homeworks[3+(self.page-1)\*7][0])
324. self.top4.show\_all()
325. **else**:
326. self.top4.hide\_all()

329. **if** len(self.homeworks) >= 5+((self.page-1)\*7):
330. **if** len(self.homeworks[4+(self.page-1)\*7][3]) > 45:
331. self.homeworks[4+(self.page-1)\*7][3]= self.homeworks[4+(self.page-1)\*7][3][:42]+"..."
332. self.top5.retranslateUi(self.homeworks[4+(self.page-1)\*7][3],
333. self.homeworks[4+(self.page-1)\*7][1] + " - Due "+self.homeworks[4+(self.page-1)\*7][2],
334. self.homeworks[4+(self.page-1)\*7][0])
335. self.top5.show\_all()
336. **else**:
337. self.top5.hide\_all()

340. **if** len(self.homeworks) >= 6+((self.page-1)\*7):
341. **if** len(self.homeworks[5+(self.page-1)\*7][3]) > 45:
342. self.homeworks[5+(self.page-1)\*7][3]= self.homeworks[5+(self.page-1)\*7][3][:42]+"..."
343. self.top6.retranslateUi(self.homeworks[5+(self.page-1)\*7][3],
344. self.homeworks[5+(self.page-1)\*7][1] + " - Due "+self.homeworks[5+(self.page-1)\*7][2],
345. self.homeworks[5+(self.page-1)\*7][0])
346. self.top6.show\_all()
347. **else**:
348. self.top6.hide\_all()

351. **if** len(self.homeworks) >= 7+((self.page-1)\*7):
352. **if** len(self.homeworks[6+(self.page-1)\*7][3]) > 45:
353. self.homeworks[6+(self.page-1)\*7][3]= self.homeworks[6+(self.page-1)\*7][3][:42]+"..."
354. self.top7.retranslateUi(self.homeworks[6+(self.page-1)\*7][3],
355. self.homeworks[6+(self.page-1)\*7][1] + " - Due "+self.homeworks[6+(self.page-1)\*7][2],
356. self.homeworks[6+(self.page-1)\*7][0])
357. self.top7.show\_all()
358. **else**:
359. self.top7.hide\_all()
361. **if** self.page == 1:
362. self.previousButton.setEnabled(False)
363. **else**:
364. self.previousButton.setEnabled(True)
366. **if** self.page < (len(self.homeworks)//7+1):
367. self.nextButton.setEnabled(True)
368. **else**:
369. self.nextButton.setEnabled(False)
371. ViewHomeworkWindow.setWindowTitle(\_translate("ViewHomeworkWindow", "View Homework", None))
372. **if** self.typeOfWindow == "Future":
373. self.typeBtn.setText(\_translate("ViewHomeworkWindow", "View\nOverdue", None))
374. **else**:
375. self.typeBtn.setText(\_translate("ViewHomeworkWindow", "View\nUpcoming", None))
376. ViewHomeworkWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
377. self.myHomeworkLabel.setText(\_translate("ViewHomeworkWindow", "MY HOMEWORK", None))
378. self.classLabel.setText(\_translate("ViewHomeworkWindow", "CLASS:", None))
379. **if** currentUser.type == "Teacher":
380. self.newHomeworkButton.setText(\_translate("ViewHomeworkWindow", "Add New Homework", None))
381. self.amountLabel.setText(\_translate("ViewHomeworkWindow",str(len(self.homeworks))+" Results ("+str((len(self.homeworks))//7+1) +" Pages)", None))
382. self.previousButton.setText(\_translate("ViewHomeworkWindow", "Previous", None))
383. self.nextButton.setText(\_translate("ViewHomeworkWindow", "Next", None))
384. self.pageNumber.setText(\_translate("ViewHomeworkWindow", str(self.page), None))
386. #This button will filter between past and future homework
387. **def** change\_type(self):
388. **if** self.typeOfWindow == "Future":
389. self.typeOfWindow = "Past"
390. **else**:
391. self.typeOfWindow = "Future"
392. self.search()
393. self.retranslateUi(self.window)

396. **def** previous\_page(self):
397. self.page -= 1
398. self.retranslateUi(self.window)
400. **def** next\_page(self):
401. self.page += 1
402. self.retranslateUi(self.window)
404. **def** new\_homework(self):
405. #Open create homework window
406. self.homeworkPage = EditWindow()
407. self.homeworkui = Ui\_HomeWorkWindow()
408. self.homeworkui.setupUi(self.homeworkPage,Homework("NULL","NULL","2001/01/01","NULL","NULL"))
409. self.homeworkPage.show()
411. **def** show\_homework(self,text):
412. self.classid = text
413. self.search()
414. self.retranslateUi(self.window)
416. **def** search(self):
417. #Getting todays date
418. today = datetime.datetime.today().strftime('%Y-%m-%d')
419. **if** self.classid != "All":
420. **if** self.typeOfWindow == "Future":
421. c.execute("SELECT \* FROM homework WHERE classid = :classid AND duedate >= :date",{"classid":self.classid,"date":today})
422. **else**:
423. c.execute("SELECT \* FROM homework WHERE classid = :classid AND duedate < :date",{"classid":self.classid,"date":today})
424. data = c.fetchall()
425. **if** data == None:
426. data = []
427. **else**:
428. data = list(data)
430. **else**:
431. data = []
432. **for** classid **in** self.data:
433. **if** self.typeOfWindow == "Future":
434. c.execute("SELECT \* FROM homework WHERE classid = :classid AND duedate >= :date",{"classid":classid[0],"date":today})
435. **else**:
436. c.execute("SELECT \* FROM homework WHERE classid = :classid AND NOT duedate >= :date",{"classid":classid[0],"date":today})
438. temp = c.fetchall()
439. **if** temp != None:
440. #Extending because there will be a fetch from each class.
441. data.extend(list(temp))
443. #Sorting by due date
444. data.sort(key=**lambda** x: time.mktime(time.strptime(x[2],"%Y-%m-%d")))
445. **if** self.typeOfWindow != "Future":
446. #If user is searching for past homeworks they will want the most recent ones first.
447. data.reverse()
448. **if** data == None:
449. data = []
450. self.homeworks = data

10

2. **class** Ui\_GradeWindow(object):
3. **def** setupUi(self, GradeWindow,homework,username,first,last):
4. self.window = GradeWindow
5. self.username = username
6. self.homework = homework
7. self.first = first
8. self.last = last
9. #Getting grade of users homework from the database
10. c.execute("SELECT "+self.homework.homeworkId+" FROM "+self.homework.classId+" WHERE Student = '"+self.username+"'")
11. data = c.fetchall()
12. grade = data[0][0]
13. **if** grade == None:
14. grade = "Not Completed"

17. #Setting window up
18. GradeWindow.setObjectName(\_fromUtf8("GradeWindow"))
19. GradeWindow.resize(304, 228)
20. self.centralwidget = QtGui.QWidget(GradeWindow)
21. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
22. GradeWindow.setStyleSheet(css)
24. #Label of users name
25. self.nameLabel = QtGui.QLabel(self.centralwidget)
26. self.nameLabel.setGeometry(QtCore.QRect(0, 10, 301, 51))
27. self.nameLabel.setFont(smalltitlefont)
28. self.nameLabel.setAlignment(QtCore.Qt.AlignCenter)
29. self.nameLabel.setObjectName(\_fromUtf8("nameLabel"))
31. #Grade combo
32. self.gradeCombo = QtGui.QComboBox(self.centralwidget)
33. self.gradeCombo.setGeometry(QtCore.QRect(130, 80, 101, 31))
34. self.gradeCombo.setObjectName(\_fromUtf8("gradeCombo"))
35. #Adding all the options to to Box
36. self.gradeCombo.addItem("Not Completed")
37. self.gradeCombo.addItem("Completed")
38. self.gradeCombo.addItem("A\*")
39. self.gradeCombo.addItem("A")
40. self.gradeCombo.addItem("B")
41. self.gradeCombo.addItem("C")
42. self.gradeCombo.addItem("D")
43. self.gradeCombo.addItem("E")
44. self.gradeCombo.addItem("F")
45. self.gradeCombo.addItem("U")
46. #Finding if the users current grade = one of the options - if so setting that as the index
47. index = self.gradeCombo.findText(grade, QtCore.Qt.MatchFixedString)
48. **if** index >= 0:
49. self.gradeCombo.setCurrentIndex(index)


