AMPLEFORTH COLLEGE

Diego Jimenez Lopez

St Cuthbert's (Y13)

A Level Latin Revision

Latin revision program

Analysis:

Introduction:

Outline description:

A revision program designed to aid teachers of classics to add questions to a database which are then selected for the students in that teacher's class. The student is then presented these questions which they answer after they have selected their chosen topic and then the questions are marked by the teacher and the results are stored in a database so both the student, the parent and the teacher can view graphs based on these scores and their meaning (such as whether the student is on target) and parents can communicate with their child's teacher. Data will be stored on a relational database.

This program is going to be centered around the fact that the decline in number of schools taking Latin has led to a decrease in revision resources for Latin, thus decreasing its popularity.

However, as I will be making a revision program, I will include the option to answer revision questions on other subjects that are not just GCSE, so it is easy to revise and add questions on other subjects.

Latin has been decreasing in popularity since it became no longer spoken. As a result, the number of schools offering Classics has been decreasing. In 1988, 16,023 students were entered for GCSE Latin. This fell to 13,408 in 1992 and 10,561 in 2000. 'Interest in Latin has grown but not to exam level'. This means that despite the government's attempt to encourage schools to teach Latin and 'the number of secondary schools teaching Latin has doubled over the last seven years', Latin may still disappear in schools as statistics show the increase in the study of Latin pre-GCSE, not GCSE and A-Level. Therefore, the changes could not be maintained in the long run as the number of A-Level candidates is significantly smaller compared to those entered for GCSE. This may have been caused by fashion/ trends rather than government intervention because students are much more likely to choose a degree if it will help for employment opportunities, not just pure interest (although passion for a course is obviously crucial). But there is always the risk that classical languages may just become an enrichment (or an extra) as opposed to an academic subject, which may ultimately cause its disappearance in all schools.

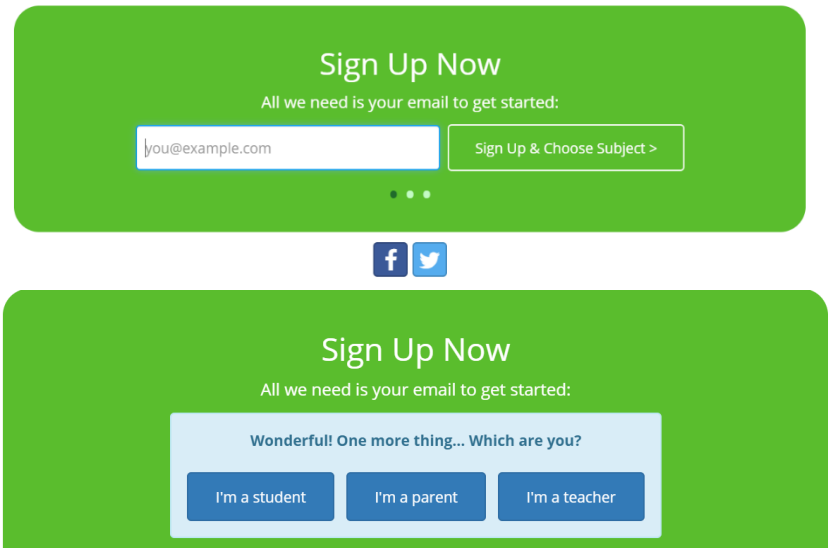
As a result of the decline in the number of schools offering Latin, the number of post-GCSE applicants will decrease because, as many schools do not offer Latin nor Greek, when students move to a different sixth form and wish to study Latin, they are limited because their previous school did not offer it, and they will not necessarily be able to afford moving far away just for one subject. So, there should not be a limit imposed to study a subject if someone really is passionate just because the subject in unpopular. "GCSE Latin is extremely hard work", students willing to put in the effort should at least be given the chance. "When confronted with Latin or German they will choose German", this is because many students would get the

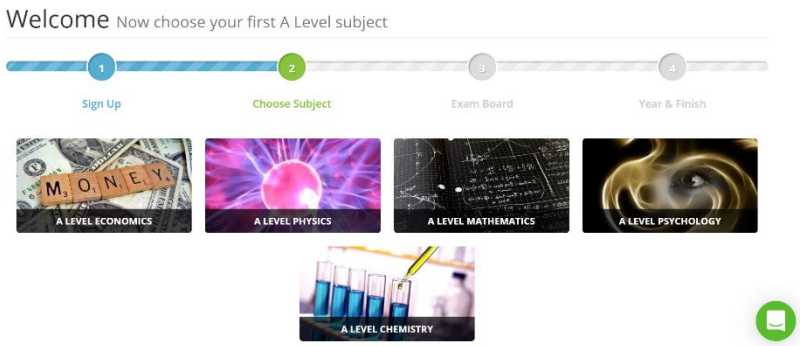
impression that Latin will not be beneficial for employment (which is wrong) but it is one of the causes contributing to the neglecting of classical languages.

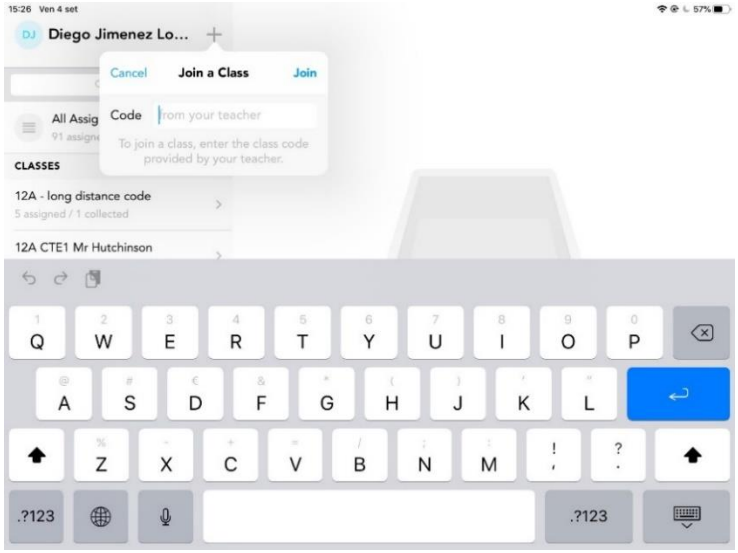
Therefore, the focus of the program is only on Latin as an academic subject (GCSE and A-Level) in a way that is user-friendly to encourage more students to study Latin, which will increase demand for the subject thus increasing its importance in schools making classical languages be valuable subjects and not simply 'dead languages'. Other than encouraging the study and reinforcing the importance of Latin, this program is designed to help students to revise so it is a revision tool (focused on Classics). To avoid limiting the program to one subject only, there will be a table in the database called 'Subject' which will store the other subjects that could potentially be included in the program, such as Greek or Classical Civilization or Ancient History.

Upon researching existing software that helps students with revision, I found a website called UpLearn.co.uk.

Existing software:







In this example, the student is prompted for the class code, and they can access the questions provided by the teacher, however, it could be modified so that there is an unlimited number of questions a student can answer.

I am making a revision program, so the questions which are added by the teacher will not necessarily be viewed by the student immediately or have a due date. Questions are generated at random so with many questions, there may be one question or more which will take a long time to be answered. Therefore, this app is not designed for a question which is meant to be answered in the short run (although, as each question belongs to one topic, this may only increase the probability of a specific question to be displayed), but the program should aid students with revision, not select induvial questions.

End users:

GCSE/ AS/ A-Level students studying Classics:

To revise by answering questions on a chosen topic and then get recommended on which areas they should work on after questions have been answered. A section will allow the student to review if the tip was successful. Other student's scores can be seen but not their names (only ID, score from lowest to highest).

Teachers teaching Classics:

To know how much progress the student is making (graphs) and know patterns in the questions the student needs more help with. Also, the ability to add more questions if appropriate and then they can review specific answers that will be saved in a database.

Parents:

Can track their son/daughter's progress in Latin with automated messages sent by teachers based on the level of progress made by the student.

Description:
Present a series of questions of increasing difficulty as the user gets them right and store the result in a relational database to help teachers keep track of where the student is at in terms of progress and give sanctions or rewards accordingly. Teachers can add questions which will be asked when the student logs in.

User needs:

Student:
The app should be engaging and user-friendly to encourage more people to study classics.

Teacher:
Should be able to see clearly how each student in their class is making progress and could potentially get recommendations for which topics they are most struggling with (more for GCSE).

Initial questions:

Is there a user that does not need to login?

No, each user has to login and register. However, there could be a supervisor such as a technician who does not need login details.

Where will data be stored?

C# Windows forms application will interact with SQL database where data will be stored/ text file.

Can the student answer a set of questions on more than one topic at once?

No, the student will select one topic for which the questions will be on, each time.

How will the questions be asked?

Some of the shorter answer questions such as single sentences will be marked by the system by comparing the answer in the text box to the one in the database. Others, such as the longer questions, will be multiple choice, if the questions are not multiple choice, the student may request the question to be checked by the teacher.

How many questions will be answered at one time?

This depends on the number of questions per exercise and the length of the question and topic.

Can the teacher view the specific answer given by each student and be able to mark it?

Yes

Is it important, which teacher added which question?

No, the questions are added and may be answered by students which are not taught by the teacher who added the question, and the system does not know who they were added by.

Which questions are selected from the database when the form of the user teacher to mark students' work is loaded?

Questions are selected based on the student id.

List of objectives:

1) Have multiple logins for the same application → more than one user can register and login on the same platform using different credentials
2) Be able to prevent a user from registering if they have not entered the information required for the specific type of user even if they have chosen a username and password
3) Prompt the student user for the topic they are finding the hardest and select questions from the database based on the topic chosen
4) Be able to type the answer to each question in windows forms by changing the label every time the student has finished answering the question while inserting the answer given to the question which corresponds to it
5) The teacher can mark the answers given by the student by inserting the number of marks and is able to see the original answer
6) Draw graphs on windows forms based on the scores (marks awarded) given by the teacher for each student in one class given teacher id
7) Store relevant information about users such as foreign keys in relevant link tables in sql based on user input (eg. data should be stored in link table student subject after a student has selected the subject)
8) Use a recursive algorithm to ask the student if they wish to add another subject and keep adding the new information to the database without affecting the previous details
9) Use statistics such as probability density function for the normal distribution given the parameter of the marks a student has achieved
10) Use aggregate sql functions to display results on windows forms graphs
11) Connect the program to a database with which the program will interact and display information from
12) Store results from sql queries in local and minimal global variables to be used in the program

As the focus of the program is going to be on students answering questions, this objective is secondary as it comes after the main scope of the program. A possible way to make it more secure, however, could be to use encryption (which I won't have time for) or possibly having a department code for teachers (to simulate a real-life example where the teacher would be asked for information that the students will not be able to get (such as a department code or teacher id given by the school, for example, and this would be validated if it were present in

the database as there would be a record of all the staff, implying that an administrator would be behind the scenes of the application). So, the department codes, in this case, are going to be added manually to the database (not by the user), to keep track of who can set predicted grades for students.

The department code will therefore be added by the administrator which means that the app is aimed towards a particular school and not one which is used by multiple schools at the same time.

Another assumption is that the app will be used by a single school.

When a question has been asked once, the system will not display the question again to avoid answering the same questions which have already been seen.

This application is aimed towards classics students and teachers, so, an assumption is going to be that the subjects used are going to be related to classics (Latin, Greek, Ancient History and Classical Civilization). A way to improve this, could be to add the subject chosen by the teacher during registration to the database if it is not already present in the database rather than just selecting the subject id. However, this is not the main scope of the program, the aim of the program is for the student to answer questions and adapt to previous scores.

The questions I will add to the database will be from the A-Level textbook as I have the answers.

Proposed solution:

Each student has one class code which is set by the teacher and given in class. To avoid limiting the program to Latin only, the database will have a table called subject which contains the student ID and class code for each subject and the students currently studying them, this is so that the program could be extended to other subjects and different levels.

This program is a revision guide to ask students questions (based on the textbook) of Latin GCSE. All questions will be asked in Latin to be translated into English to begin with, questions which are added by the teacher during the academic year can be in English to be translated into Latin. When the user first opens the program, the system will ask the user to login, if they are not registered, there is the option to register, which adds a new profile to the database, or reset the password, which modifies the details in the database. If the user is a student, the system will ask which topic they are struggling with the most (presented as a list checkbox containing the chapters of the textbook and some of the things it will ask (for example: complex sentences), if the student chooses more than one topic, questions will only be on those topics and not on other topics.

As this is an application which has the main user of students and teachers, in a real-world situation, this would mean that the different types of users would need to enter specific details such as class code as this is an application aimed at schools using external resources for revision, therefore, it is assumed that the students will be able to use it hereby providing teachers relevant information regarding their progress. So, in the database, there is an

attribute called 'Class Code' which has the purpose to give teachers relevant information about the students in their class and not the ones from outside classes. When the teacher first registers, they will be prompted for a class code which they can choose (to make this easier for users). When the student wishes to create an account, the system will ask for the teacher's class code, if the student is using this resource independently outside the classroom, the system will still ask if they are sure they wish to continue without class code, they will still be able to view other students' scores as details are about the marks gained by all students within the program. The difference is that there is not a teacher aware of their progress and communication with parent or guardian. These details can be modified after the student has registered successfully so they do not need to create another account.

When the user first opens the program, the system will prompt for username, password and their position at school, if the user does not have an account, they can click on the button 'register' which will open a form asking for name, surname, username, password, mail address and phone number. If the username chosen already exists in the database or one of the fields have been left blank, the system will print an error message. Depending on the type of user (student, parent, teacher) selected in the register page, the system will ask for different details in-order to create an account to be able to login and to do this, a different form will open which will ask for certain details. For example, if the user is a student, the system will ask for parent, teacher name, class code and subjects taken (which can be changed later on as a student may drop a subject or choose a different one). If the user is a teacher, the system will ask for the number of students, subjects taught and if the user is a parent, the system will ask for the number of kids and names and home address.

The teacher will be able to see how the student is making progress on each chapter, not individual aspects of it. For example, chapter 1 will be on cases, but the assumption is that the questions will be answered on the whole of chapter 1.

When a student has logged in successfully, the system will ask which subject they wish to start with (as a list of the subjects currently taken by students will be added to the subject combo box). As the aim of this application is towards GCSE Latin, once the student (in this example) has chosen the subject from the combo box, a form called answer questions will open. Before completing other tasks available to the student, they are prompted to answer a set of questions (each set consisting of ten questions each). Although the number of topics will vary depending on the level, the user should not be allowed to select more than three at a time. The questions asked will all be asked on the same form with a label adapting to the next question which is to be asked. If a question has already been asked, the system will not ask the question again unless the student wishes to answer it more than one time, this will be done by holding a Boolean variable for each question which will become false after the question is answered. If at least one set of exercises has been completed having scored fifty per cent or more, the system will give the option to gain access to their page. While answering questions, the student has the option to ask for help regarding a specific question or more general clues. The student page will consist of viewing previous scores and graphs, viewing other students' scores (only with profile ID not their names (only the teacher can see their names)).

If a parent has logged in, they have the option to view their son's or daughter's progress (graphs) or communicate with the teacher. In the database, the parent table has the

Evidence of analysis (reason I am interested):

Questions for the users regarding the solution:

Student/teacher survey:

What do you need the program to do?

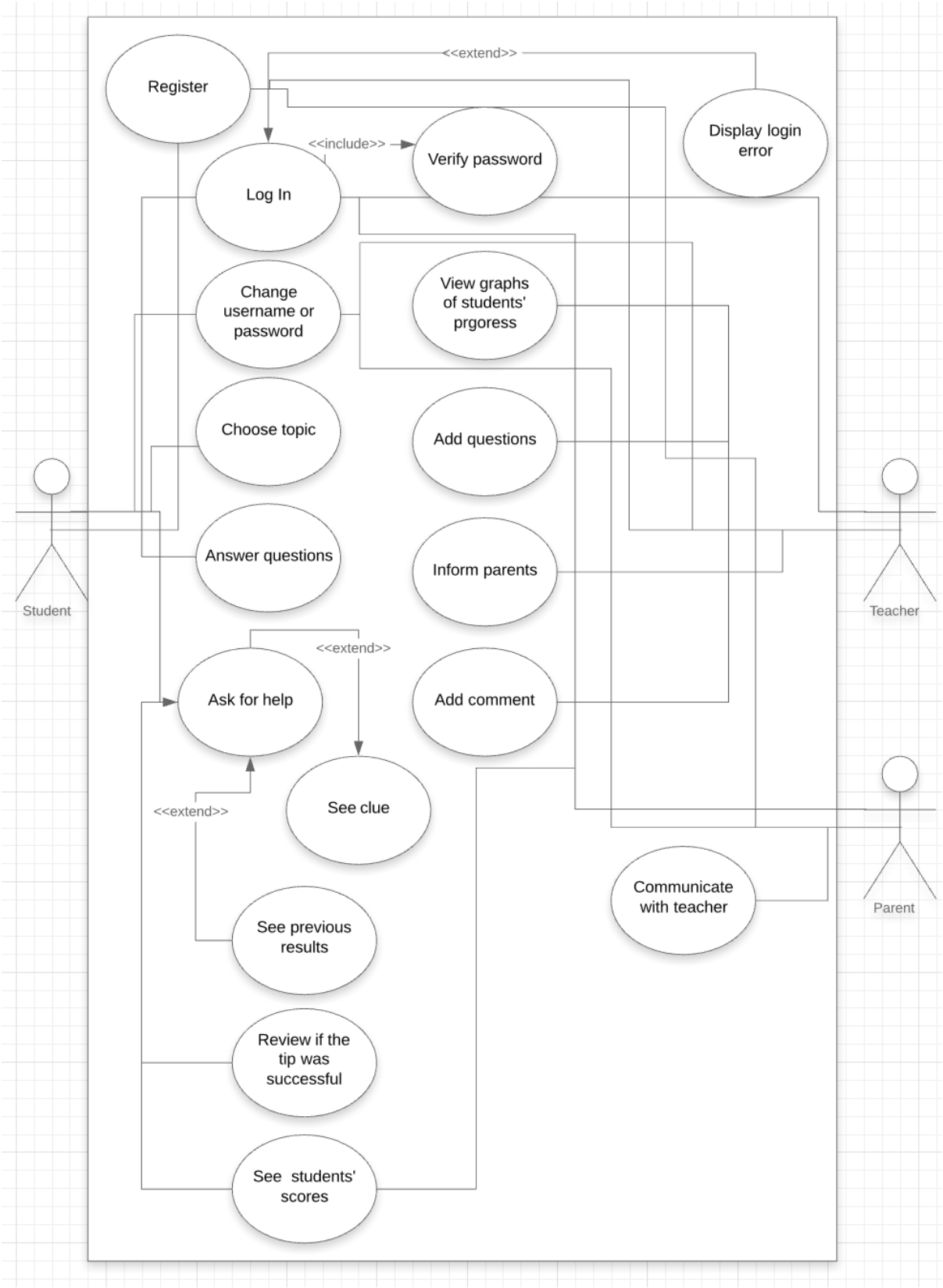
I need to be able to choose a topic and answer questions without having to search for each chapter individually.

Needs to have graphs of marks awarded and should be able to mark each question having the actual answer to it so it can be compared and see probability for normal distribution.

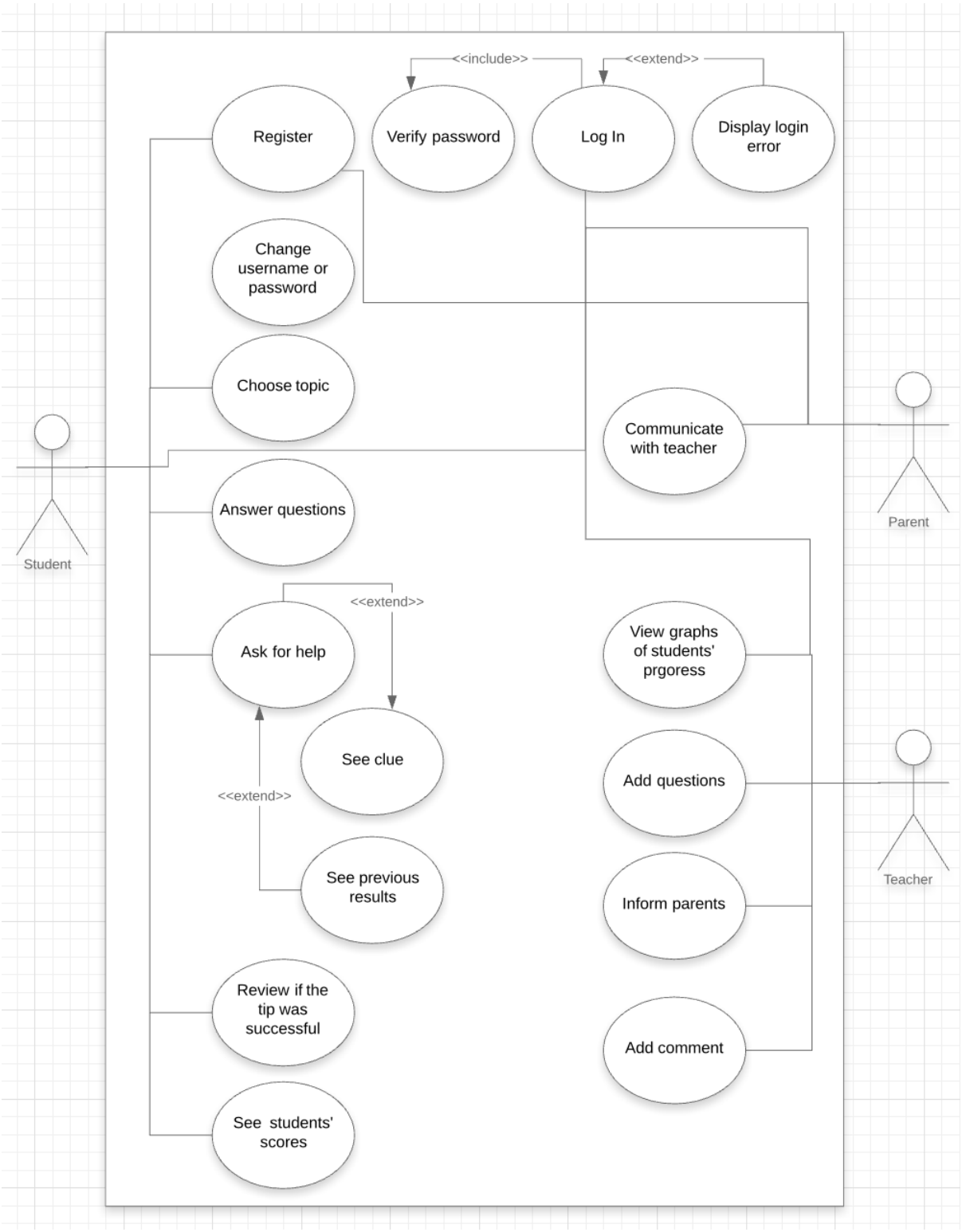
Key concepts/ objectives:

- SQL database connection and interaction
- Converting the values retrieved from the database to a string or int to be used inside the program and the user interface.
- Relational database
- Database normalization
- Exception handling
- Graphical user Interface
- Questions generated at random of increasing difficulty of the topic the student needs more help with, adapting to previous scores.
- Emailing users
- User interface
- Multiple logins
- Telling teacher which topics each student is struggling with by generating graphs.
- Priority queue
- Array lists
- Normal distribution and standard deviation to interpret data (students' scores)

Initial UML to show how each user will use the program:



Refined use case diagram (UML) to show actions of users:


<<include>>
<<extend>>
Register
Verify password
Log In
Display login error
Change username or password
Choose topic
Communicate with teacher
Parent
Student
Answer questions
<<extend>>
Ask for help
View graphs of students' prgoress
See clue
<<extend>>
Add questions
Teacher
See previous results
Inform parents
Review if the tip was successful
Add comment
See students' scores

Analysis dictionaries:

The 3 actors (student, parent, teacher) will have the following tasks to choose from (some just for a particular type of user):

Use Case ID:	1
Use Case Name:	Register
Description:	Option to register
Primary actor:	Student, Teacher, Parent
Preconditions:	User is not registered in the system
Postconditions:	Account is created
Main Flow:	System asks whether they are a student, parent, or a teacher and to choose their username and password.
Alternative Flows:	If the user details are already in the system, they will get an error message.

Use Case ID:	2
Use Case Name:	Log In
Description:	The system asks for their login details. They can login for 3 times, they can reset password if they want to.
Primary actor:	Student, Teacher, Parent
Preconditions:	They need to register before they can login.
Postconditions:	They are logged-in into the system
Main Flow:	- Student, parent, or teacher enters username and password and enters. - System validates and verifies login attempt and prints error message if login details are incorrect. - System displays page according to the type of user (student, parent, teacher) - Use case ends
Alternative Flows:	- Missing username or password 1. System prompts for ID/password. 2. Ask for details again. - Wrong password or username after 3 attempts 1. Display (maximum number of trials) - Invalid username/password

	1. Display “wrong password, try again) 2. System prompts for ID and password

Use Case ID:	3
Use Case Name: (include 2)	Verify password and username
Description:	After entering password, the user has three attempts before they are asked if they need to register or forgot password
Primary actor:	Student, Teacher, Parent
Preconditions:	User has entered login details and registered
Postconditions:	The password and username are verified
Main Flow:	If the user enters correct username and password, they are taken to their set of actions.
Alternative Flows:	If the username and password are incorrect, the system prints an error message.

Use Case ID:	4
Use Case Name: (extend 2)	Display login error
Description:	Error message displayed if login details are incorrect
Primary actor:	Student, Teacher, Parent
Preconditions:	Having entered wrong ID or password
Postconditions:	An error message is displayed.
Main Flow:	None
Alternative Flows:	None

Use Case ID:	5
Use Case Name:	Choose topic
Description:	The student selects which topic they wish to revise.
Primary actor:	Student
Preconditions:	Login successful and they have entered predicted grade.
Postconditions:	The topic has been selected
Main Flow:	The system asks the student which topic they are finding the hardest.

Alternative Flows:	None

Use Case ID:	6
Use Case Name: (include 5)	Evaluate student
Description:	The system works out each of the students' predicted grades after they have answered at least one set of questions.
Primary actor:	Teacher
Preconditions:	The student has answered at least ten questions.
Postconditions:	The teacher can see the students' predicted grades based on the questions they have answered.
Main Flow:	The system calculates each student' predicted grade considering the topic, difficulty and the number of questions answered.
Alternative Flows:	If the student has not completed any questions/ exercises, there will not be a predicted grade.

Use Case ID:	7
Use Case Name:	Answer questions
Description:	Present a series of randomly generated questions of increasing difficulty.
Primary actor:	Student
Preconditions:	Topic has been chosen
Postconditions:	Next question
Main Flow:	The system presents a set of questions (each containing ten) for the student to answer, if one question is left blank, the student gets a warning message asking if they still want to continue.
Alternative Flows:	- Ask for help

Use Case ID:	8
Use Case Name:	Ask for help
Description:	If they are stuck on a question or get it wrong, they can ask for clues. When they answer the questions, the score and topic score is saved.

Primary actor:	Student
Preconditions:	To have put something down for the question (not left it blank)
Postconditions:	The student gets advice on how to answer a question.
Main Flow:	None
Alternative Flows:	None

Use Case ID:	9
Use Case Name: (extend)	See clue
Description:	If a student is stuck on a question, they always have the option to ask for help (look for a clue) depending on the type of question or a general piece of information useful for most questions. For example, some parts of a word (word endings) indicate what type of verb it is which may be used as an indicator for answering questions.
Primary actor:	Student
Preconditions:	At least one question completed, and something is written for current question.
Postconditions:	A clue is shown that can be specific to an individual question or for most questions.
Main Flow:	None
Alternative Flows:	None

Use Case ID:	10
Use Case Name: (extend)	See previous results
Description:	Individual student scores are stored on a relational database for them to view at a later stage.
Primary actor:	Student
Preconditions:	Completing at least one set of questions
Postconditions:	Previous results can be seen
Main Flow:	When the student completes a set of exercises, the results are saved so they can view them later.
Alternative Flows:	None

Use Case ID:	11
Use Case Name:	See students' scores

Description:	A list of the average of each student's scores is an option that can be viewed either by the student or the teacher, only IDs are shown, not their names (in case of students, teachers can see their names).
Primary actor:	Student, Teacher
Preconditions:	Login successful
Postconditions:	The student can see other students' scores only with their ID and the teacher can view their names.
Main Flow:	None
Alternative Flows:	None

Use Case ID:	12
Use Case Name:	Review tip
Description:	At the end of each set of questions (10 or they can end session before to quit) they will be asked if now they understand the topic or feel more confident about answering questions, otherwise they can just not write anything for this section. Reviews are multiple choice and should change the types of questions that are asked in the future. The tips are then saved.
Primary actor:	Student
Preconditions:	Having completed a set of exercises
Postconditions:	Average of student's reports change the type of questions asked in the future.
Main Flow:	The system asks the student if the clue they were shown (if they have not asked for help this will mean, how they have found the exercises in terms of difficulty) and the answer is then saved in the database.
Alternative Flows:	None

Use Case ID:	13
Use Case Name:	View teacher's comment
Description:	The student can view teacher's comment on their progress or how they have answered questions.
Primary actor:	Student

Preconditions:	Having answered at least a set of questions.
Postconditions:	They can view the teacher's comment.
Main Flow:	After they have viewed the comment, it is assumed that they will close the application.
Alternative Flows:	None

Use Case ID:	14
Use Case Name:	View graphs of students' progress
Description:	Graphs indicate how the student is getting on. If the student gets most questions right, the teacher will be advised to ask harder questions
Primary actor:	Teacher
Preconditions:	Login successfully
Postconditions:	The teacher sees the progress the student has been making.
Main Flow:	None
Alternative Flows:	None

Use Case ID:	15
Use Case Name:	Add questions
Description:	If they want to, they can put more questions that will be asked at random, they also must put the answer for the question. The system then checks if the question is already in the database, if the question is similar or worded differently, the system does not recognize it as a different question.
Primary actor:	Teacher
Preconditions:	None
Postconditions:	New question is added
Main Flow:	If the question is already in the system, an error is displayed, otherwise, the question is added
Alternative Flows:	If the question is already present in the database, the system outputs an error message.

Use Case ID:	16
Use Case Name:	Inform parents

Description:	An option will be to send a message to parents in addition to the standard description of their son/daughter's progress.
Primary actor:	Teacher
Preconditions:	Select (inform parents choice or 3)
Postconditions:	Ability to send a message to a parent
Main Flow:	The teacher can send a comment to parents.
Alternative Flows:	None

Use Case ID:	17
Use Case Name:	Add comment
Description:	Leave a comment on how the student is doing which will be displayed on the relevant student's page when they login
Primary actor:	Teacher
Preconditions:	None
Postconditions:	The system sends the comment to the student page.
Main Flow:	None
Alternative Flows:	None

Use Case ID:	18
Use Case Name:	Communicate with teacher
Description:	Send a message to teacher which is shown on their page when they login
Primary actor:	Parent
Preconditions:	Login successful
Postconditions:	Option to communicate with teacher
Main Flow:	Parents can communicate with the teacher.
Alternative Flows:	None

Analysis data dictionaries for each situation:

Login:

Data item	Data type	Validation	Sample data
Username	varchar(100)	datatable.Rows[0][0].ToString() == ("1") (checks if a record with the same information exists in the database)	doughnut

Password	varchar(100)	Same as username	economics
Role (position)	varchar(20)	Same as password	Student

Register:

Data item	Data type	Validation	Sample data
Username	varchar(100)	datatable.Rows.Count >= 1 (checks if username and password already exist once in the database)	doughnut
Password	varchar(100)	Same as username	economics
Role (position)	varchar(20)	Same as password	Student
Name	varchar(40)	Textbox != “”	Jimmy
Surname	varchar(40)	Textbox != “”	Page
Mail	varchar(100)	Contains “@”	example@mail.org
Telephone	varchar(30)	>11	12345678
Class code (student)	int	Textbox != “” and if field exists in database	23950
Parent name (student)	varchar(40)	Textbox != “” and no numbers	Name of parent
Subjects studying (student)	varchar(100)	Textbox!= “”	Computer Science, Economics, Latin
Class code (teacher)	int	<11 && Textbox!= “”	23950
Department (teacher)	varchar(200)	Textbox!= “”	Classics
Number of kids (parent)	int	Must be an int	3

Student logged in:

Data item	Data type	Validation	Sample data
Answer given	varchar(300)	Cannot leave empty	ceteris paribus
Student feedback	varchar(200)	Different to null	Opinion on the app
Student enquiry about questions	varchar(200)	Different to “”	Aspect they have found confusing

Teacher logged in:

Data item	Data type	Validation	Sample data
Difficulty (teacher (regarding new questions))	float	<10 (according to GCSE grades)	6 or 6.2
Topic (teacher)	varchar(100)		Complex sentences
Question name (teacher)	varchar(300)		Translate this sentence: “”
Question answer (teacher)	varchar(300)		exempli gratia
Number of marks (teacher)	int	Range (1,70)	40
Teacher’ comment on student’s progress	varchar(200)		WWW/ EBI

Teacher's communication with parent	varchar(200)		Student x has…

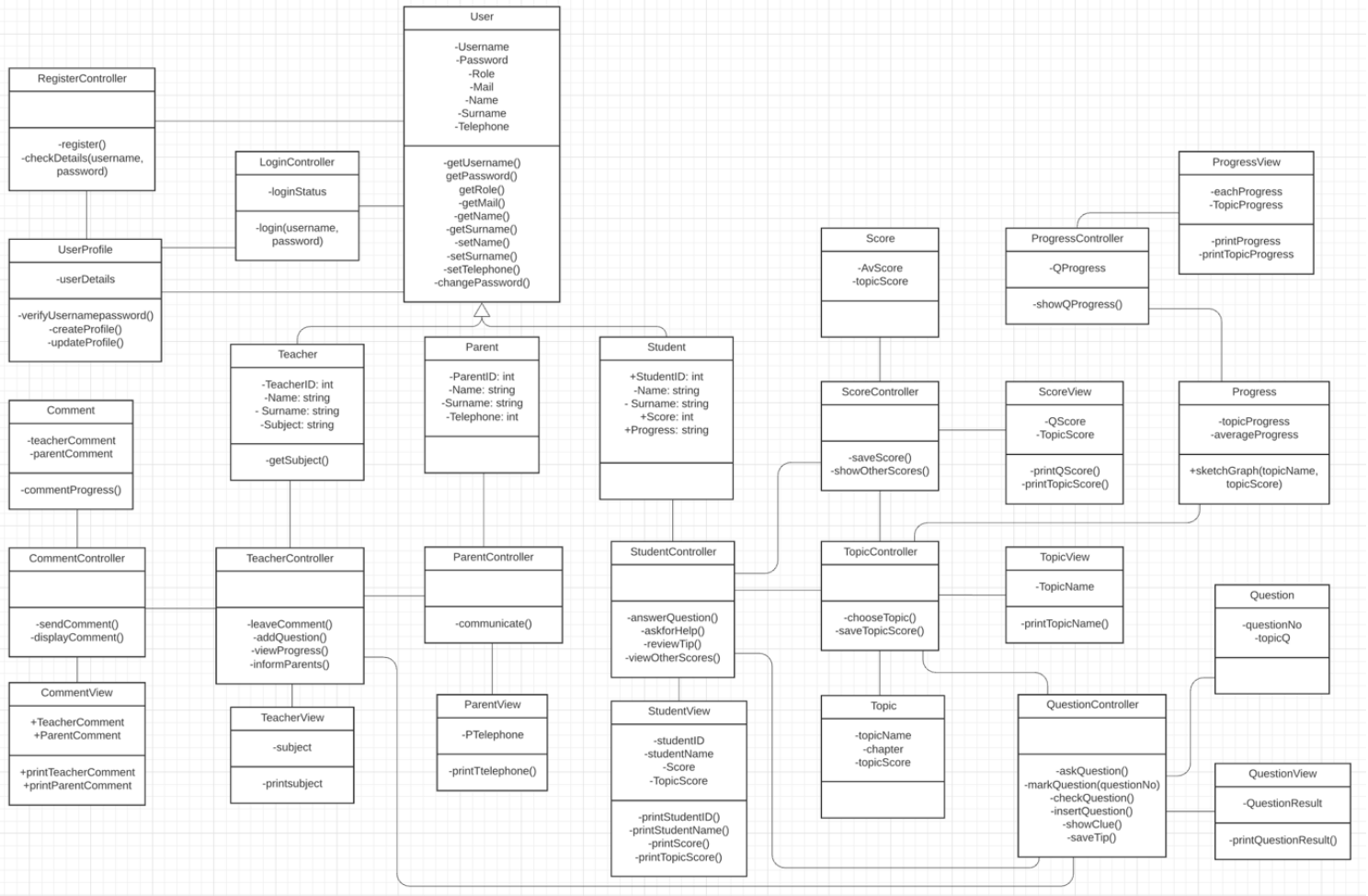
Parent logged in:

Data item	Data type	Validation	Sample data
Parent's communication with teacher	varchar(200)		Do you recommend any books?

