Computing Practical Project

Oscar Daniel

Febuary 2015

Contents

Chapter 1

Analysis

1.1 Background and Definition of Problem

At Queen Elizabeth Grammar School, pupils in year 13 are taught several calculations for their science subjects. They are tested on these every month or so, after the unit has been taught and students are assumed to understand the topics. Students are set questions during teaching time, however these results are not recorded and sometimes a struggling student may slip under a teacher's radar.
Examples of A Level Calculations:

Rate of reaction equations
A set of reaction equations used to deduce the rate of a reaction, the concentrations of reactants, their orders and the rate constant: k.
A reaction of reactants $A + B \rightarrow D$ where A has the concentration $0.0023\,\mathrm{mol\,dm^{-3}}$ and the power 1. B has the concentration $0.0154\,\mathrm{mol\,dm^{-3}}$ with the power 2. Whilst the rate of reaction is $0.000\,045\,\mathrm{mol\,s}-1$.
An example question.

Standard deviation
A statistical value that is used ceaselessly in biology.
You and your friends have just measured the heights of your dogs (in millimetres): The heights (at the shoulders) are: 600mm, 470mm, 170mm, 430mm and 300mm. Find the standard deviation
Find out the Mean, the Variance, and the Standard Deviation.
Your first step is to find the Mean: The mean (average) height is 394 mm
Now we calculate each dog's difference from the Mean. To calculate the Variance, take each difference, square it, and then average the result:
So, the Variance is 21,704.

And the Standard Deviation is just the square root of Variance, so:
Standard Deviation: $\sqrt{21,704} = 147.32 = 147$ (to the nearest mm)
An Example Question

Ideal Gas Equation
A calculation that is used in both chemistry and physics A level.
The ideal gas equation is PV = nRT. P is pressure in Pa, V is volume in cubic metres, n is number of moles and T is temperature in degrees Kelvin. R is the ideal gas constant which is measured as 8.314 Joules per Kelvin per Mole.
STP = 273K and 100kPa
Problem 1: Determine the volume of occupied by 2.34 grams of carbon dioxide gas at STP.
Our first job is to rearrange the equation to find V: V = nRT/P
Then find moles of CO_2: 2.34/44 = 0.0532
Then apply these to the equation.
$(0.0532 * 8.314 * 273)/100,000 = 0.0012074921m^3$
Problem 2: A sample of argon gas at STP occupies 56.2 litres. Determine the number of moles of argon and the mass in the sample.
Rearrange to find n: n=PV/RT
56.2 litres = 0.0000562 m^3
(100,000*0.0000562)/(8.314*273) = 0.00247607416 moles. $Ar = 18g/Mole$
$18 * 0.00248 = 0.0446g$
An Example Question

Hardy – Weinberg Equation
A statistical test that is used to estimate the proportions of different genotypes in a population.
1. If 98 out of 200 individuals in a population express the recessive phenotype, what percent of the population would you predict would be heterozygotes? (a) I have given you information on the frequency of the homozygous recessive (or q^2). So start by determining q2 and then solving for q. $98/200 = q^2 = 0.49$ $\sqrt{0.49} = 0.7 = q$
(b) Now that you have q, you can solve for p. Remember there are only two alleles in the population, so if you add the frequency of the two alleles, you have accounted for all possibilities and it must equal 1. So
$p + q = 1$ Therefore: $p = 1 - q = 0.3$
(c) Now what is the formula for heterozygotes? Think back to the Hardy-Weinberg equation – it is dealing with the genotypes of individuals in the population. $2pq = 2 * 0.3 * 0.7 = 0.42$
(d) Now that you have figured out the percentage of heterozygotes, can you

figure out the percentage of homozygous dominant? Does the percentage of homozygous dominant, heterozygotes and homozygous recessive individuals add up to 100 percent? If not, you have made an error. Those are the only three genotypes possible with only two alleles and a simple dominant and recessive relationship. $1 - (0.49 + 0.42) = p^2 = 0.09$ or 0.3^2
An Example Question

1.2 Users and the current system

Mr Fascione is the head of Science at QEGS. He focusses on student grades and student support alongside being a teacher.

Currently the only way for pupils' scores from homework to be tracked is by them being entered manually into a spreadsheet after being recorded on paper. There is also no way for pupils to have their problems solved, or explained step by step, other than referring to a text book, which is not interactive. Student feedback is also only monitored for tests, with this system every piece of homework could be used for student support.

Pupils can only access a few problems that have been devised by teachers or examiners. These can also be hard for students to grasp, so a program which goes through these at the students' personal pace would be of great use. Teachers also have no digitally stored questions that can be easily sent to students, they must hand out photocopies and these are always being lost. A computer system would allow work to be set easily, without risk of students losing their things, as well as teachers not losing score sheets.

From my feedback and interviews from my end users I have been able to select the appropriate calculations for the computer package to use, and also have a framework to try and emulate, to best please my end user.

1.2.1 Data Flow Diagrams

To give a better understanding of how data flows in the current and proposed systems, I produced a data flow diagram for each. They highlight the areas of concern already mentioned and show how a copmuter package would eliminate them.