53. #Grade Label
54. self.gradeLabel = QtGui.QLabel(self.centralwidget)
55. self.gradeLabel.setGeometry(QtCore.QRect(-20, 70, 141, 51))
56. self.gradeLabel.setFont(labelfont)
57. self.gradeLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
58. self.gradeLabel.setObjectName(\_fromUtf8("gradeLabel"))
60. #Save Button
61. self.saveBtn = QtGui.QPushButton(self.centralwidget)
62. self.saveBtn.setGeometry(QtCore.QRect(210, 140, 75, 27))
63. self.saveBtn.setFont(labelfont)
64. self.saveBtn.setObjectName(\_fromUtf8("saveBtn"))
65. self.saveBtn.clicked.connect(self.save)

68. GradeWindow.setCentralWidget(self.centralwidget)
69. self.menubar = QtGui.QMenuBar(GradeWindow)
70. self.menubar.setGeometry(QtCore.QRect(0, 0, 304, 21))
71. self.menubar.setObjectName(\_fromUtf8("menubar"))
72. GradeWindow.setMenuBar(self.menubar)
73. self.statusbar = QtGui.QStatusBar(GradeWindow)
74. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
75. GradeWindow.setStatusBar(self.statusbar)
77. self.retranslateUi(GradeWindow)
78. QtCore.QMetaObject.connectSlotsByName(GradeWindow)
80. **def** retranslateUi(self, GradeWindow):
81. GradeWindow.setWindowTitle(\_translate("GradeWindow", "Grade Window", None))
82. self.nameLabel.setText(\_translate("GradeWindow", self.first + " " + self.last, None))
83. self.gradeLabel.setText(\_translate("GradeWindow", "GRADE:", None))
84. self.saveBtn.setText(\_translate("GradeWindow", "Update", None))
85. GradeWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
87. **def** save(self):
88. grade = self.gradeCombo.currentText()
89. **if** grade != "Not Completed":
90. c.execute("UPDATE "+self.homework.classId+" SET "+self.homework.homeworkId+" = :grade WHERE student = :student",
91. {"grade":grade,"student":self.username})
92. conn.commit()
93. QtGui.QMessageBox.question(self.window,"Saved","Save Successful",
94. QtGui.QMessageBox.Ok)

97. **class** Ui\_PointsWindow(object):
98. **def** setupUi(self, MainWindow,typeOfWindow,student):
99. self.window = MainWindow

6

1. self.typeOfWindow = typeOfWindow
2. self.student = student
3. MainWindow.setObjectName(\_fromUtf8("MainWindow"))
4. MainWindow.resize(449, 310)
5. self.centralwidget = QtGui.QWidget(MainWindow)
6. self.centralwidget.setObjectName(\_fromUtf8("centralwidget"))
7. MainWindow.setStyleSheet(css)
9. #Save Button
10. self.saveBtn = QtGui.QPushButton(self.centralwidget)
11. self.saveBtn.setGeometry(QtCore.QRect(366, 250, 75, 27))
12. self.saveBtn.setFont(labelfont)
13. self.saveBtn.setObjectName(\_fromUtf8("saveBtn"))
14. self.saveBtn.clicked.connect(self.save)

17. #Reason Text Edit/Input
18. self.reasonEdit = QtGui.QPlainTextEdit(self.centralwidget)
19. self.reasonEdit.setGeometry(QtCore.QRect(90, 60, 351, 181))
20. self.reasonEdit.setObjectName(\_fromUtf8("reasonEdit"))
22. #Reason Label
23. self.reasonLabel = QtGui.QLabel(self.centralwidget)
24. self.reasonLabel.setGeometry(QtCore.QRect(-60, 50, 141, 51))
25. self.reasonLabel.setFont(labelfont)
26. self.reasonLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
27. self.reasonLabel.setObjectName(\_fromUtf8("reasonLabel"))
29. #Points Label
30. self.pointsLabel = QtGui.QLabel(self.centralwidget)
31. self.pointsLabel.setGeometry(QtCore.QRect(-60, 0, 141, 51))
32. self.pointsLabel.setFont(labelfont)
33. self.pointsLabel.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
34. self.pointsLabel.setObjectName(\_fromUtf8("pointsLabel"))
36. #Points Edit
37. self.pointEdit = QtGui.QSpinBox(self.centralwidget)
38. self.pointEdit.setGeometry(QtCore.QRect(90, 10, 351, 31))
39. self.pointEdit.setObjectName(\_fromUtf8("pointEdit"))
40. self.pointEdit.setMaximum(9)
41. self.pointEdit.setMinimum(1)

44. MainWindow.setCentralWidget(self.centralwidget)
45. self.menubar = QtGui.QMenuBar(MainWindow)
46. self.menubar.setGeometry(QtCore.QRect(0, 0, 449, 21))
47. self.menubar.setObjectName(\_fromUtf8("menubar"))
48. MainWindow.setMenuBar(self.menubar)
49. self.statusbar = QtGui.QStatusBar(MainWindow)
50. self.statusbar.setObjectName(\_fromUtf8("statusbar"))
51. MainWindow.setStatusBar(self.statusbar)

54. self.retranslateUi(MainWindow)
55. QtCore.QMetaObject.connectSlotsByName(MainWindow)
57. **def** retranslateUi(self, MainWindow):
58. **if** self.typeOfWindow == "achievement":
59. MainWindow.setWindowTitle(\_translate("MainWindow", "Add Achievement Points", None))
60. **else**:
61. MainWindow.setWindowTitle(\_translate("MainWindow", "Add Behaviour Points", None))
62. self.saveBtn.setText(\_translate("MainWindow", "Save", None))
63. self.reasonLabel.setText(\_translate("MainWindow", "REASON:", None))
64. self.pointsLabel.setText(\_translate("MainWindow", "POINTS:", None))
65. MainWindow.setWindowIcon(QtGui.QIcon('robertsmyth.png'))
67. **def** save(self):
68. desc = self.reasonEdit.toPlainText()
69. points = self.pointEdit.value()

72. #Adding points to the database.
73. **if** self.typeOfWindow == "achievement":
74. self.student.achievementpoints += points
75. c.execute("INSERT INTO achievementpoints VALUES (:teacher,:student,:points,:reason)",
76. {"teacher":currentUser.username,"student":self.student.username,"points":points,"reason":desc})
77. c.execute("UPDATE student SET achievementpoints = :points WHERE username = :username",
78. {"points":self.student.achievementpoints,"username":self.student.username})
79. **else**:
80. self.student.behaviourpoints += points
81. c.execute("INSERT INTO behaviourpoints VALUES (:teacher,:student,:points,:reason)",
82. {"teacher":currentUser.username,"student":self.student,"points":points,"reason":desc})
83. c.execute("UPDATE student SET behaviourpoints = :points WHERE username = :username",
84. {"points":self.student.achievementpoints,"username":self.student.username})
85. conn.commit()
86. self.saved\_window()
87. self.window.hide()
89. **def** saved\_window(self):
90. QtGui.QMessageBox.question(self.window,"Saved","Save Successful",
91. QtGui.QMessageBox.Ok)
92. #Use this as MainWindow for the close event popup window
93. **class** EditWindow(QtGui.QMainWindow):
94. **def** closeEvent(self,event):
95. result = QtGui.QMessageBox.question(self,
96. "Close The Window",
97. "Are you sure you want to exit? \nAny unsaved data will be lost.",
98. QtGui.QMessageBox.Yes| QtGui.QMessageBox.No)
99. event.ignore()
101. **if** result == QtGui.QMessageBox.Yes:
102. event.accept()


106. #Hashing
107. **def** hashing(password):
108. """Using a sha-512 hashing algorithm to hash a string"""
109. hashed = hashlib.sha512(password.encode('utf-8'))
110. **return** hashed.hexdigest()

1

1. #This is so that if main.py is imported then the program would not actually be ran.
2. **if** \_\_name\_\_ == "\_\_main\_\_":
3. **import** sys
4. #Creating the window
5. app = QtGui.QApplication(sys.argv)
6. MainWindow = QtGui.QMainWindow()
7. ui = Ui\_MainWindow()
8. ui.setupUi(MainWindow)
9. MainWindow.show()
10. sys.exit(app.exec\_())