Documented design:

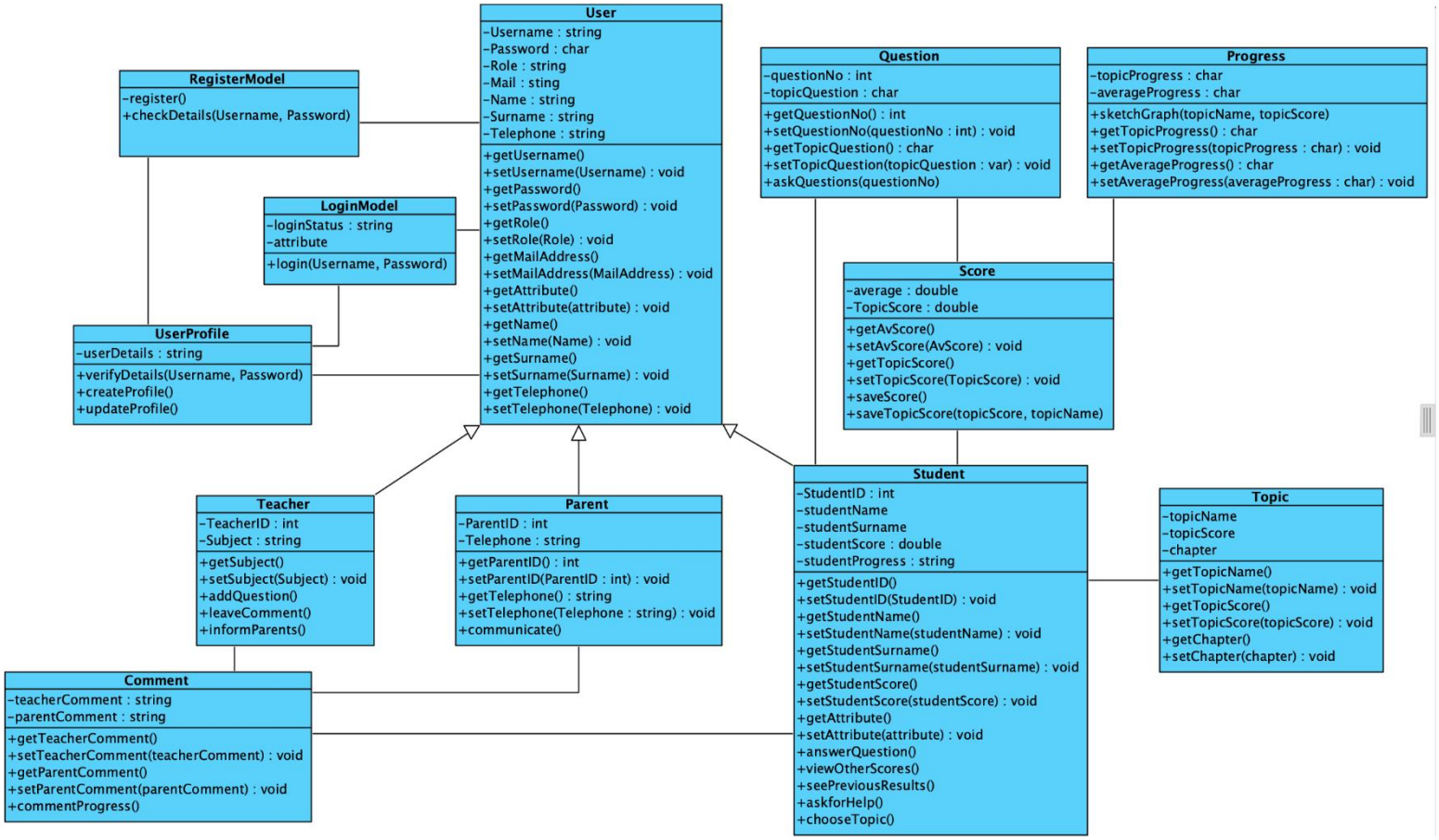
Class diagram:

I have used the MVC pattern (Model-View-Controller) which is a pattern used to develop user interfaces. The classes Student, Teacher, Parent, Comment, Topic, Score, Question, Progress each have their view and controller class. The view is to print details about its class (visualization), the controller communicates with other controllers and is connected to both the model and the view (updates the view and keeps model and view separate). The Student, Parent, Teacher models inherit the methods of the user class, the Teacher class has one different method to get their subject which should be to differentiate between a student simply typing 'Teacher' when asked what the role is.

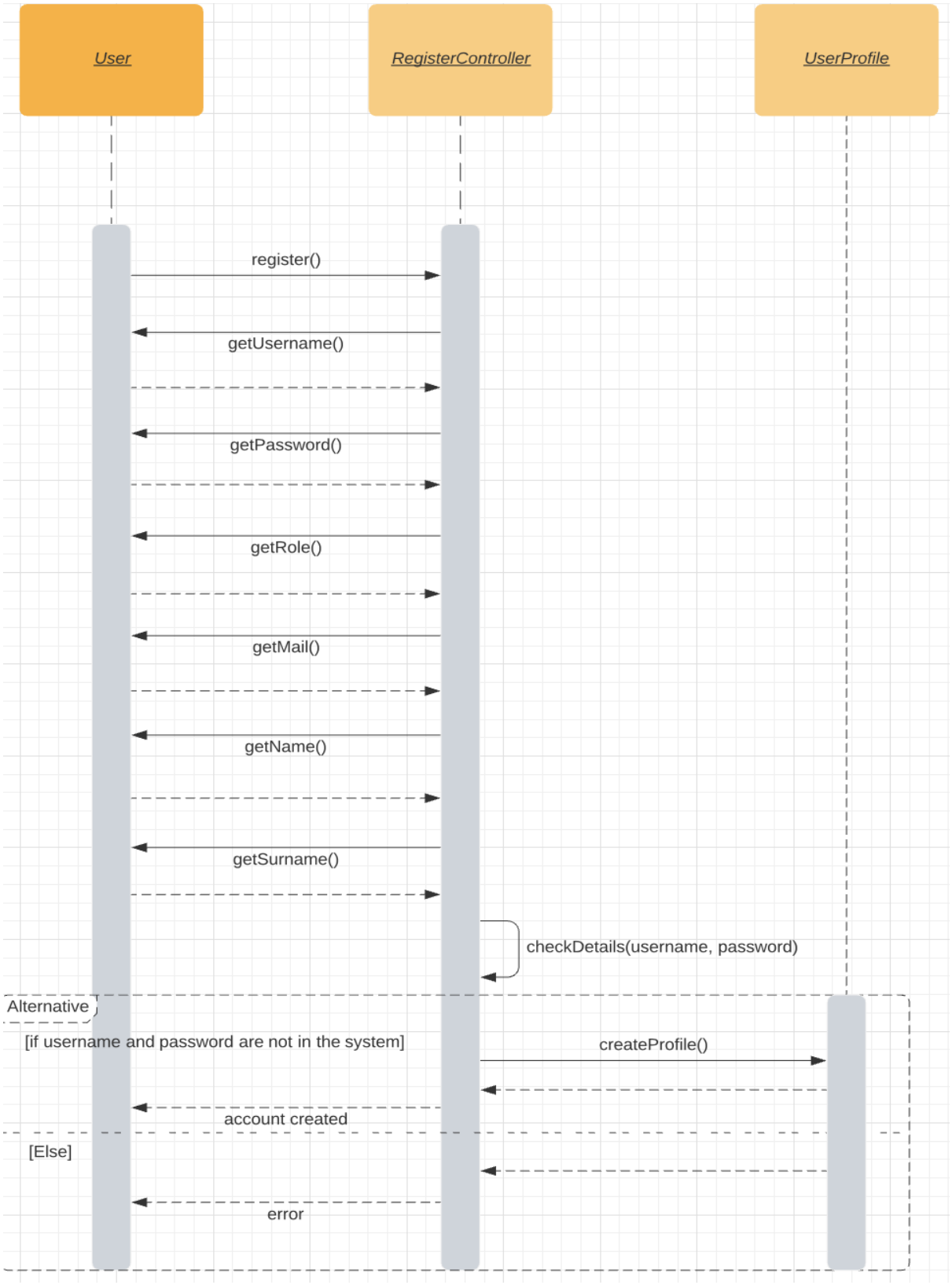


*the class ‘parent’ does not have an arrow to inherit from the superclass ‘user’ as the number of items used exceeded the limit of 60 for the free account

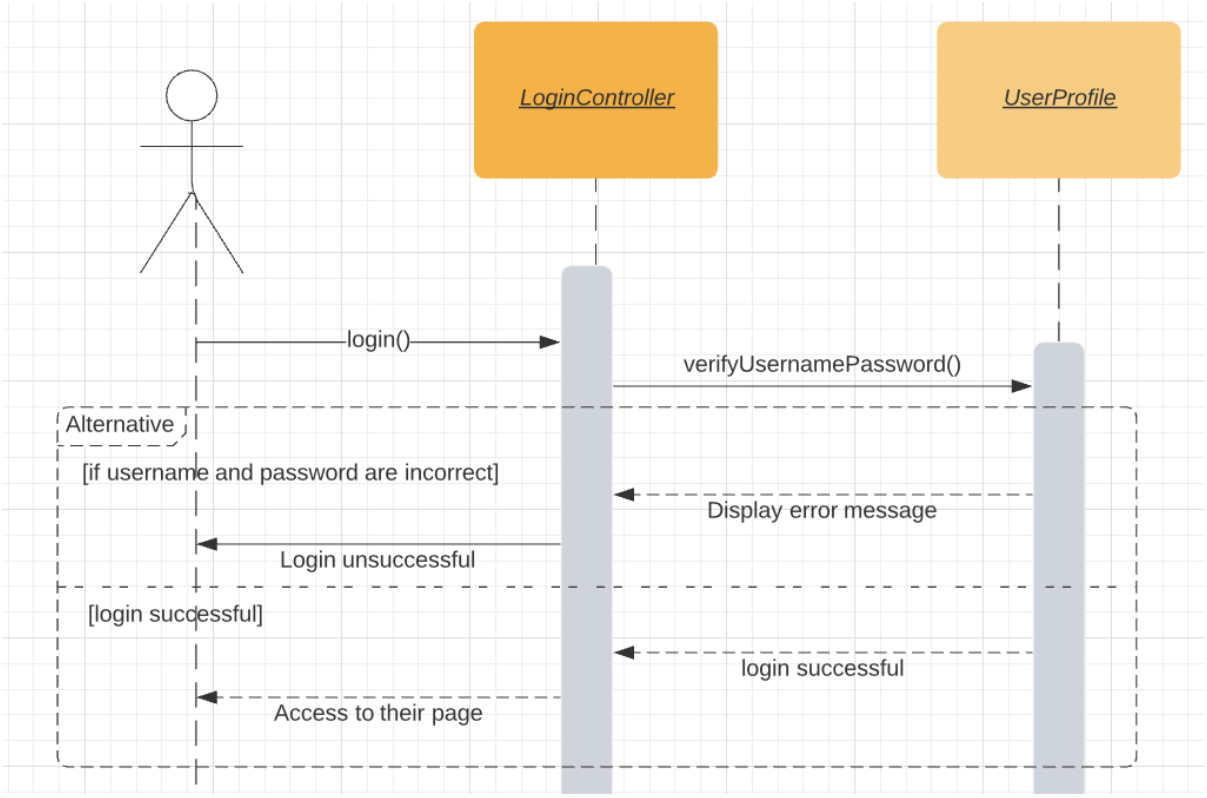
Class diagram designed with visual paradigm without using the MVC pattern:

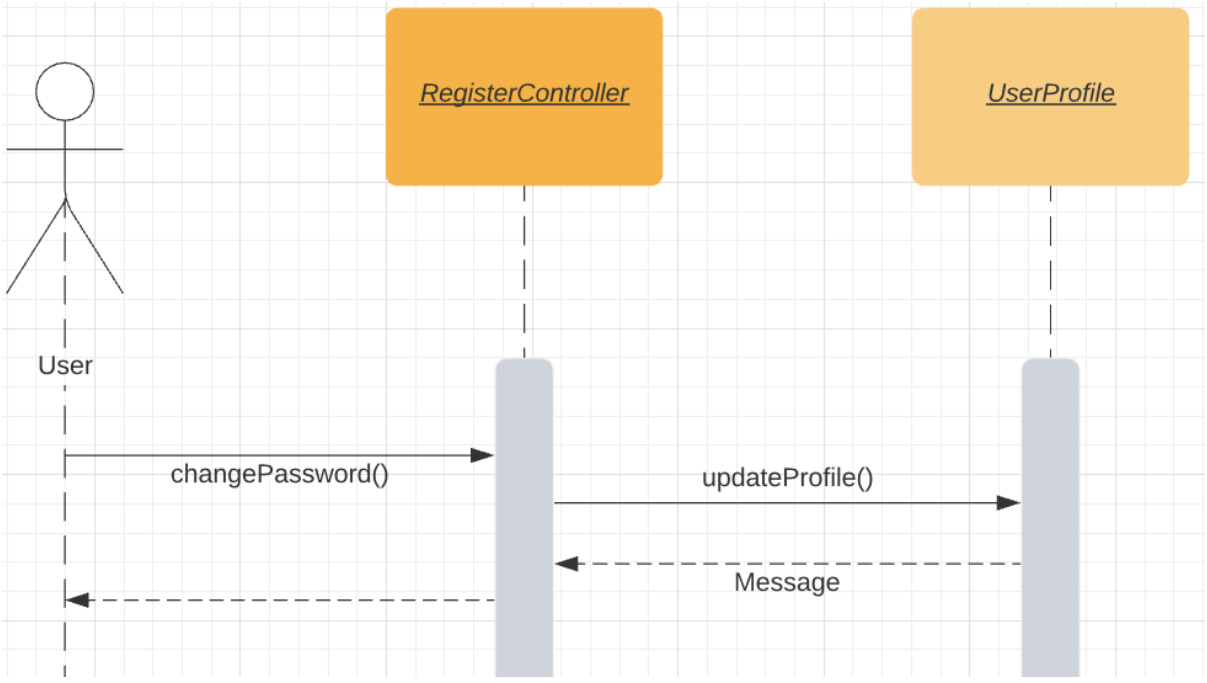


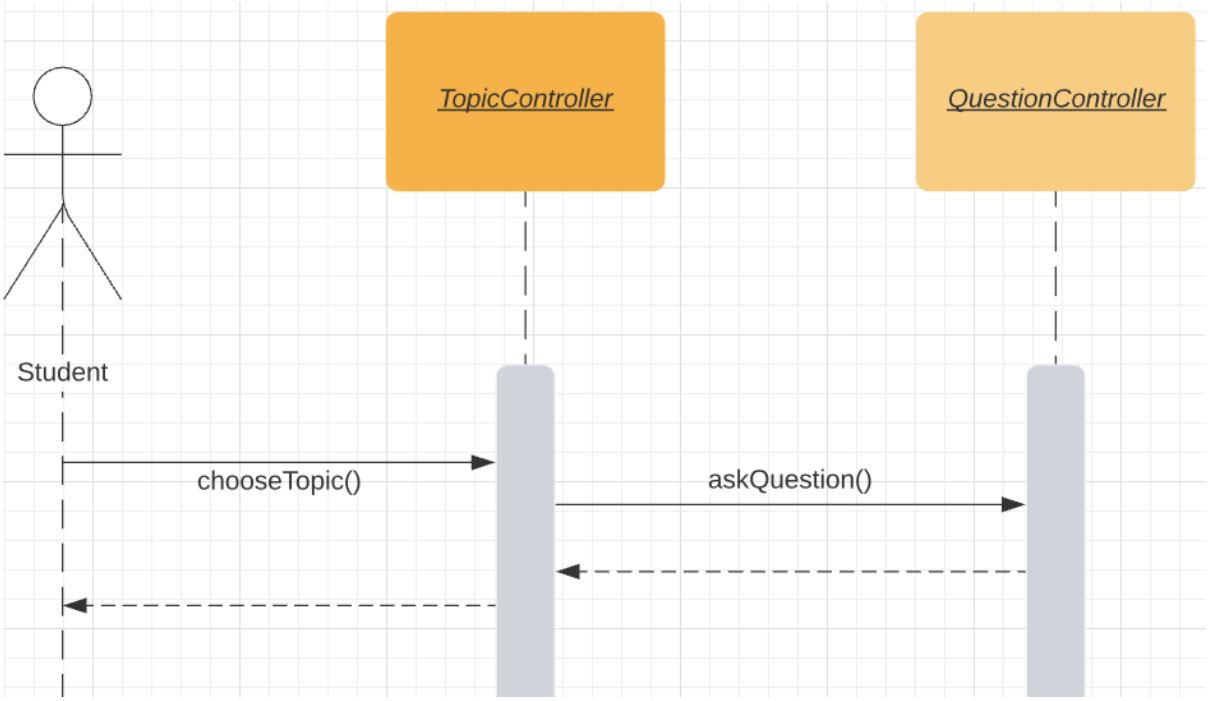
Sequence diagrams to show when each method is called and by which class:



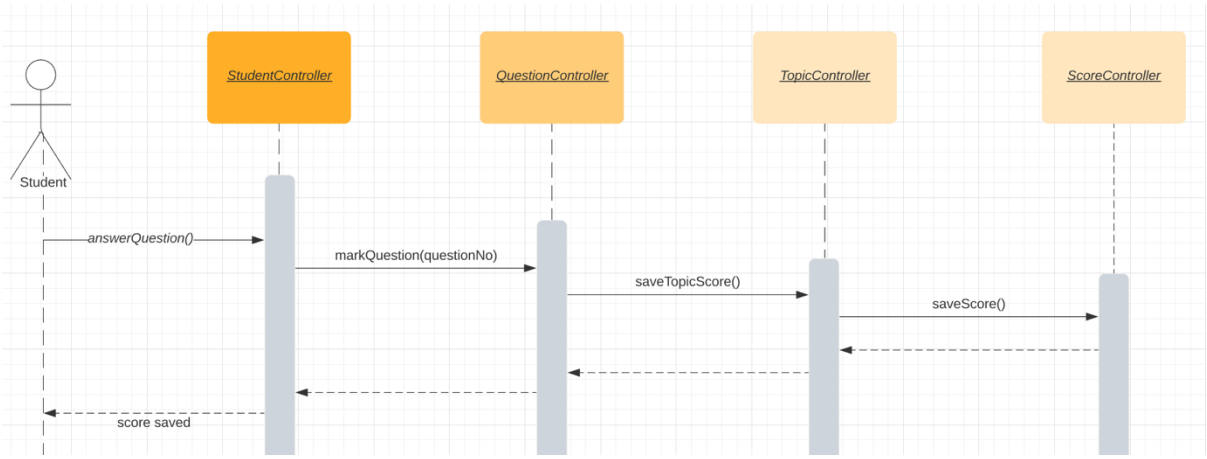
When user clicks the button ‘register’, the method register() of RegisterController is called which will call getUsername() of the class User, to get details (username, password) to store in the database, and so on. If the details already exist in the database, the system will show the user an error message saying that they are already registered, otherwise, a new profile is created with the createProfile() method of UserProfile.

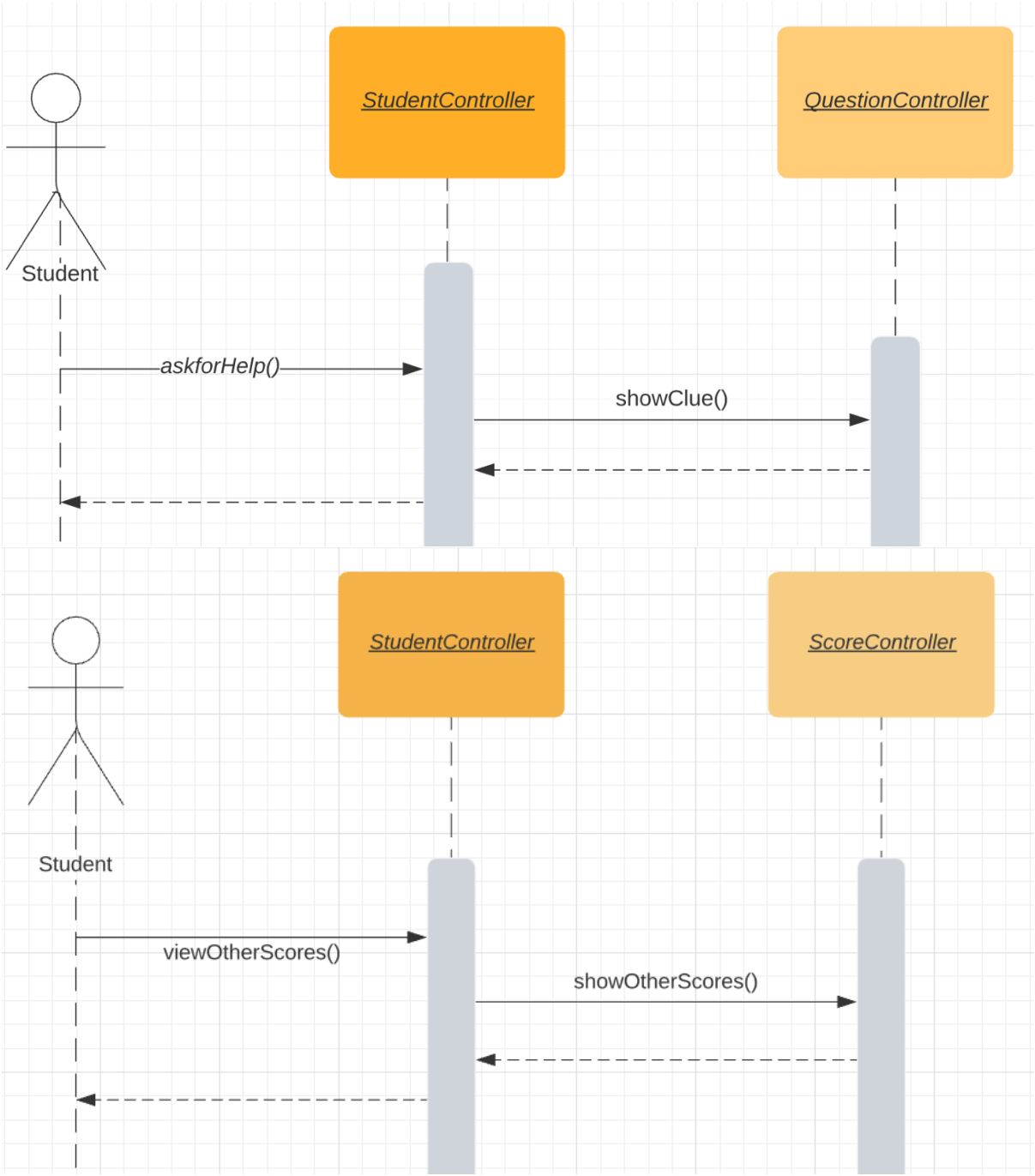




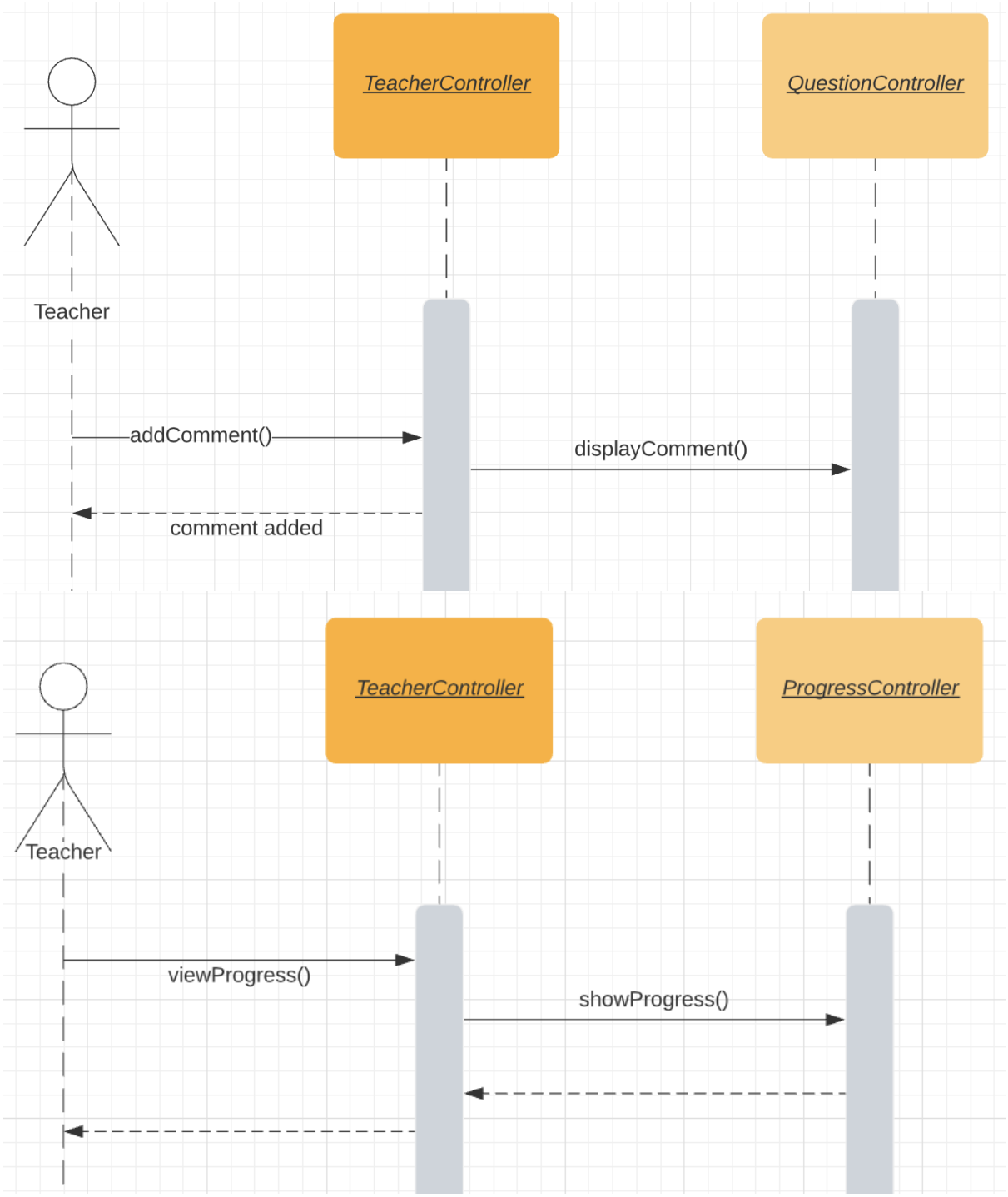


Initially only one topic should be chosen but as the program expands, more than one topic may be chosen, as this is to prioritize the chosen topic, but questions will be asked on other topics as well.