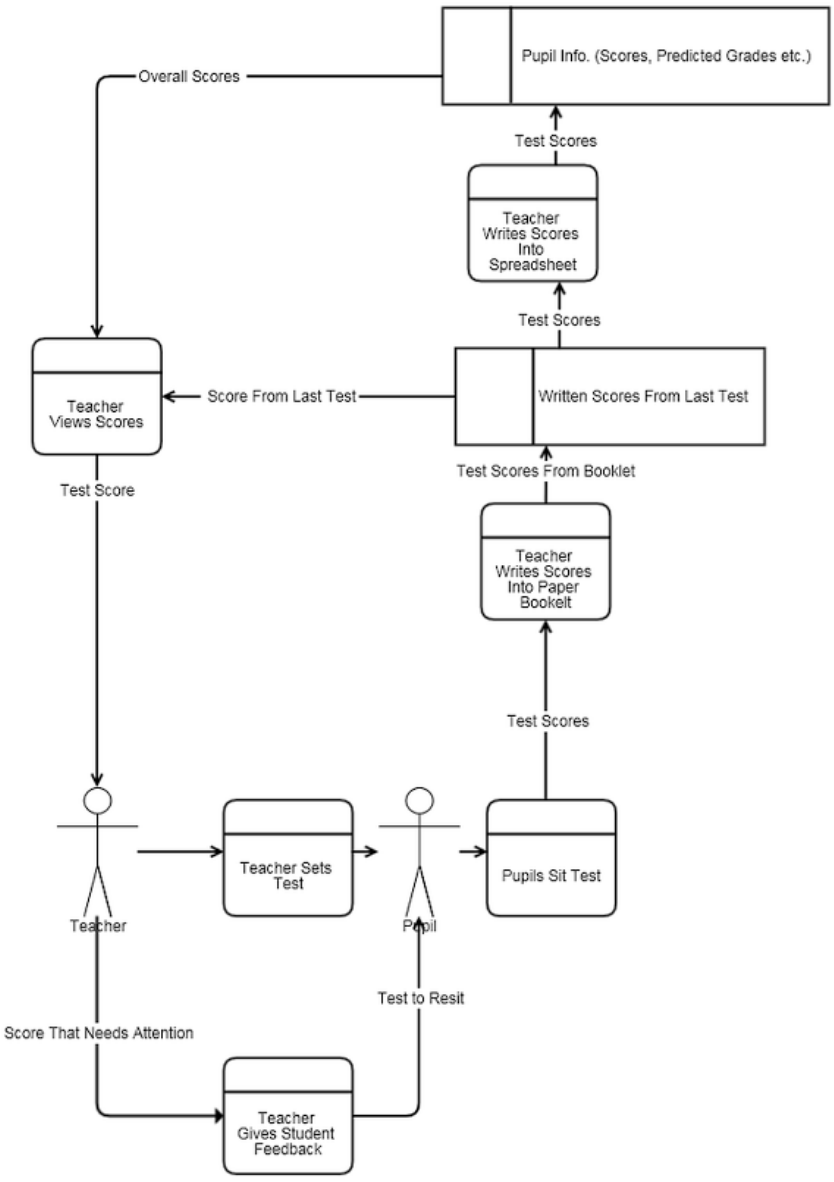


Figure 1.1: The Current System

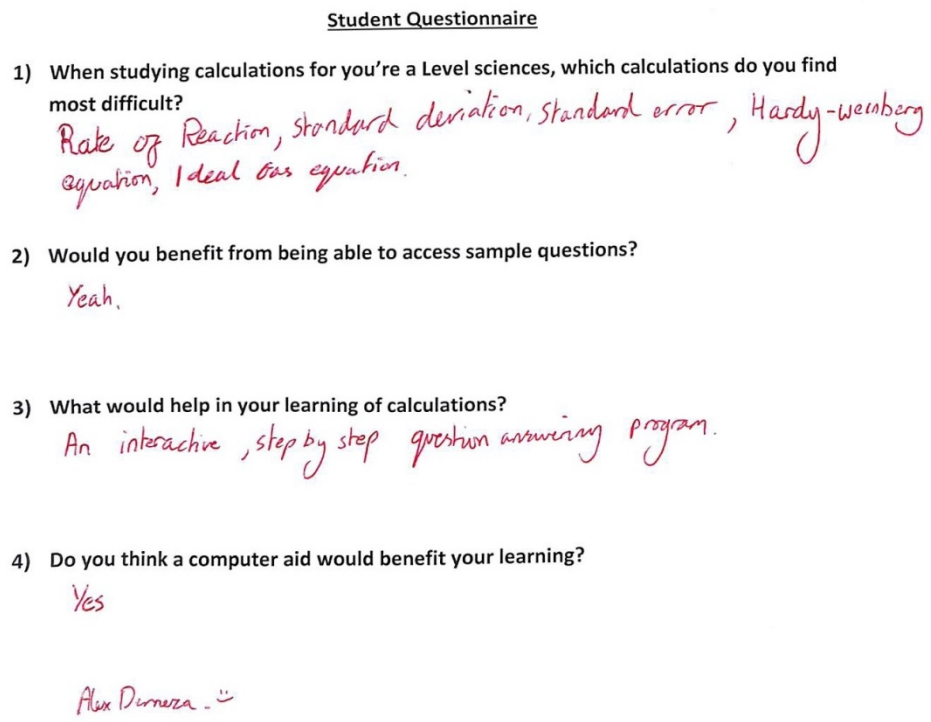
Figure 1.2: New System 1

Figure 1.3: New System 2

Figure 1.4: New System 3

1.3 Feedback Reports

To get a better idea of exactly what type of system I should make, and what requirements that system would need to have, I asked my end users a few questions.



Student Questionnaire

1) When studying calculations for you're a Level sciences, which calculations do you find most difficult?

Rate of Reaction, standard deviation, standard error, Hardy-weinberg equation, Ideal Gas equation.

2) Would you benefit from being able to access sample questions?

Yeah.

3) What would help in your learning of calculations?

An interactive, step by step question answering program.

4) Do you think a computer aid would benefit your learning?