## Userclasses.py

1. **class** User:

3

1. """A sample user superclass"""
3. **def** \_\_init\_\_(self,first,last,username,password,dob,email,typeOfUser,pic):
4. self.first = first
5. self.last = last
6. self.username = username
7. self.password = password
8. self.dob = dob
9. self.email = email
10. self.type = typeOfUser
11. self.pic = pic

14. @property
15. **def** fullname(self):
16. **return** "{} {}".format(self.first,self.last)
18. @property
19. **def** dobyear(self):
20. **return** self.dob[6:]
22. @property
23. **def** dobmonth(self):
24. **return** self.dob[3:5]
26. @property
27. **def** dobday(self):
28. **return** self.dob[:2]

3

1. **class** Student(User):
2. """A students basic attributes"""
4. **def** \_\_init\_\_(self,first,last,username,password,dob,email,typeOfUser,yeargroup,pic,achievementpoints,behaviourpoints):
5. super(Student,self).\_\_init\_\_(first,last,username,password,dob,email,typeOfUser,pic)
6. self.yeargroup = yeargroup
7. self.achievementpoints = achievementpoints
8. self.behaviourpoints = behaviourpoints
10. **class** Class:
11. """A sample class"""
13. **def** \_\_init\_\_(self,yearGroup,teacher,subject,id,lesson1,lesson2,lesson3,lesson4):
14. self.yearGroup = yearGroup
15. self.teacher = teacher

8

1. self.subject = subject
2. self.id = id
3. self.lesson1 = lesson1
4. self.lesson2 = lesson2
5. self.lesson3 = lesson3
6. self.lesson4 = lesson4

9. **class** Subject:

7

1. """A sample subject"""
2. **def** \_\_init\_\_(self,fullname,shortname,head):
3. self.fullname = fullname
4. self.shortname = shortname
5. self.head = head

8. **class** Behaviourpoint:
9. """A sample behaviour point"""
10. **def** \_\_init\_\_(self,teacher,student,points,reason):

6

1. self.teacher = teacher
2. self.student = student
3. self.points = int(points)
4. self.reason = reason
6. **class** Achievementpoint(Behaviourpoint):
7. """A sample achievement point"""

6

1. **pass**
3. **class** StudentClass:
4. """A class that links students and their classes"""
5. **def** \_\_init\_\_(self,student,currentClass):
6. self.student = student
7. self.currentClass = currentClass
8. self.predicted = none
10. **class** Homework:
11. **def** \_\_init\_\_(self,homeworkId,classId,due,title,desc):
12. self.homeworkId = homeworkId
13. self.classId = classId
14. self.due = str(due)

11

1. self.title = title
2. self.desc = desc
4. @property
5. **def** dueday(self):
6. **return** self.due[8:]
8. @property
9. **def** duemonth(self):
10. **return** self.due[5:7]
12. @property
13. **def** dueyear(self):
14. **return** self.due[:4]

# Screenshots

## Login Form

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidence

2

1

## Main Windows

A picture containing screenshot, person

Description generated with very high confidence

3

A screenshot of a person

Description generated with very high confidence

3

A screenshot of a cell phone

Description generated with very high confidence

3

## Creating and Editing Users

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidence

4

4

4

4

4

A screenshot of a cell phone

Description generated with very high confidence

4

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceView Classes

5

5

4

4

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidence

5

5

A screenshot of a cell phone

Description generated with very high confidence

5

## Behavior and Achievement Points

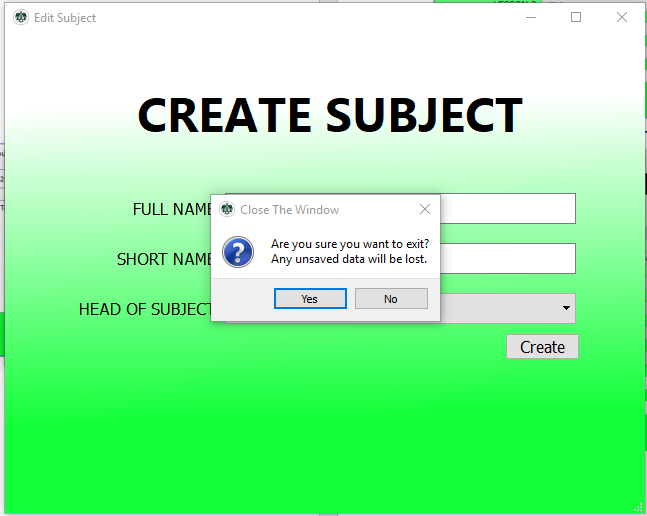
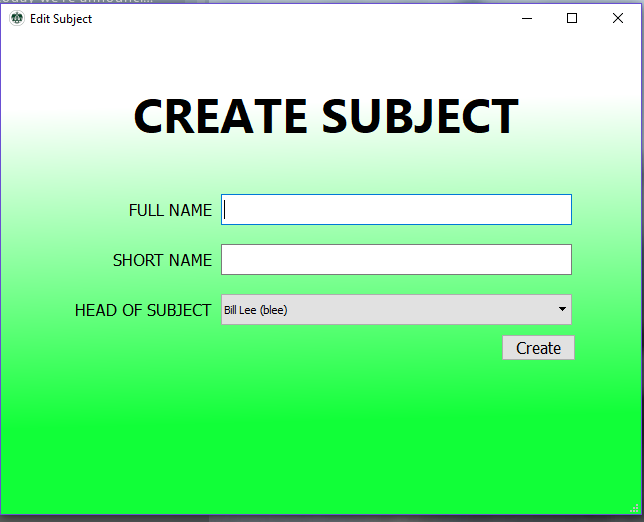
A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceCreate Subject