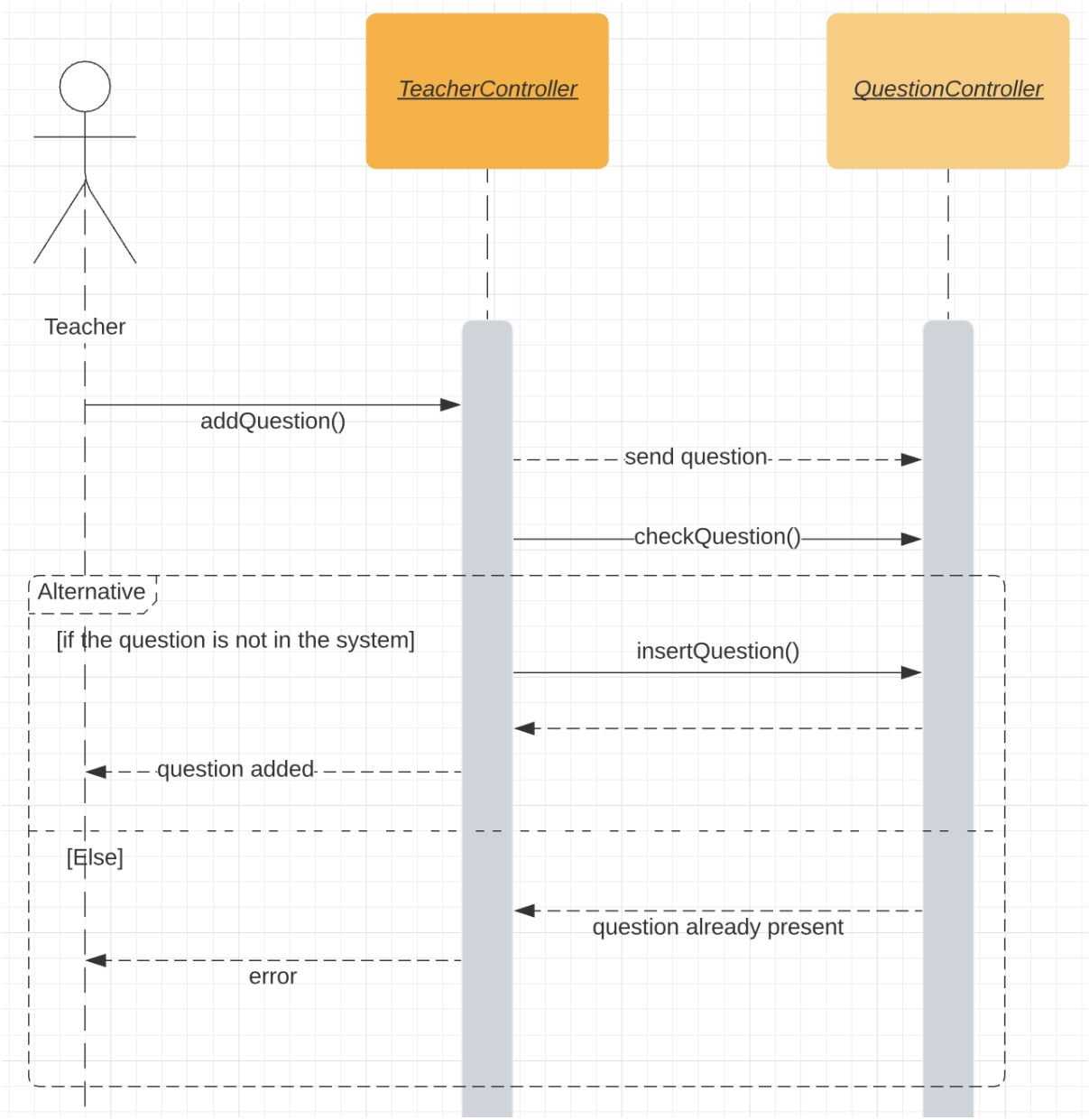




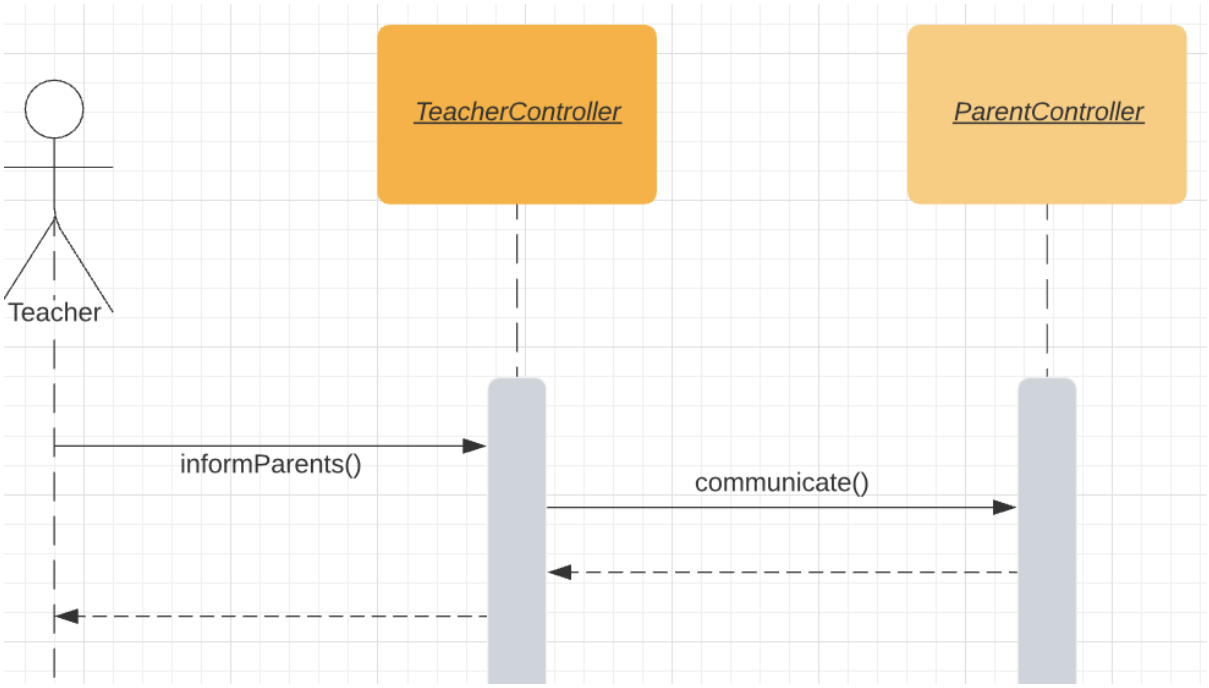
Scores of other students will be saved to the database and can be viewed by other students (only with their student ID), the teacher can also see students' scores with their names

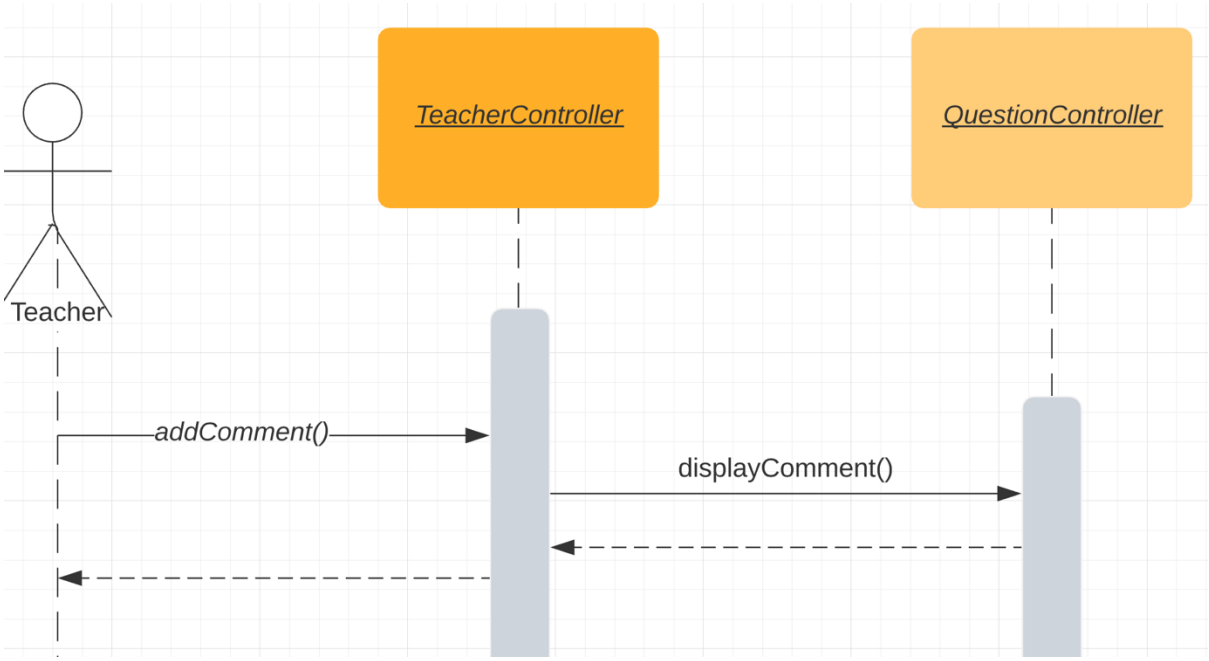


The progress means that a bar graph or a doughnut chart will be generated based on the scores stored on SQL tables and then the teacher can view them.

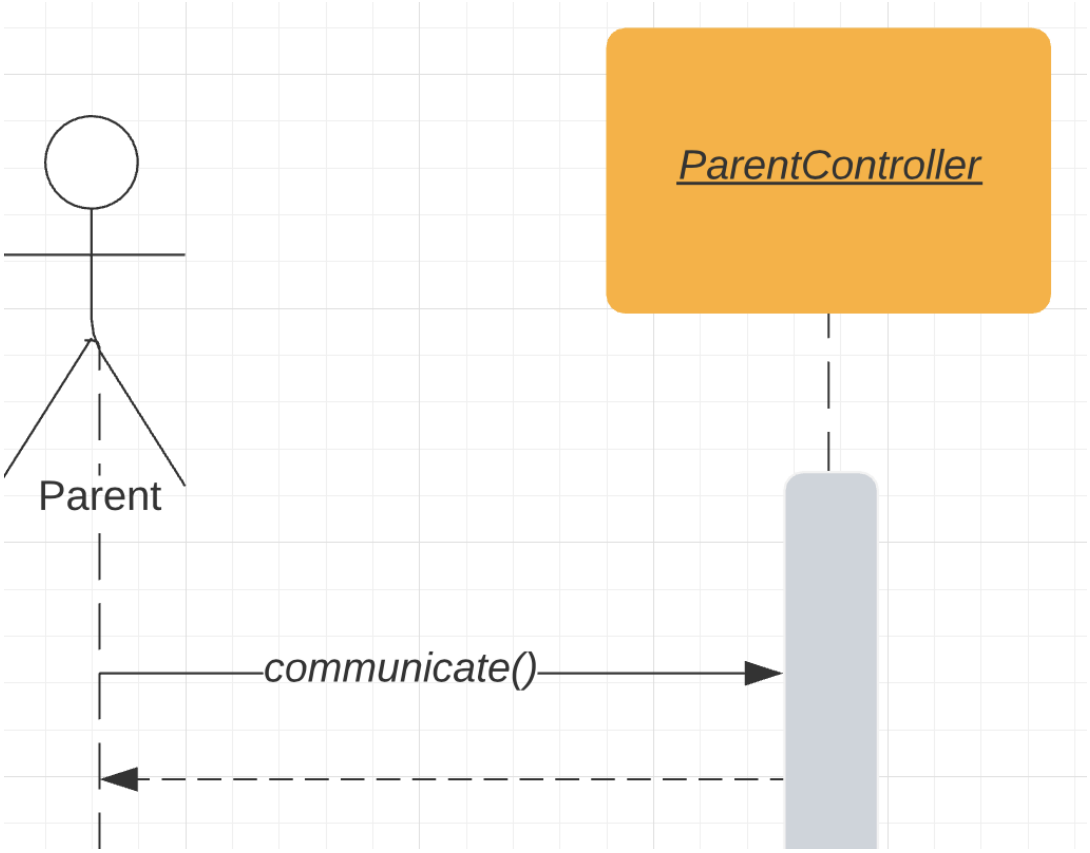


There is the option to add questions on the teacher's page, if they choose this option, the method addQuestion() is called from TeacherController, the system asks the teacher to type a question, and then the answer, and estimated level of difficulty (this means approximately which grade it is (9-1)). This will then be stored in the database and included in the questions that the system will present to the student to answer.





The method addComment () of TeacherController will take a string parameter which will be displayed on the student's page.



Parent can communicate with their son or daughter's teacher; the dotted arrow is a response back to that class (in this case parent) after they have done a task (in this case having communicated with the teacher). The black arrows are always methods, they point towards the class to which the method belongs to (the method communicate () is in the class Parent Controller). The message sent will then be displayed on the teacher's page when they log back in.

In windows forms, these methods will be button clicks.

To store data about the users, a database is more efficient than storing it locally in the program as it is not static data, but this is the class library for the classes specified above:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Project
{
    public class User
    {
        private string Username;
        private string Password;
```

```
private string Role;
private string Mail;
private string Name;
private string Surname;
private string Telephone;
public void getUsername()
{
    // TODO - implement User.getUsername
}

public void setUsername(int Username)
{
    // TODO - implement User.setUsername
}

public void getPassword()
{
    // TODO - implement User.getPassword
}

public void setPassword(string Password)
{
    // TODO - implement User.setPassword
}

public void getRole()
{
    // TODO - implement User.getRole
}
public void setRole(int Role)
{
    // TODO - implement User.setRole
}

public void getMailAddress()
{
    // TODO - implement User.getMailAddress
}

public void setMailAddress(int MailAddress)
{
    // TODO - implement User.setMailAddress
}

public void getAttribute()
{
    // TODO - implement User.getAttribute
}

public void setAttribute(int attribute)
{
    // TODO - implement User.setAttribute
}
```

```
    public void getName()
    {
        // TODO - implement User.getName
    }

    public void setName(int Name)
    {
        // TODO - implement User.setName
    }
    public void getSurname()
    {
        // TODO - implement User.getSurname
    }

    public void setSurname(int Surname)
    {
        // TODO - implement User.setSurname
    }
    public void getTelephone()
    {
        // TODO - implement User.getTelephone
    }

    public void setTelephone(int Telephone)
    {
        // TODO - implement User.setTelephone
    }
}
public class Student : User
{
    private int StudentID;
    private int studentName;
    private int studentSurname;
    private double studentScore;
    private string studentProgress;
    public void getStudentID()
    {
        // TODO - implement Student.getStudentID
    }

    public void setStudentID(int StudentID)
    {
        // TODO - implement Student.setStudentID
    }
    public void getStudentName()
    {
        // TODO - implement Student.getStudentName
    }

    public void setStudentName(int studentName)
    {
        this.studentName = studentName;
    }
```

```
    public void getStudentSurname()
    {
        // TODO - implement Student.getStudentSurname
    }

    public void setStudentSurname(int studentSurname)
    {
        this.studentSurname = studentSurname;
    }
    public void getStudentScore()
    {
        // TODO - implement Student.getStudentScore
    }

    public void setStudentScore(int studentScore)
    {
        // TODO - implement Student.setStudentScore
    }
    public void getAttribute()
    {
        // TODO - implement Student.getAttribute
    }

    public void setAttribute(int attribute)
    {
        // TODO - implement Student.setAttribute
    }
}
public class StudentController
{
    private Student student;
    private StudentView view;

    public StudentController(string name, string id)
    {

    }
    public void answerQuestion()
    {
        // TODO - implement StudentController.answerQuestion
    }
    public void askForHelp()
    {
        // TODO - implement StudentController.askForHelp
    }
    public void viewOtherScores()
    {
        // TODO - implement StudentController.viewOtherScores
    }
}
public class StudentView
{
    private double topicScore;
```

```
    public void printScores()
    {
        // TODO - implement StudentView.printScores
    }
    public void printStudentID()
    {
        // TODO - implement StudentView.printStudentID
    }
}
public class Teacher : User
{
    private int TeacherID;
    private string Subject;
    public void getSubject()
    {
        // TODO - implement Teacher.getSubject
    }

    public void setSubject(int Subject)
    {
        // TODO - implement Teacher.setSubject
    }
}
public class TeacherController
{
    public void leaveComment()
    {
        // TODO - implement TeacherController.leaveComment
    }
    public void addQuestion()
    {
        // TODO - implement TeacherController.addQuestion
    }
    public void viewProgress()
    {
        // TODO - implement TeacherController.viewProgress
    }
    public void informParents()
    {
        // TODO - implement TeacherController.informParents
    }
}
public class TeacherView
{
    private string subject;
    public void printsubject()
    {
        // TODO - implement TeacherView.printsubject
    }
}
public class Parent : User
{
    private int ParentID;
```

```
    private string Telephone;
    public Parent(int getParentID, string getTelephone)
    {
        ParentID = getParentID;
        Telephone = getTelephone;
    }
    public int getParentID()
    {
        // TODO - implement Parent.getParentID
        return ParentID;
    }

    public void setParentID(int ParentID)
    {
        // TODO - implement Parent.setParentID
    }
    public string getTelephone()
    {
        // TODO - implement Parent.getTelephone
        return Telephone;
    }

    public void setTelephone(string Telephone)
    {
        // TODO - implement Parent.setTelephone
    }
}
public class ParentController
{
    public void communicate()
    {
        // TODO - implement ParentController.communicate
    }
}
public class ParentView
{
    private string ParentTelephone;
    public void printTelephone()
    {
        // TODO - implement ParentView.printTelephone
    }
}
public class UserProfile : User
{
    public int profileID;
    public string userDetails;

    public void verifyDetails(int Username, int Password)
    {
        // TODO - implement UserProfile.verifyDetails
    }
    public void createProfile()
    {
```

```
        // TODO - implement UserProfile.createProfile
    }
    public void updateProfile()
    {
        // TODO - implement UserProfile.updateProfile
    }
}
public class LoginController
{
    private string loginStatus;
    private int attribute;

    public void login(string Username, char Password)
    {
        // TODO - implement LoginController.login
    }
}
public class Question
{
    private int questionNo;
    private char topicQuestion;
    public int getQuestionNo()
    {
        return this.questionNo;
    }

    public void setQuestionNo(int questionNo)
    {
        this.questionNo = questionNo;
    }
    public char getTopicQuestion()
    {
        return this.topicQuestion;
    }

    public void setTopicQuestion(string topicQuestion)
    {
        // TODO - implement Question.setTopicQuestion
    }
}
public class QuestionController
{
    public void askQuestion()
    {
        // TODO - implement QuestionController.askQuestion
    }

    public void markQuestion(int questionNo)
    {
        // TODO - implement QuestionController.markQuestion
    }
    public void questionExists()
    {
```

```
        // TODO - implement QuestionController.questionExists
    }
    public void insertQuestion()
    {
        // TODO - implement QuestionController.insertQuestion
    }
    public void showClue()
    {
        // TODO - implement QuestionController.showClue
    }
    public void saveTip()
    {
        // TODO - implement QuestionController.saveTip
    }
}
public class QuestionView
{
    private int questionResult;
    public void getQuestionResult()
    {
        // TODO - implement QuestionView.getQuestionResult
    }

    public void setQuestionResult(int questionResult)
    {
        this.questionResult = questionResult;
    }
    public void printQuestionResult()
    {
        // TODO - implement QuestionView.printQuestionResult
    }
}
public class Score
{
    private double average;
    private double TopicScore;
    public void getAvScore()
    {
        // TODO - implement Score.getAvScore
    }

    public void setAvScore(int AvScore)
    {
        // TODO - implement Score.setAvScore
    }
    public void getTopicScore()
    {
        // TODO - implement Score.getTopicScore
    }

    public void setTopicScore(int TopicScore)
    {
        // TODO - implement Score.setTopicScore
```

```
    }
}
public class ScoreController
{
    public void saveScore()
    {
        // TODO - implement ScoreController.saveScore
    }
    public void showOtherScores()
    {
        // TODO - implement ScoreController.showOtherScores
    }
}
public class ScoreView
{
    private double QuestionScore;
    public void getQuestionScore()
    {
        // TODO - implement ScoreView.getQuestionScore
    }

    public void setQuestionScore(int QuestionScore)
    {
        // TODO - implement ScoreView.setQuestionScore
    }
    public void printscores()
    {
        // TODO - implement ScoreView.printscores
    }
}
public class Progress
{
    private char topicProgress;
    private char averageProgress;

    public void sketchGraph(int topicName, int topicScore)
    {
        // TODO - implement Progress.sketchGraph
    }
    public char getTopicProgress()
    {
        return this.topicProgress;
    }

    public void setTopicProgress(char topicProgress)
    {
        this.topicProgress = topicProgress;
    }
    public char getAverageProgress()
    {
        return this.averageProgress;
    }
```

```
    public void setAverageProgress(char averageProgress)
    {
        this.averageProgress = averageProgress;
    }
}
public class ProgressController
{
    private string questionsProgress;
    public string getQuestionsProgress()
    {
        return this.questionsProgress;
    }

    public void setQuestionsProgress(string questionsProgress)
    {
        this.questionsProgress = questionsProgress;
    }
}
public class ProgressView
{
    private string progressEach;
    private string topicProgress;
    public void printProgress()
    {
        // TODO - implement ProgressView.printProgress
    }
    public void printTopicProgess()
    {
        // TODO - implement ProgressView.printTopicProgess
    }
}
public class Comment1
{
    private string teacherComment;
    private string parentComment;
    public void getTeacherComment()
    {
        // TODO - implement Comment.getTeacherComment
    }

    public void setTeacherComment(int teacherComment)
    {
        // TODO - implement Comment.setTeacherComment
    }
    public void getParentComment()
    {
        // TODO - implement Comment.getParentComment
    }

    public void setParentComment(int parentComment)
    {
        // TODO - implement Comment.setParentComment
    }
```

```
    public void commentProgress()
    {
        // TODO - implement Comment.commentProgress
    }
}
public class CommentController
{
    public void sendComment()
    {
        // TODO - implement CommentController.sendComment
    }
    public void displayComment()
    {
        // TODO - implement CommentController.displayComment
    }
}
public class CommentView
{
    public void printComment()
    {
        // TODO - implement CommentView.printComment
    }
}
public class Topic
{
    private int topicName;
    private int topicScore;
    private int chapter;
    public void getTopicName()
    {
        // TODO - implement Topic.getTopicName
    }

    public void setTopicName(int topicName)
    {
        this.topicName = topicName;
    }
    public void getTopicScore()
    {
        // TODO - implement Topic.getTopicScore
    }

    public void setTopicScore(int topicScore)
    {
        this.topicScore = topicScore;
    }
    public void getChapter()
    {
        // TODO - implement Topic.getChapter
    }
    public void setChapter(int chapter)
    {
        this.chapter = chapter;
```