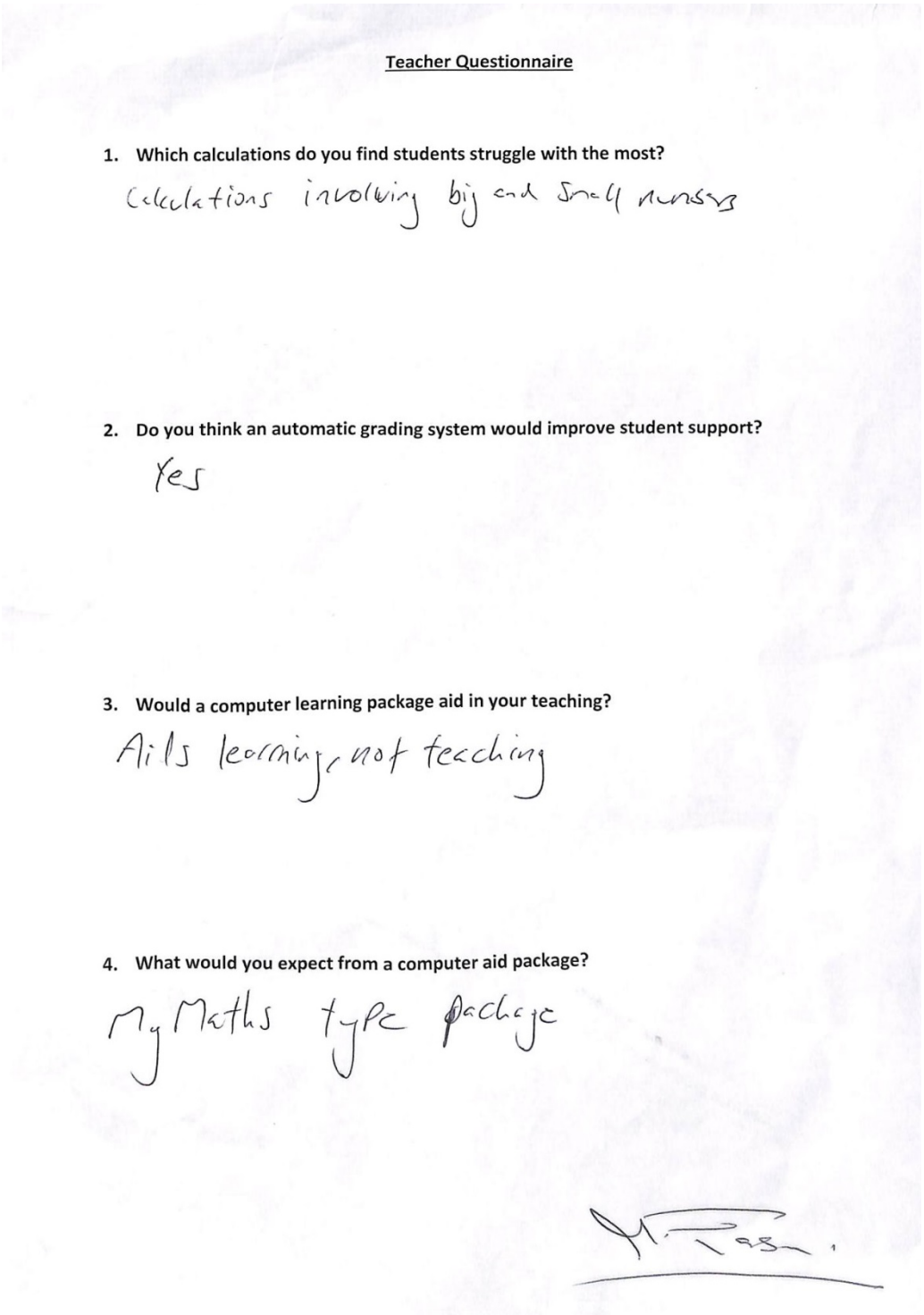
Yes

Alex Dimeza - ☺

Figure 1.5: Student Questionnaire

1.4 Proposed System

Pupils will need to be able to input their own problems and have them be solved in a step by step process where they can enter their answer for each step. The program will tell them if they were right and instructing them as to what they should be doing.
They will also be able to store their own problems and solutions in the central database.
Teachers will be allowed to set work to be done and see the scores.



Teacher Questionnaire

1. **Which calculations do you find students struggle with the most?**

Calculations involving big and small numbers

2. **Do you think an automatic grading system would improve student support?**

Yes

3. **Would a computer learning package aid in your teaching?**

Aids learning, not teaching

4. **What would you expect from a computer aid package?**

My Maths type package

Figure 1.6: Teacher Questionnaire

Teachers and pupils will also have unique accounts with teachers having the ability to see pupil scores and set questions. If pupils are having difficulties then they can send a message to the teacher and vice versa. Teachers should be able to make new pupil and teacher accounts and delete old ones.
Allows students and staff to log in.
The program will generate its own questions but may not display questions in the correct format.
Only 100 questions will be stored at once as the memory taken up would be too high.
Pupil accounts would have to be entered in manually as school will not give me access to the student detail database.

1.5 Entity Relationship Diagrams

These are the entity relationships that I propose to implement in my new System:

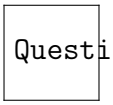


Figure 1.7: The relationship between Questions and Question Sets

Figure 1.8: The relationship between Students and Forms

1.6 Objectives

Essential Objectives

1. Example Based Demonstration – The program should be able to provide a clear demonstration of how any of the equations work, by executing them for the user. It should be able to carry out this demonstration either on data entered by the user, or on data stored within the program/database.

2. Interactive System allowing Users to Solve Equations – The user should be able to choose to attempt to solve the equations themselves – if this is the case the program should point out (or correct) any mistakes they may have made.

3. Unique User Accounts- Teachers and pupils should have different accounts with different privileges.

4. A System To Allow Users to Input and Store Questions – The users should be able to input their own questions, that the program can solve or walk them through, and have the option to store them.

5. A Database Holding Question Data- the program should run off of a back end database holding around 100 questions in total, which can randomly select a question, or allow a teacher to set specific questions for the pupil to answer.

6. A System For Teachers to Set Questions and Students To Attempt them – The program should allow teachers to select questions from the database to give to students. Students should be able to then complete the questions and have them removed from their questions to do.

General Objectives

1. Teachers should be able to create or delete user accounts.

2. A central database should collate all information into a practical form.

3. The program should be self-explanatory and not overly complex for the user.

4. Should use less than 100MB of storage.

5. A log in system with passwords and usernames

Possible Extension

1. Background Explanation Of Problems- The program should be able to give a small amount of detail initially if the student should request it.

2. A Student Feedback System- Students should be able to highlight questions that they are struggling with and send the question to the teacher.

3. Grade scores- Highlights students who are below their target grades.

4. A Database Holding Pupil Performance- Teachers could have access to the questions that pupils. answered correctly and incorrectly so that they can monitor their performance and therefore provide support.

Acceptable Limitations

1. To allow the program to generate questions properly with the correct units would require the use of libraries and a lot more time than I have available.

2. As my school will not allow me access to the student details database, students must be entered in manually.

3. Only 100 questions stored at one time to control database size. As there may be hundreds of students inputting questions, the database may have thousands of questions at one time, which would make it cumbersome to use.

4. As the school already has a functioning email system, it would be unnecessary to recreate this system in my project.

5. As having a function to solve equations could lead to cheating, I will leave this out.

1.7 Data Information

Instead of using an SQL database to store information, as I am using an object oriented system I will use the Pickle function in Python to store my objects and classes as .pickle binary files. This allows for compact storage and easy data storage manipulation within python.

1.7.1 Data Volumes

For a fully up and running version of my system, I anticipate that there will be around 1500 student accounts and 150 teacher accounts. Each of these contains around 30 characters of text, depending on name length and amount of questions set, so are minuscule in size, around 230 bytes each. I anticipate a working library of around 100 questions. Each question contains at most 3 floats and 2 integers so a maximum of 32 bytes each. Totalled up this means the whole volume of the system will add up to around 4.86 MB, this is well within my acceptable limit.

1.7.2 Data Dictionary

Data	Source	Destination	Type
Student Name	Input manually	Student PICKLE File	String
Student ID	Generated by Database	Student PICKLE File	Integer
Student Password	Input manually	Student PICKLE File	String
Student Target Grade	Input manually	Student PICKLE File	String
Teacher Name	Input manually	Teacher PICKLE File	String
Teacher Password	Input manually	Teacher PICKLE File	String
Questions	Input manually	Question PICKLE File	Arrays of Integers and Strings

1.8 Object Orientated Programming Planning

In my program, I have used decided to use object orientated programming as it will easily allow me to create many questions and users, as well as easily manage them.

I intend to have a class to store information about Students, this would be called "Student" and contain information like name, form group and so on, all stored as integers. It would also store question objects in an array to allow work to be set to the student. There will also be a class called teacher which would store simple data like name and password. Each of these would also have an attribute called "account type" which would allow the program to distinguish them from each other.

I would then make a class for each question type which would include all of the maths to calculate the correct answer and show the question to the student. They would be numbered with an ID and have a question type attribute. Questions would then be collated into another class called "Question Sets" which will have a name attribute and store question objects in an array.

1.8.1 Student Class

Field Name	Datatype	Description

Username	String	The students username.
Password	String	The sudents password.
Studentid	Int	The students specific ID number.
Forename	String	The students forename
Middlename	String	The students middle name
Surname	String	The students surname
Form	String	The students form
Targetgrade	String	The students target grade
Questionstoanswer	Array	The questions that a student has been set by their teacher.
Answered	Array	A list of all the questions a student has answered
Percentage	Float	The percentage of questions the student has answered correctly
AnsweredCorrectly	Array	A list of questions a student has answered correctly

1.8.2 Teacher

Field Name	Data Type	Description
ID	Int	The ID number of the specific question.
questiontype	String	Tells the program which type of question the
values	Array	An array of numbers that, when the question type is known, the program will use to
self.conc_of_A	Float	A number used in the rate constant calculation
self.conc_of_B	Float	A number used in the rate constant calculation
self.order_of_A	Int	A number used in the rate constant calculation

Self.order_of_B	Int	A number used in the rate constant calculation
Totalpop	int	A number used in the hardy-weinberg calculation.
K	float	A number used in the rate constant calculation
Intermediary	float	A number used in the rate constant calculation
q_sqrdaspop	int	A number used in the hardy-weinberg calculation.