6

6



7

## Create/Edit Class

A screenshot of a cell phone

Description generated with very high confidence

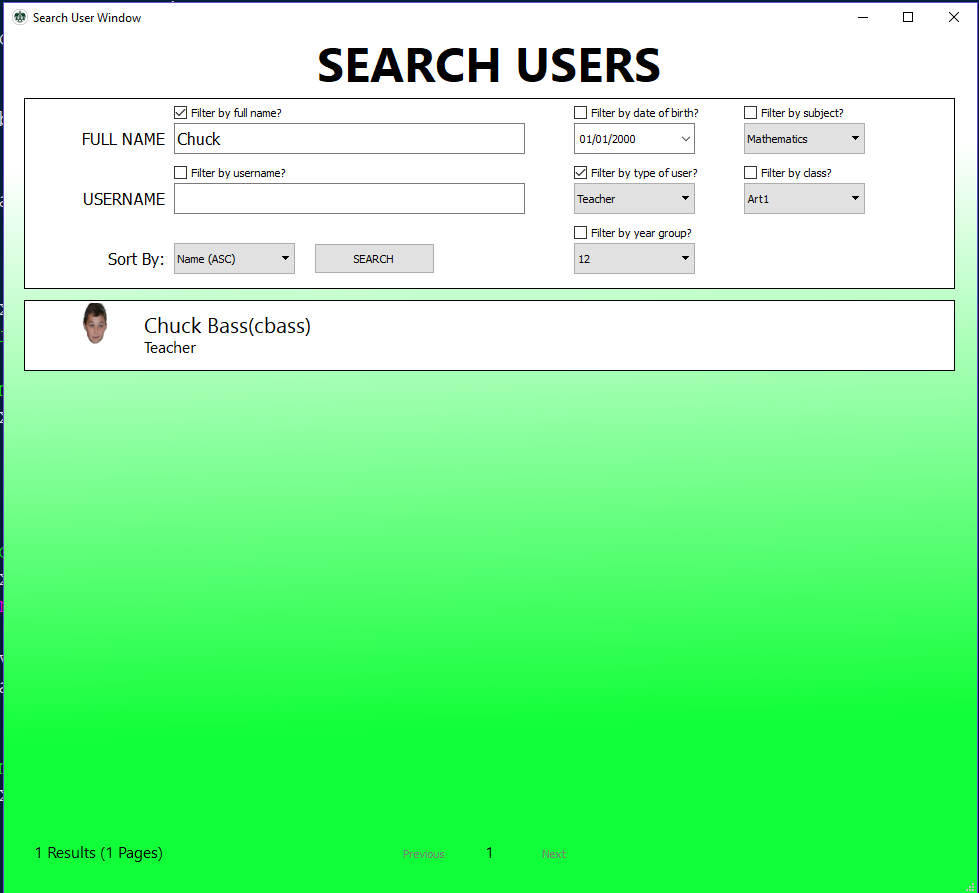
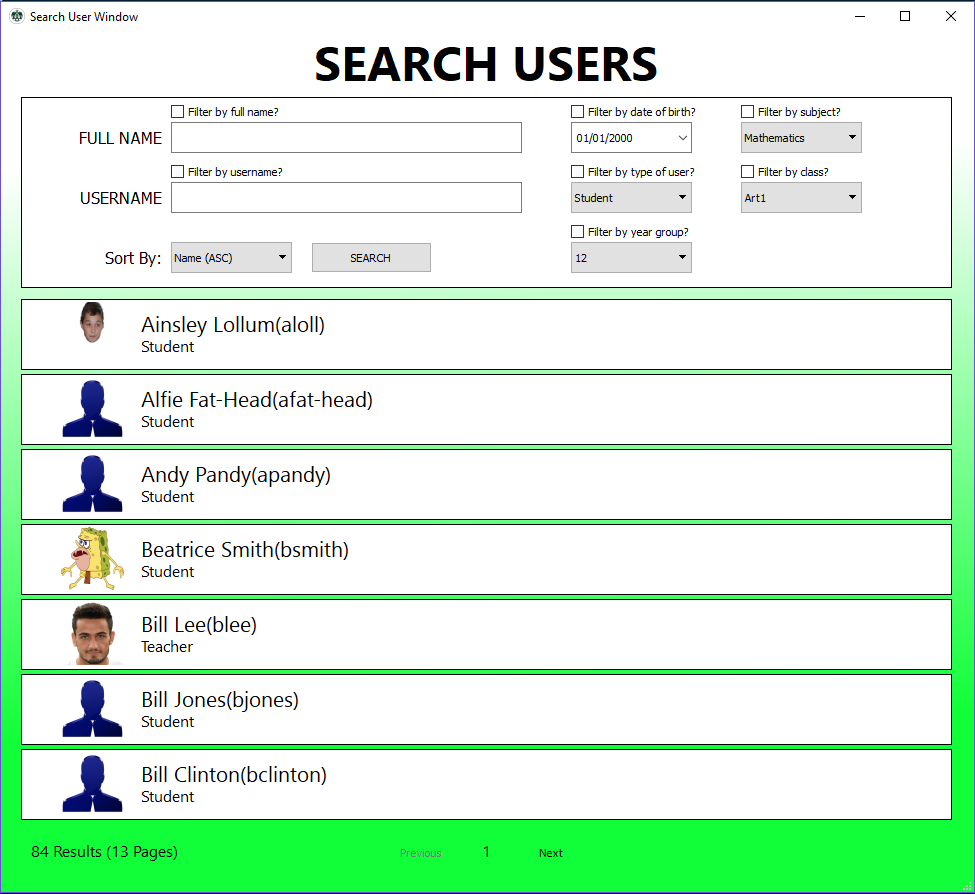
8

A screenshot of a cell phone

Description generated with very high confidence

8

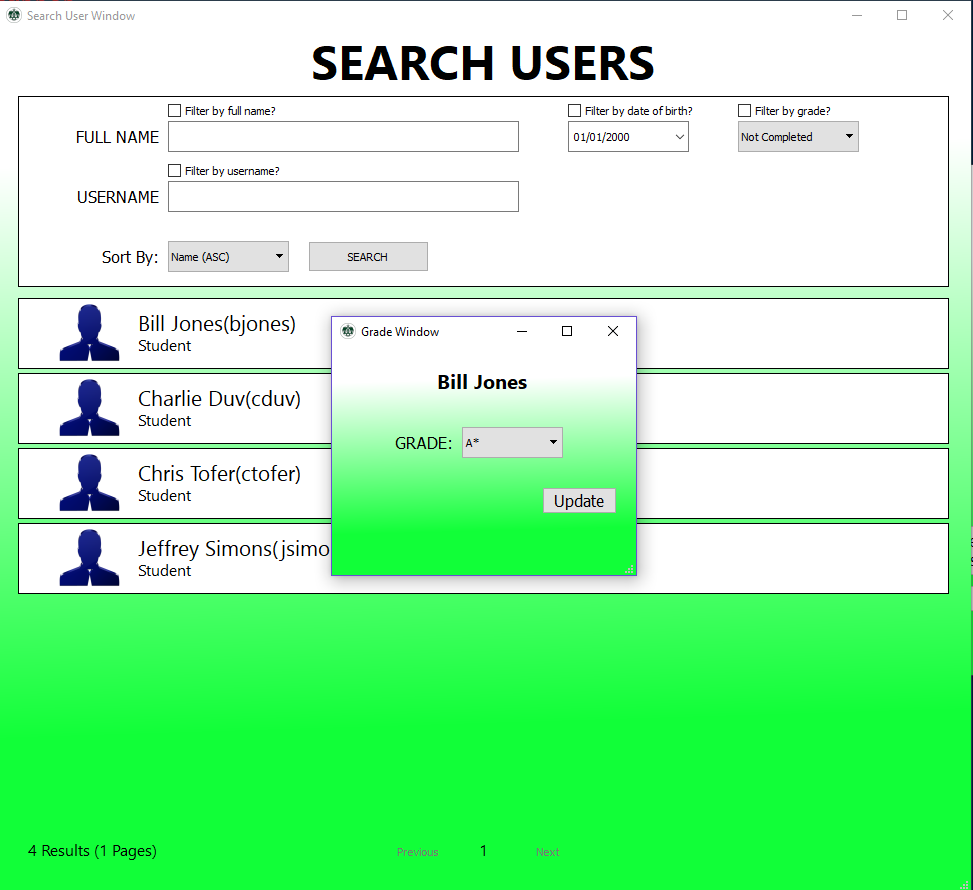
## Search Users



9

9

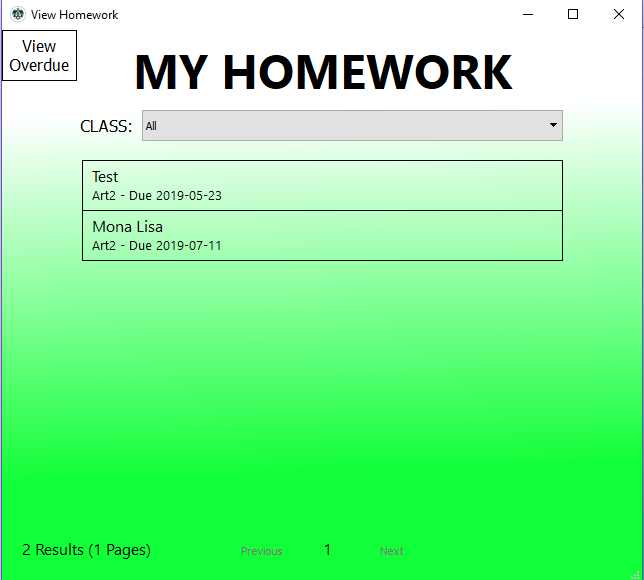
## Update Grades



10

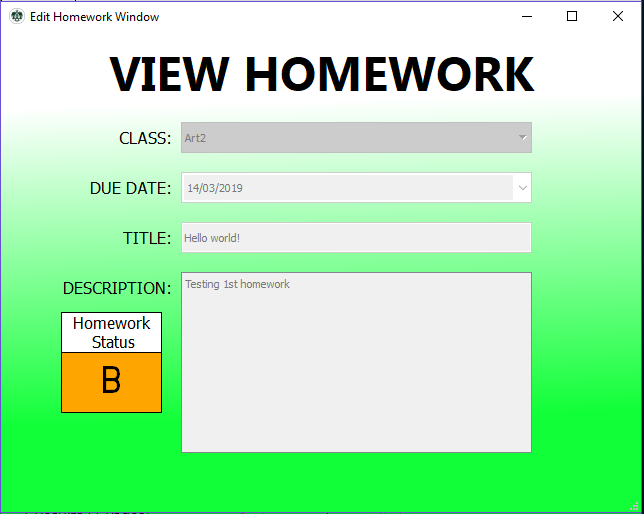
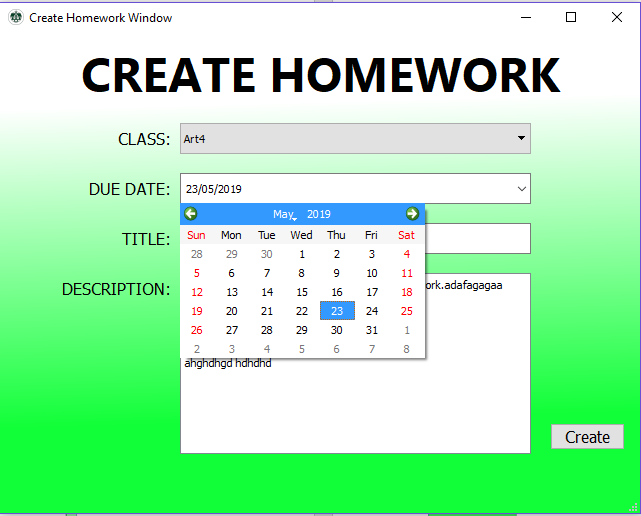
9