```
        }
    }
    public class TopicController
    {
        public void chooseTopic()
        {
            // TODO - implement TopicController.chooseTopic
        }
        public void saveTopicScores()
        {
            // TODO - implement TopicController.saveTopicScores
        }
    }
    public class TopicView
    {
        private string TopicName;
        public void printTopicName()
        {
            // TODO - implement TopicView.printTopicName
        }
    }
}
```

Object oriented programming principles used:

Inheritance:

Student, parent, and teacher inherit from the table user profile because they have the attributes of username and password… in common.

Methods:

The methods in windows forms are based on the dependencies of the visual forms with which the user interacts so a method means the user has entered or made an input to the form.

Constructors:

These will be used to pass information regarding the users such as individual id and name across each form when a new instance of that class (form) is created.

Partial classes:

These inherit from their dependency of the visual form and are treated as a class would be in a console application except in windows forms there is interaction with the visual form.

Access modifiers:

Public, private. The attributes declared in each partial class will be either an int, string or array list because it can be accessed using an index. Nearly or all these attributes can be private because they are needed within that class.

IPSO chart:

Input	Process	Storage	Output
Login details (username and password) and position (student, parent, or teacher)	Username and password retrieved from database and compared to the ones provided	SQL table: user profile	If the user has entered their position and username and password are correct (exist in the database), the relevant form according to their position will open. If the details provided do not exist in the database, the system will output an error message.
New login details to register	Check if the username provided already exists in the database, otherwise insert the fields into the table user profile. Validate details, if valid, store in the database.	SQL table: user profile	If the username is taken, the system will output an error message, if it is not, the system will prompt for extra login information by opening a new form.
Student: extra details (class code, level, and subject)	Validate details, if valid, store information given and profile id in the database	SQL table: student	If the user has provided valid information, this means they have registered successfully so the system will output a message box 'account created'. Else, the system will show an error message box.
Parent: postcode, address, number of kids, name of child	Validate details, check if the name of child provided exists in the database (in the case where their child has not yet registered, they would need to do so first)	SQL table: parent	Message box 'account created' if details have been validated, else, the system will output an error box.
Teacher: department, subjects teaching, class code	Validate details, if information provided is in the correct format, insert into database	SQL table: teacher	Message box 'account created' is details have been validated, else, error box shown.
Student' answer to a given question	Insert the answers submitted at the end of the ten questions, update tables answer, score and progress.	SQL tables: answer, score, progress	Output whether each question answered was either right or wrong always with the option to request the question to be remarked by the teacher. At the end of ten questions, ask if the user wishes to continue answering questions or else go back to their main menu.
Student's feedback about their experience	Feedback inserted into database	SQL	Message box 'feedback received', 'thank you!'

with the application			
Student's enquiry to their teacher about a particular question	Comment stored in database and shown in the teacher's page	SQL	Message box 'message sent'
Teacher (information about a new question to be added)	Question inserted to database	SQL table: question	Message box 'question has been added'
Teacher's feedback to student	Comment shown on student's page	SQL	Message box ' message sent'
Teacher communication with a parent	Comment shown on parent's page	SQL	Message box ' message sent'
Parent communication with their child's teacher	Comment shown on teacher's page (deleted every school term)	SQL	Message box 'message sent'

Modular design:

Form number	Name	Description
1	Login	Prompts user for username, password, and position
2	Register	Asks user for new login details
3	Student page	Student menu includes the options of the student in the app
4	Answer questions	The student selects the topic they want to start with and answers the questions underneath
5	View scores	Shows students are making progress
6	Comment	Option to comment
7	Teacher page	Buttons containing the actions a teacher would do in the app (UML)
8	Add questions	Fields contain information about the new question to be stored
9	Mark students' work	Select students' name from the class and the question to be marked and enter the number of marks awarded
10	Parent page	Actions a parent would carry out in the application
11	View scores	Present the graphs of students' progress

Login (first form) (user details)

- if the user clicks 'login'

- check if record exists in database

- if present, open relevant form

- if the user clicks 'register'
- Register (second form) (registration details)
- Depending on the type of user (student, parent, or teacher) selected in combo box:
- Student
- Prompt for student details and deactivate parent and teacher details

- Parent
-Prompt for parent details and deactivate student and teacher details

- Teacher
- Prompt for teacher details and deactivate parent and student details

- check if username is already present in database
- if no matching record is found
- validate details
- if valid
- insert into database
- Login (first form)

- if the user is a student
- Student menu (third form)

- if the option is 'answer questions'
- Choose topic depending on the subject chosen
- Answer questions (fifth form) present questions
- generate questions at random of increasing difficulty from database

- if the option is 'view previous scores'
- select previous scores from the student from database
- Previous scores (sixth form) (usually presented as graphs)

- if the option is 'view other students' scores'
- select scores from database and ID
- Other scores (sixth form) (only IDs for each student in the database shown (graphs))

- if the option is 'send feedback about application'
- Individual feedback (seventh form) (will prompt the user for a comment)
- store in database

- if the option is 'Ask question to the teacher'
- Student enquiry (seventh form)
- (store database), display on teacher's page when they login

- if the user is a teacher
- Teacher menu (ninth form)
- if the option is 'view students' progress'
- Other scores (sixth form) (graphs)
- select scores for each student or only from the individual class and their name

- if the option is 'add question'
- Question information (tenth form) (prompt for details about the question eg. topic
- Ask if they wish to add more questions, if yes
- Question information (tenth form)
- else go back to teacher menu

- if the option is 'comment on student's progress'
- ask for whose' progress they wish to comment on
- prompt for comment (seventh form)
- display/ show in relevant student's page when they login

- if the option is 'communicate with parent'
- (seventh form)
- display comment on parent's page

- if the option is ‘mark students’ work.’
- (form)
- select name of first student in the class
- select question they wish to mark and view the student’s answer and input the number of marks awarded

- if the user is a parent
- Parent menu (thirteenth form)

- if the option is 'view son's or daughter's progress'
- select scores only from student with the parent that has just logged in
- (sixth form)

- if the option is 'communicate with teacher'
- (seventh form)
- select teacher ID where their son or daughter has the same class code
- send comment to teacher' page

***the number of marks is not entirely accurate to the textbook (and neither is the difficulty), this is just for the purpose of the program to prioritize questions based on their difficulty.**

Questions and topics (A-Level):
Chapter 1 (cases, tenses, complex relative clauses, jussive subjunctive):

omnia verba semper discite, amici et amicae!
Romae duo annos habitabamus
puer arborem antea ascenderat
sonum undarum audire possum
haec femina me sapientiam docuit
senator, vir magnae virtutis, civibus persuasit
rex amicus est mihi
consul plura verbis quam gladio effecit
et nos et servi diligenter laboramus
ille miles dux factus est

Always learn all the words, friends male and female!
We lived in Rome for two years.
The boy had climbed the tree before.
I can hear the sound of the waves.
This woman taught me wisdom.
The senator, a man of great virtue, persuaded the citizens.
The king is my friend.
The consul accomplished more with his words than with his sword.
Both we and the slaves are working carefully.
That soldier was made leader

hunc montem cras ascendemus
quinque dies contendebamus; sexto ad oram advenimus
puellam quam antea conspexeram salutavi
opus diligenter facio, sed mox confecero
omnis cibus iam consumptus est
si navem conspicies, statim clama
patriam relinquebam; subito tamen revocatus sum
tam audax facinus in Italia numquam visum erat
ubi Romam rediremus, domum tuam invenire conabimur
dum haec geruntur, nova clades nuntiata est

We shall climb this mountain tomorrow.
We were marching for five days; on the sixth we reached the coast.
I greeted the girl whom I had caught sight of before.
I am doing the work carefully, but I shall soon have finished it.
All the food has now been eaten.
If you catch sight of a ship, shout immediately.
I was leaving my homeland; suddenly however I was called back.
So bold a crime had never been seen in Italy.
When we return to Rome, we shall try to find your house.
While these things were being done, a new disaster was reported

novum librum legamus
si liberi adhuc adsunt, exeant omnes
vivamus atque amemus

ne hoc a magistro audiatur
gaudeamus igitur, iuvenes dum sumus

Let us read the new book!
If the children are still here, let them all go out!
Let us live and let us love!
May this not be heard by the teacher!
Let us rejoice therefore, while we are young men!

Chapter 2 (principal parts, indirect statement, ablative absolute, time):
legatos mittam pacem petitum
imperator arcessitus est patriam defensum
domum dormitum eo
ad Graeciam navigavimus hostes oppugnatum
omnia verba (mirabile dictu) tandem didici
nuntii advenerunt questum iniurias
accidit res non modo visu mirabilis sed etiam auditu
socii ad castra contenderunt rogatum auxilium
veni monitum vos, non punitum
spectatum veniunt , veniunt spectentur ut ipsae

I shall send envoys to seek peace.
The general was summoned to defend his country.
I am going home to sleep.
We sailed to Greece to attack the enemy.
I have at last (wonderful to tell!) learned all the words.
Messengers arrived to complain about injustices.
A thing happened that was wonderful not only to see but also to hear about.
The allies marched to the camp to ask for help.
I come to warn you, not to punish you.
The women come to watch, and they come so that they may themselves be watched

nautae dixerunt naves hostium appropinquare
puella intellegit se ab omnibus laudari
puto servos opus nunc confecisse
puer dixit amicum suum mox adventurum esse
credo nos montem ascendere posse
servus promisit se opus celeriter confecturum esse
audivimus pedites tandem victos esse
nuntius respondit se nihil de clade cognovisse
non putamus hunc captivum vera dicere
sensi me ab inmico spectari

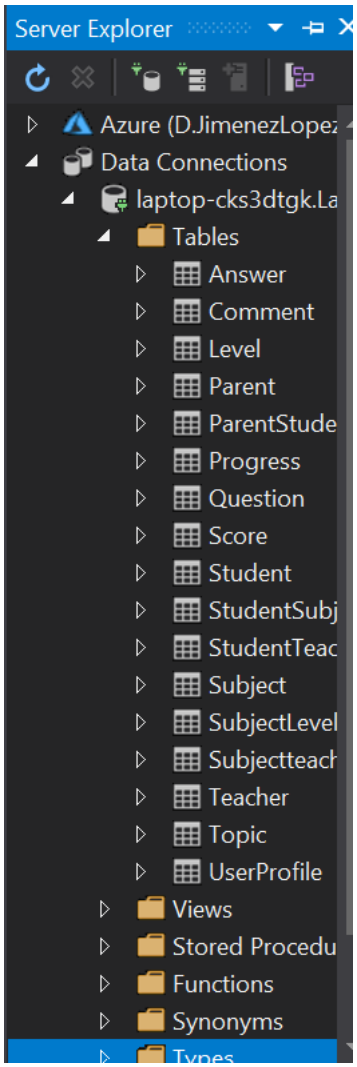
The sailors said that the enemy's ships were approaching.
The girl understands that she is being praised by everyone.
I think that the slaves have now completed the task.
The boy said that his friend would soon arrive.

I believe we can climb the mountain.
The slave promised that he would complete the work quickly.
We heard that the infantry had been defeated at last.
The messenger replied that he had found out nothing about the disaster.
We do not think that this prisoner is telling the truth.
I felt that I was being watched by an enemy

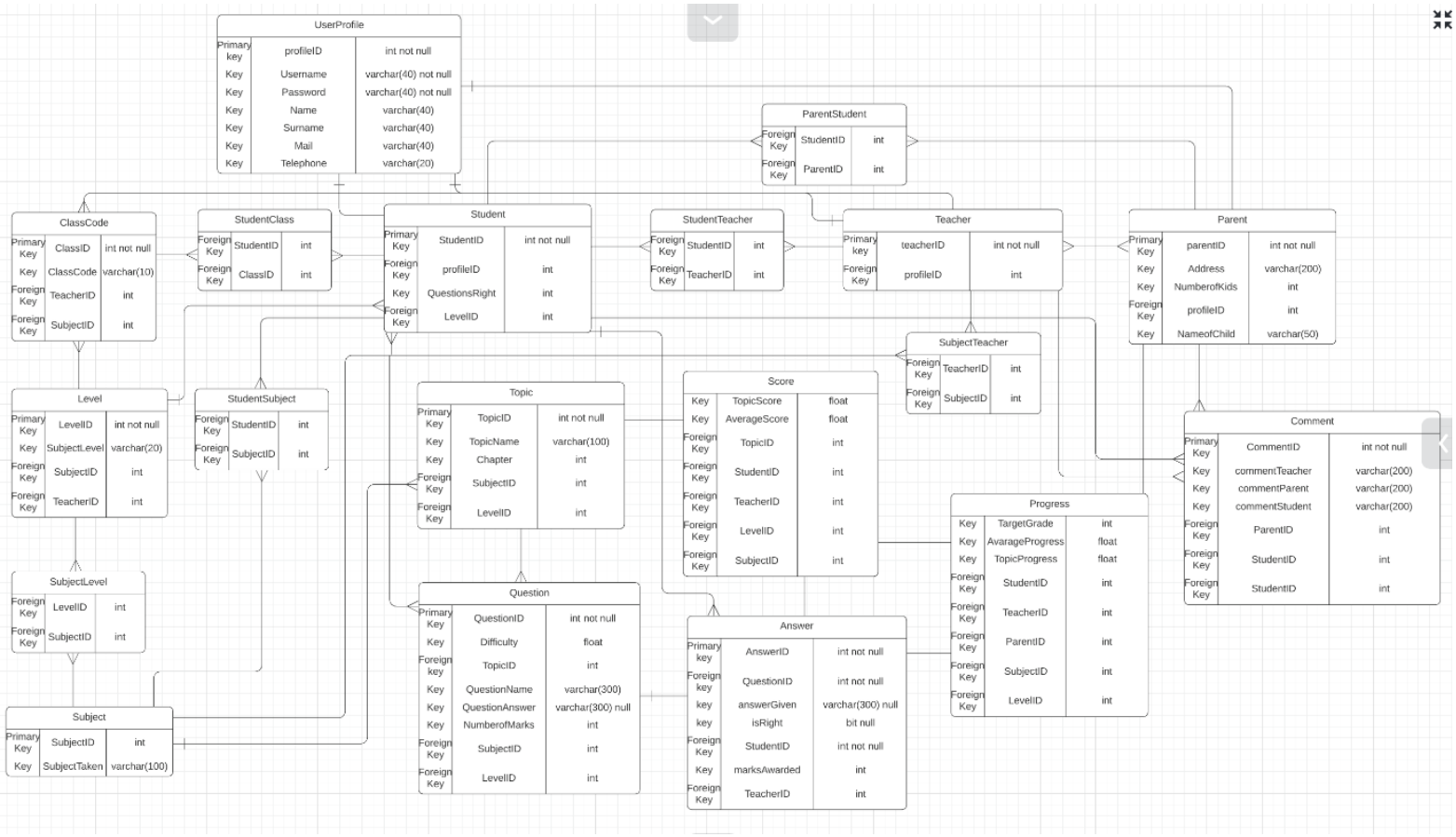
Chapter 3 (predicative dative, gerunds, and gerundives):

Chapter 4 (unseen translation passages, English to Latin):

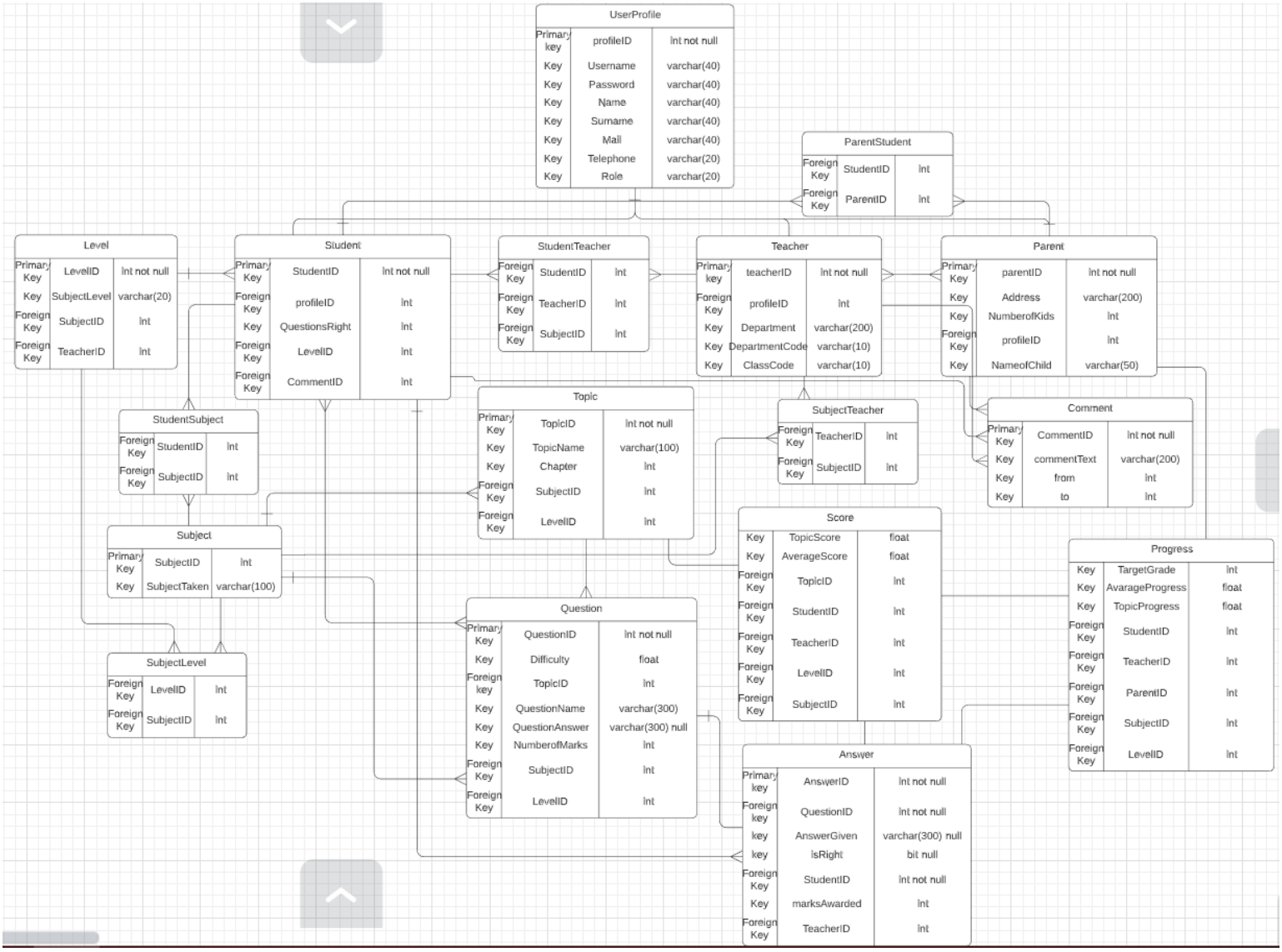
Proof of database connection:



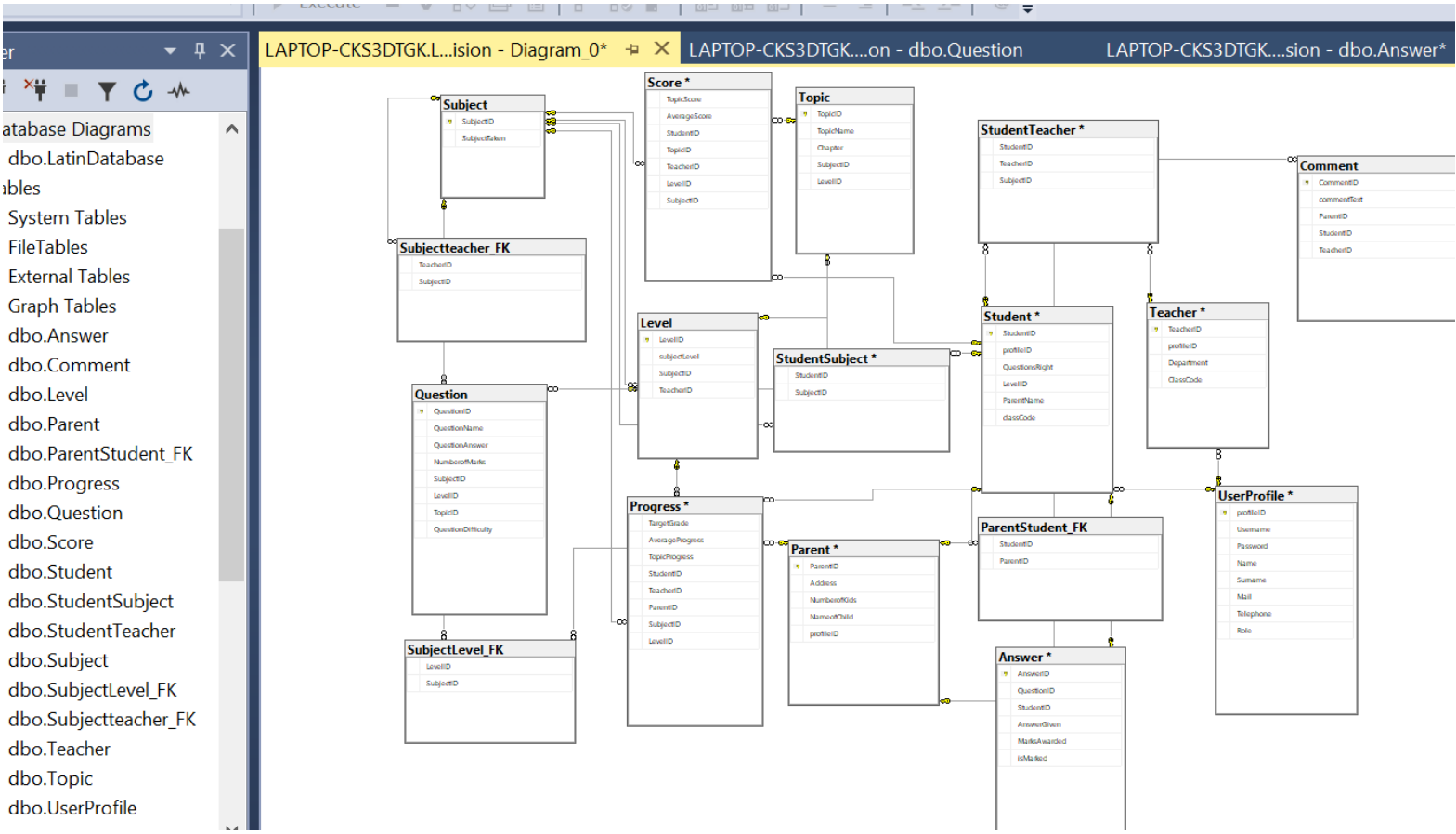
First draft of the database:



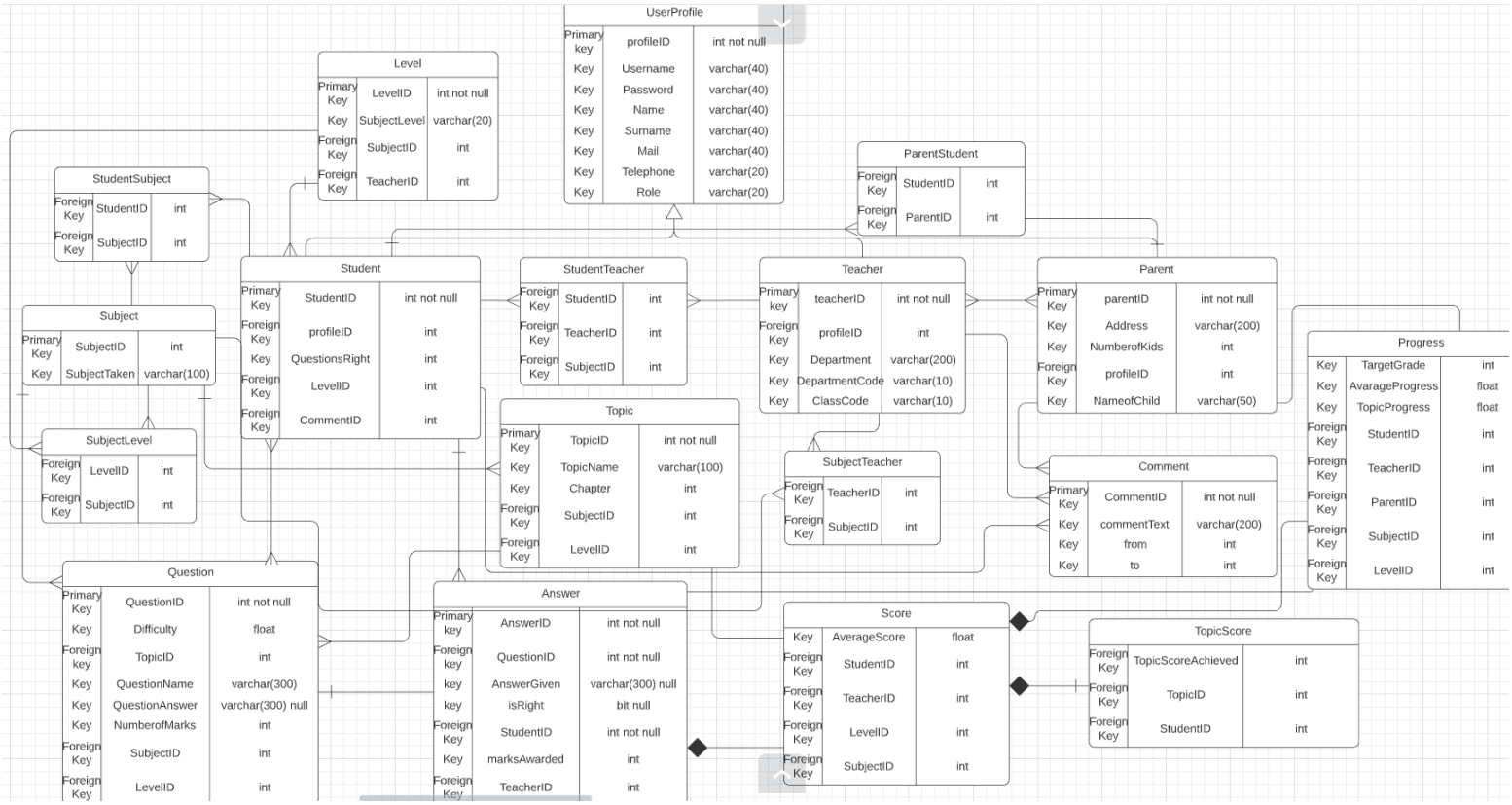
Database design:



From microsoft sql server management studio:



Final database design (relational database) with several interlinked tables which is connected to the coded solution:



Inheritance: student, parent and teacher from user profile

Composition: answer and score – if there is no answer submitted, there cannot be a score based on it which means that without the answer table, there wouldn't be a table to hold the scores, which in turn would destroy tables: topic score and progress.

Table:

- Topic score:
- TopicScoreAchieved - this the added total of the number of marks awarded by the teacher which are stored in the table called answer, using the aggregate sql function sum()
- Score:
- Average score – this is the average (using an sql aggregate function avg()) of all the number of marks gained by an individual student
- Progress:
- Target grade – user input from the teacher
- Average progress – user input
- Topic progress – user input

These tables will send information to the form ViewScores:

- Student:

- Will be shown their previous scores for each topic
- Will be able to see other students' scores with student id
- Teacher:
- Will see what each student has scored across each topic
- Parent:
- Will see their child's performance across each topic

There cannot be a table called 'class code' because each class ID cannot have a unique identifier, since more than one student can have the same class code. This would cause a problem as there would be an SQL error 'subquery returned more than one value', since class code value is not unique, therefore they would have different class IDs if there were a table. However, the relationships:

1) Student- class:
- (many-to-many)
- One class has many students, and one student attends many classes.

And,

2) Teacher- class:
- (one-to-one)
- One teacher teaches one class, and one class is taught by one teacher.

Still exist, conceptually, but not implemented with tables.

Database tables:

Database name	Tablename	TableDescription
LatinRevision	UserProfile	Table containing general information about users such as password which does not need to be repeated
LatinRevision	Student	Specific details regarding the student eg. Subjects currently studying
LatinRevision	Teacher	Specific details regarding the teacher eg. Department
LatinRevision	Parent	Specific details regarding the parent eg. Postocode
LatinRevision	ClassCode	Unique field which is set by the teacher to identify by whom students are taught
LatinRevision	Level	GCSE/AS/A-Level, allows the option to ask questions that are not just GCSE
LatinRevision	Subject	Option to extend the program to present questions on other subjects as well as Latin