Field Name	Datatype	Description
Account_Type	String	"T" to signify the account type.
Username	String	The teachers username.
Password	String	The teachers password.
Forename	String	The teachers forename
Middlename	String	The teachers middle name
Surname	String	The teachers surname
Pupils	Array	The pupils that a teacher teaches

1.8.3 Question

1.8.4 Question Set

Chapter 2

Design

2.1 Navigation

- Log in menu
  - Main Menu
    - Set Questions *
    - Select Student/s*
    - Select Question/s*
    - See Set Work ^
    - Attempt Set Questions^
    - Practice
    - Finding K Questions
    - Select question from list / Randomly
    - Finding Rate Of Reaction Questions
    - Select question from list / Randomly
    - Messages
      - Open Messages
      - Create Message
        Recipient Content

* = *Teacher account only*
^ = *Student account only*

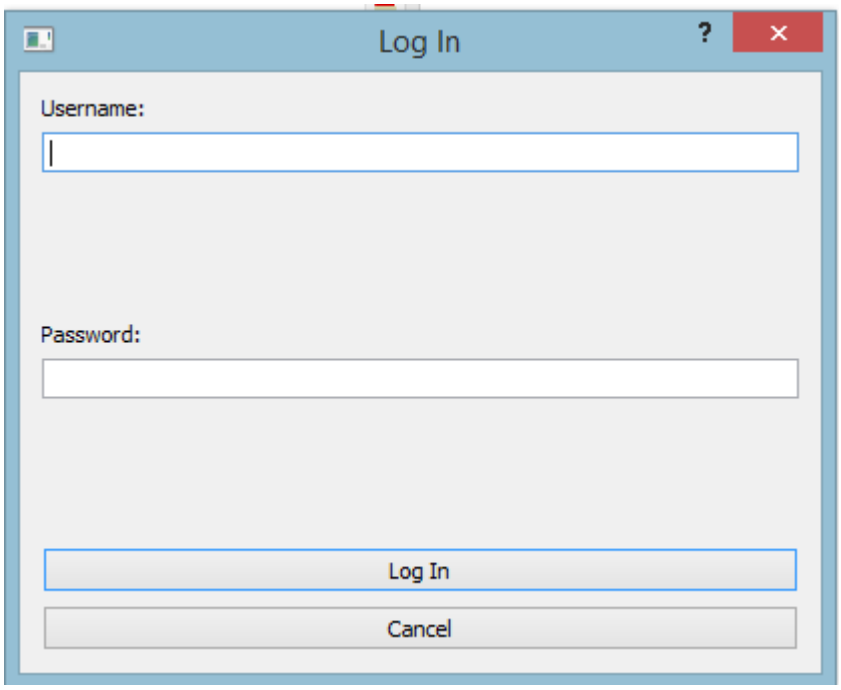


Figure 2.1: Log In Page

2.1.1 GUI Navigation

To make my program more user friendly, and useing user feedback, I decided that a graphical user interface would be more appropriate. These forms show general navigation of the program.

2.2 Validation

Field Name	Validation Checks	Description	Error Message	Data Input
Gender	Listed Data (“M”, “F”)	Only allow M or F	Please select legal gender.	Radio Button

Name	Presence, Data Type.	Only allow names with letter characters, apostrophes or hyphens, make sure a name is present.	Please enter a name. / Please remove invalid characters from field.	F0rd Pr3f3ct
Form Group	Presence, Listed Data.	Allow only listed form groups.	Please select a form group from the list.	Drop Down
Password	Presence, correct password format.	Any string of ascii characters, including numbers and special characters.	Please enter a valid password.	Slartibartfast
Target Grade	A, B, C, D, E, F or U	A single ASCII upper case character.	Please enter a target grade.	A
Parent Email	Correct email format.	String.string'@'string.'string	Please enter a valid email address.	myEmail@email.net

The student class which I have designed incorporates all of the information above, apart from email and gender as it is going to be used at an all-boys school which stores parental contact information securely. Therefore those two data fields would not be appropriate to store in my program. The pupil class accepts input in this format: (studentid, password, forename, surname, middlename, age, form, targetgrade, questionsanswered, questionstoanswer, answeredcorrectly)

The questionsanswered, questionstoanswer and answeredcorrectly values are worked out by other functions in the program and are set to 0 when a new student is created.

To track a students' performance, when they answer questions the percentage() findgrade() functions take their answered questions and find out if they are keeping to their current target grade.

2.3 Overall System Design

Input	Process	Storage	Output
Question Data	Store Question	Question Pickle	Question stored successfully.
Question Data	Answer Question	Question Pickle	Question Answer.
Student Data	Add Student	Pupil Pickle	Pupil Data stored successfully.
Question ID, Student ID	Set Question	Pupil Pickle	Questions sent to Pupil
Student Answer	Answer Question	Retrieve from answer box in window, check answer. Record if wrong or right. Send to pupil pickle.	Question answered in-/correctly. OR Answer Sent.
Student/Teacher Password	Log In	Retrieve from Pupil / Teacher pickle	Logged in as "person" OR Username password combination is incorrect.

Table 2.1: System Process Input / Output

2.4 Record Structure

I plan to use pickle tool to store my objects as binary files.

The pickle module implements binary protocols for serializing and de-serializing a Python object structure. "Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

Pickle accepts inputs in this format: pickle.dump(obj, file, protocol,)

As an example:

```
class container():
    def __init__(self, name):
        self.objects = []
        self.name = name

    def sayName(self):
        return self.name

    def listObjects(self):
        for thing in self.objects:
            print (thing.sayName())

    def addObject(self,item):
        self.objects.append(item)

with open('data.pickle', 'wb') as f:
                        pickle.dump(box, f, pickle.HIGHEST_PROTOCOL)
```

^ Stores class and instance as a binary file (.PICKLE)

```
with open('data.pickle', 'rb') as f:
```

The protocol version used is detected automatically, so we do not have to specify it.

```
    box = pickle.load(f)
```

^ Reads in the information from the file and decodes it.

2.5 Data Transformation

Hardy - Weinberg
Uses the Hardy-Weinberg equation to find 2pq. Two variations of the class are used. One makes random numbers and one uses values from a question class.