## Homework



12

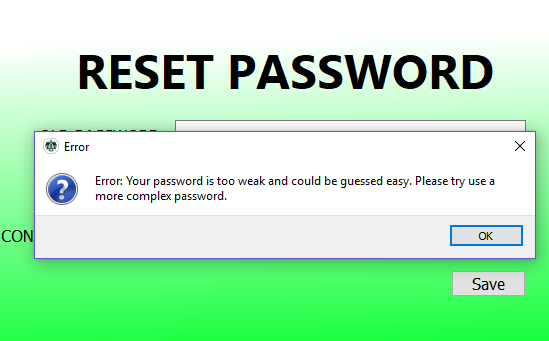
11



11

11

## Reset Password



13

13

## Database

A screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidenceA screenshot of a cell phone

Description generated with very high confidence

7

4

11

8

10

6

A screenshot of a cell phone

Description generated with very high confidence

7

## Objectives

As stated in my analysis my objectives were for:

• “Administrators to be able to add, edit and remove students, teachers and subjects from the database”. This objective has been completed. As shown in the screenshots above, admins are able to create and edit users, classes and subjects. There is no ability to remove users – however I do not feel that it is an important feature that was necessary to complete the project.

• “Teachers to be able to see all their classes in a timetable, to see the details including a list of students etc. and the ability of adding homework, behaviour/achievement points and to add grades for assignments, mini-tests etc.”. Most of this objective was completed. As shown by *number 11,* the teacher is able to set homework and grade users. This would work for tests as well. Behaviour points and achievement points are also added as seen by *number 6*. The structure to create a timetable is there – with different lessons of each class being stored in the database – however it would have been too long to add a timetable. This would be a feature I would add in the future.

• “For students, the plan is so that they can see their timetable and lesson details, they should have access to homework, and news feeds etc. as well as the details such as how they are performing.” Students are able to see homework, grades and class details. As mentioned previously, I was not able to add a GUI for a timetable.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Number | Description | TEX (Typical, Erroneous, Extreme) | Expected  Outcome | Actual  Outcome | Comments and Corrective Actions | Cross Reference |
| 1 | Logging In | T: Entered a user's details | Welcome window to open | As expected. | Tested and working for all types of users | Screenshot 1 |
|  |  | E: Incorrect details | Error: User not found | As expected. |  |  |
|  |  | E: Empty fields | Error: User not found | As expected. |  |  |
| 2 | Reset Password | T: Entered the users current password and a random string of 10 letters. | Save Successful | As expected. |  | Screenshot 13 |
|  |  | E: Confirm password does not match | Error pop up window – saying passwords do not match | As expected. |  |  |
|  |  | E: Entering easy to guess password ie. password123 | Error pop up window – stating password is too weak. | As expected. |  |  |
|  |  | E: Empty new password fields | Error pop up window stating password is too weak | Python Error when testing strength of password. | Updated code so when empty string is passed through it will give correct error message. |  |
| 3 | Creating User | T: All fields filled in | Adds to database and opens edit user window | As expected. |  | Screenshot 4 |
|  |  | E: Use of invalid characters ie. Symbols in name field | Symbols are not added to the field | As expected. |  |  |
| 4 | Saving User | T: Updating fields | Appending changes to database and saved window pop up message | As expected. |  | Screenshot 4 |
|  |  | T: Uploading profile picture | Picture to change. | Mostly as expected. | Sometimes picture goes blank – but seems to be a problem with the library. |  |
| 5 | Adding behaviour/ achievement points | T: Adding 4 points | Adding 4 points to the database | As expected. |  | Screenshot 6 |
|  |  | X: Adding 10 and 1 points | Adds to database | As expected. |  |  |
|  |  | E: 54 points | Does not allow to go that high | As expected. |  |  |
| 6 | Search Users | T: Search with filters and sort | Shows users it applies for. Once clicked – open edit user. | As expected. |  | Screenshot 9 |
|  |  | E: Filters on but with empty fields | Show all users anyway | As expected. |  |  |
| 7 | Create Subject | T: Fill in fields and click create | Add to database and open edit subject window | As expected. |  | Screenshot 7 |
| 8 | Add Class | T: Fill in fields and click create | Add to database and open edit class window | As expected. |  | Screenshot 8 |
| 9 | Show classes | T: Filter by a subject and click on a class | Show all classes and open class window. | As expected. |  | Screenshot 5 |
| 10 | View homework | T: List all future homework filtered by class. | Show homework and click to open homework window | As expected. |  | Screenshot 12 |
| 11 | Create homework | T: Filling in all the fields | Add to database then open edit homework window | As expected. |  | Screenshot 11 |
|  |  | E: Entering a due date in the past | Should not be possible | As expected. |  |  |
|  |  | X: Setting the due date today | Add to database and open edit homework window | As expected. |  |  |
| 12 | Update Grades | T: Select user and grade | Should update the database | As expected. |  | Screenshot 10 |
| 13 | Navigating Through Pages | T: Login and open create user | N/A | As expected. |  |  |
|  |  | T: Login and open view all homework and then open view homework | N/A | As expected. |  |  |
|  |  | E: Login as student and open create a user | Impossible | As expected. |  |  |