LatinRevision	Question	Stores the question string and the answer that corresponds to it
LatinRevision	Answer	Table which stores the answer given by the student so the teacher can view the specific answer submitted
LatinRevision	Progress	Informs teachers and parents (and students respectively) about the progress being made
LatinRevision	Comment	Allows parents to communicate with their son's or daughter's teacher
LatinRevision	Score	Table containing scores obtained by each student on each chapter and in total
LatinRevision	Topic	Table containing the topic name and chapter and LevelID to inform the teacher on which topics a student is struggling the most with

Relationship tables:

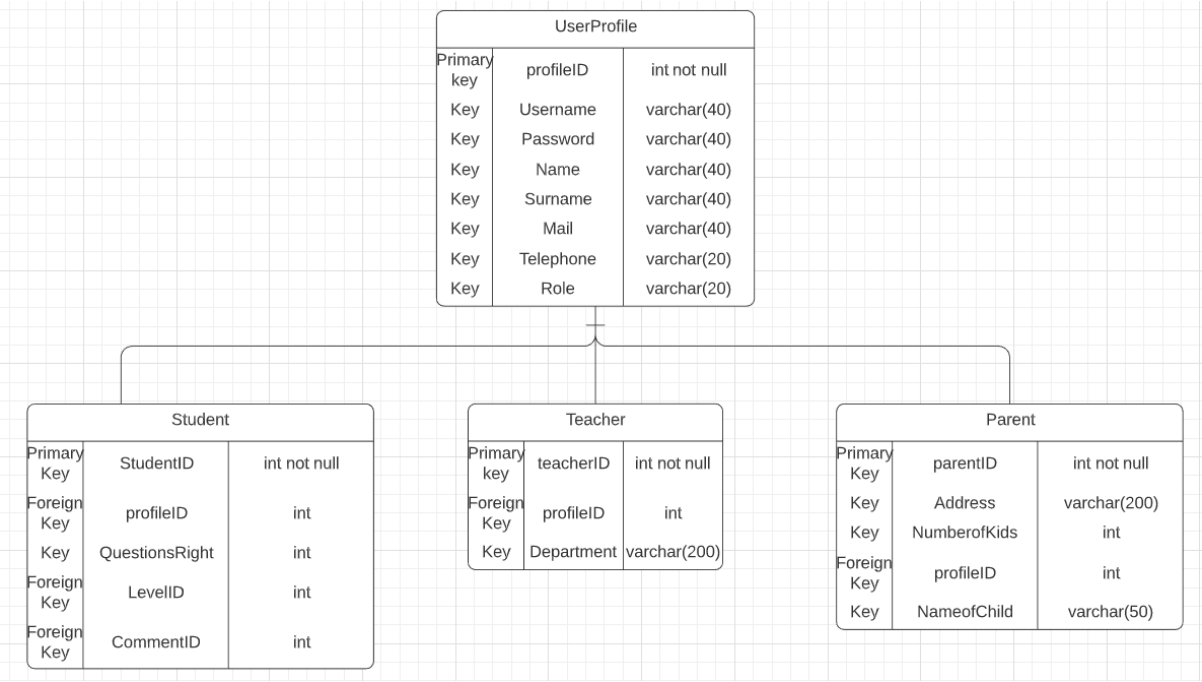
TableName	Description
SubjectLevel	Links tables subject and level
StudentClass	Links student to their class
ParentStudent	Links students to their parents
StudentTeacher	Links students to teacher
SubjectTeacher	Links subject to teacher
StudentSubject	Links student to subject

Generalisation: in this case, student, parent, and teacher can be generalised to be a user.

Database relationships:

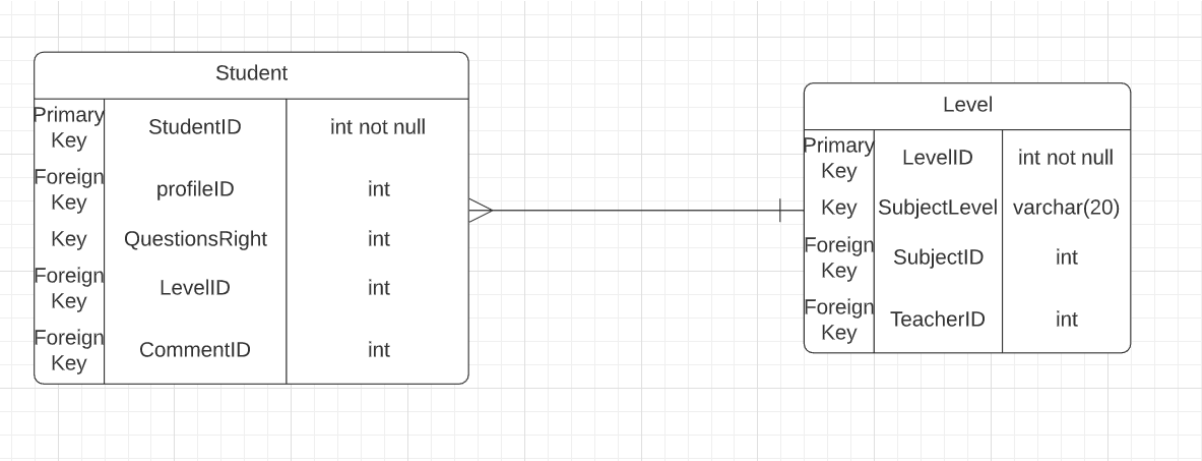
3) User profile- student, parent, teacher:

- (one-to-one)
- One user has one profile



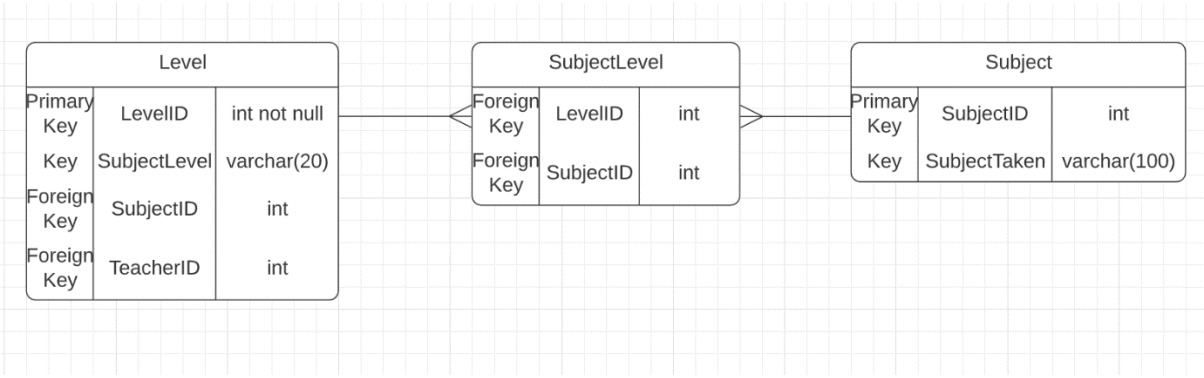
4) Student- level:

- (many-to-one)
- One student can study a subject at one level and one level has many students



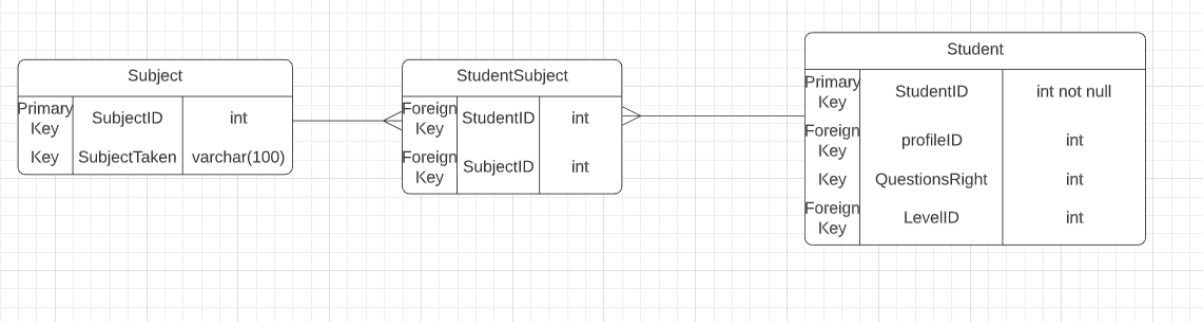
5) Level- subject:

- (many-to-many)
- One level, for example GCSE, has many subjects and one subject can be studied at many levels



6) Student-subject:

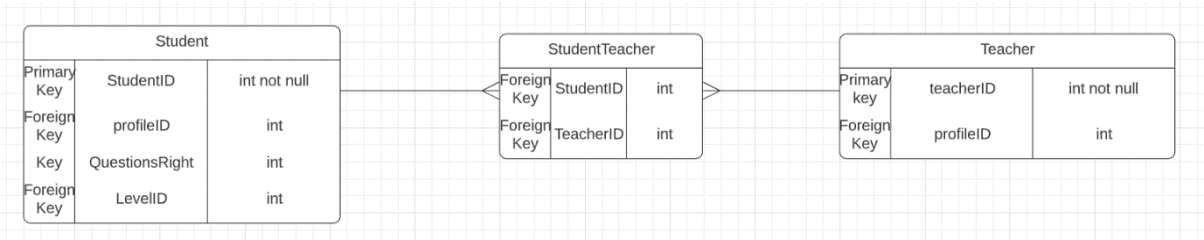
- (many-to-many)
- One student takes many subjects, and one subject is studied by many students.



Since this application is aimed towards classical subjects, the values in the subject table will be inserted beforehand and not by the user (Latin, Greek, Ancient History, Classical Civilization).

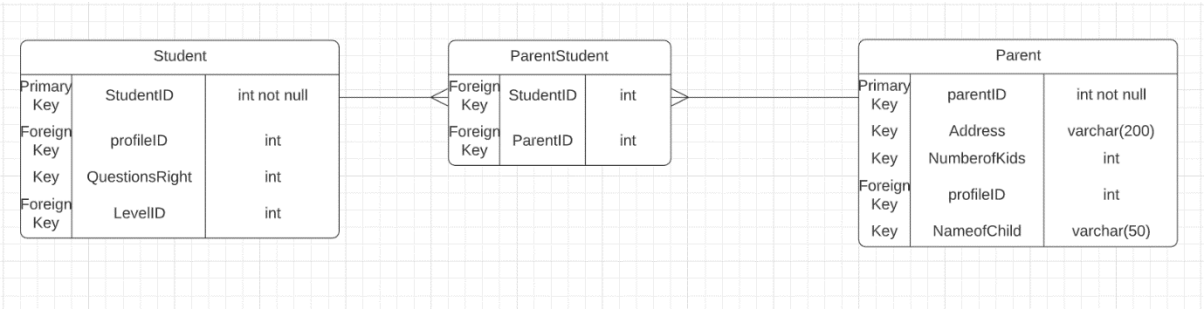
7) Student- teacher:

- (many-to-many)
- One student can be taught by many teachers and one teacher teaches many students



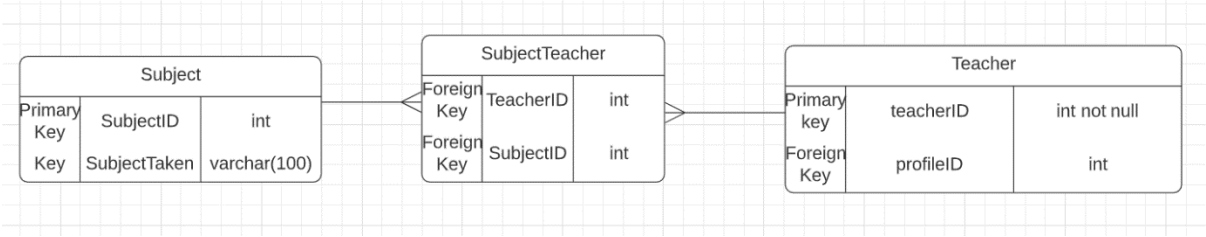
8) Student- parent:

- (many-to-many)
- One student has more than one parent (this is so that the application could be extended to guardians as well not only parents depending on their situation) and one parent has more than one child



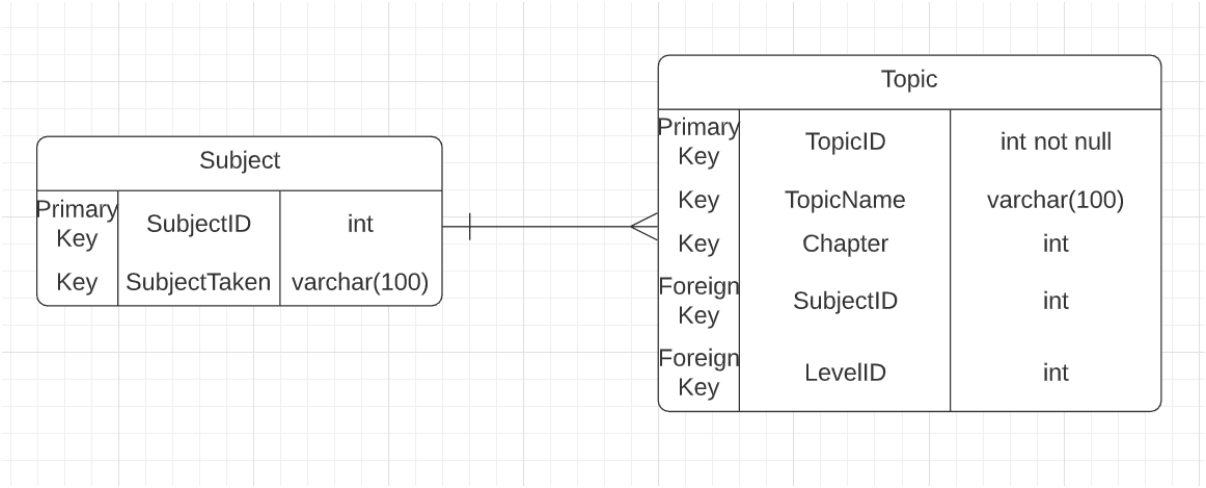
9) Subject- teacher:

- (many-to-many)
- One subject is taught by more than one teacher and one teacher can teach more than one subject



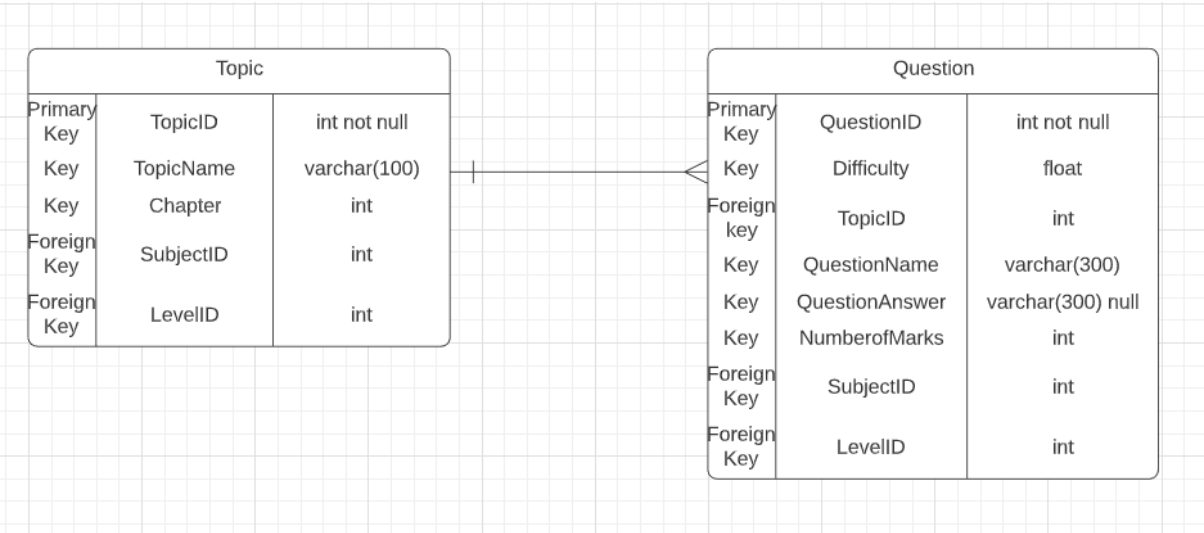
10) Subject- topic:

- (many-to-one)
- One subject has many topics, but one topic belongs to one subject



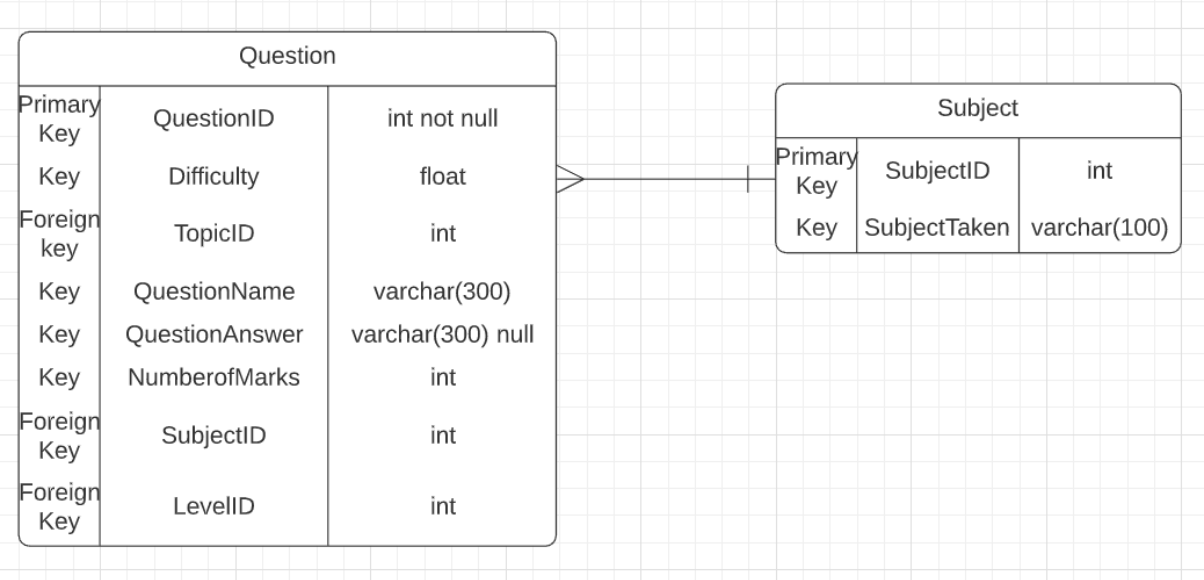
11) Topic- question:

- (one-to-many)
- One topic has many questions, and one question belongs to one topic.



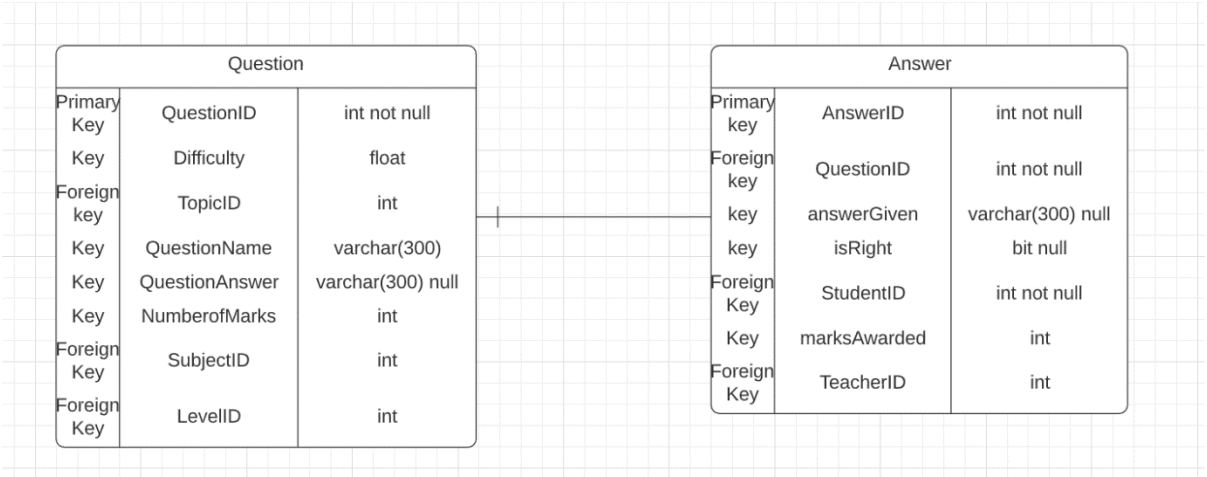
12) Subject- question

- (one-to-many)
- One subject has many questions, and one question belongs to one subject only



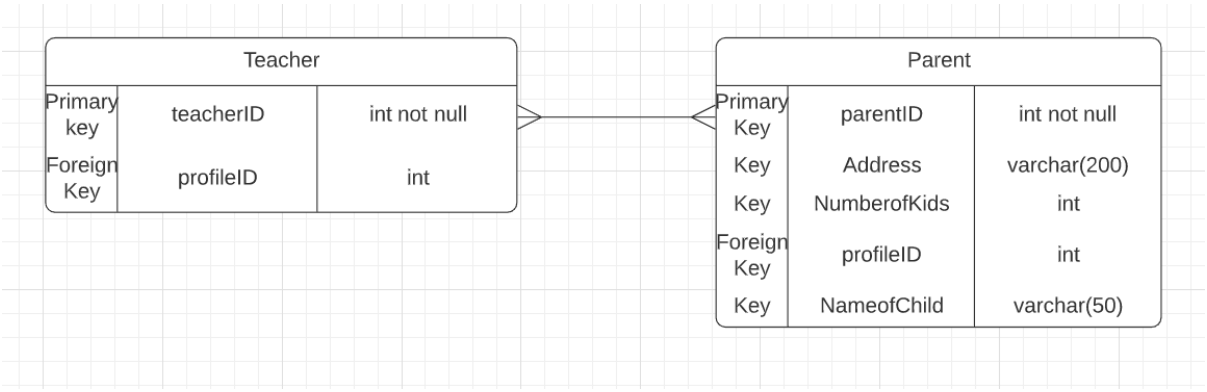
13) Question- answer:

- (one-to-one)
- One question has one answer submitted and one answer belongs to one question



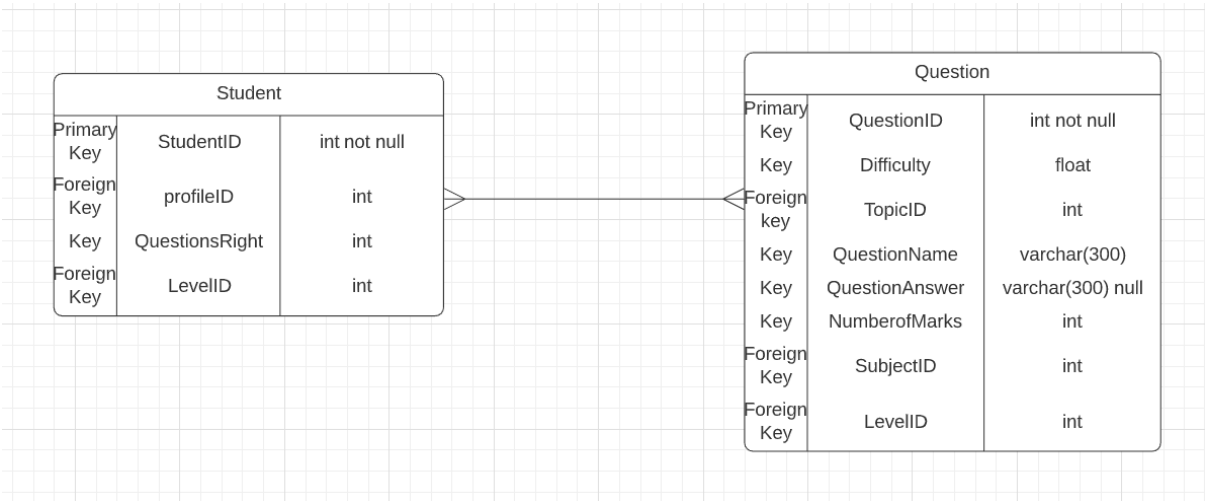
14) Teacher- parent:

- (many-to-many)
- One teacher communicates with many parents and one parent communicates with many teachers



15) Question- student

- (many-to-many)
- One question is answered by many students and one student answers many questions



16) Comment- student, parent, teacher:

- (many-to-one)
- One comment can be written either by one student, parent or teacher which can write many comments

All possible combinations are:

- 1) Student' comment (feedback) regarding how they have found the application (stored in the database)
- 2) Student's enquiry to their teacher about the questions
- 3) Teacher's comment to all students regarding their work
- 4) Teacher's comment to a parent regarding their child
- 5) Parent's communication with the teacher

4):
And to do this, I will add two int columns to the database table dbo.Comment, one called sender id, which will be the user's profile id and one receiver id which will be the profile id of the receiver and will be passed to the constructor of the form comment as a priority queue. And the receiver id will be retrieved as a list of values using an inner join command.

However, once this has been done for one type of user, it is not necessary to repeat the process for other combinations as the method to do this has already been demostrated once.

Regardless of whether the database is normalised or not, the link tables are going to be the same.

Logical sequence of data when each user registers (which information goes in which table and when):

- The first user to register should be the teacher as the student should have a class code, although this is not necessary, but it is more logical. If the student registers first, they can add their class code later.

1) Teacher id stored on teacher table, and department
2) The information provided (class code) will be stored on the class code table
3) The link table subject teacher will hold the teacher id and the subject id for the chosen subject to know how many teachers there are for each subject

- Student registration:

4) Student id stored on student table, and level id
5) Student id and subject id are held in link table student subject
6) Link table student teacher

- Parent registration, after the student is registered

7) Link table parent student

Database normalisation (following the techniques from the A Level textbook for unit 2):

The database is normalised because:

There are no repeating groups:

All attributes are retrieved by a query as a simple value as there are no repeating groups, no row or column contains multiple values.

Values in cells are atomic:

Each value is as simple as it can be, cannot be broken down further without losing meaning.

The link tables mean that there are no repeating attributes as entities are fully normalised from which fully normalised relations follow. Therefore, the database is consistent.

This means that it allows all possible relationships and avoids unnecessary duplication which avoids wasting spaceand does not increase run-time unnecessarily (no redundant data).

The database is normalised to third normal form because it is in first and second normal form.

First normal form:

Every non primary key attribute is functionally dependent on the primary key:

- Repeating groups removed
- Partial dependencies removed (second normal form)
- Transitive dependencies removed (third normal form) → all attributes are dependent on the key only (except link tables which do not need a primary key)

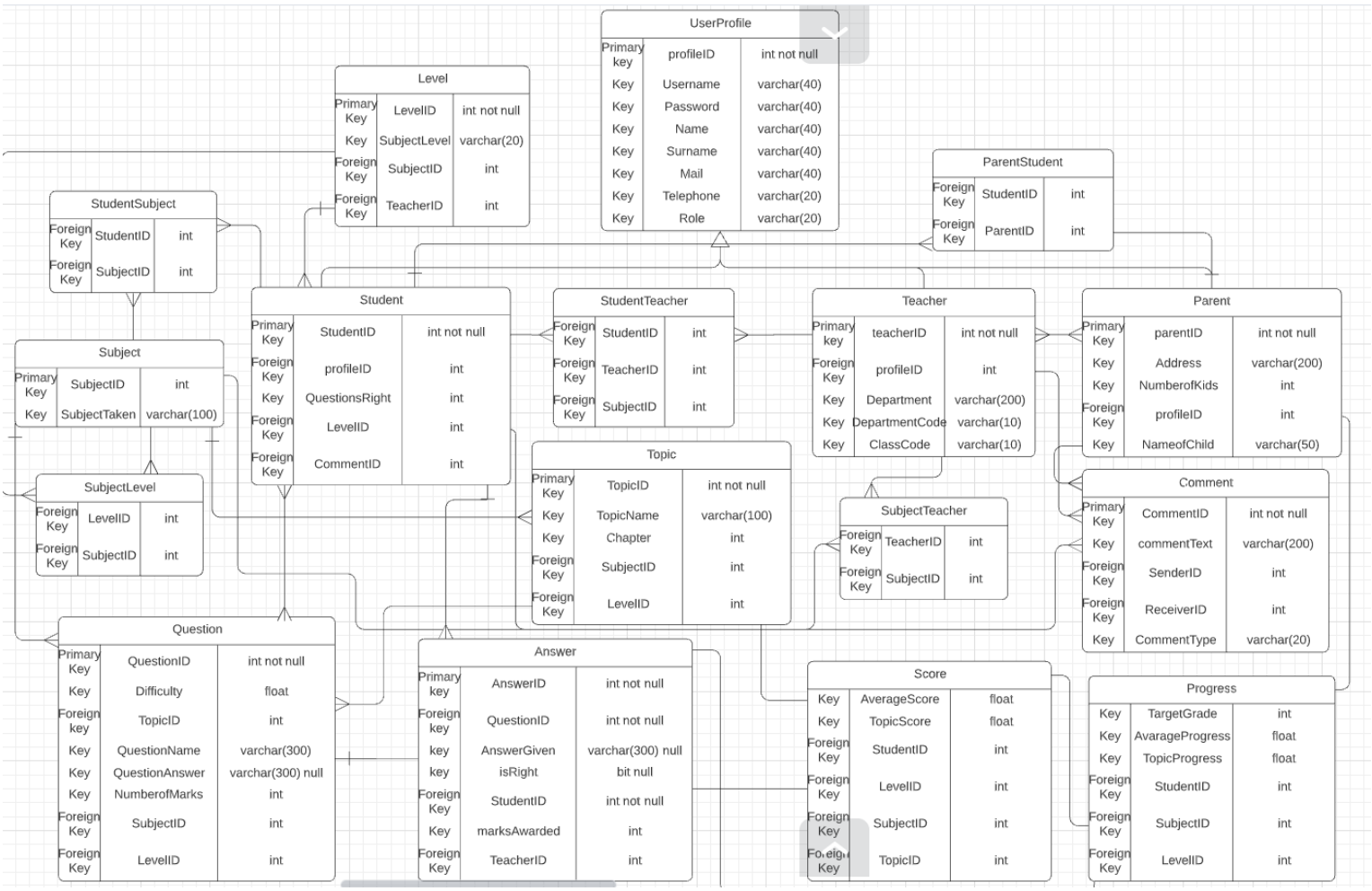
Second normal form:

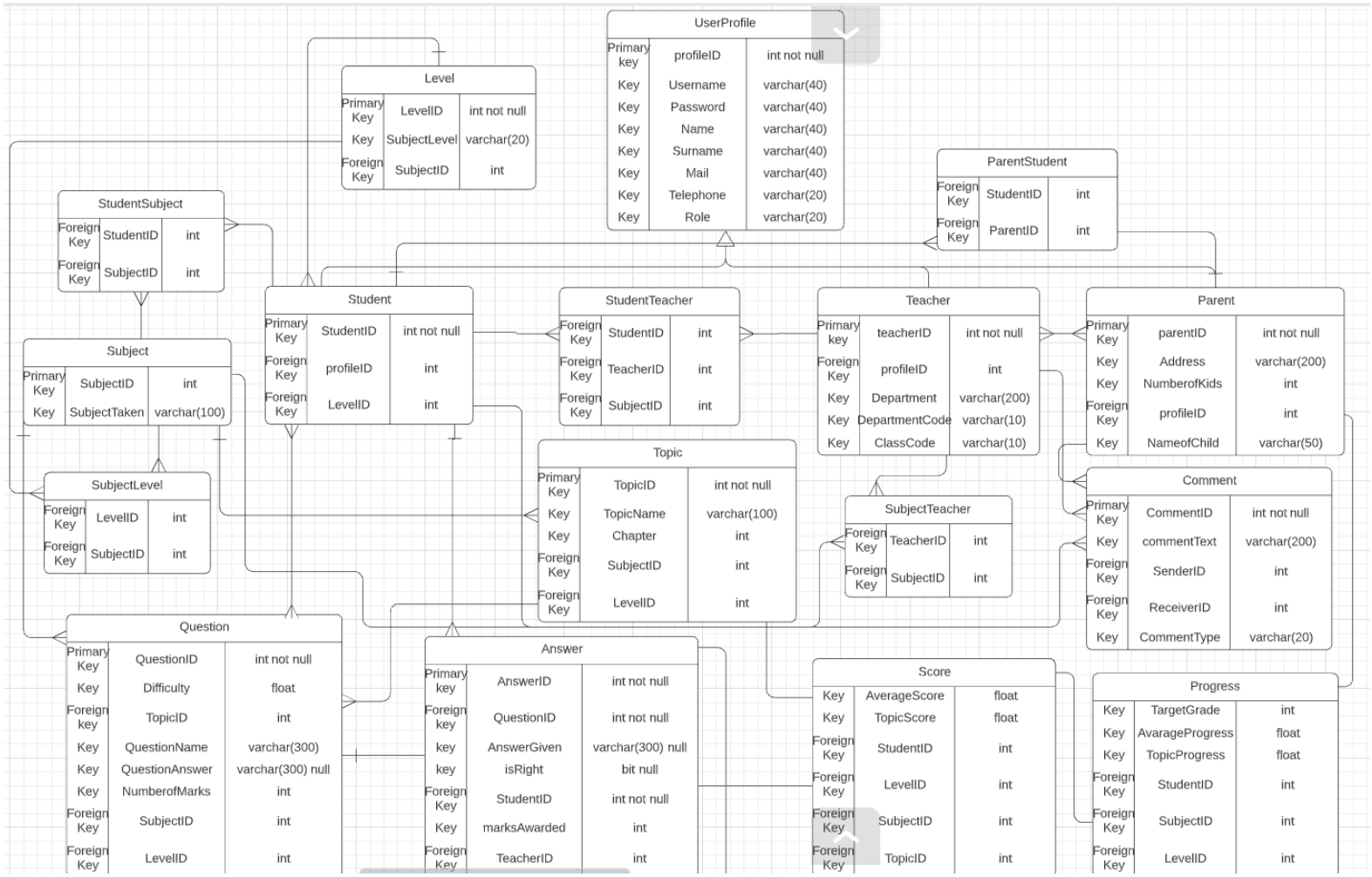
There are no non primary key attributes that are not a fact about the whole primary key to separate relations to get there → attributes can be retrieved by using given primary keys.

Third normal form:

There are no non primary key attributes that are both a fact about the key and other non key attributes to separate relations to get there → every non key attribute is a fact about the key.

Final normalised design of the database:


UserProfile
Primary key profileID int not null
Key Username varchar(40)
Key Password varchar(40)
Key Name varchar(40)
Key Surname varchar(40)
Key Mail varchar(40)
Key Telephone varchar(20)
Key Role varchar(20)
Level
Primary Key LevelID int not null
Key SubjectLevel varchar(20)
Foreign Key SubjectID int
Foreign Key TeacherID int
ParentStudent
Foreign Key StudentID int
Foreign Key ParentID int
StudentSubject
Foreign Key StudentID int
Foreign Key SubjectID int
Student
Primary Key StudentID int not null
Foreign Key profileID int
Key QuestionsRight int
Foreign Key LevelID int
Foreign Key CommentID int
StudentTeacher
Foreign Key StudentID int
Foreign Key TeacherID int
Foreign Key SubjectID int
Teacher
Primary key teacherID int not null
Foreign Key profileID int
Key Department varchar(200)
Key DepartmentCode varchar(10)
Key ClassCode varchar(10)
Parent
Primary Key parentID int not null
Key Address varchar(200)
Key NumberofKids int
Foreign Key profileID int
Key NameofChild varchar(50)
Subject
Primary Key SubjectID int
Key SubjectTaken varchar(100)
Topic
Primary Key TopicID int not null
Key TopicName varchar(100)
Key Chapter int
Foreign Key SubjectID int
Foreign Key LevelID int
SubjectLevel
Foreign Key LevelID int
Foreign Key SubjectID int
SubjectTeacher
Foreign Key TeacherID int
Foreign Key SubjectID int
Comment
Primary Key CommentID int not null
Key commentText varchar(200)
Foreign Key SenderID int
Foreign Key ReceiverID int
Key CommentType varchar(20)
Question
Primary Key QuestionID int not null
Key Difficulty float
Foreign key TopicID int
Key QuestionName varchar(300)
Key QuestionAnswer varchar(300) null
Key NumberofMarks int
Foreign Key SubjectID int
Foreign Key LevelID int
Answer
Primary key AnswerID int not null
Foreign key QuestionID int not null
key AnswerGiven varchar(300) null
key isRight bit null
Foreign Key StudentID int not null
Key marksAwarded int
Foreign Key TeacherID int
Score
Key AverageScore float
Key TopicScore float
Foreign Key StudentID int
Foreign Key LevelID int
Foreign Key SubjectID int
Foreign Key TopicID int
Progress
Key TargetGrade int
Key AvarageProgress float
Key TopicProgress float
Foreign Key StudentID int
Foreign Key SubjectID int
Foreign Key LevelID int



Database testing:

Definition of database tables (values used for testing):

UserProfile:

profileID	Username	Password	Name	Surname	Mail	Telephone	Role
379	538 ...	a ...	Bertrand ...	Russell ...	western@. ...	123456 ...	Student ...
380	234 ...	a ...	Aristotle ...	Philosop...	ancient@ ...	9875432 ...	Student ...
381	642 ...	a ...	Plato ...	Academy...	ratio@ ...	6424532...	Student ...
382	532 ...	b ...	Seneca ...	Poems ...	latin@ ...	234134 ...	Teacher ...
383	342 ...	b ...	Cicero ...	Law ...	roman@ ...	4232345...	Teacher ...
384	566 ...	b ...	Livy ...	Greek ...	heli@ ...	3452342 ...	Parent ...

Student:

StudentID	profileID	LevelID
142	379	2
143	380	2
144	381	2

Teacher:

TeacherID	profileID	Departm...	ClassCode
88	382	Classics ...	15
89	383	Classics ...	20

Parent:

ParentID	Address	NumberofKids	NameofChild	profileID
45	9128 ...	3	abc ...	384

Topic:

TopicID	TopicName	Chapter	SubjectID	LevelID
1	Chapter 1 (cases, tenses, complex relative clauses, jussive subjunctive)...	1	10	2
2	Chapter 2 (principal parts, indirect statement, ablative absolute, time) ...	2	10	2
3	Chapter 3 (predicative dative, gerunds, and gerundives) ...	3	10	2
4	Chapter 4 (unseen translation passages, English to Latin) ...	4	10	2
5	Chapter 5 (unseen prose: Livy, scansion) ...	5	10	3
6	Chapter 6 (Readings) ...	6	10	3

Score:

TopicScore	AverageScore	StudentID	TopicID	LevelID	SubjectID
231	331	142	1	2	10
421	234	143	1	2	10
100	50	144	1	2	10

Comment:

CommentID	commentText	SenderID	ReceiverID	CommentType
13	abc ...	382	379	StudentProgress

Progress:

TargetGrade	AverageProgress	TopicProgress	StudentID	SubjectID	LevelID
A	122	112	142	10	2
B	123	5423	143	10	2
C	5643	12311	144	10	2

Level:

LevelID	subjectLevel
1	GCSE ...
2	AS Level ...
3	A-Level ...

Question:

QuestionID	QuestionName	QuestionAnswer	Number...	SubjectID	LevelID	TopicID	QuestionDifficulty
18	omnia verba semper discite, amici et amicae! ...	Always learn all the words, frien...	3	10	2	1	A
19	Romae duo annos habitabamus ...	We lived in Rome for two years ...	3	10	2	1	B
20	puer arborem antea ascenderat ...	The boy had climbed the tree b...	3	10	2	1	B
21	sonum undarum audire possum ...	I can hear the sound of the wav...	2	10	2	1	A
22	haec femina me sapientiam docuit ...	This woman taught me wisdom...	1	10	2	1	A
23	senator, vir magnae virtutis, civibus persuasit ...	The senator, a man of great virt...	1	10	2	1	C
24	rex amicus est mihi ...	The king is my friend. ...	1	10	2	1	B
25	consul plura verbis quam gladio effecit ...	The consul accomplished more ...	2	10	2	1	C
26	et nos et servi diligenter laboramus ...	Both we and the slaves are wor...	4	10	2	1	A
27	ille miles dux factus est ...	That soldier was made leader ...	1	10	2	1	A
28	hunc montem cras ascendemus ...	We shall climb this mountain to...	2	10	2	1	B
29	quinque dies contendebamus; sexto ad oram advenimus ...	We were marching for five days...	3	10	2	1	A
30	puellam quam antea conspexeram salutavi ...	I greeted the girl whom I had c...	4	10	2	1	C
31	opus diligenter facio, sed mox confecero ...	I am doing the work carefully, b...	2	10	2	1	B
32	omnis cibus iam consumptus est ...	All the food has now been eate...	2	10	2	1	A
33	si navem conspicies, statim clama ...	If you catch sight of a ship, sho...	3	10	2	1	A
34	patriam relinquebam; subito tamen revocatus sum ...	I was leaving my homeland; su...	4	10	2	1	C

Answer:

AnswerID	QuestionID	StudentID	AnswerGiven	MarksAwarded	isMarked
515	18	142	abcdef ...	3	True
516	19	143	12345 ...	1	True
517	20	144	abc123 ...	3	True
518	21	142	abcd5421 ...	1	True
519	22	143	4321 ...	4	True
520	23	144	abc76 ...	2	True

Subject:

SubjectID	SubjectTaken
9	Ancient History ...
10	Latin ...
11	Greek ...
12	Classical Civilization...

(link tables)

StudentSubject:

StudentID	SubjectID
142	10
143	10
144	10

StudentTeacher:

StudentID	TeacherID	SubjectID
142	88	10
143	88	10
144	88	10

SubjectLevel:

LevelID	SubjectID
2	10

SubjectTeacher:

TeacherID	SubjectID
88	10
89	10

ParentStudent:

StudentID	ParentID
142	45

Every query will be written and updated from microsoft sql server management studio:

Test ID (query number)	**Description**	**Expected results**	**Pass/ fail**
1	Select numberof students for one subject given subject id	As expected	Pass
2	Select number of students in the database	As expected	Pass
3	Select number of teachers for a given subject	As expected	Pass
4	Select number of students for a given level	As expected	Pass
5	Select level id for a given level	As expected	Pass
6	Select the student ids for the students in a particular class given the class code	As expected	Pass

7	Select profile id for a given hash value (ciphertext) of a username	As expected	Pass
8	Select the subject for a given subject id	As expected	Pass
9	Select number of students for a given subject	As expected	Pass
10	Select the name of a user given ciphertext of username	As expected	Pass
11	Select all questions for which the difficulty is below a certain letter	As expected	Pass
12	Select all questions for a given topic name	As expected	Pass
13	Select 5 questions at random and order by difficulty	As expected	Pass
14	Select 5 questions at random starting with the lowest difficulty	As expected	Pass
15	Update answer given by student once the teacher has marked it for a specific answer and student id	As expected	Pass
16	Select the names of the students in one class given teacher id	As expected	Pass
17	Select question, answer, marks and answer given for the questions that have been marked for a student given student id	As expected	Pass
18	Select the sum of marks awarded for a given topic and student id	As expected	Pass
19	Select sum of marks awarded given topic name and student id	As expected	Pass
20	Select sum of marks awarded for a given level id	As expected	Pass
21	Select sum of number of marks available for given topic id	As expected	Pass
22	Select total marks awarded for a given student id	As expected	Pass
23	Select average of marks awarded given student id	As expected	Pass
24	Select average marks awarded given student and topic id	As expected	Pass
25	Select all profile ids of the students in a class given teacher id	As expected	Pass

1)