Pseudocode 1

```
FUNCTION calculateAnswer(self):
    totalpop <- totalpop
    recessiveInpop <- recessiveInpop
    q_sqrdaspop <- Round_To_n(recessiveInpop, 1) # now rounded
    qsqrd <- Round_To_n(q_sqrdaspop / totalpop, 2)
    # Finds q squared as a decimal.
    OUTPUT "recessives", q_sqrdaspop
    OUTPUT "qsqrd", qsqrd
    q <- Round_To_n(math.sqrt(qsqrd), 2)
    # finds the number of reccessive alleles
    p <- Round_To_n(1 - q, 3) #finds the number of dominant alleles
    psqrd <- Round_To_n(p * p, 3) #finds the number of homozygous dominant
    OUTPUT "psqrd", psqrd
    pq <- Round_To_n(p * q, 3) # finds heterozygotes
ENDFUNCTION
```

Pseudocode 2

```
FUNCTION generate_H_W(self):
     totalpop <- random.randint(10,1000)
    IF  totalpop % 2 = 0:
         percentageReccessive <- random.randrange(0, 99, 2)
    ELSE:
         percentageReccessive <- random.randrange(1, 99, 2)
    ENDIF
     calculateAnswerfromRand()
     updateForm(hardy_weinberg)
ENDFUNCTION
```

```
    FUNCTION calculateAnswerfromRand(self):
        percentageReccessive <-  percentageReccessive
        totalpop <-  totalpop
        recessiveInpop <- ((totalpop * percentageReccessive
            ) / 100)
        #unrounded.Makes a percentage of the population
            have recessive alleles.
        q_sqrdaspop <- Round_To_n(recessiveInpop, 1)  # now
             rounded
        qsqrd <- Round_To_n(q_sqrdaspop / totalpop, 3)
        # Finds q squared as a decimal.
        OUTPUT "recessives", q_sqrdaspop
        OUTPUT "qsqrd", qsqrd
        q <- Round_To_n(math.sqrt(qsqrd), 3)
        #finds the number of reccessive alleles
        p <- Round_To_n(1 - q, 3)
        #finds the number of dominant alleles
        psqrd <- Round_To_n(p * p, 3)
        #finds the number of homozygous dominant
        OUTPUT "psqrd", psqrd
        pq <- Round_To_n(p * q, 3) # finds heterozygotes
    ENDFUNCTION
```

Python Code 1

```
def calculateanswer(self)
 recessiveInpop = ((totalpop * percentageReccessive) / 100)
 # unrounded.Makes a percentage of the population have
    recessive alleles.

 q_sqrdaspop = Round_To_n(recessiveInpop, 1) # now rounded

 qsqrd = Round_To_n(q_sqrdaspop / totalpop, 3) # Finds q
    squared as a decimal.

 q = Round_To_n(math.sqrt(qsqrd), 3)
 # finds the number of reccessive alleles and rounds

 p = Round_To_n(1 - q, 3)#finds the number of dominant
    alleles and rounds

 psqrd = Round_To_n(p * p, 3)#finds the number of
    homozygous dominant

 pq = Round_To_n(p * q, 3) # finds heterozygotes
```

Python Code 2

```
def generate_H_W(self):
```

```
        self.totalpop = random.randint(10,1000)
        if self.totalpop % 2 == 0:
            self.percentageReccessive = random.randrange(0,
                99, 2)
        else:
            self.percentageReccessive = random.randrange(1,
                99, 2)
        self.calculateAnswerfromRand()
        self.updateForm(hardy_weinberg)

    def calculateAnswerfromRand(self):
        percentageReccessive = self.percentageReccessive
        totalpop = self.totalpop
        recessiveInpop = ((totalpop *
        percentageReccessive) / 100)  # unrounded.
        Makes a percentage of the population have
        recessive alleles.
        q_sqrdaspop = Round_To_n(recessiveInpop, 1)
        #now rounded
        qsqrd = Round_To_n(q_sqrdaspop / totalpop, 3)
        #Finds q squared as a decimal.
        print("recessives", q_sqrdaspop)
        print("qsqrd", qsqrd)
        q = Round_To_n(math.sqrt(qsqrd), 3)
        #finds the number of reccessive alleles
        p = Round_To_n(1 - q, 3) #finds the number of
            dominant alleles
        psqrd = Round_To_n(p * p, 3) #finds the number of
            homozygous dominant
        print("psqrd", psqrd)
        pq = Round_To_n(p * q, 3) # finds heterozygotes
        print("p", p, "q", q)

        print("2pq", pq * 2)
```

Find k

Takes values sent by random generation or from a question and finds the answer. It then will update the question form.

Psuedocode

```
FUNCTION findK(self, window):
            QtGui.QApplication.processEvents()
            n <- 0
            intermediary <- pow( conc_of_A,
            order_of_A) * (pow( conc_of_B,
```

```
            order_of_B))
             k <-  rateOfReaction /  intermediary
            correct <- False
            window.label.setText(_translate("Form", "The
               conc of reactant A is"
            + str("%.2E" %  conc_of_A) +
            " moldm^-3. The conc of B is " + str("%.2E" %
            conc_of_B) + " moldm^-3\n" "The order of A is "
                + str(order_of_A) + " AND the order of B
               is " + str( order_of_B) + ". The rate of
               reaction is " + str("%.2E" % rateOfReaction
               ), None))
ENDFUNCTION
```

Python Code

```
def findK(self, window):

        QtGui.QApplication.processEvents()
        n = 0
        self.intermediary = pow(self.conc_of_A,
        self.order_of_A) * (pow(self.conc_of_B,
        self.order_of_B))
        self.k = self.rateOfReaction/self.intermediary
        correct = False
        window.label.setText(_translate
        ("Form", "The conc of reactant A is " + str("%.2E"
           %
        self.conc_of_A) +
        " moldm^-3. The conc of B is "
        + str("%.2E" % self.conc_of_B) + "moldm^-3\n"
        "The order of A is " + str(
        self.order_of_A) + " and the order of B is "
        + str(self.order_of_B) + ". The rate of reaction is
             " + str("%.2E" %
        self.rateOfReaction) + " Find k of this
        reaction.", None))
```

2.6 Class Diagrams

As I have decided to use objects in my program and have finalised how I think they should work, I have made inheritance diagrams that contains all of my classes.

2.6.1 Student

Figure 2.2: The Student Class

Field Name	Datatype	Description
Account Type	String	“S” to signify the account type.
Username	String	The students username.
Password	String	The sudents password.
Studentid	Int	The students specific ID number.
Forename	String	The students forename
Middlename	String	The students middle name
Surname	String	The students surname
Form	String	The students form
Targetgrade	String	The students target grade
Questionstoanswer	Array	The questions that a student has been set by their teacher.
Answered	Array	A list of all the questions a student has answered
Percentage	Float	The percentage of questions the student has answered correctly
AnsweredCorrectly	Array	A list of questions a student has answered correctly

Method Name	Description
__init__	Stores all of the variables sent to the class when an object is made.
percentage	Finds the percentage of questions the student has answered correctly.
findgrade	Compares the percentage of questions answered correctly with a set of standards to find the students grade.
questionAnswered	Removes a question from a students “questionstoanswer” once it has been correctly answered.