# Testing

### Trace Tables

|  |  |  |  |
| --- | --- | --- | --- |
| **Generate Password** | | | |
| **Description**  When an admin creates a user – the admin can’t think of a password for them – but equally the user needs to be able to log on with a secured account to set their password. Hence – the option for the admin to create a temporary password. As shown by the generate\_password function. This function initially creates an empty string – and then generates a random character out of lowercase, uppercase and digits. This is then looped for 9 times (chosen 9 as it will be impossible to get through brute force and not too long that a user can actually enter it). This will then ask if the user is sure they want to update the password – and any previous password would be lost. If the yes button is pressed then a new pop up (password\_popup) appears – and this will update the database and inform the user of it being saved. | | | |
| **Code** | | | |
| **Test inputs**   1. Self.user.username = “trowbotham” | | | |
| **Expected Results:**  Any random 9 length password. | | | |
| i | Change | Input | Output |
|  | Self.password = “” |  |  |
| 0 | Self.password = “9” |  |  |
| 1 | Self.password = “9U” |  |  |
| 2 | Self.password = “9U3” |  |  |
| 3 | Self.password = “9U3h” |  |  |
| 4 | Self.password = “9U3hN” |  |  |
| 5 | Self.password = “9U3hNo” |  |  |
| 6 | Self.password = “9U3hNoL” |  |  |
| 7 | Self.password = “9U3hNoLM” |  |  |
| 8 | Self.password = “9U3hNoLMe” |  |  |
|  |  | Yes | Pop Up “Generate Password?","Are you sure you would like to create a new password? Any previous password will be lost” Yes/No |
|  |  |  | Pop Up  The generated password is: 9U3hNoLMe  Make a note of this password as you will not be able\nto access this again. |
|  | Self.password = “602296A95D55B9AC68F27C7CD86175649ED213AFE82D5507D4818EA5CA1F22347D02D058F006FFD3CCAC58D16E28D3BF37ED95C85A34E1D384BB62C27AAF8DD9” |  |  |
|  | Saved to database |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | Change | Input | | Output | |
|  | Self.password = “” |  | |  | |
| 0 | Self.password = “9” |  | |  | |
| 1 | Self.password = “9U” |  | |  | |
| 2 | Self.password = “9U3” |  | |  | |
| 3 | Self.password = “9U3h” |  | |  | |
| 4 | Self.password = “9U3hN” |  | |  | |
| 5 | Self.password = “9U3hNo” |  | |  | |
| 6 | Self.password = “9U3hNoL” |  | |  | |
| 7 | Self.password = “9U3hNoLM” |  | |  | |
| 8 | Self.password = “9U3hNoLMe” |  | |  | |
|  |  | No | | Pop Up “Generate Password?","Are you sure you would like to create a new password? Any previous password will be lost” Yes/No | |
| **List Homework** | | | | | |
| **Description**  When a user is logged on the will be a window that lists the users homework. This can be used by teachers and students. The user will then be able to filter by class or show all homeworks in one list. | | | | | |
| **Code** | | | | | |
| **Test inputs**   1. Self.classid = “Art2”, self.typeOfWindow = “Future” 2. Self.classid = “All”, self.data = [“Art1”,”Art2”,”Art3”,”Art4”], self.typeOfWindow = “Past” | | | | | |
| **Expected Results:** | | | | | |
| Change | | | Input | | Output |
|  | | | Self.classid = “Art2”  Self.typeOfWindow = “Future” | |  |
| Today = “2019-04-06” | | |  | |  |
| Data = Fetch from database any value greater than “2019-04-06” | | |  | |  |
| Data = ((“Art2\_4”,”Art2”,”2019-04-25", “Read article...”,” “),(“Art2\_3”,”Art2”,”2019-05-23", “Test”,”FSAGF...”),(“Art2\_2”,”Art2”,”2019-07-11",”Mona Lisa”,”Test”)) | | |  | |  |
| Data =(“Art2\_4”,”Art2”,”2019-04-25", “Read article...”,” “),(“Art2\_3”,”Art2”,”2019-05-23", “Test”,”FSAGF...”),(“Art2\_2”,”Art2”,”2019-07-11",”Mona Lisa”,”Test”)] | | |  | |  |
| Self.homeworks = [(“Art2\_4”,”Art2”,”2019-04-25", “Read article...”,” “),(“Art2\_3”,”Art2”,”2019-05-23", “Test”,”FSAGF...”),(“Art2\_2”,”Art2”,”2019-07-11",”Mona Lisa”,”Test”)] | | |  | | Self.retranslateUi() |

|  |  |  |  |
| --- | --- | --- | --- |
| i | Change | Input | Output |
|  |  | Self.classid = “All”  Self.typeOfWindow = “Past”  Self.data = [“Art1,”Art2”,”Art3”,”Art4”] |  |
|  | Today = “2019-04-06” |  |  |
|  | Data = [] |  |  |
| Classid = “Art1” | Data = [] |  |  |
| Classid = “Art2” | Data = [(“Art2\_1”,”Art2”,”2019-02-12",”Hello world!”,”Testing 1st homework”)] |  |  |
| Classid = “Art3” | Data = [(“Art2\_1”,”Art2”,”2019-02-12",”Hello world!”,”Testing 1st homework”)] |  |  |
| Classid = “Art4” | Data = [(“Art2\_1”,”Art2”,”2019-02-12",”Hello world!”,”Testing 1st homework”),(“Art4\_1”,”Art4”,”2019-03-15",”ABCDEFGHI”,”Testtttttt”)] |  |  |
|  | Data = [(“Art2\_1”,”Art2”,”2019-02-12",”Hello world!”,”Testing 1st homework”),(“Art4\_1”,”Art4”,”2019-03-15",”ABCDEFGHI”,”Testtttttt”)] |  |  |
|  | Data = [(“Art4\_1”,”Art4”,”2019-03-15",”ABCDEFGHI”,”Testtttttt”), (“Art2\_1”,”Art2”,”2019-02-12",”Hello world!”,”Testing 1st homework”)] |  |  |
|  | Self.homeworks = [(“Art4\_1”,”Art4”,”2019-03-15",”ABCDEFGHI”,”Testtttttt”), (“Art2\_1”,”Art2”,”2019-02-12",”Hello world!”,”Testing 1st homework”)] |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Create User** | | | |
| **Description**  At the start – for users to be able to log on and use their accounts they must be actually created and stored in the database. The following function gathers the necessary details for that to happen and then appends to the database. The intention is to only get the essential details at first – and then at a later point will the profile picture and password be added. | | | |
| **Code** | | | |
| **Test Inputs**   1. Self.user.first = “Tom” Self.user.last = “Rowbotham” self.user.dob = “23/07/2001” self.user.email = “[rowbotom@gmail.com](mailto:rowbotom@gmail.com)" self.user.type = “Student” self.user.yeargroup = “13” 2. Self.user.first = “John” Self.user.last = “Canoe” self.user.dob = “25/06/1960” self.user.email = “[canoejohn@outlook.com](mailto:canoejohn@outlook.com)" self.user.type = “Teacher” | | | |
| i | Change | Input | Output |
|  |  | Self.user.first = “Tom”  Self.user.last = “Rowbotham” |  |
|  | Self.user.username = “trowbotham”  Length = 10  Data = [(“trowbotham”), (“trowbotham1”), (“trowbotham2”), (“trowbotham3”), (“trowbotham4”)] Usernames = [] |  |  |
| (“trowbotham”) | Usernames = [“trowbotham”] |  |  |
| (“trowbotham1”) | Usernames =  [“trowbotham”,”trowbotham1”] |  |  |
| (“trowbotham2”) | Usernames = [“trowbotham”, “trowbotham1”, “trowbotham2”] |  |  |
| (“trowbotham3”) | Usernames = [“trowbotham”, “trowbotham1”, “trowbotham2”, “trowbotham3”] |  |  |
| (“trowbotham4”) | Usernames = [“trowbotham”, “trowbotham1”, “trowbotham2”, “trowbotham3”, “trowbotham4”] |  |  |
|  | Chosen = False |  |  |
|  | Self.user.username = “trowbotham1” |  |  |
|  | Self.user.username = “trowbotham2” |  |  |
|  | Self.user.username = “trowbotham3” |  |  |
|  | Self.user.username = “trowbotham4” |  |  |
|  | Self.user.username = “trowbotham5” Chosen = True |  |  |
|  |  | self.user.dob = “23/07/2001” |  |
|  |  | Self.user.email = “rowbotom@gmail.com” |  |
|  | Self.user.password= “NULL” |  |  |
|  | Self.user.pic = “NULL” |  |  |
|  | \*appends to base user to database\* |  |  |
|  | Self.user.yeargroup = “13” |  |  |
|  | \*appends to student database\* |  |  |
|  | \*opens edit user window\* |  | Saved window |

## Testing Navigation

As navigation is hard to test through trace tables – I have ensured to thoroughly test almost every possible combination of windows ensuring I do not miss out on anything. So – instead, I chose to record my program working an me going through each window. This can be found at <https://youtu.be/81GsRPy--go> . As shown in the video, the navigation works fine - with the user being able to access all pages through the menu bar and buttons on pages.