```
Select count(*) from StudentSubject where SubjectID = '10'
```

114 %

Results | Messages

	(No column name)
1	3

2)

```
Select count(*) as NumStudent from Student
```

14 %

Results | Messages

	NumStudent
1	3

3)

```
Select count(*) from Subjectteacher_FK where SubjectID = (Select SubjectID from Subject where SubjectTaken = 'Latin')
```

03 %

Results | Messages

	(No column name)
1	2

4)

```
Select count(StudentID), LevelID from Student group by LevelID
```

03 %

Results | Messages

	(No column name)	LevelID
1	3	2

5)

```
Select LevelID from Level where subjectLevel = 'AS Level'
```

103 %

Results | Messages

	LevelID
1	2

6)

```
Select StudentID from StudentTeacher
inner join Teacher on Teacher.TeacherID = StudentTeacher.TeacherID and Teacher.ClassCode = '15'
```

03 %

Results | Messages

	StudentID
1	142
2	143
3	144

7)

```
Select profileID from UserProfile where Username = '538'
```

03 %

Results | Messages

	profileID
1	379

8)

```
Select SubjectTaken from Subject where SubjectID = '10'
```

103 %

Results | Messages

	SubjectTaken
1	Latin

9)

```
Select count(*) StudentID from StudentSubject where SubjectID = (Select SubjectID from Subject where SubjectTaken = 'Latin')
```

03 %

Results | Messages

	StudentID
1	3

10)

```
Select Name from UserProfile where profileID = '379'
```

103 %

Results | Messages

	Name
1	Bertrand

11)

```
Select QuestionName from Question where QuestionDifficulty < 'B'
```

103 %

Results | Messages

	QuestionName
1	omnia verba semper discite, amici et amicae!
2	sonum undarum audire possum
3	haec femina me sapientiam docuit
4	et nos et servi diligenter laboramus
5	ille miles dux factus est
6	quinque dies contendebamus; sexto ad oram adveni...
7	omnis cibus iam consumptus est
8	si navem conspicies, statim clama

12)

```
Select QuestionName from Question where TopicID =
(Select TopicID from Topic where TopicName = 'Chapter 1 (cases, tenses, complex relative clauses, jussive subjunctive)')
```

35 %

Results | Messages

	QuestionName
1	omnia verba semper discite, amici et amicae!
2	Romae duo annos habitabamus
3	puer arborem antea ascenderat
4	sonum undarum audire possum
5	haec femina me sapientiam docuit
6	senator, vir magnae virtutis, civibus persuasit
7	rex amicus est mihi
8	consul plura verbis quam gladio effecit
9	et nos et servi diligenter laboramus
10	ille miles dux factus est
11	hunc montem cras ascendemus
12	quinque dies contendebamus; sexto ad oram adveni...
13	puellam quam antea conspexeram salutavi
14	opus diligenter facio, sed mox confecero
15	omnis cibus iam consumptus est
16	si navem conspicies, statim clama
17	patriam relinquebam; subito tamen revocatus sum

13)

```
Select top 5 QuestionName from Question order by QuestionDifficulty
```

114 %

Results | Messages

	QuestionName
1	omnia verba semper discite, amici et amicae!
2	haec femina me sapientiam docuit
3	sonum undarum audire possum
4	et nos et servi diligenter laboramus
5	ille miles dux factus est

14)

```
Select top 5 QuestionName from Question order by QuestionDifficulty desc
```

114 %

Results | Messages

	QuestionName
1	senator, vir magnae virtutis, civibus persuasit
2	consul plura verbis quam gladio effecit
3	puellam quam antea conspexeram salutavi
4	patriam relinquebam; subito tamen revocatus sum
5	puer arborem antea ascenderat

15)

```
Update Answer set MarksAwarded = '2' where StudentID = '142' and AnswerID = '515'
```

114 %

Messages

```
(1 row affected)

Completion time: 2021-03-15T15:05:45.6409365+00:00
```

16)

```
Select UserProfile.Name
from ((UserProfile
inner join Student on UserProfile.profileID = Student.profileID)
inner join StudentTeacher on Student.StudentID = StudentTeacher.StudentID and StudentTeacher.TeacherID = '88')
```

114 %

Results | Messages

	Name
1	Bertrand
2	Aristotle
3	Plato

17)

```
Select Question.QuestionName, Question.QuestionAnswer, Question.NumberofMarks, Answer.AnswerGiven
from Question
inner join Answer on Question.QuestionID = Answer.QuestionID and Answer.isMarked = 'True' and Answer.StudentID
= '142'
```

114 %

Results | Messages

	QuestionName	QuestionAnswer	NumberofMarks	AnswerGiven
1	omnia verba semper discite, amici et amicae!	Always learn all the words, friends male and fem...	3	abcdef
2	sonum undarum audire possum	I can hear the sound of the waves	2	abcd5421

18)

```
Select Sum(Answer.MarksAwarded) as "Marks Awarded" from Answer, Question where Question.QuestionID = Answer.QuestionID
and Question.TopicID = '1' and Answer.StudentID = '144'
```

103 %

Results | Messages

	Marks Awarded
1	5

19)

```
Select Sum(MarksAwarded)
from Answer
inner join Question on Answer.QuestionID = Question.QuestionID and Question.TopicID =
(Select TopicID from Topic where TopicName = 'Chapter 1 (cases, tenses, complex relative clauses, jussive subjunctive)')
and Answer.StudentID = '142'
```

Results | Messages

	(No column name)
1	3

20)

```
Select Sum(MarksAwarded)
from Answer
inner join Question on Question.QuestionID = Answer.QuestionID and Question.LevelID = '2'
```

Results | Messages

	(No column name)
	13

21)

```
Select Sum(NumberofMarks) as "Marks" from Question where TopicID = '1'
```

Results | Messages

	Marks
1	41

22)

```
Select Sum(MarksAwarded) from Answer where StudentID = '142'
```

Results | Messages

	(No column name)
1	3

23)

```
Select Avg(MarksAwarded) from Answer where StudentID = '142'
```

Results | Messages

	(No column name)
1	1,5

24)

```
Select Avg(MarksAwarded)
from Answer
inner join Question on Question.TopicID = '1' and Answer.StudentID = '143'
```

103 %

Results | Messages

	(No column name)
1	2,5

25)

```
Select UserProfile.profileID
from UserProfile
inner join Student on Student.profileID = UserProfile.profileID
inner join StudentTeacher on StudentTeacher.StudentID = Student.StudentID and StudentTeacher.TeacherID = '88'
```

03 %

Results | Messages

	profileID
1	379
2	380
3	381

Every query used in the program and to show the purpose of the database:

```
select count (*) from StudentSubject where SubjectID = '6';
--- query to show the number of students for a particular subject

select count (*) as NumStudent from Student
--- select the total number of students in the database

select count (*) from Subjectteacher_FK where SubjectID =(select SubjectID from Subject where SubjectTaken = 'Latin');
--- show the number of teachers for a given subject

select count (StudentID), LevelID from Student group by LevelID;
--- select every student for each level
```

```
select LevelID from Level where subjectLevel = 'A-Level'
--- select LevelID for a given level to insert into student table
```

```
SELECT StudentID FROM StudentClass WHERE ClassID = (SELECT ClassID FROM ClassCode WHERE CLassCode = '9')
--- select student id to see which class they are in (from link table student class) for a given class code
```

```
select profileID from UserProfile where Username = 'calculus'
--- select profileID for a given username (to insert in either tables student parent or teacher

select SubjectTaken from Subject where SubjectID = '5'
--- select the name of the subject for a given ID

select count(*) StudentID from StudentSubject where SubjectID =
(select SubjectID from Subject where SubjectTaken = 'Greek')
--- select total number of students for a given subject

select Name from UserProfile where profileID = '8'
--- select name for a given profileID
```

```
select QuestionName from Question where Difficulty < 7
--- select the question for which the difficulty is below a given number
```

```
Select QuestionName from Question where TopicID = (Select TopicID from Topic where TopicName
= 'imperfect subjunctive, complex sentences, cum clauses')
--- select the question for a given topic
```

```
SELECT TOP 5 QuestionName FROM Question
ORDER BY Difficulty
--- select 5 questions at random, ordered by their difficulty
```

```
CREATE PROCEDURE dbo.priority
AS
BEGIN
SELECT 1 QuestionName FROM Question
ORDER BY QuestionDifficulty DESC
DELETE FROM Question
OUTPUT DELETED.*;
END
GO
```

```
select top 5 QuestionName from Question
order by QuestionDifficulty desc
--- select 5 questions at random of increasing difficulty
```

```
--- update the table 'answer' when the teacher has marked a certain question
--- insert the number of marks awarded to a particular field for a given studentID
update Answer set MarksAwarded='2' where StudentID = '4'
```

```
--- select each student's name in the class for a given teacher id
Select UserProfile.Name
from ((UserProfile
inner join Student on UserProfile.profileID = Student.profileID)
inner join StudentTeacher on Student.StudentID = StudentTeacher.StudentID and StudentTeacher.TeacherID = '39'
```

The query below also selects only the questions that have not been already marked:

```
--- select the question answered and the original answer to be compared with the answer given by the student and the marks
Select Question.QuestionName, Question.QuestionAnswer, Question.NumberofMarks, Answer.AnswerGiven
from Question
inner join Answer on Question.QuestionID = Answer.QuestionID and Answer.isMarked = 'False' and Answer.StudentID = '111'
```

```
--- aggregate sql function sum() to select the added total of the marks awarded for the questions answered by one student for a given topic
Select Sum(Answer.MarksAwarded) as "Marks awarded" from Answer, Question where Question.QuestionID = Answer.QuestionID
and Question.TopicID = '1' and Answer.StudentID = '112'
```

```
--- select the sum (aggregate SQL) of marks awarded for a chosen topic for a given student using inner join
Select  Sum(MarksAwarded)
from Answer
inner join Question on Answer.QuestionID = Question.QuestionID
and Question.TopicID = (Select TopicID from Topic where TopicName = ('Chapter 1 (cases, tenses, complex relative clauses, jussive subjunctive)'))
and Answer.StudentID = '112'
```

```
Select Sum(MarksAwarded)
from Answer
inner join Question on Answer.QuestionID = Question.QuestionID
and Question.TopicID = (Select TopicID from Topic where TopicName = ('Chapter 1 (cases, tenses, complex relative clauses, jussive subjunctive)'))
and Answer.StudentID = (Select StudentID from Student where profileID = (Select profileID from UserProfile where Name = 'ABC1'))
```

```
--- target defined as the individual score across each topic as a percentage of all scores achieved by each student for that level
--- select the total number of marks awarded for each student of a certain level eg. AS Level
Select Sum(MarksAwarded)
from Answer
inner join Question on Question.QuestionID = Answer.QuestionID and Question.LevelID = '2'
```

```
Select Sum(NumberofMarks) as 'Marks' from Question where TopicID = '1'

Select Sum((Answer.MarksAwarded / '" + markResult + "') * 100) as 'Avg' from Answer where StudentID = '112'
```

Or, the target grade could be user input from the teacher and the value above could be the average progress field in the sql table progress and target grade user input.