Figure 2.3: The Teacher Class

Figure 2.4: Universal Question Class

2.6.2 Teacher

To make my code more concise I made a universal question window class, along with individual classes for each type of question. This class would be initialised every time a student wanted to do a question that they had been set, then make an individual object from the class that the question was.

2.6.3 Universal Question Window

Field Name	Data Type	Description
ID	Int	The ID number of the specific question.
questiontype	String	Tells the program which type of question the
values	Array	An array of numbers that, when the question type is known, the program will use to
self.conc_of_A	Float	A number used in the rate constant claculation
self.conc_of_B	Float	A number used in the rate constant claculation
self.order_of_A	Int	A number used in the rate constant claculation
Self.order_of_B	Int	A number used in the rate constant claculation
Totalpop	int	A number used in the hardy-weinberg calculation.
K	float	A number used in the rate constant claculation
Intermediary	float	A number used in the rate constant claculation

q_sqrdaspop	int	A number used in the hardy-weinberg calculation.

Method	Description
__init__	Stores all of the variables sent to theclass when an object is made
Kquestion	Sets the numbers from "values" to the correct variables for the rate constant question.
hardy_weinberg	Sets the numbers from "values" to the correct variables for the Hardy-Weinberg calculation.
Findk	Changes the text on a page to the rate constant question text, including the numbers from "values".
HWwindow	Changes the text on a page to the hardy-Weinberg question text, including the numbers from "values".

2.7 Security and Integrity

As Python is an interpreted language, students could in theory open it as a text file. An easy way around this would to use the current controls in schools to not allow students access to the program files. However the program requires read write access to files in order to work. Also pickle files are binary files so they are not easily human readable.

However data can be backed up to a secure location once a week. As the files are very small, each time the data is backed up it can be assigned a date.

2.8 Prototype

I showed my end user an initial skeleton program that I had created. I responded to their requests and added a function to give the user an option to immediately have another question also, when their answer is incorrect then they are told it was incorrect and are asked to try again. With further testing I thought the user should be able to have the program to take them through the problem, as it is only a practice.

2.8.1 Command Line Prototype Code

```
author = 'Oscar'
import random
import decimal

def getChoice():
    choice = (input("Please choose an option. "))

    if choice == "1":
        print("  ")
        loadQuestion()
    if choice == "2":
        pass
    if choice == "4":
        pass
    return (choice)


def menu():
    choice = 0


    while choice != "q":
        print("1.Do a practice K question.")
        print("2.See a K example. ")
        print("3. Find units of K. ")
        print("4. Calculate the initial rate of reaction.")
        print("Press q to quit. ")
        choice = getChoice()


        def format_e(n):
            a = '%E' % n
            return a


        class findKreaction():
            def __init__(self, conc_of_A, conc_of_B,
             order_of_A, order_of_B, rateOfReaction):
                self.conc_of_A = conc_of_A
                self.conc_of_B = conc_of_B
                self.order_of_A = order_of_A
                self.order_of_B = order_of_B
                # self.rateNumber = rateNumber
                # self.rateExponent = rateExponent
                self.rateOfReaction = rateOfReaction
                self.noOfMoles = (order_of_A + order_of_B)
```

```
        def walk_throughK(self):
            print("To calculate the rate constant, k,
               we must first
             multiply the concentrations of our
                reactants together."
                  "However the concentrations must be
                     raised to the power
             of their orders."
                  "The concentrations of A and B are",
                     self.conc_of_A, "and",
             self.conc_of_B, "respectively."
                  "The order of A is", self.order_of_A,
                      "and the order of B is",
             self.order_of_B, ".")
            print("This means our intermediary value is
               ", self.intermediary, "."
                  "We then divide our rate of reaction
                     by this number. The reate
             of reaction is", self.rateOfReaction,".")
            print("This equals", self.k)
            print("Next we must find the units of
             k")
            print("To find the units of k we must
               cancel the number of moles

    per dm cubed on both sides of the equation. "
                  "We do this by adding the powers of
                     the two reactants
                  and putting them into the formula:
                     mol^-x dm^+x*3.
                   Where x is the powers added together
                       minus 1."
                  "Unless the number of moles is
                   one, then there are no units.")
            if self.noOfMoles > 1:
                x = int((self.noOfMoles) - 1)

                print("The units are mol^-", x, "dm^+",
                    3 * x, "s^-1")
            else:
                print("No units")



        def findK(self):
            n = 0
```

```
self.intermediary = pow(self.conc_of_A,
    self.order_of_A) *
(pow(self.conc_of_B, self.order_of_B))
self.k = self.rateOfReaction / self.
    intermediary
correct = False
print("The conc of reactant A is ", self.
    conc_of_A, "moldm^-3.
The conc of B is", self.conc_of_B, "moldm
    ^-3")
print("The order of A is ", self.order_of_A
    , "and the order of B
is", self.order_of_B)
print("The rate of reaction is", self.
    rateOfReaction)
while correct == False:
    k = self.k
    k = ("%.3G" % (k))


    studentAnswer = input("Enter in your
        value of k
    to 3 sig fig: ")
    if studentAnswer == k:
        correct = True
        print("Well done, that was correct!
            ")
        another = input("Would you like to
            try again? y/n")
        if another == "y":
            print("  ")
            loadQuestion()
        if another == "n":
            print("Goodbye!")
    if studentAnswer != k:
        n += 1
        print("Sorry that's not right, try
            again.")
        if n > 2:
            walkthrough = input("Would you
                like a walk-through?
                  y/n")
            if walkthrough == "y":
                print("  ")
                findKreaction.walk_throughK
                    (self)
```

```
            def findunitsofK(self):
                print("To find the units of k we must
                   cancel the number of moles
                per dm cubed on both sides of the equation.
                    "
                      "\We do this by adding the
                       powers of the two reactants and
                       putting them into the formula: "
                      "\nmol^-x dm^+x*3. Where x is
                       the powers added together minus 1."
                       "\nUnless the number of moles is one
                          , then
                        there are no units.\n ")
                if self.noOfMoles > 1:
                    x = int((self.noOfMoles) - 1)

                    print("The units are mol^-", x, "dm^+",
                        3 * x, "s^-1")
                else:
                    print("No units")


                    # def getRateOfReaction(self):
                    # rateOfReaction = self.k*(pow(10,self.
                        rateExponent))
                    # return rateOfReaction


        def loadQuestion():
            Q1 = findKreaction(0.4, 0.4, 2, 1, 0.00000585)
            Q2 = findKreaction(0.2, 0.3, 1, 2, 0.0000657)
            Q3 = findKreaction(0.2, 0.3, 1, 2, 0.0000257)
            Q4 = findKreaction(0.2, 0.3, 1, 2, 0.0000457)
            Q5 = findKreaction(0.2, 0.3, 1, 2, 0.0000127)
            Q6 = findKreaction(0.2, 0.3, 1, 2, 0.0100687)
            Q7 = findKreaction(0.2, 0.3, 1, 2, 0.0000757)
            Q8 = findKreaction(0.2, 0.3, 1, 2, 0.0000357)
            Q9 = findKreaction(0.2, 0.3, 1, 2, 0.0000257)
            Questions = [Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9
                ]
            x = random.randrange(9)
            y = Questions[x]
            findKreaction.findK(y)
            # findKreaction.findunitsofK(y)


        menu()
\end{verbatim}
```

```
\clearpage
\textbf{Screenshots of CLI Prototype:}\\
\bigskip
\begin{figure}[h!]
\centering
\includegraphics{CLI01}
\caption{CLI Main Menu}
\label{fig:CLImm}
\end{figure}

\begin{figure}[h!]
\centering
\includegraphics{CLI02}
\caption{Attempting a question in CLI}
\label{fig:CLIQ}
\end{figure}

\begin{figure}[h!]
\centering
\includegraphics{CLI03}
\caption{Entering an Incorrect Answer}
\label{fig:my_label}
\end{figure}


After this I decided that if the student was only
   practising questions, they should not use up questions
   that teachers are marking them on. Therefore I created
   a function that generates random numbers, in a specific
    range, and rounds them to suit the question. This way,
    a student will never have encountered the questions
   they would be assessed on.

\begin{lstlisting}


def generateK():
    sig = 3


    concofa = random.uniform(0.001, 0.9)
    concofA = round(concofa, sig-int(floor(log10(concofa)))
       -1)
    concofb = random.uniform(0.001, 0.9)
    concofB = round(concofb, sig-int(floor(log10(concofb)))
       -1)
    orderofA = random.randrange(1,3)
    orderofB = random.randrange(1,3)
    rate = random.uniform(0.00000001, 0.0009)
```

```
    rateofreaction = round(rate, sig-int(floor(log10(rate))
        )-1)
    y = findKreaction (concofA, concofB, orderofA, orderofB
        , rateofreaction)
    return y
```