# Evaluation

## Evaluation of system objectives

|  |  |  |
| --- | --- | --- |
| **Objective** | **Met?** | **Comment?** |
| Administrators to be able to add, edit and remove students, teachers and subjects from the database | Mostly | Administrators are able to add, edit and more with users, classes and subjects. However – I was not able to add the ‘remove user function’ – as I did not enough time and I did not believe this was a vital enough objective. The user is still able to delete records by using a database viewer program such as *DB browser for SQL lite.* Implementing the remove option would be simple – a simple button would require to be implemented on the edit user page connecting to an SQL statement such as “DELETE FROM users WHERE username = *username”.* The problem, however, would be removing all previous records of the user which would require searching every other table. This is possible but would take a bit more time. |
| Teachers to be able to see all their classes in a timetable, to see the details including a list of students etc. and the ability of adding homework, behavior/achievement points and to add grades for assignments, mini-tests etc. | Mostly | Teachers are able to see all their classes as well as the details of the classes. The framework for creating a timetable is in the database, but the GUI that would need setting up – which I did not have enough time to set up. To do that I would have needed to fetch all the individual lessons for each period from the user’s classes and ensure there is no overlap. The teacher and admin both have access to a search function where they can search users – and select users to add grades, achievement points and behaviour points. The grades will be from the homework window which can act as mini-test results too. |
| For students, the plan is so that they can see their timetable and lesson details, they should have access to homework, and news feeds etc. as well as the details such as how they are performing. | Mostly | As mentioned previously, I was unable to implement the timetable GUI due to time. However, all the other requirements were met. The user is able to view both past and present homework – see their grades. Homework can also be used as a news feed for example a title “Reminder - coursework in for this Friday”. As for details on how a student is performing – they are able to click on each homework and view what grade they got – with a colour indicator depending on how good it was. |

## Discussion of possible improvements

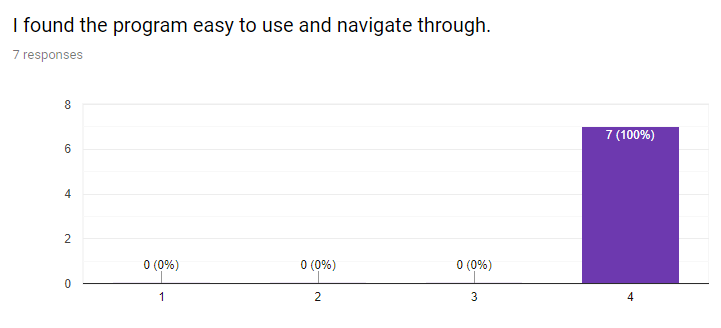
If I was to carry on there would be a lot more functions and improvements I would add:

* As mentioned in my objectives – I would ensure to add the timetable GUI and the ability to remove users.
* One improvement I would make is my validation. Though throughout my program I have ensured a certain level of validation through the use of regex – I would like to add extra requirements such as fields are not empty – or some are fields are required and some are not. In the future – there are other functions that could be created – such as grouping classes/sets of people together – I.e. Posting a UCAS homework for all of year 13 etc.
* Another potential function that could be added in the future would be messages – which would be helpful in cases for example if a student is stuck with homework and needs to message the teacher.
* If I had more time – I would hopefully venture towards using other programming languages. This would perhaps make it a lot easier sometimes especially with the difficulty with Pythons UI libraries – compared to languages such as java where a screen could be made a lot easier.
* I would also hope to in the future – expand beyond just a desktop application and create things such as a phone app/website - where this would hopefully teach me a new set of skills.

## User feedback and analysis of feedback

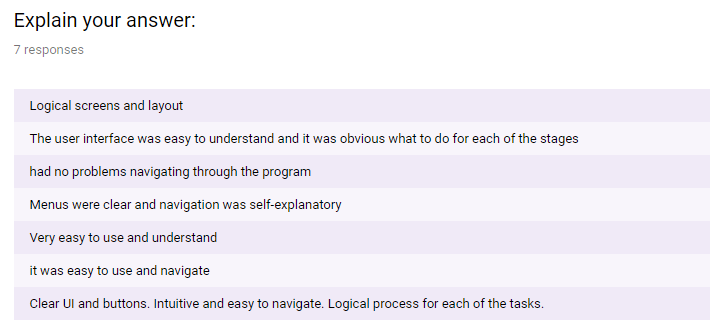
I gave my program and a feedback form to a few people in my school to get feedback on my project. The full google forms sheet can be seen in Appendix B.

**Question 1a – Ease of Use**



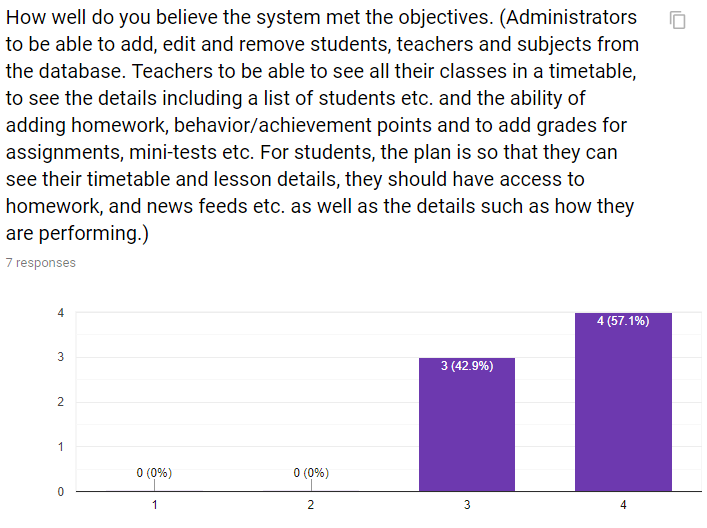
The first question gained an overwhelmingly positive response with everyone rating the ease of use and navigation 4/4. This question was to ensure that my program would be usable and understandable by all users – even those that are not used to using technology. By ensuring my program had clear titles, headings, menu names and buttons – it has clearly helped my test users navigate through my program and helped achieve all 4/4 responses.

**Question 1b – Ease of Use Explanation**

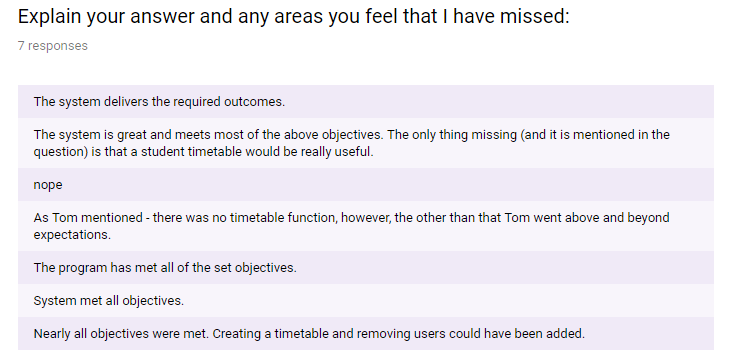


**Question 2a – Objectives Met**

With the next question, I was hoping that if there were any problems – they would be identified in the above question. However, as mentioned previously – nobody seemed to have any problems with using my system.

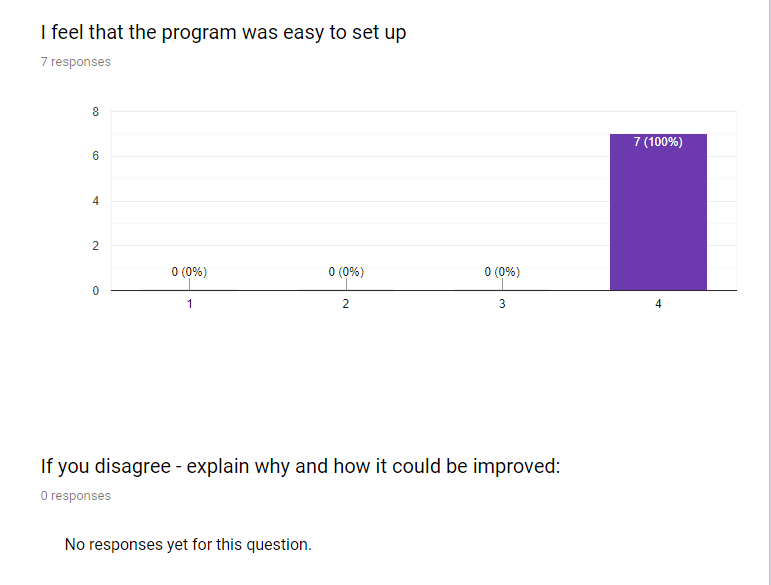


**Question 2b – Objectives Met / Improvements**

The next question was aimed to ensure that I had achieved what I originally said I was going to do and achieve my objectives. For this question – I was expecting quite a few responses with three out of four answers. This is because as mentioned at the start of this section – there were two main objectives I was not able to complete and this was due to a misjudgement of how the time and skills required to articulate my system. However, I was able to create a working version of my system that met all the important objectives – and also a few functions I had not mentioned before such as my reset password and search function. The results were not surprising and there were no answers that I was not expecting.   