```
--- divide the value below which is the total number of marks scored by a student by the value above and multiply by 100 (in the program)
Select Sum(MarksAwarded) from Answer where StudentID = '112'
```

```
Select Avg(MarksAwarded)
from Answer where StudentID = '112'
```

```
--- select the average of the questions answered by a student for a particular topic or in general
Select Avg(MarksAwarded)
from Answer
inner join Question on Question.TopicID = '1' and Answer.StudentID = '112'
```

```
--- select the profile id of the students in a class for a given teacher id
Select UserProfile.profileID
from UserProfile
inner join Student on Student.profileID = UserProfile.profileID
inner join StudentTeacher on StudentTeacher.StudentID = Student.StudentID and StudentTeacher.TeacherID = '40'
```

The last queries have different backgrounds because I wrote them in visual studio rather than in microsoft sql server.

Efficiency:

The aspect of the project which makes the program most efficient as possible is the database as it is central to the windows forms application. The database is normalized to occupy the least amount of space as possible in RAM by avoiding repeating attributes in extra tables where these attributes are not needed. The first example of this is each type of user's name, the concept of inheritance can be used for this because, by having the table user profile, this avoids having an attribute which stores the name of the user all in the student, parent and teacher tables which saves space.
Link tables such as student teacher which says which students are in which class, helps to save space by holding foreign keys to identify a many-to-many relationship.
This means that the design of the database ensures the program is the most efficient it can be. The use of the database itself for this type of project makes the program more efficient because not using a database would imply storing usernames, passwords and questions in arrays or array lists, or even queues which would significantly increase run time and the storage occupied in RAM which would decrease efficiency.

Security:

To keep the users' login credentials safe, I will use a hash function from the A-Level textbook (in pseudo-code) and will use it in C#. I will use this hashing algorithm to keep the username safe so that it cannot be recognised or guessed by looking at the tables in the database.

The pseudo code for the hashing algorithm is:

```
Create procedure hashUsername(string username)
    int sum → 0
    string hash → ""
    StringBuilder sb → new StringBuilder()
    byte[] ascii = Encoding.ASCII.GetBytes(username)
    foreach byte b in ascii
        sb.Append(b)
        hash = sb.ToString()
    end foreach

    for i → 0 i  to sb.Length
        sum += hash[i] * hash[i]
    end for

    int x = sum % 523
    return Convert.ToString(x)
end procedure
```

The efficiency of this algorithm is $O(n^2)$ because… this means that as the value of the username gets larger, the n^2 term increases faster than other terms (stringbuilder).

Deriving Big-O (four assignments, 2 for loops (statements) and strinbuilder grows exponentially).
$4+a^n+2n^2$

Permutations	i	sum	sb	username	X (hash value of username)
1	2	6274	97	a	521
2	4	12548	9797	aa	519
3	6	19157	979899	abc	329
4	2	6385	98	b	109
5	15	42442	979899100495051	abcd123	79
6	21	58217	999710899117108117115	calculus	164

In this case, the x value is the length of the stringbuilder sb (so int i) → as the value of the username gets larger, x tends to infinity by polynomial order

Method to deal with collisions:

This hashing algorithm provides a number which hides the value of the username selected by the user. However, if a user were to choose a username with the same letters in a different order.

- A user cannot select a username that is already in use, to avoid this, the system checks that the hash value of the username chosen does not match any of the ones in the database
- However, it is clear that the subroutine hashUsername(string) could return a value identical to the previous one if the argument given to the constructor contains the same letters in a different order which means that the use has not chosen a username that is already in use but that there has been a hashing collision

One method to solve this is to use rehashing (linear rehash):

The rehash function is defined as:
Address ← (address + 1) mod 523

In this case, hash[i+1] mod 523

To do this, I will need to store the actual usernames in a local datatable which will contain primary keys (hash addresses) and the actual username string.

This datatable will be used when the hash value of a username matches the new one selected by the user → if the usernames are different, the system will apply the rehashing function to the new username.

Issues and their significance:

Values hashing to same address/ username school:
The first issue is that if a user selects a username which has the same letters in a different order to one that has been already selected, the system will recognise them as the same username because they both produce the same result.

For example:

```
0 references
static void Main(string[] args)
{
    Console.WriteLine("Enter username");
    string h = Console.ReadLine();
    int sum = 0;
    string hash = string.Empty;
    StringBuilder sb = new StringBuilder();
    byte[] ascii = Encoding.ASCII.GetBytes(h);
    foreach (byte b in ascii)
    {
        sb.Append(b);
        hash = sb.ToString();
    }

    int i;

    for (i = 0; i < sb.Length; i++)
    {
        sum += hash[i] * hash[i];
    }

    int x = sum % 523;
    Console.WriteLine("The hash value of the username is: " + x);
}
```

```
Microsoft Visual Studio Debug ...
Enter username
abc
The hash value of the username is: 329

C:\Users\diego\source\repos\ConsoleApp1\Co
oleApp1\bin\Debug\netcoreapp3.1\ConsoleApp
exe (process 17200) exited with code 0.
To automatically close the console when de
gging stops, enable Tools->Options->Debugg
g->Automatically close the console when de
gging stops.Press any key to close this wi
ow . . .
```

```
0 references
static void Main(string[] args)
{
    Console.WriteLine("Enter username");
    string h = Console.ReadLine();
    int sum = 0;
    string hash = string.Empty;
    StringBuilder sb = new StringBuilder();
    byte[] ascii = Encoding.ASCII.GetBytes(h);
    foreach (byte b in ascii)
    {
        sb.Append(b);
        hash = sb.ToString();
    }

    int i;

    for (i = 0; i < sb.Length; i++)
    {
        sum += hash[i] * hash[i];
    }

    int x = sum % 523;
    Console.WriteLine("The hash value of the username is: " + x);
}
```

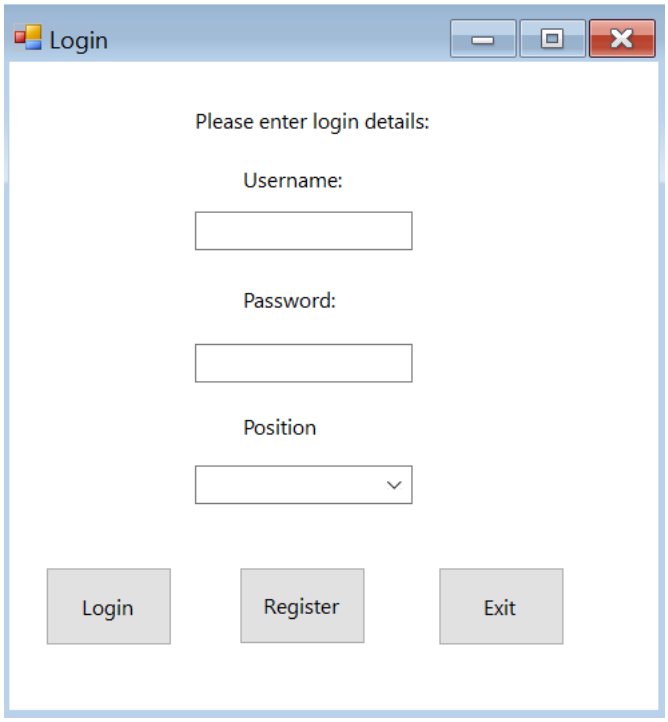
```
Enter username
bca
The hash value of the username is: 329

C:\Users\diego\source\repos\ConsoleApp1
oleApp1\bin\Debug\netcoreapp3.1\Console
exe (process 16876) exited with code 0.
To automatically close the console when
gging stops, enable Tools->Options->Deb
g->Automatically close the console when
gging stops.
Press any key to close this window . .
```

However, if a user were to guess a username, they would still need to know the password which is why this is not a very significant problem. In school login, for example, where the username is one's name, anyone can know another username but not their password. And since this is a school app and not a bank account, the focus is not on the risk of username thefy but on the database. Hence, this does not diminish the purpose of the program.

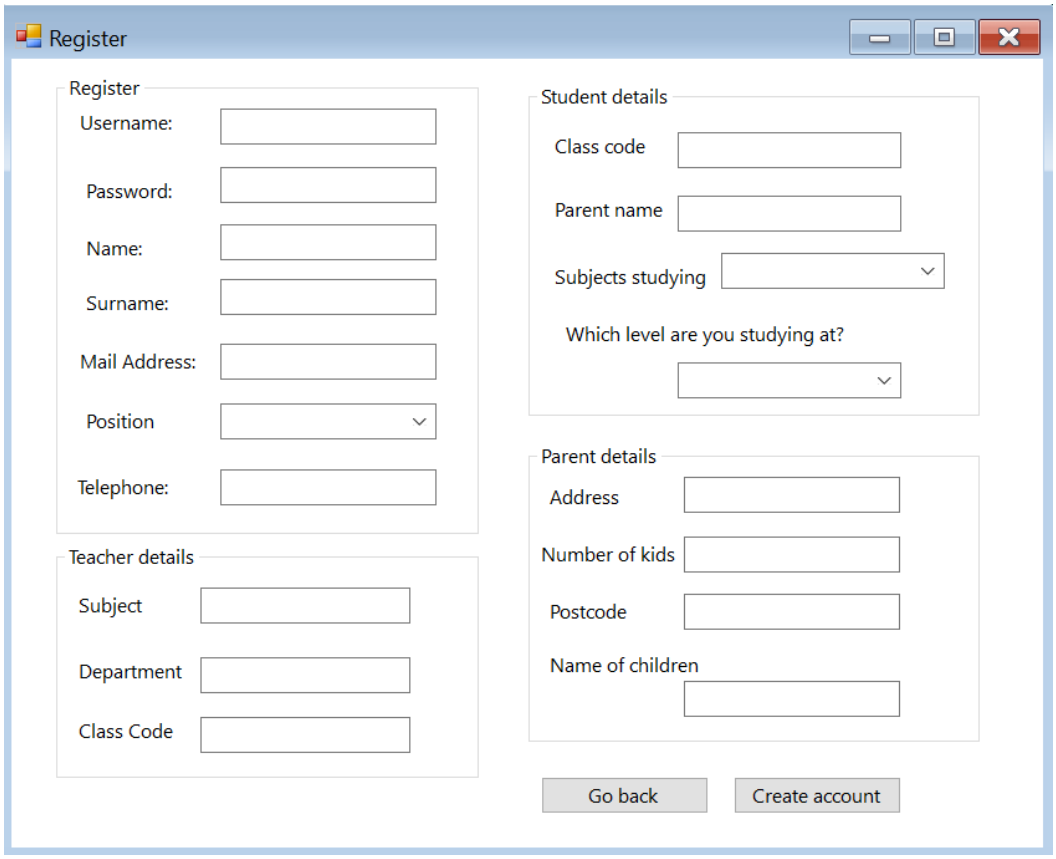
User interface design:

The program will use windows forms, data will be stored on Microsoft SQL Server Management Studio database.
Users will use the app at one time on one application.
The initial form that the system will open when the user opens the program will look like this, if this form is closed, the rest of the application will close as well:



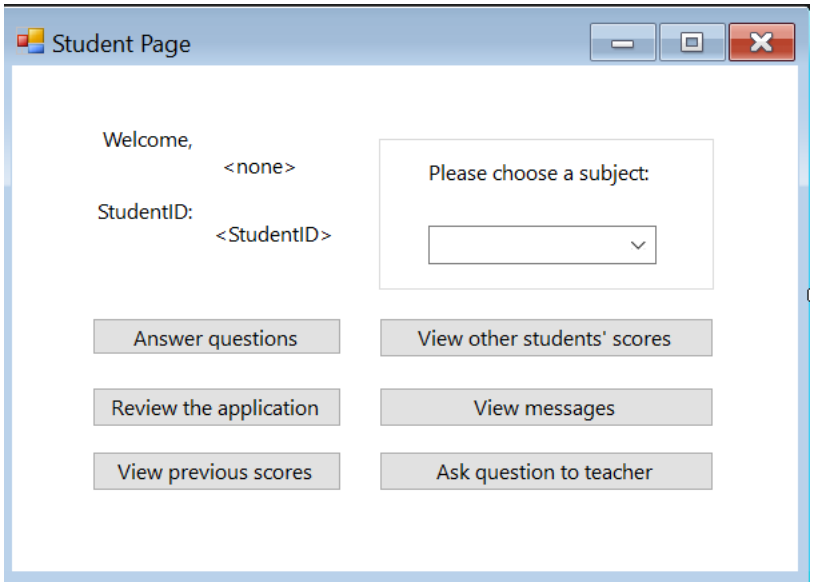
The exit button will exit the application.

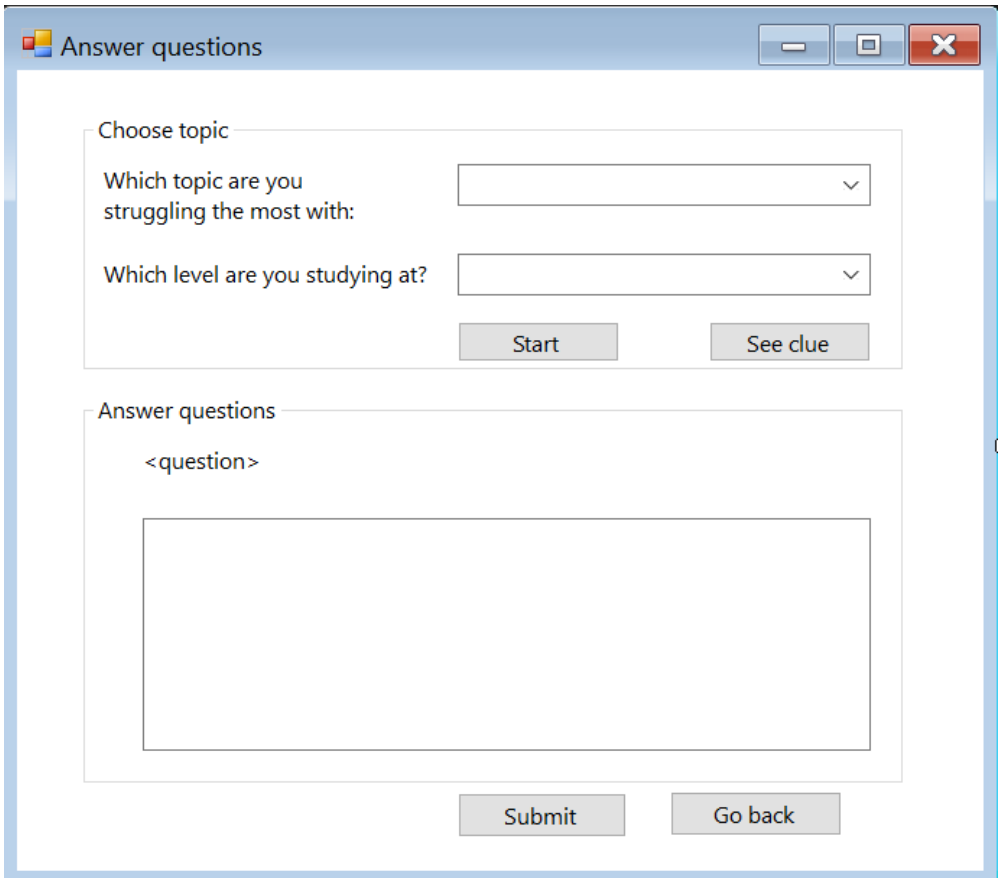
If the user clicks the register button, this form will open which will add their login details to the database unless the username already exists in the database (which will cause the system to display an error):



For example, if the user is a student, the group boxes parent and teacher will be disabled. Having all fields in the same form means username and password are not repeated on different forms and inserting the same attributes from more than one form on the same table. Once they have entered their login details, they can click the create account button to create a new profile in the database to have access to their page when they login. If the login is successful, this form will not be opened.

If the login details are correct and the option student is selected as the role, this form will open, and this is the first thing the system will ask before they are asked questions:





Alternative way to display questions on the form (using a timer instead) but isn't used in the program as students can spend the amount of time they wish on a question:

```
SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("Select QuestionName, NumberofMarks
from Question where TopicID = (Select TopicID from Topic where TopicName = '" +
TopicComboBox.SelectedItem.ToString() + "')", sqlConnection);
            List<string> questionsSeen = new List<string> { };
```

```
            DataTable questionsTotal = new DataTable();
            sqlDataAdapter.Fill(questionsTotal);
            Queue<string> priority = new Queue<string> { };
            for (int i = 0; i < questionsTotal.Rows.Count; i++)
            {
                string question =
Convert.ToString(questionsTotal.Rows[i]["QuestionName"]);
                priority.Enqueue(question);
                questionLabel.Text = priority.Peek();

                int numMarks =
Convert.ToInt32(questionsTotal.Rows[i]["NumberofMarks"]);
                MessageBox.Show("time", (numMarks * 60).ToString());

                questionsTimer.Start();
                questionsTimer.Tick += new EventHandler(questionsTimer_Tick);

                if (questionsTimer.Interval == numMarks)
                {
                    questionsTimer.Stop();
                    MessageBox.Show("Time's up");
                    questionLabel.Text = "";
                }

                int timeAllowed = 1;
                if (MessageBox.Show("Continue running?", "count" + timeAllowed,
MessageBoxButtons.YesNo) == DialogResult.Yes)
                {

                }
            }

            SqlCommand sqlCommand = new SqlCommand("Select QuestionName from Question
where TopicID = (Select TopicID from Topic where TopicName = '" +
chooseChapterchecbox.Text + "')", sqlConnection);
            string[] array;
            int index = 0;
            array = new String[10];
            SqlDataReader reader1 = sqlCommand.ExecuteReader();
            while (reader1.Read())
            {
                array[index] = reader1.GetString(0);
                index++;
                List<string> seensQ = new List<string> { };
                string s = reader1["QuestionName"].ToString();
                seensQ.Add(s);
            }

            reader1.Close();
```

Merge sort could be used to sort questions to display on the student form:

Pseudo-code:

- Store the questions in an array
- Define a recursive subroutine called merge sort which sorts the questions and pass the array as an argument to the constructor

- The algorithm could be used to present the questions alphabetically according to the first letter of the question, or they could be stored according to the difficulty associatede with the question which is in the database
- Or, it could be used to sort the array of student names in another form

```
def mergeSort(questions):
    i ← start
    j ← middle
   index ← start
  while index < end
       if (j >= end) or (I < middle and A[i] < A[j])
         then
            B[index] ← A[i]
            i ← i + 1
         else
           B[index] ← A[j]
           j ← j + 1
       endif
       index ← index + 1
endwhile
endsubroutine
```

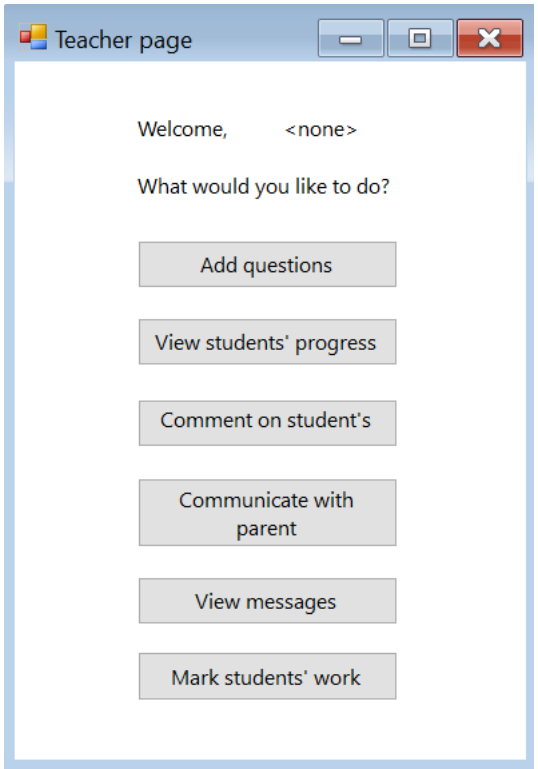
The time complexity of this algorithm is $O(n \log_2 n)$.

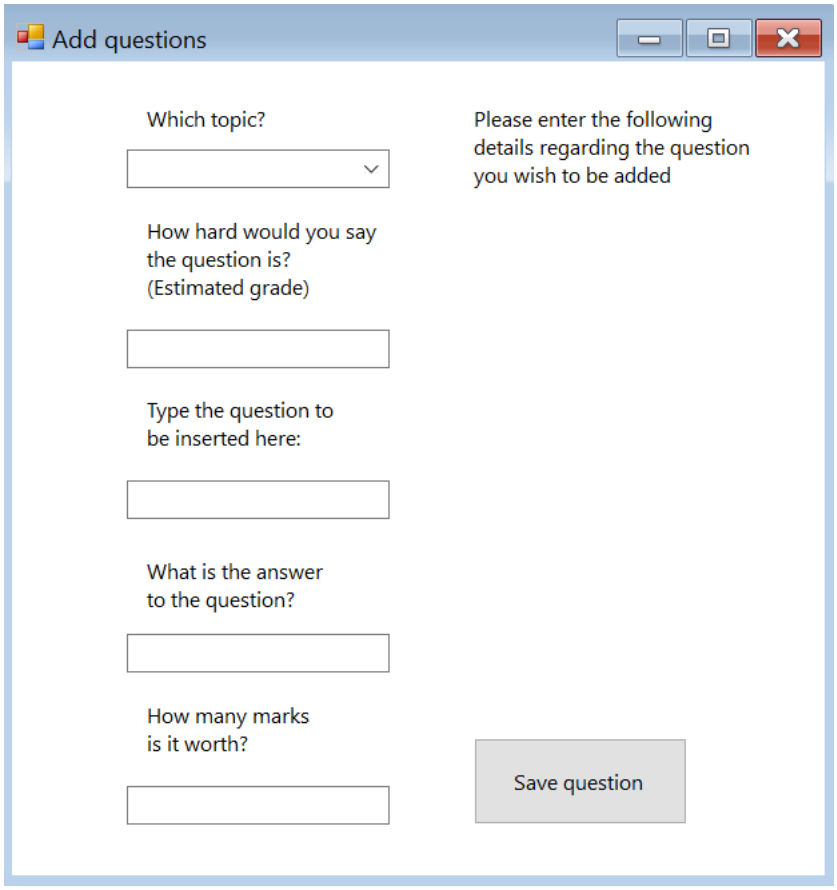
An alternative could be to use quick sort instead:

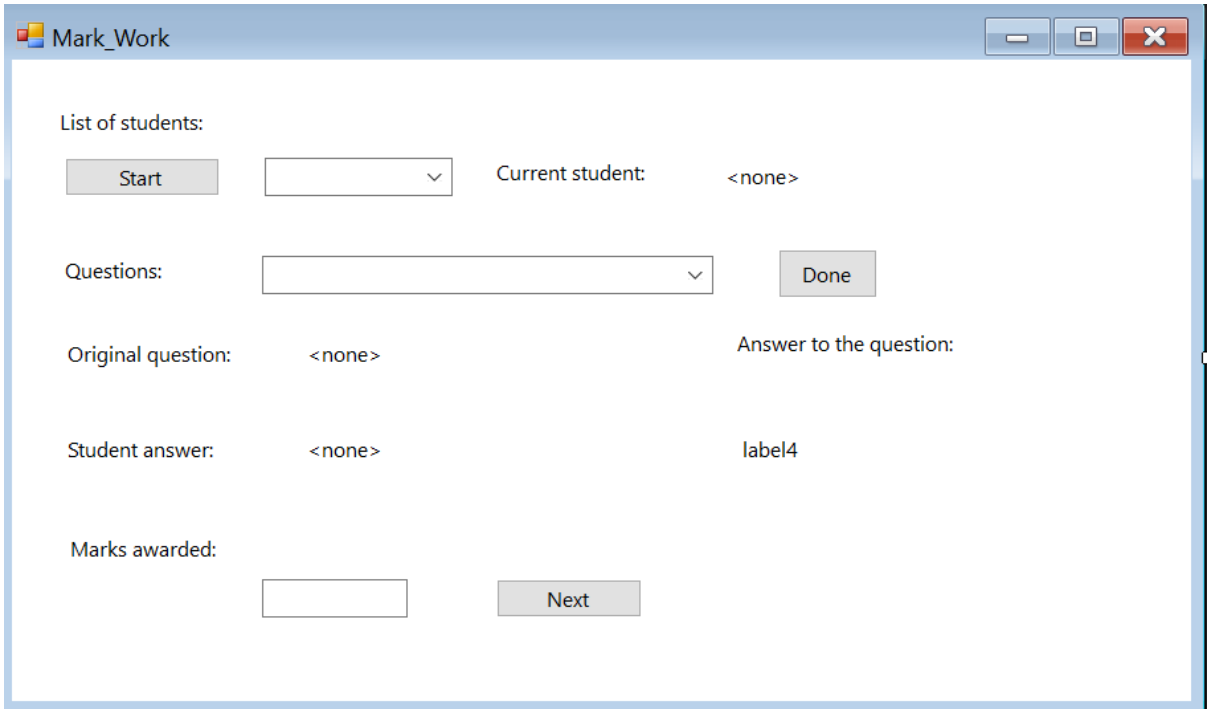
Pseudo-code:

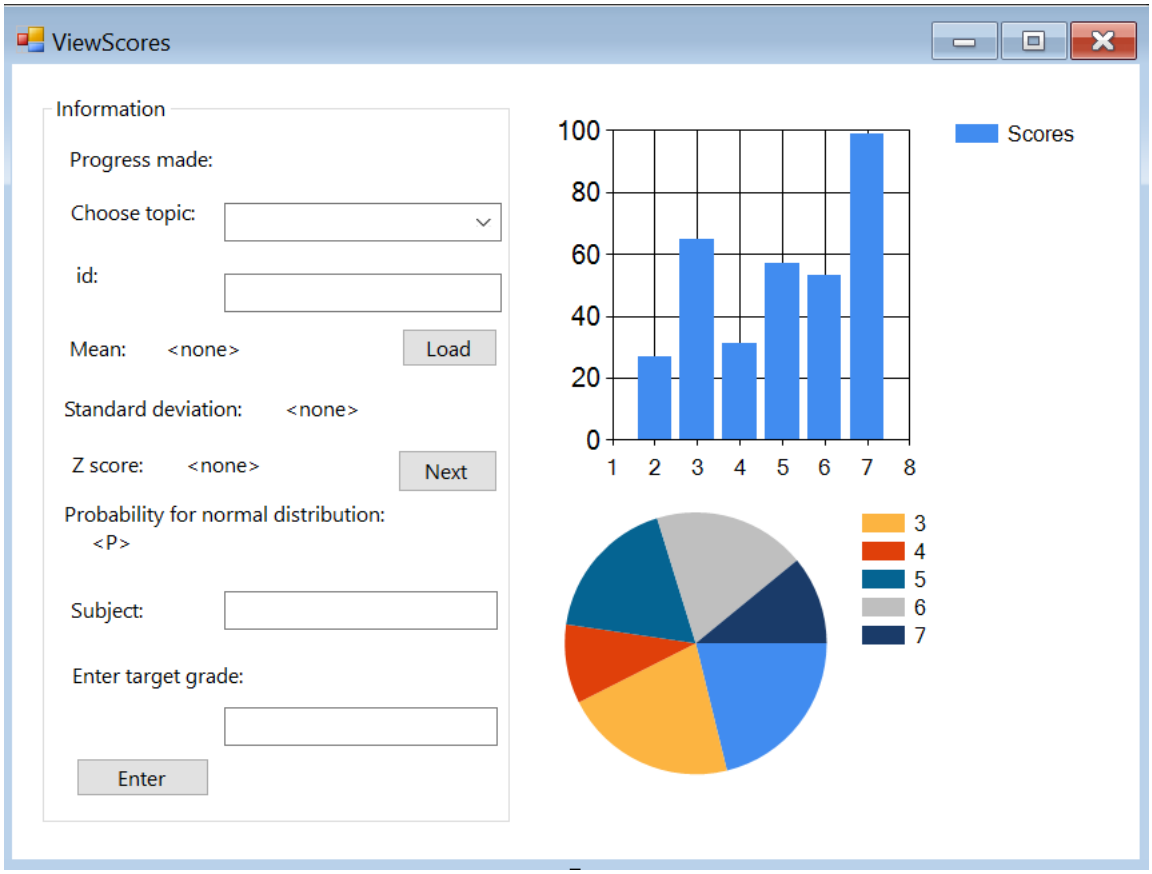
```
if (registerButton_Click)
{
    open RegisterForm
    get user details
    generate ID
    save data to database
}
if (Role == "Student")
{
    open ChooseTopicForm
    select Topic
}
if (Role == "Teacher")
{
    getSubject
}
if (Role == "Parent")
{
    getPostcode
}
```

The <none> label will display the user's name which will be retrieved from the database once the user has logged in successfully.

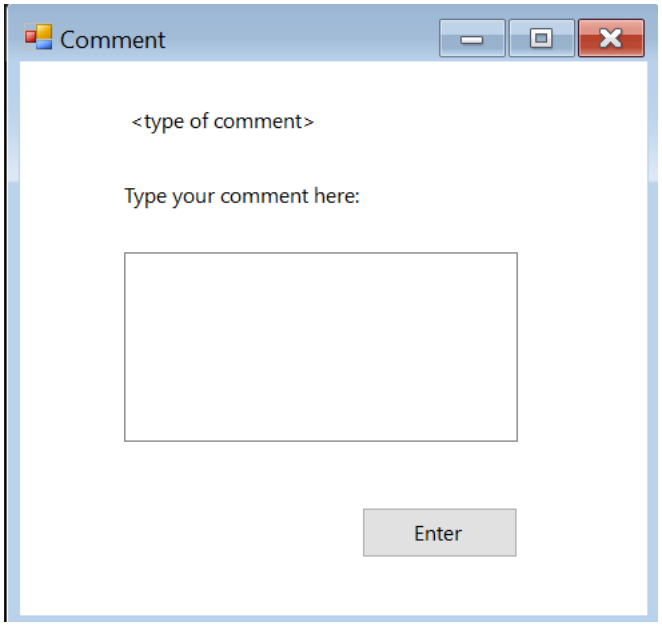


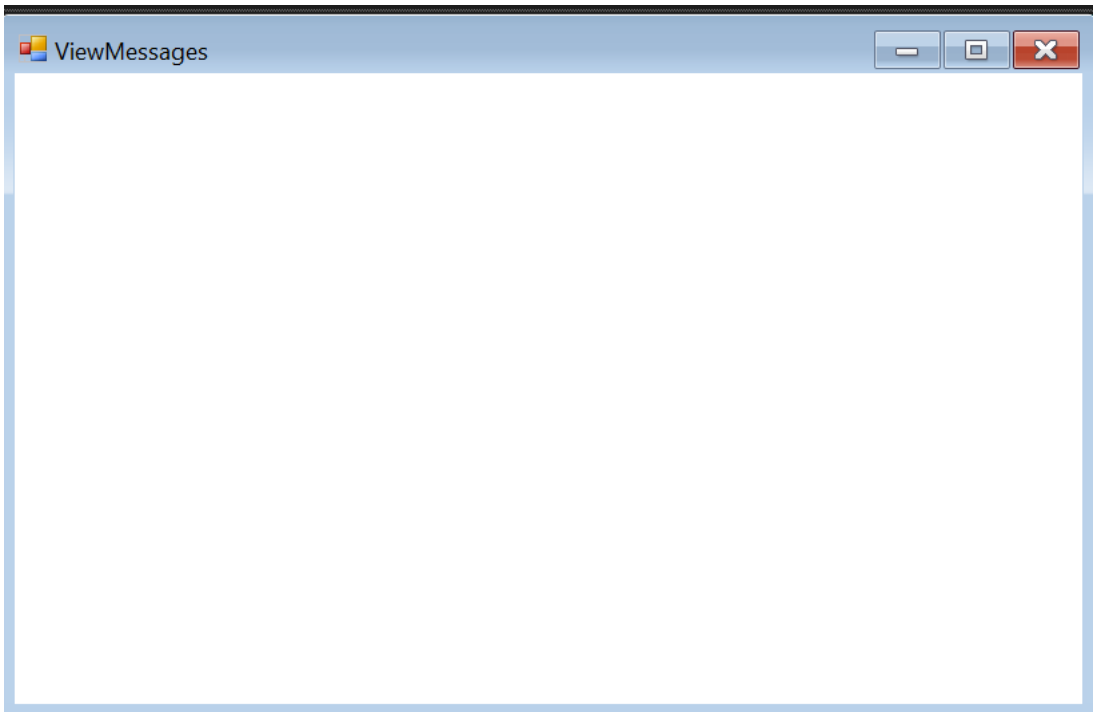

Add questions
Which topic?
Please enter the following details regarding the question you wish to be added
How hard would you say the question is? (Estimated grade)
Type the question to be inserted here:
What is the answer to the question?
How many marks is it worth?
Save question


Mark_Work
List of students:
Start
Current student:
<none>
Questions:
Done
Original question:
<none>
Answer to the question:
Student answer:
<none>
label4
Marks awarded:
Next

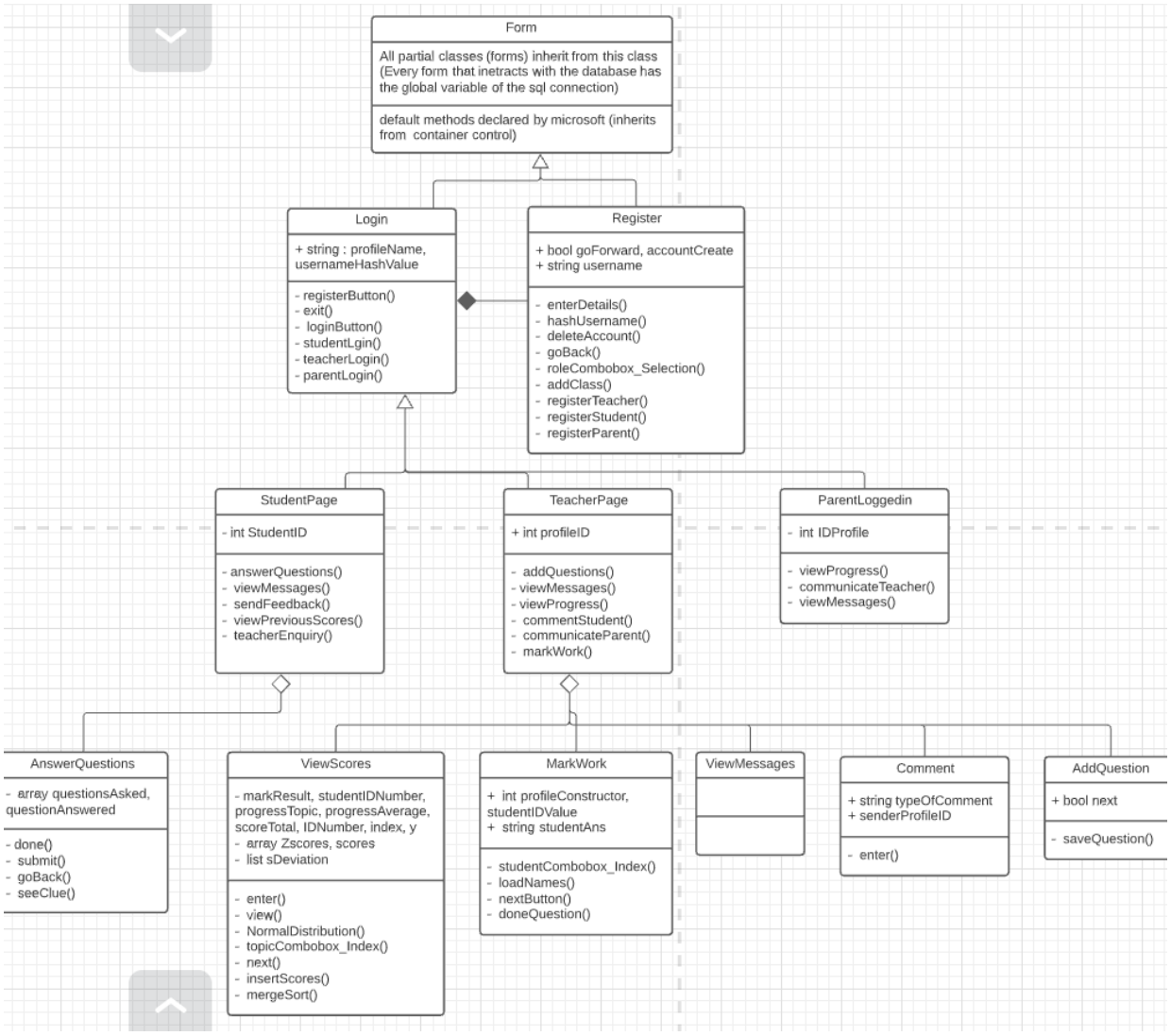

ViewScores
Information
Progress made:
Choose topic:
id:
Mean: <none>
Load
Standard deviation: <none>
Z score: <none>
Next
Probability for normal distribution:
<P>
Subject:
Enter target grade:
Enter
100
80
60
40
20
0
1 2 3 4 5 6 7 8
Scores
3
4
5
6
7


Parent page
Welcome, <none>
What would you like to do?
Communicate with teacher
View messages
View son's or daughter's progress





Class diagram for windows forms solution/ hierarchy diagram:



Techniques used:

Concept: (introduction)
I will use the normal distribution from A Level Mathematics to represent and analyse data, in this case, students' scores.

The normal distribution is a pattern of data whenever you measure a population or set of scores with some randomness.

The normal distribution is a good choice for this project because the data in the SQL tables, progress, answer, and score, is continuous (regarding students' scores). The scores are whole, positive integers represented by the set $\{S \in \mathbb{Z} \geq 0\}$ (S for scores).

- Mean, mode, and median should all be equal

Every student is going to have different scores, and some will be lower than others → since one of the objectives is to learn with which topics the student needs more help, representing

the scores graphically is going to help spot patterns in how questions are being answered, for example, when the shape of the bell curve is different and not symmetric (negatively skewed).

If there is an element with some randomness to it, you musn't get a curve which is different to the original. Patterns that aren't supposed to be, in this project, are examples when the average gets really low, which would imply that students aren't answering well .

SQL tables: score, progress, answer (x axis- mean of score(average score), y- f(x))
Answer- contains how many questions the student has got right
Score- topic/ average score
Progress- target grade, topic progress, average progress → these are measured based on the number of questions being aswered and probabilities from the normal distribution.

Formulas and data:

Mean = $\bar{x} = \frac{\sum x}{n}$ or Mean = $\bar{x} = \left(\frac{\sum f}{\sum f}\right)$

Standard deviation = $\sigma = \sqrt{\frac{\sum f x^2}{\sum f} - \left(\frac{\sum f}{\sum f}\right)^2}$

Median = middle data value when all data values are placed in order of size (scores placed in ascending order)

Mode = most frequently occurring data value (it is rare but not impossible that more than one student gets the same score, this indicates the score that most students have scored)

Standard deviation and variance measure how spread out the data is from the mean, the bigger the variance, the more spread-out readings are.

Since the normal distribution looks at the whole of the population (what every student has scored) → this is to show teachers how students in general are making progress, this is measure across 6 chapters.
Students should not be able to choose the topic on which they want to answer questions, the more the students answer questions, the more data there will be (no names).

Characteristics:
Symmetrical about μ (mean)
Asymptotic to the x axis

We can tell how high/ low a score is based on:

- How far it is above / below the mean μ
- In terms of the standard deviation σ

This measurement is called the z-score and it is defined as $z = \frac{x-\mu}{\sigma}$

the data is going to be distributed from the marks awarded by the teacher which are stored in the database in the table called answer. However, each user is going to see a different graph:

- Student:
- Will see the individual marks they got for which question (similarly to when the teacher has to mark the questions
- A bar plot of the total of the scores they achieved for each individual topic, with the aid of the database tables, score and progress which will include targets based on scores
- A bar plot of other students' scores without names

- Teacher:
- Will see a normally distributed bell curve generated from the scores of the students in their class and what it means (on target, performance and whether there is under performance)
- A bar plot of the students' scores with their names
- Probabilities using the probability density function for the normal distribution and their meaning in context (analysis)

- Parent:
- Same as the student

Probability density function for normal distribution:

$$\boldsymbol{f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}}$$

Pseudo-code for probability of the normal distribution:

The values in the constructor are retrieved from sql using aggregate functions.