This generates the numbers for the equation, creates a class using them, then returns it to the main program.
After consulting my end user again, and by using their request of something "MyMaths"-esque, I have implemented a GUI into my design using PyQt, which is event driven. This improves the ease of use of the program and also makes it more user friendly for people who do not use command line regularly.

2.8.2 GUI Prototype

To prototype the GUI I hand drew some of my windows and annotated functions:

Figure 2.5: Drawing of Log In page

Figure 2.6: Drawing of Attempt Question page

Figure 2.7: Drawing of Main Menu

Figure 2.8: Drawing of Student Performance page

I also made an intermediate main menu before I decided on the best layout;

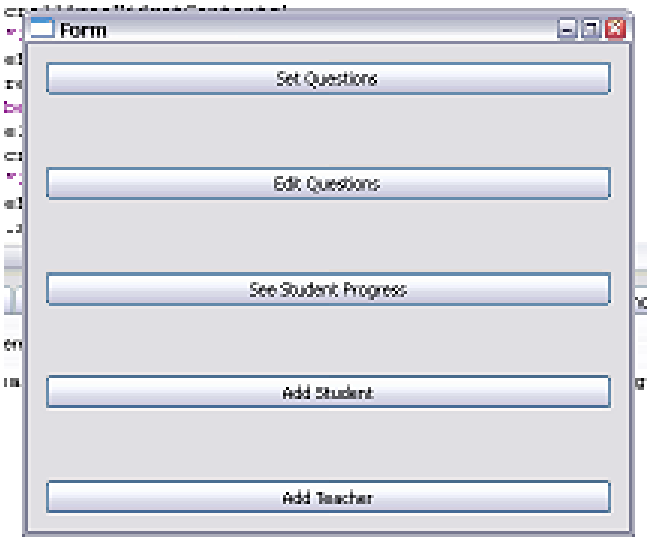


Figure 2.9: Early teacher main menu

As part of my program, there is a form for the input of new users to the system. This window sends its' variables to a function which adds the new student and re-pickles the list. This function is only available to teacher accounts. They can create new teacher accounts and new student accounts.