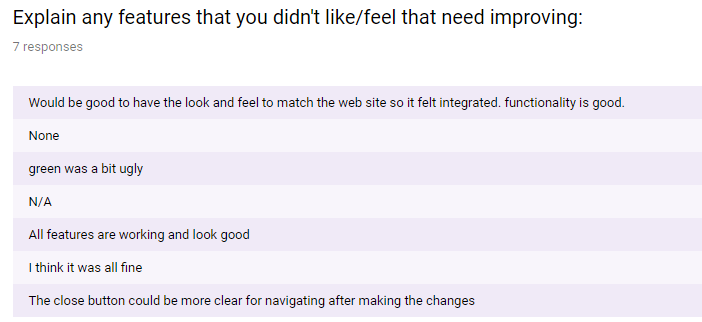
Similarly, to what I had previously said, this question I was expecting my respondents to mention the missing timetable and remove user functions. There were no areas mentioned that I was not already aware of – so overall, I believe the feedback for this question was a success.

**Question 3 – Ease of Set-Up**



This question is similar to my first question – but was more aimed at how frightening the experience is to a new user that has never seen my system before. However – from these results I am happy that everyone found it easy to set up and from the start system makes clear the functions the user is able to control.

**Question 4 - Improvements**



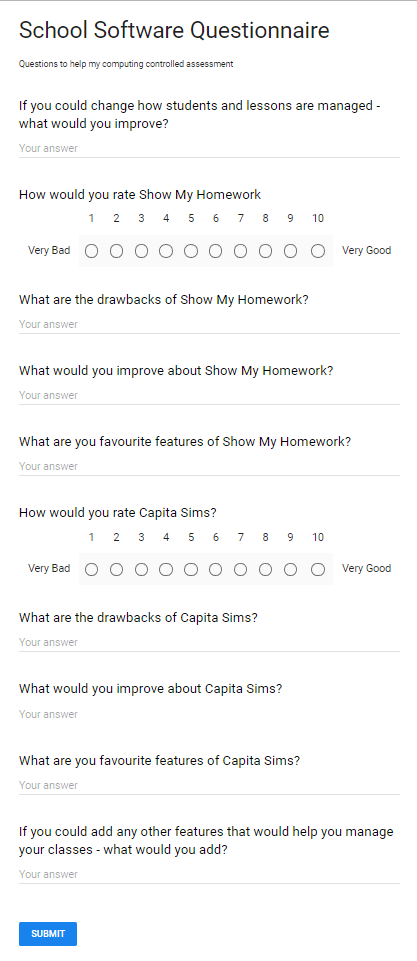
Finally – this question was aimed to catch any other problems or improvements that were not directly asked for before. The responses included:

* The first response mentioned how integrating a matching website would be good. As mentioned in my possible improvements for the future - this would definitely be on my list on what to do next. The best part of having a website - is that basically every platform would be able to run it.
* The next response mentioned the green colour scheme. Though I understand that green to white gradient background may not be the typical background – the green and white colour scheme is because Robert Smyth Academy is part of the “Tudor Grange Academies Trust” and hence has a lot of green and white contained in their websites so I was inspired to do something similar. More specifically – there is a similar background on the school site portal with a darker green to white gradient used ([*https://cse-trinidad.co.uk/core/login?signin=f82eac0759e131f13316aefe94dd4b9b*](https://cse-trinidad.co.uk/core/login?signin=f82eac0759e131f13316aefe94dd4b9b)). Clearly if this was a common complaint about the system, I would change it – however, currently I do not see any issue with the background.
* The final issue was after making changes I.e. To a user – they wanted an easier way to return to windows after making the changes. Every time a window is opened the previous one remains open. This is often useful when comparing two students – as you will be able to hold two windows side by side. So, to close the windows – you have to press the red “X” at the top right of that window and the windows will stay forever open until you have pressed the X button (unless it is the login page). It would be helpful in the future so that when the user logs back out, every window closes – not just the main menu. I would need to investigate if this could be done with the UI library.

## Conclusion

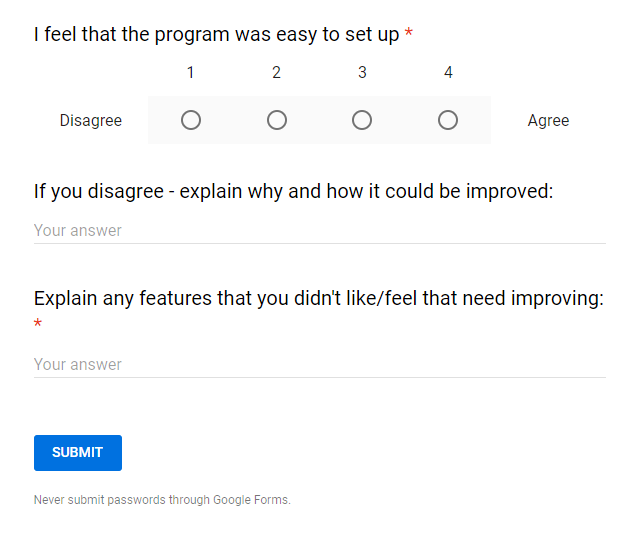
In conclusion, I feel that I have created my project to a standard I am very pleased with. Six months ago, I would not have had any clue on how to create my system and it is great to see how far I have come and how much I have learnt. If I had extra time, I would definitely complete the areas that I was not able to implement in the current version of the system. I have enjoyed the creation of my project and have gained a lot of skills and knowledge of different libraries, database structures etc, which has definitely made this project worthwhile.

# Appendix A – Questionnaire



[*https://docs.google.com/forms/d/e/1FAIpQLSeB4rIBygKiJJWhCPXezRTH2-IDNXLnGwj\_p-YxEK1857DiCw/viewform*](https://docs.google.com/forms/d/e/1FAIpQLSeB4rIBygKiJJWhCPXezRTH2-IDNXLnGwj_p-YxEK1857DiCw/viewform)

# Appendix B – Feedback Form



[*https://docs.google.com/forms/d/1QOj6EVqgK3Tdm4o\_yERemQxn\_B1xVBcM4K7rzUHzj1M/edit#responses*](https://docs.google.com/forms/d/1QOj6EVqgK3Tdm4o_yERemQxn_B1xVBcM4K7rzUHzj1M/edit)

# Bibliography

iSams. (2018, November 1). *The Platform*. Retrieved from iSams: https://www.isams.com/platform/the-platform/

Raspberry Pi Foundation. (2018, July 13). *Object-oriented Programming in Python*. Retrieved from Future Learn: https://www.futurelearn.com/courses/object-oriented-principles

Satchel, T. (2018, September 17). *Show My Homework*. Retrieved from Show My Homework: https://www.showmyhomework.co.uk

Schafer, C. (2018, July 15). *Python SQLite Tutorial: Complete Overview - Creating a Database, Table, and Running Queries*. Retrieved from Youtube: https://www.youtube.com/watch?v=pd-0G0MigUA&t=520s

Sentdex. (2015). *PyQt Python GUI Application Development with Python*. Retrieved from Youtube: https://www.youtube.com/playlist?list=PLQVvvaa0QuDdVpDFNq4FwY9APZPGSUyR4

Sims, C. (2018, September 17). *Sims Teacher App*. Retrieved from Capita Sims: https://www.capita-sims.co.uk/products-and-services/sims-teacher-app

Ssj6. (2017). *Python PyQt Login Form*. Retrieved from YouTube: https://www.youtube.com/playlist?list=PL9OFx3Fy8bijqr9hQ5vsSCE\_k0itKOjyt

Torres, A. (2018, September 1). *Hashing strings with Python*. Retrieved from Python Central: https://www.pythoncentral.io/hashing-strings-with-python/

Wheeler, D. L. (2018, September 1). *zxcvbn: Low-Budget Password Strength Estimation*. Retrieved from Usenix: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler

1. **iSams. 2018** [↑](#footnote-ref-1)
2. **Team Satchel. 2018.** [↑](#footnote-ref-2)
3. **Raspberry Pi Foundation. 2018.** [↑](#footnote-ref-3)
4. **Schafer, Corey. 2018.** [↑](#footnote-ref-4)
5. **Wheeler, Daniel Lowe. 2018.** [↑](#footnote-ref-5)
6. **Andres Torres. 2018.** [↑](#footnote-ref-6)
7. **Sentdex. 2015** [↑](#footnote-ref-7)
8. **Ssj6. 2017** [↑](#footnote-ref-8)
9. **Capita Sims, 2018** [↑](#footnote-ref-9)
10. [↑](#endnote-ref-1)
11. https://www.g2crowd.com/products/isams/reviews [↑](#footnote-ref-10)