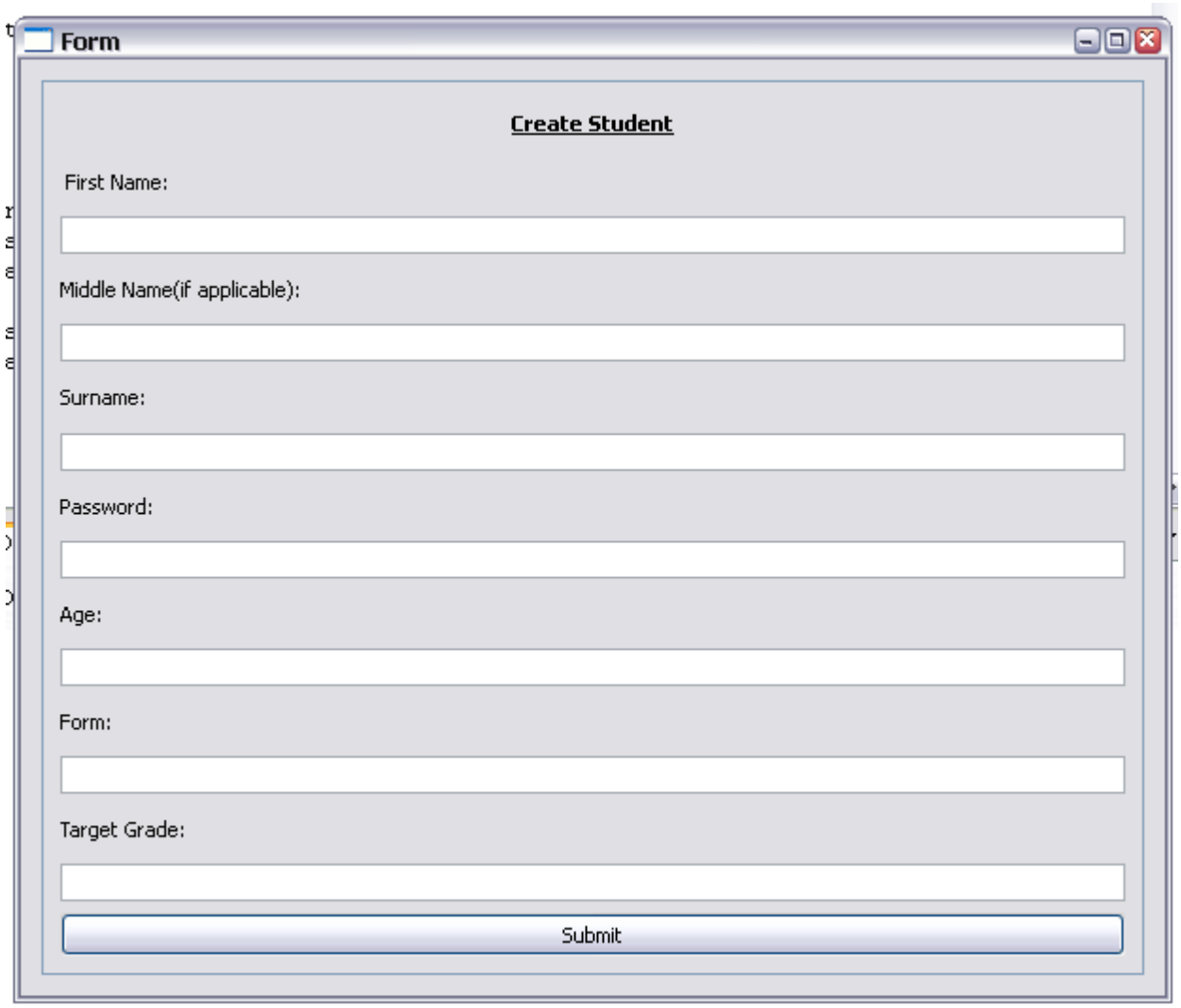


Figure 2.10: The initial create student window

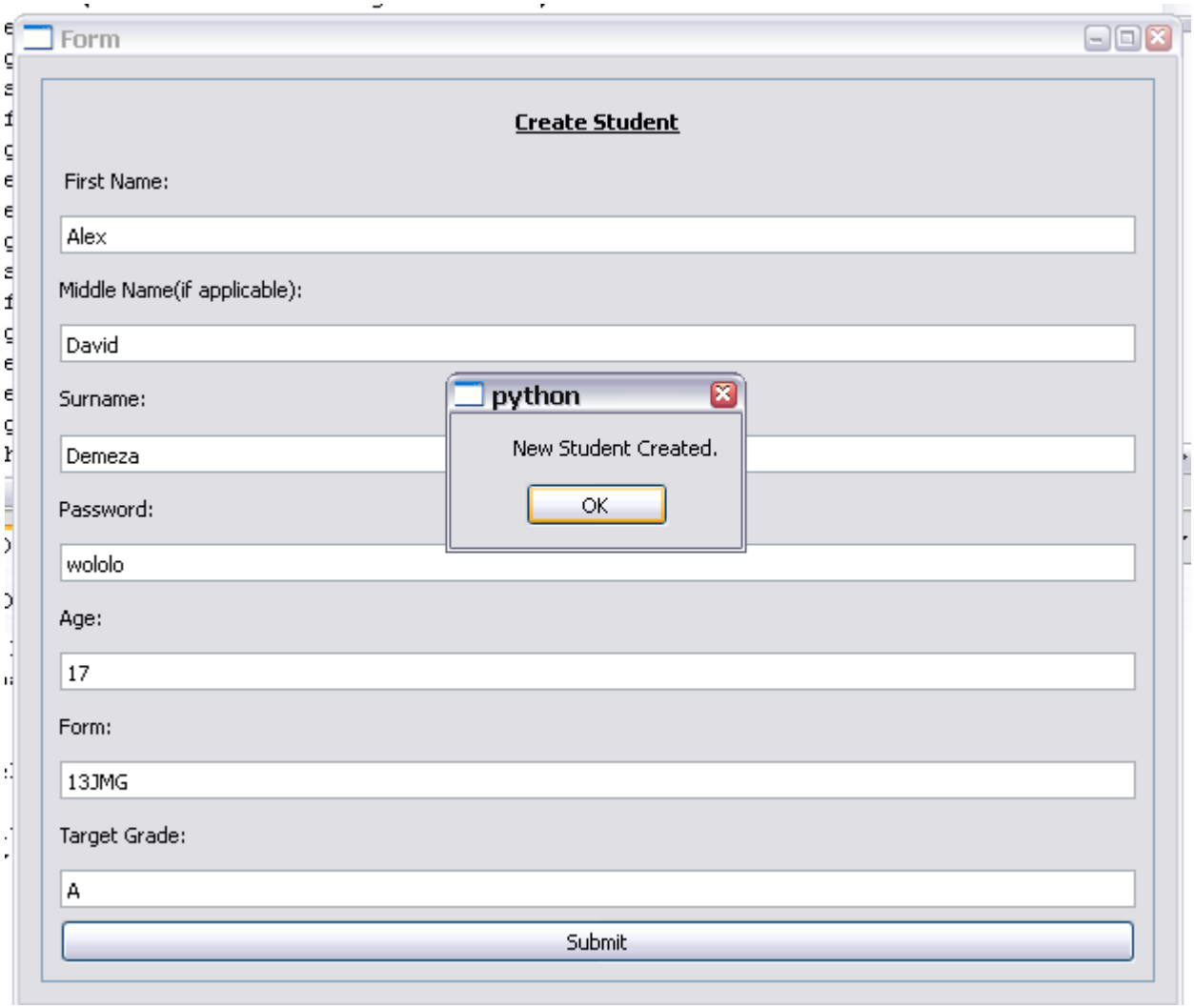


Figure 2.11: An account being created

PyQT

To create my GUI I used a python package called "PyQT". This comes with a designer app which provides a drag and drop interface to place buttons, boxes, labels etc onto a form, then export them to a python file.

SCREENSHOTS HERE

The python code looks like this:

```
    def retranslateUi(self, Form):
        Form.setWindowTitle(_translate("Form", "Logged in
            as " + username, None))
        self.pushButton_3.setText(_translate("Form", "Set
            Questions", None))
        self.pushButton_2.setText(_translate("Form", "Add
            Questions", None))
        self.pushButton.setText(_translate("Form", "See
            Student Progress", None))
        self.pushButton_4.setText(_translate("Form", "Add
            New Account", None))
        self.editAcc.setText(_translate("Form", "Edit
            Accounts", None))
        self.pushButton_6.setText(_translate("Form", "Make
            QSet", None))
        self.pushButton_5.setText(_translate("Form", "Log
            Out", None))
        self.pushButton_4.clicked.connect(self.
            showCreateNewAccountWindow)
        self.pushButton_5.clicked.connect(self.close)
        self.pushButton_5.clicked.connect(self.
            showLoginPage)
        self.pushButton_2.clicked.connect(self.showNewq)
        self.pushButton_3.clicked.connect(self.
            showSetQuestions)
        self.pushButton_6.clicked.connect(self.makeQSet)
        self.editAcc.clicked.connect(self.openEditStudents)
```

2.9 Test Strategy

Chapter 3

Testing

To test my program I will use a combination of black-box and white-box testing. However to get the most information and therefore tackle bugs or improve the efficiency of my code, I will use mostly white box testing. To do this, I will include print commands into my *hardy_weinberg* and *Practice_K_Question* classes to see if the calculations are being done correctly, and to make sure that the program can cope with the values given. I will also test if my navigation works, if user account creation has acceptable validation.

known error of overflow with certain values in rate constant equations, combated by testing in question creation.

```
 try:
     pow(FirstVal, ThirdVal) * (pow(SecondVal, FourthVal))

except:
      reply = QtGui.QMessageBox()
      reply.setText("Please use smaller concentrations.")
      reply.setIcon(3)
      reply.addButton(QtGui.QPushButton('OK'), QtGui.
         QMessageBox.YesRole)

      ret = reply.exec_()
      valid = False
```

This means that if a number will cause an overflow error the program will inform you and not store these values.

know error of rounding for Hardy Weinberg because rounding in python goes for the nearest even value.

test for no pickle files.

3rd party testing