```
def standardDeviation():
    float s = sumX / length

def normalDistribution(double sigma, x, mean):
    probability = float((1/ (sigma * sqrt.2 * pi)) * exp.((-x-mean) ** 2) / 2 *sigma ** 2)
```

Statistics testing:

Based on the scores provided in the database, these values are the ones which are calculated by the calculator, casio fx cg50/ fx991 ex:

Sample data set used	
A	**X**
1	3
2	2
3	2
4	3
5	4
6	5
7	64
8	3
9	3
10	3
11	55
12	3
13	3
14	2
15	4
16	3
17	4
18	5

For this I will use the spreadsheet function on the calculator because it helps to recognise who the score belongs to rather than all in one list:

StudentID	Score
112	2
112	2
112	3
112	4
112	64
112	3
112	3
112	3
112	55
112	3
112	3
112	2
112	4
113	5
135	3
135	4
135	5

Values calculated in the program				
X value (mark awarded)	**Mean**	**Standard deviation**	**Normal distribution**	**Z score**

3	9.5	17.7615377212172	0.0210062144913	-0.2535644191
2	9.5	17.7615377212172	0.0205452416397	-0.4222607365
4	9.5	17.7615377212172	0,021409557388	-0.309657873
64	9.5	17.7615377212172	0.0002027373815	3.0684280187

Values from the calculator				
X value (mark awarded)	**Mean**	**Standard deviation**	**Normal distribution**	**Z score**
3	9.5	17.7615377	0.0210062144	-0.36595930
2	9.5	17.7615377	0.0205452416	-0.42226073
4	9.5	17.7615377	0.0214095573	-0.30965787
64	9.5	17.7615377	2.02737381E-4 (in scientific notation)	3.068428022

Absolute and relative errors (from the A Level specification) for the probability density function:

For x = 3 , the absolute error is the actual value (from the calculator) minus the closest value that can be represented:

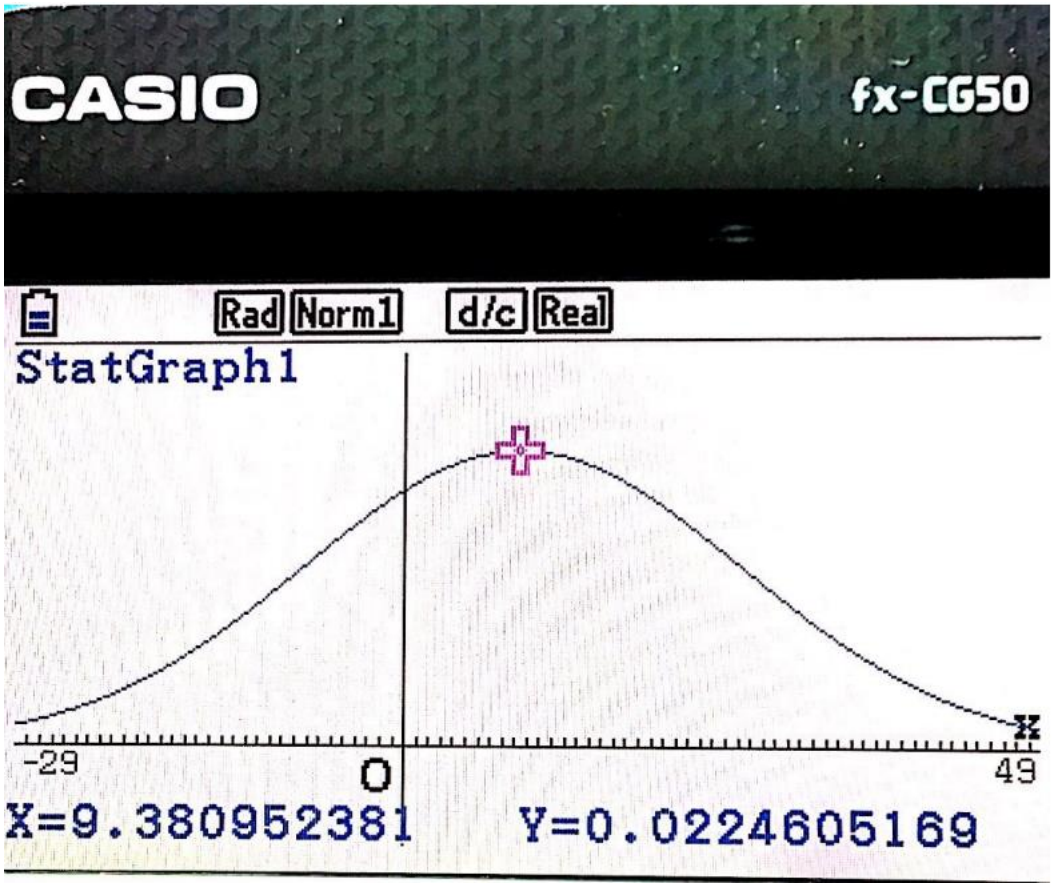
0.0210062144 – 0.210062144913 = 2.17968 x 10-11

The relative error is absolute error divided by the actual value times 100:

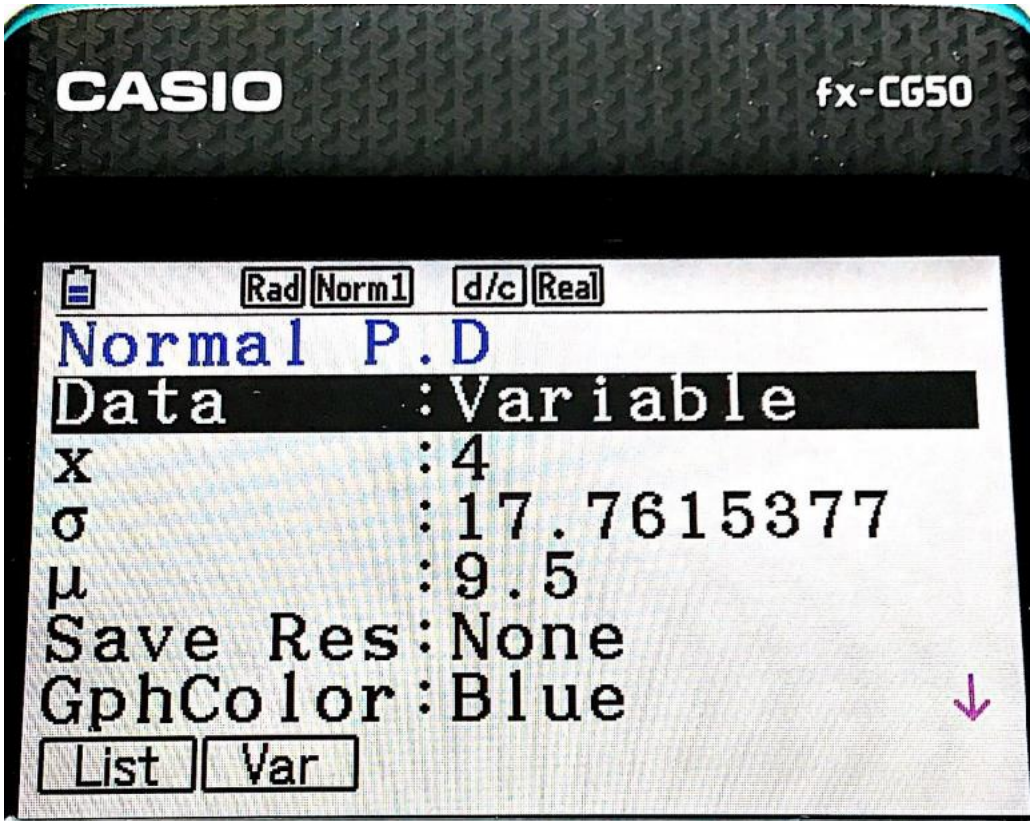
(2.17968 x 10-11 / 0.210062144) * 100 = 1.0376357 x 10

Graphs from the calculator:

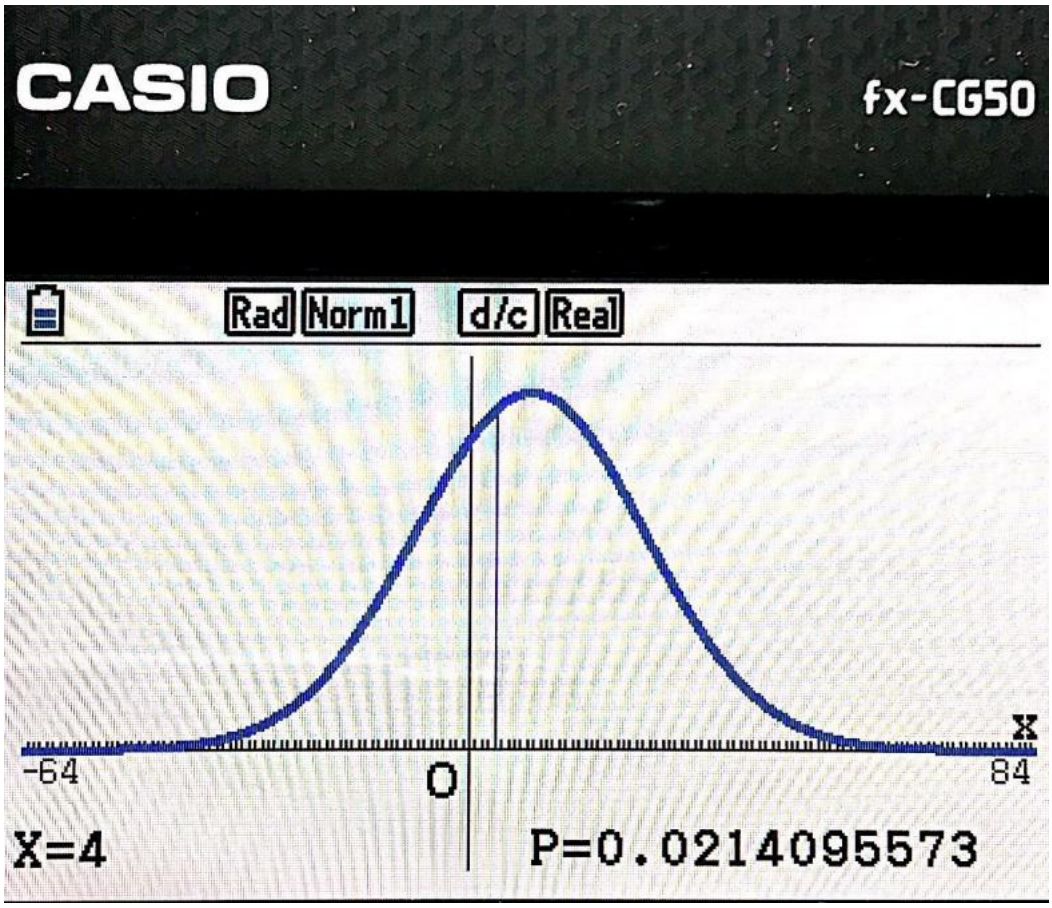
Statistical graph set to normal distribution with the parameter of the list of scores specified above:



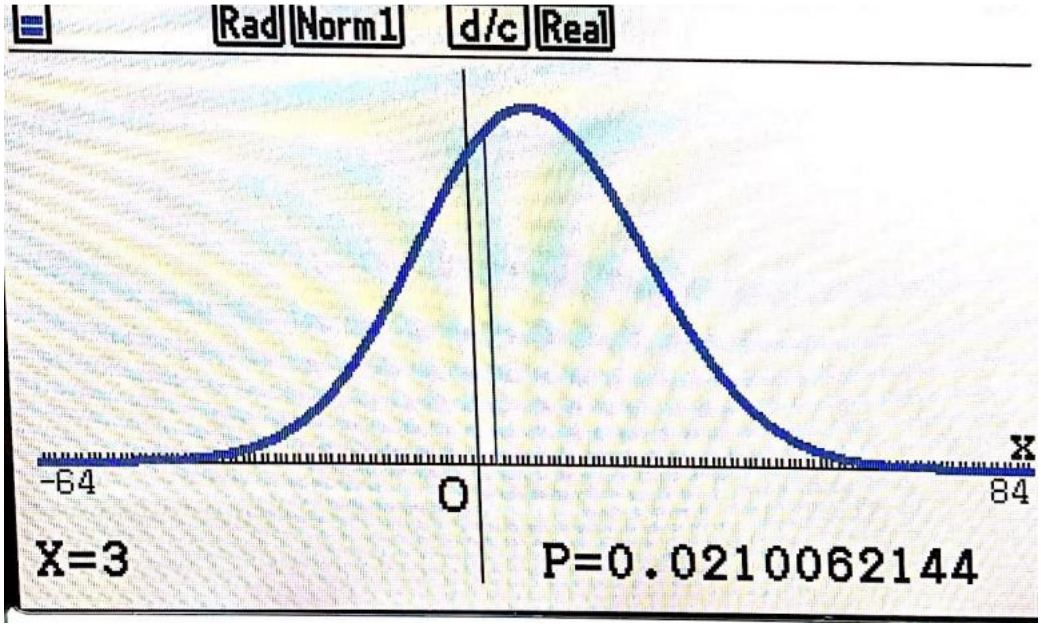
1 variable calc (mean, standard deviation and value of x):

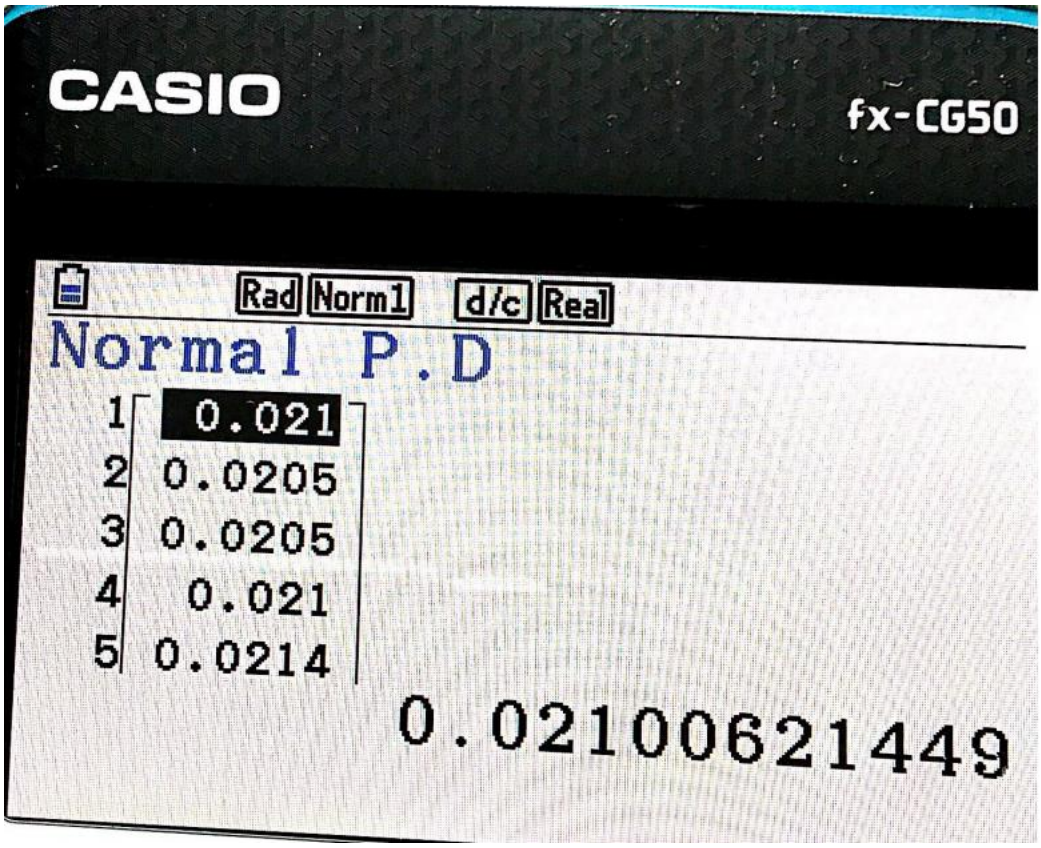


Normal distribution curve for a specfic value of x:

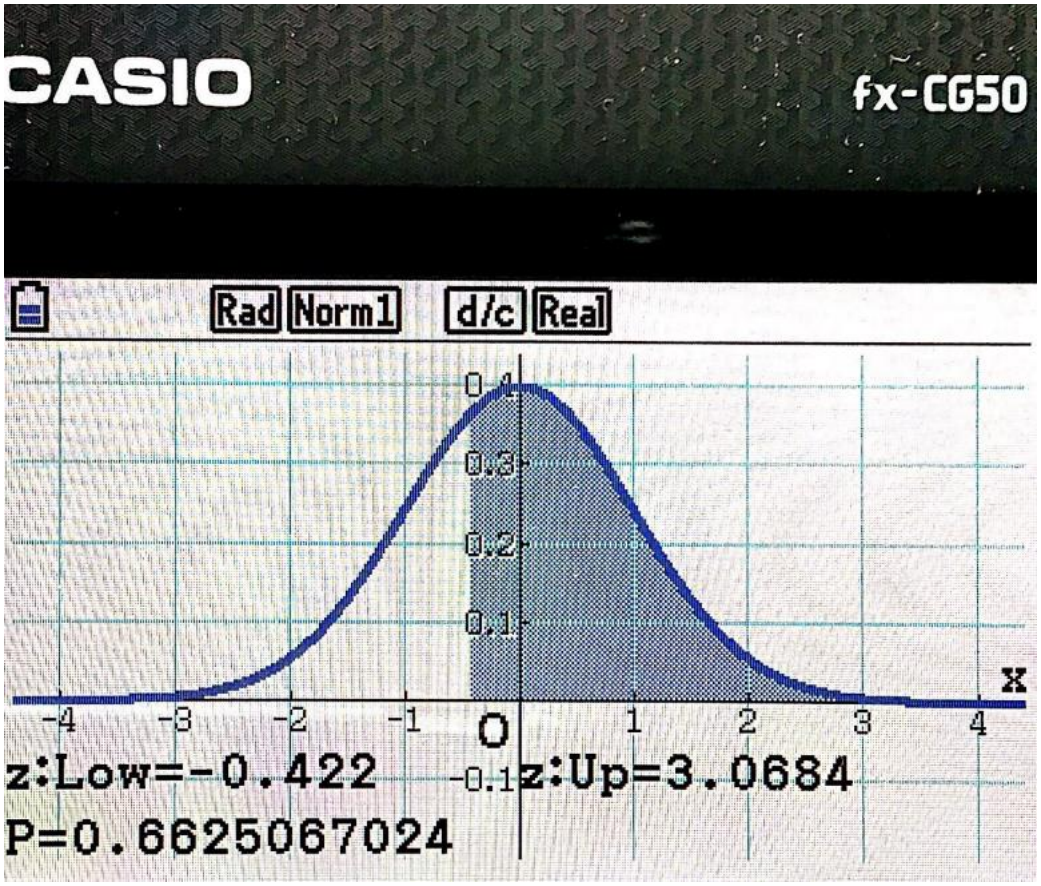


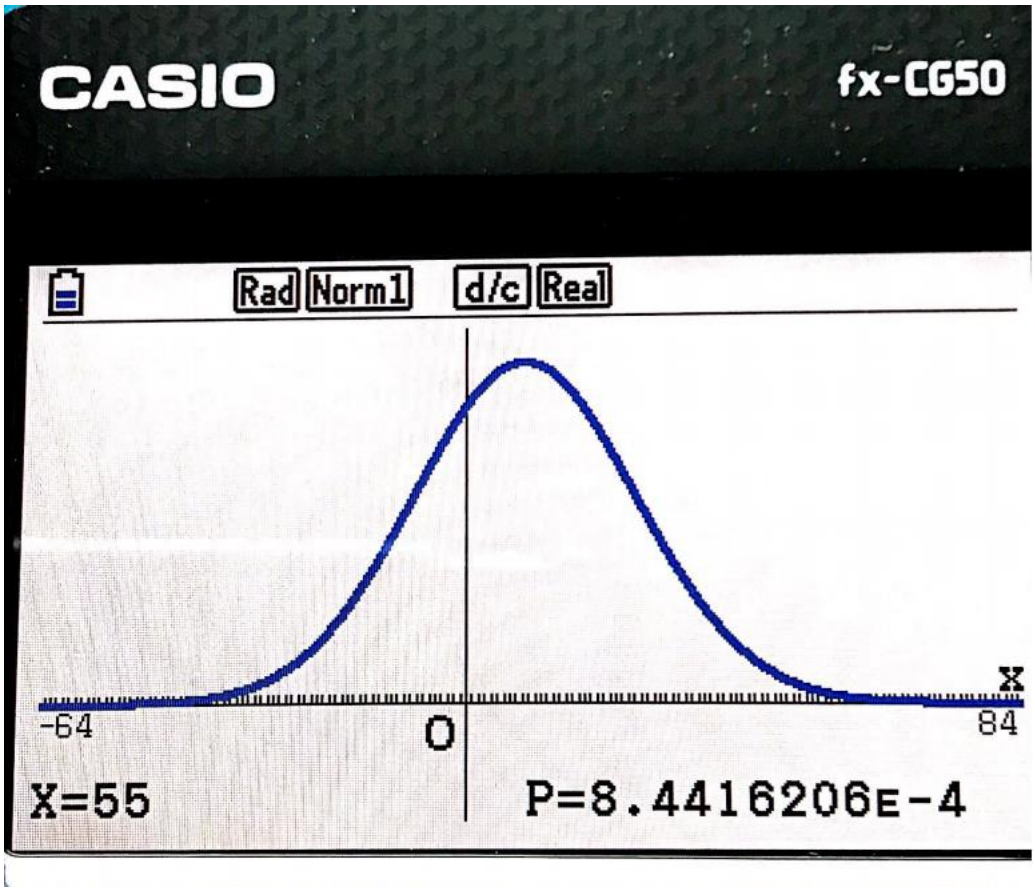
Different value of x:

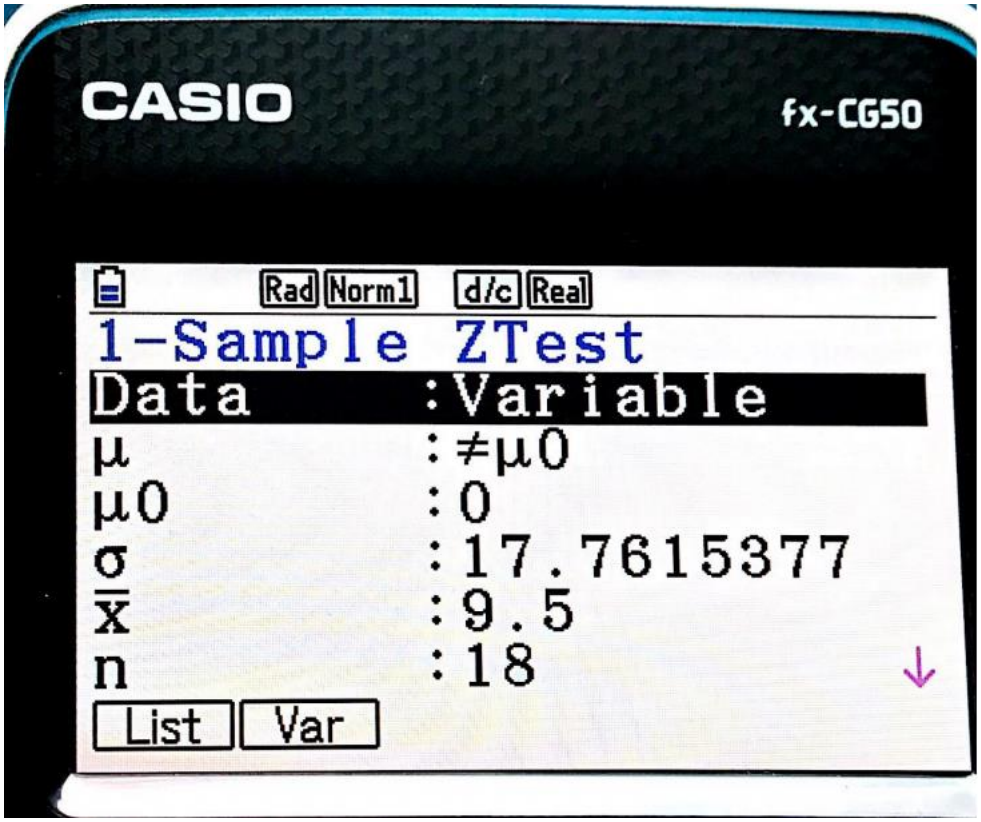


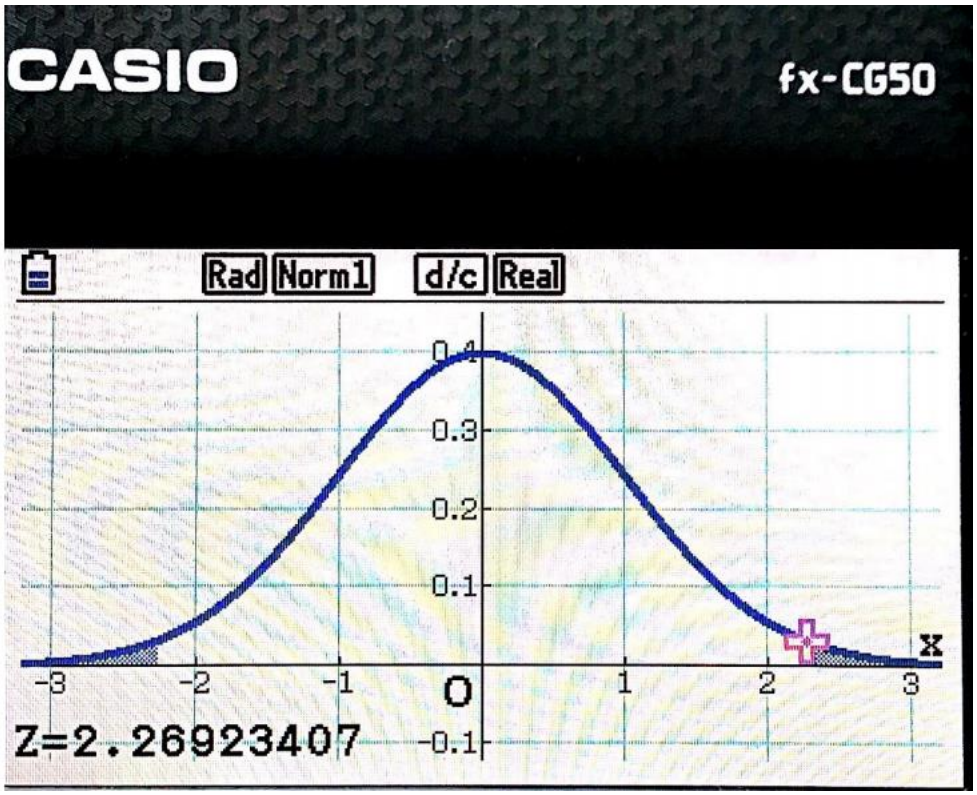


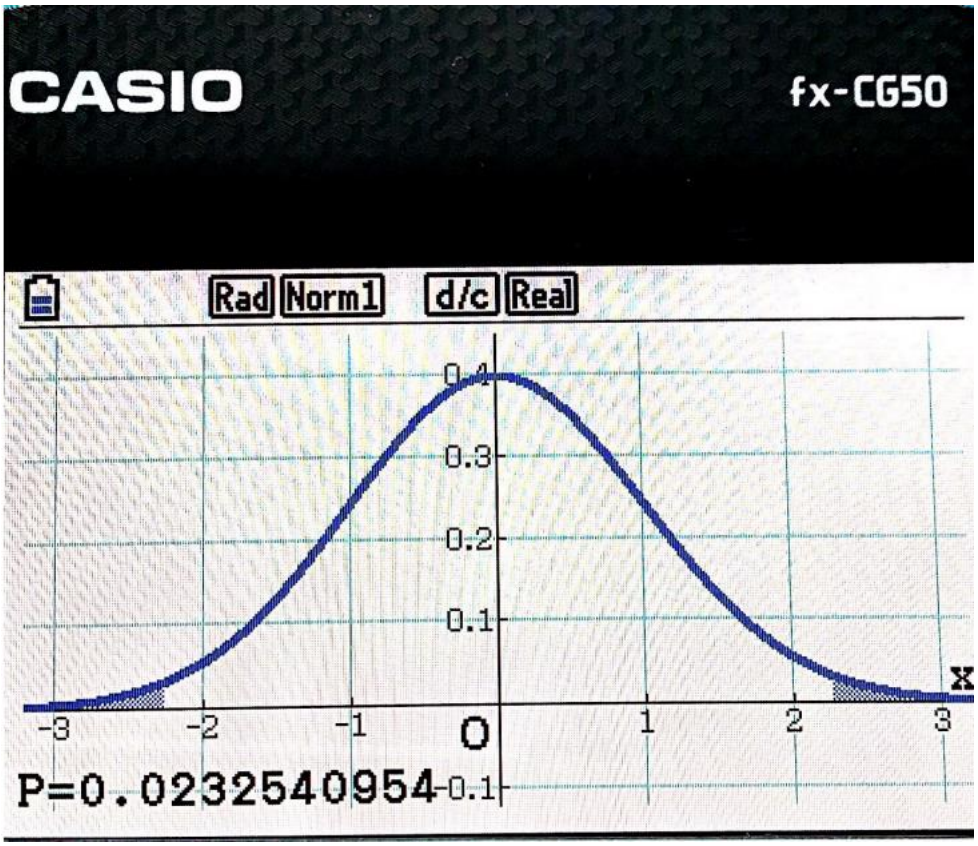
Upper = 64
Lower = 2:

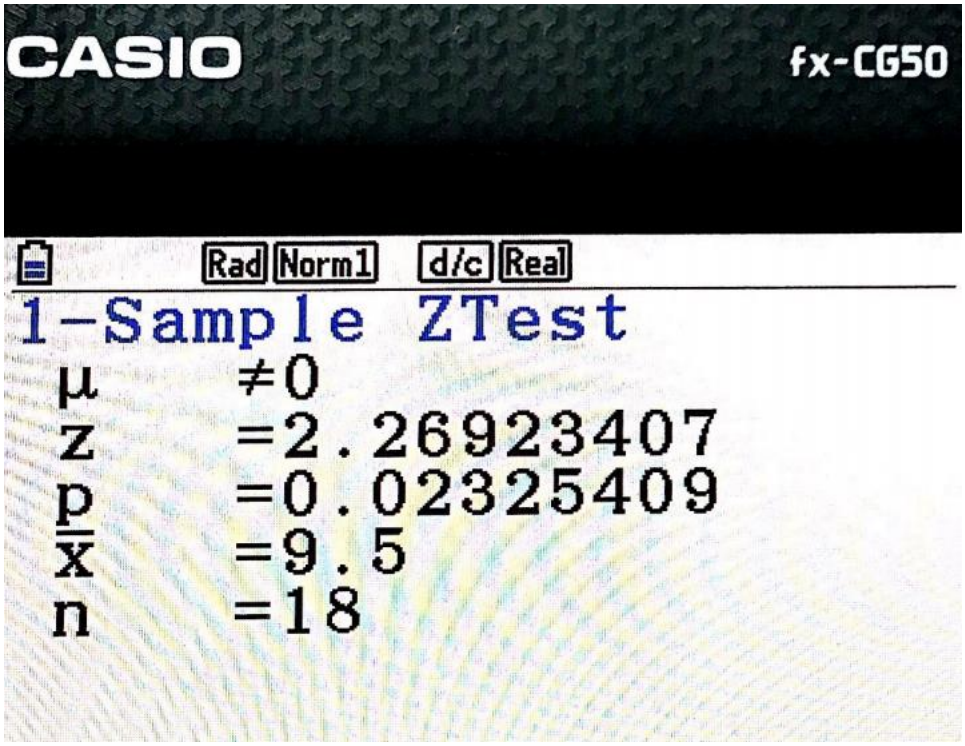



CASIO
fx-CG50
Rad Norm1 d/c Real
-64
0
84
X
X=55
P=8.4416206E-4

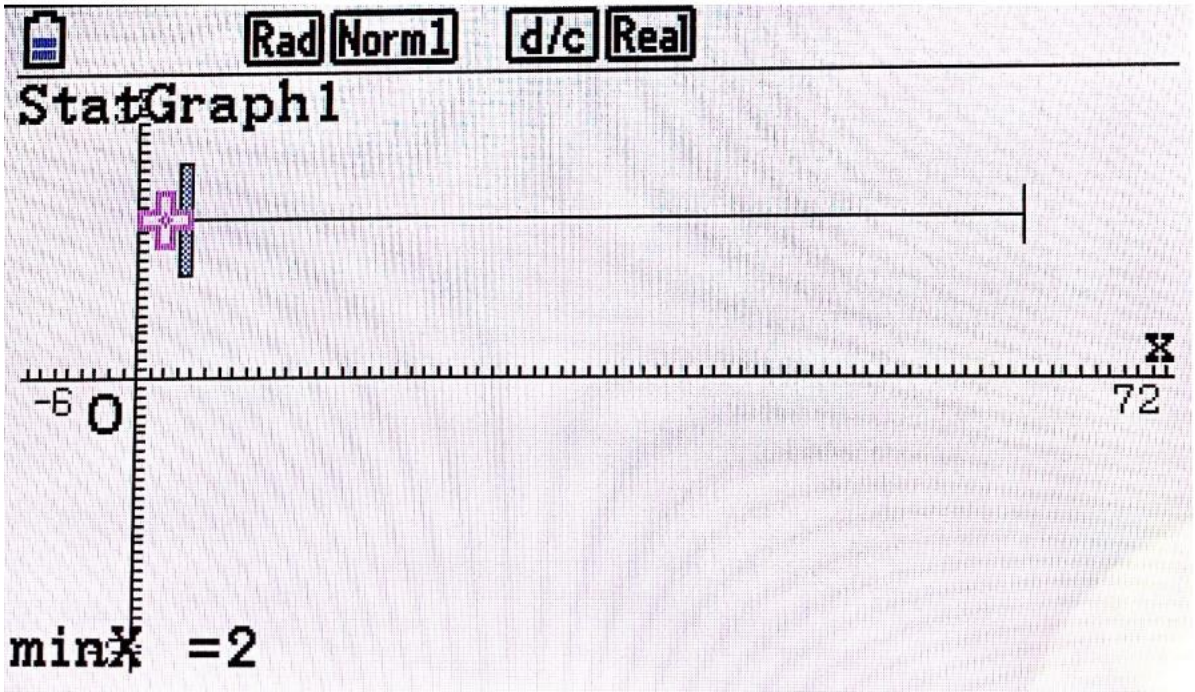

CASIO
fx-CG50
Rad Norm1 d/c Real
1-Sample ZTest
Data :Variable
μ :≠μ0
μ0 :0
σ :17.7615377
x̄ :9.5
n :18
List Var

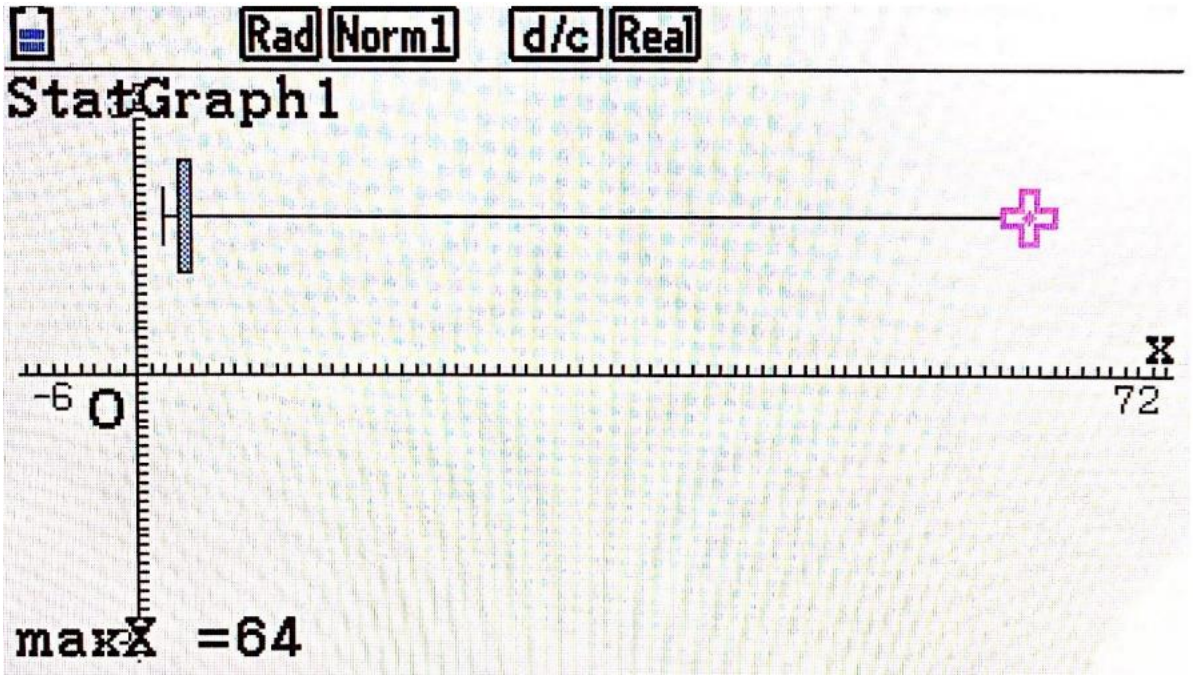

CASIO
fx-CG50
Rad Norm1 d/c Real
0.4
0.3
0.2
0.1
-3 -2 -1 O 1 2 3 X
-0.1
Z=2.26923407


CASIO
fx-CG50
Rad Norm1 d/c Real
0.4
0.3
0.2
0.1
-3 -2 -1 O 1 2 3 X
-0.1
P=0.0232540954

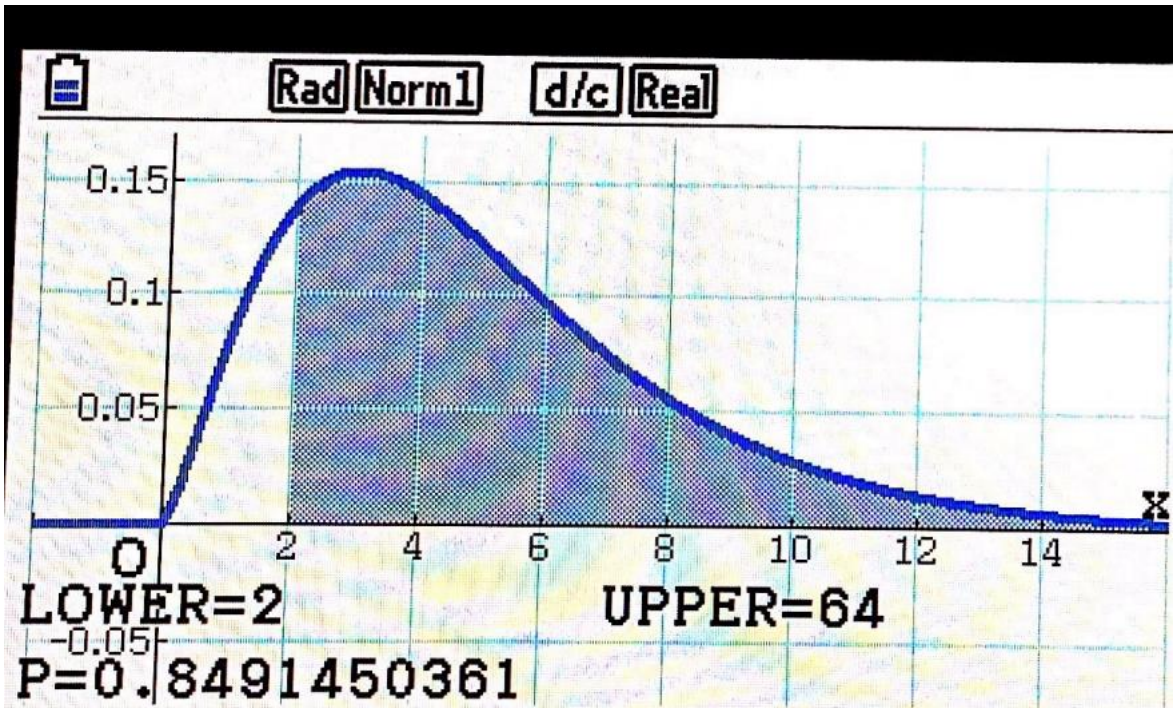

CASIO
fx-CG50
Rad Norm1 d/c Real
1-Sample ZTest
μ ≠0
z =2.26923407
p =0.02325409
x̄ =9.5
n =18

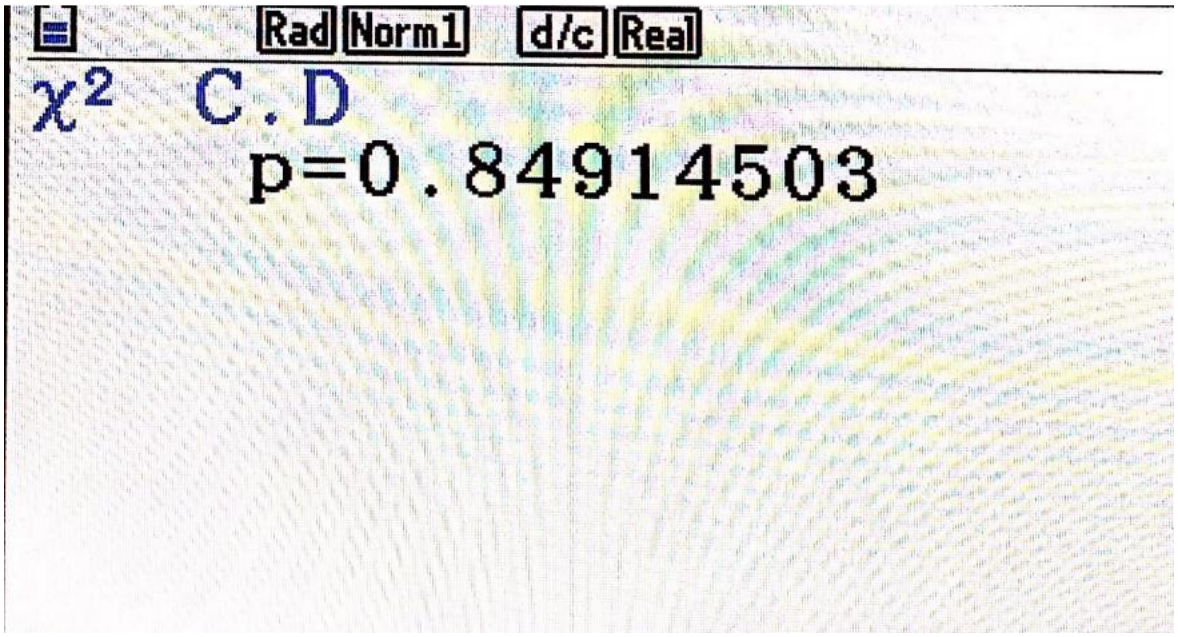
Box plot:

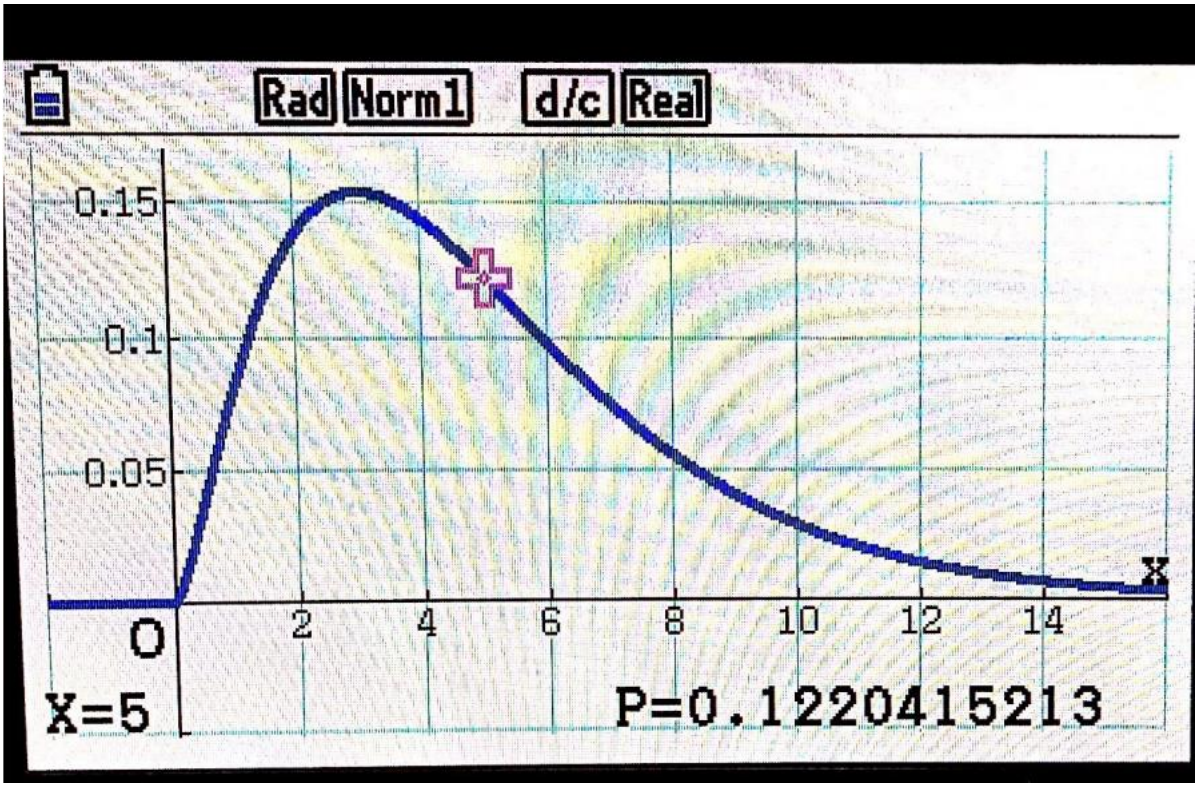

Rad Norm1 d/c Real
Stat Graph1
x
-6
0
72
minX =2



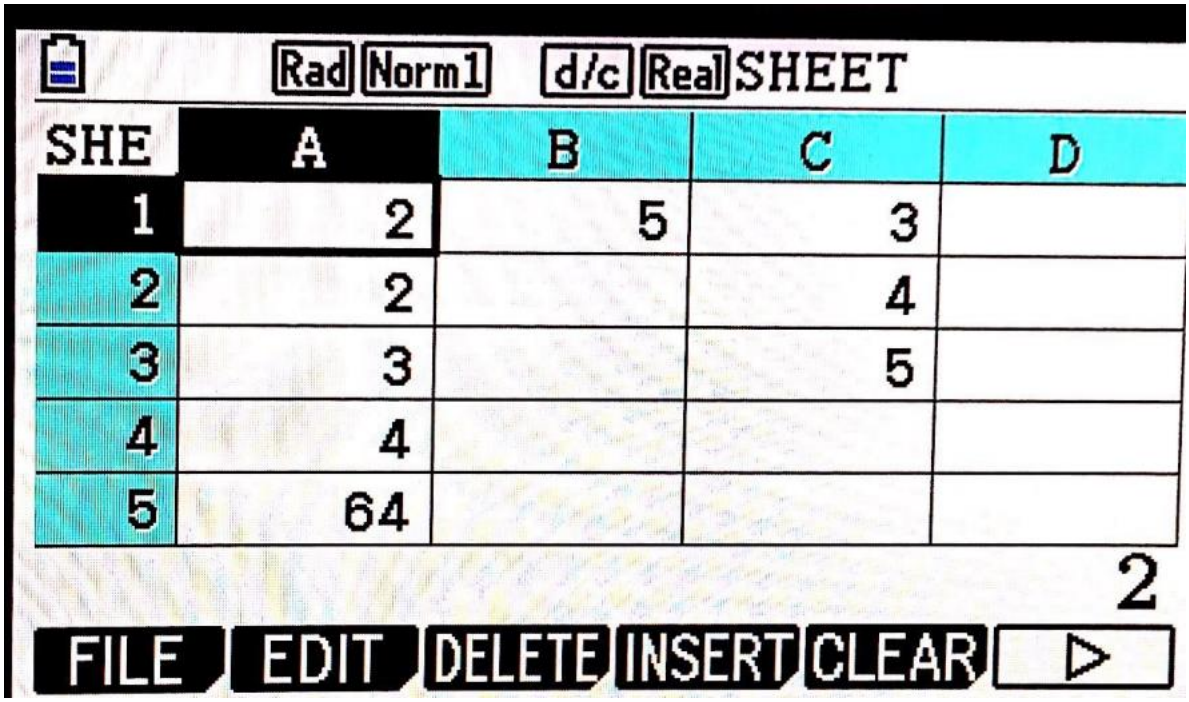
Chi squared test:

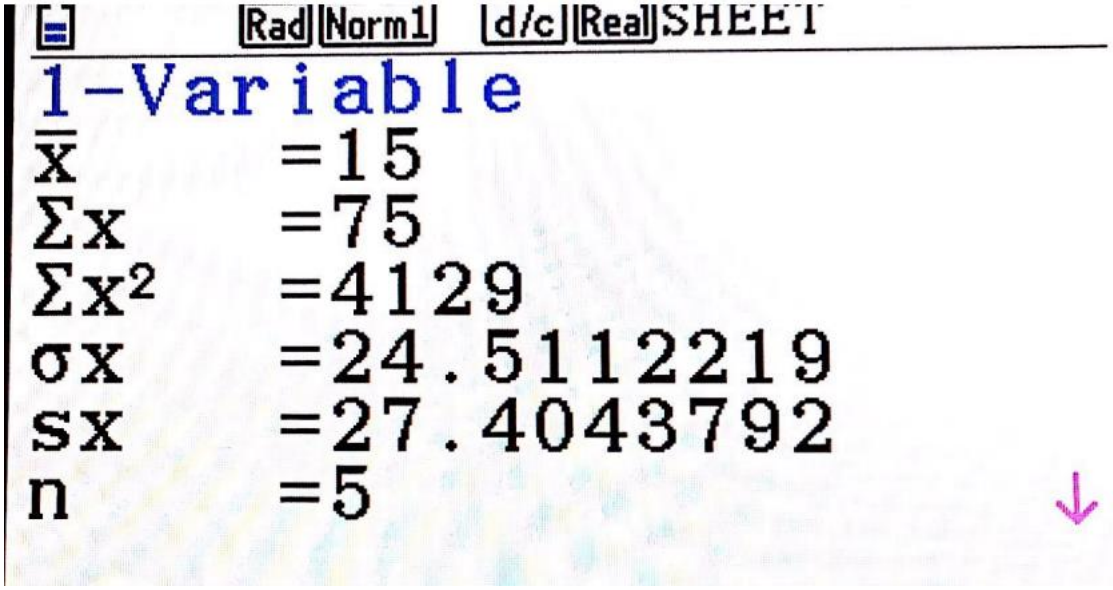


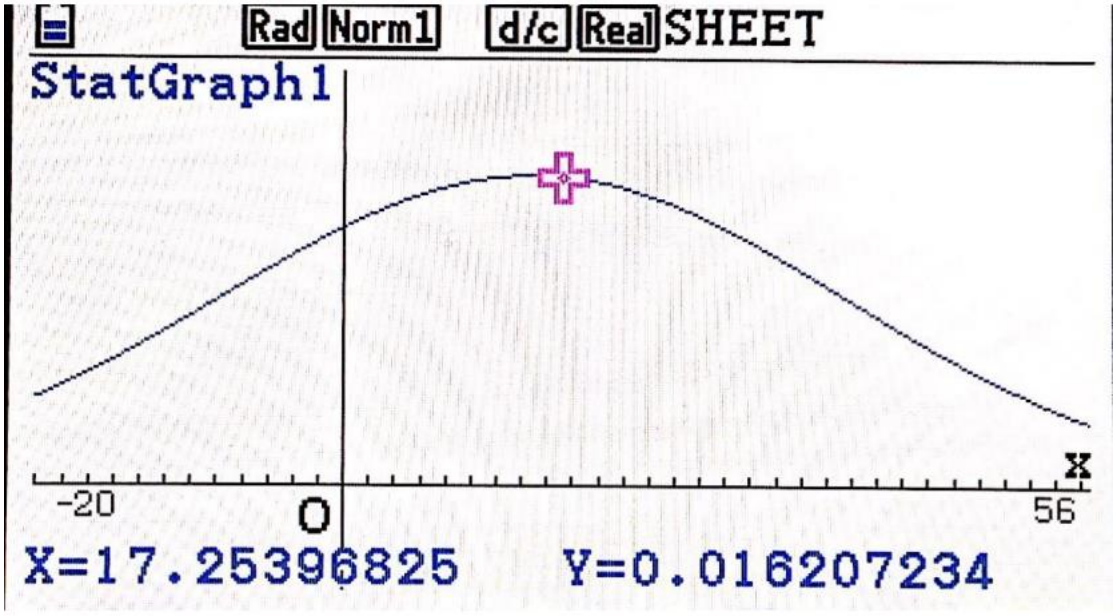




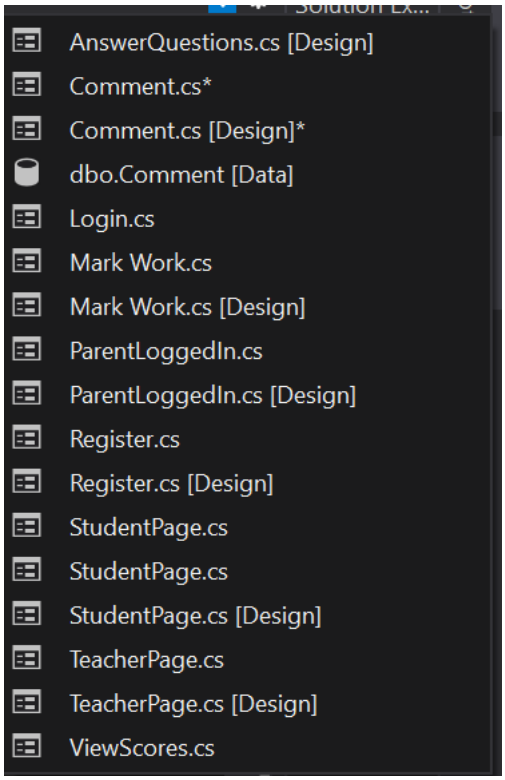
Spreadsheet:


Rad Norm1 d/c Real SHEET
SHE A B C D
1 2 5 3
2 2 4
3 3 5
4 4
5 64
2
FILE EDIT DELETE INSERT CLEAR ▷


Rad Norm1 d/c Real SHEET
1-Variable
x̄ =15
Σx =75
Σx² =4129
σx =24.5112219
sx =27.4043792
n =5



Files organised for direct access:



Technical solution:

Code for the first form that is opened:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Project
{
    public partial class Login : Form
    {
        // create new connection
        SqlConnection loginConnection = new SqlConnection("Data Source =
LAPTOP-CKS3DTGK; Initial Catalog=LatinRevision; Integrated
Security=True");
        // meaningful identifier names
        public string profileName, usernameHashValue;
        public Login()
        {
            InitializeComponent();
        }
        private void Login_Load(object sender, EventArgs e)
        {
            loginConnection.Open();
        }
        private void registerButton_Click(object sender, EventArgs e)
        {
            // open the register form
            Register position = new Register();
            position.Show();
            this.Hide();
        }
        private void exitButton_Click(object sender, EventArgs e)
        {
            // will exit the application
            Application.Exit();
        }
        private void loginButton_Click(object sender, EventArgs e)
        {
            // get username and password from database and compare them
against the values given by the user
            // select count(*) means that the number of records is
selected
            // the count function is an aggregate SQL function

            // select the name of the student and pass it as an argument
```

```
to the constructor of StudentPage
            // this is to identify which student is answering the
questions each time

            Register register = new Register();
            usernameHashValue =
register.hashUsername(usernameTextboxLogin.Text);
            SqlDataAdapter adapter = new SqlDataAdapter("Select count(*)
from UserProfile where Username = '" + usernameHashValue + "' and
Password = '"+passwordTextboxLogin.Text+"' and Role =
'"+roleComboboxLogin.Text+"'", loginConnection);

            SqlCommand name = new SqlCommand("Select Name from
UserProfile where profileID = (Select profileID from UserProfile where
Username = '" + usernameHashValue + "')", loginConnection);
            name.ExecuteNonQuery();
            SqlDataReader sqlData = name.ExecuteReader();
            while(sqlData.Read())
            {
                profileName = sqlData["Name"].ToString();
            }
            sqlData.Close();

            DataTable datatable = new DataTable();
            adapter.Fill(datatable);

            // checks if username and password match the ones saved to
the database
            // this means that there is one record which contains the
provided parameters
            if (datatable.Rows[0][0].ToString() == ("1"))
            {
                try
                {
                    if
(roleComboboxLogin.SelectedItem.ToString().Contains("Student"))
                    {
                        // use of object oriented programming
                        // call specific methods depending on the user
which is more efficient than having all the code on the same method
which is not necessary
                        studentLogin();
                    }

                    if
(roleComboboxLogin.SelectedItem.ToString().Contains("Teacher"))
                    {
                        teacherLogin();
                    }

                    if
(roleComboboxLogin.SelectedItem.ToString().Contains("Parent"))
                    {
```

```
                        parentLogin();
                    }
                }

                catch (Exception)
                {
                    MessageBox.Show("Enter login details");
                }
            }

            else
            {
                MessageBox.Show("Incorrect details", "Try again",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void studentLogin()
        {
            // select both student and teacher ID to pass the value as
an argument to the constructor of the next form to be opened
            SqlCommand selectID = new SqlCommand("Select StudentID from
Student where profileID = (Select profileID from UserProfile where
Username = '" + usernameHashValue + "')", loginConnection);
            int idProfile = 0;
            SqlDataReader studentIDReader = selectID.ExecuteReader();
            while (studentIDReader.Read())
            {
                idProfile = Int32.Parse((String.Format("{0}",
studentIDReader["StudentID"])));
            }
            studentIDReader.Close();

            // open a specific form if the user is a student, parent or
teacher
            StudentPage studentPage = new StudentPage(profileName,
idProfile);
            studentPage.Show();
            this.Hide();
        }

        private void teacherLogin()
        {
            SqlCommand selectTeacherID = new SqlCommand("Select
TeacherID from Teacher where profileID = (Select profileID from
UserProfile where Username = '" + usernameHashValue + "')",
loginConnection);
            int ProfileID = 0;
            SqlDataReader Reader = selectTeacherID.ExecuteReader();
            while (Reader.Read())
            {
                ProfileID = Int32.Parse((String.Format("{0}",
Reader["TeacherID"])));
```

```
            }
            Reader.Close();

            teacherLoggedin teacher = new teacherLoggedin(profileName,
ProfileID);
            teacher.Show();
            this.Hide();
        }
        private void parentLogin()
        {
            SqlCommand selectParentID = new SqlCommand("Select ParentID
from Parent where profileID = (Select profileID from UserProfile where
Username = '" + y + "')", loginConnection);
            int IDParent = 0;
            SqlDataReader Reader = selectParentID.ExecuteReader();
            while (Reader.Read())
            {
                IDParent = Int32.Parse((String.Format("{0}",
Reader["ParentID"])));
            }
            Reader.Close();

            // give the name of the user to the other forms using the
same string in which the name retrieved from the database is stored
            ParentLoggedin parent = new ParentLoggedin(profileName,
IDParent);
            parent.Show();
            this.Hide();
        }
    }
}
```

Code for the second form (register):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Collections;

namespace Project
{
    public partial class Register : Form
```

```
    {
        SqlConnection connection = new SqlConnection("Data Source =
LAPTOP-CKS3DTGK; Initial Catalog=LatinRevision; Integrated
Security=True");

        public bool goForward = false, accountCreate;
        public string username;
        public int usernameHash, profileIDValue;
        public Register()
        {
            InitializeComponent();
        }
        private void Register_Load(object sender, EventArgs e)
        {
            // hide the button to add a new class for the student as
this is shown only if the student wishes to add a new class
            addClassButton.Hide();;
            connection.Open();
        }
        private void enterDetailsButton_Click(object sender, EventArgs
e)
        {
            // print message box if values are left empty
            if (nameTextboxRegister.Text == "" ||
surnameTextboxRegister.Text == "" || usernameTextboxRegister.Text == ""
|| passwordTextboxRegister.Text == "" || mailAddressTextbox.Text == ""
|| roleComboBox.Text == "" || telephoneTextboxRegister.Text == "")
            {
                MessageBox.Show("Enter all details");
            }

            else
            {
                SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter("Select Username from UserProfile where Username = '" +
hashUsername(usernameTextboxRegister.Text) + "'", connection);
                DataTable dataTable = new DataTable();
                sqlDataAdapter.Fill(dataTable);

                // check to see if username already exists in the
database by looking for an existing record in the database equal to the
one given by the user
                if (dataTable.Rows.Count >= 1)
                {
                    MessageBox.Show("Username already exists");
                }

                else
                {
                    // insert user login details into the generic table
UserProfile
                    SqlCommand command = new SqlCommand(@"INSERT INTO
[dbo].[UserProfile]
```

```
                    ([Username]
                    ,[Password]
                    ,[Name]
                    ,[Surname]
                    ,[Mail]
                    ,[Telephone]
                    ,[Role])
                VALUES
                ('" + hashUsername(usernameTextboxRegister.Text) + "', "
+ "'" + passwordTextboxRegister.Text + "','" + nameTextboxRegister.Text
+ "', '" + surnameTextboxRegister.Text + "',  '" +
mailAddressTextbox.Text + "', " +
                    "'" + telephoneTextboxRegister.Text + "', '" +
roleComboBox.Text + "')", connection);
                    command.ExecuteNonQuery();

                    // query to select profile ID for the username which
has just been created
                    SqlCommand profile = new SqlCommand("Select
profileID from UserProfile where Username = '" +
hashUsername(usernameTextboxRegister.Text) + "'", connection);
                    SqlDataReader sqlDataReader =
profile.ExecuteReader();
                    while(sqlDataReader.Read())
                    {
                        profileIDValue =
Int32.Parse((String.Format("{0}", sqlDataReader["profileID"])));
                    }
                    sqlDataReader.Close();

                    // create specific account
                    // the class code inserted by the teacher is assumed
to not be an existing one, so it will be added to the database in the
class code table
                    // the class code has therefore to be unique so the
system will check for existing class code before inserting to sql

                    if
(roleComboBox.SelectedItem.ToString().Contains("Teacher"))
                    {
                        // call specific method for each user which will
store their details in the database and handle exceptions to avoid
having all references to sql tables in the same method which is not
needed and would be less efiicient as only one user can register at a
time
                        registerTeacher();
                    }

                    else if
(roleComboBox.SelectedItem.ToString().Contains("Student"))
                    {
                        registerStudent();
                    }
```

```
                    else if
(roleComboBox.SelectedItem.ToString().Contains("Parent"))
                    {
                        registerParent();
                    }

                    if(accountCreate == false)
                    {
                        // this method is only if the user has not
registered correctly so there is no need to have the code in this method
                        deleteAccount();
                    }

                    if (goForward == false)
                    {
                        // open new login form if the user has
registered successfully
                        // most frequent and likely case
                        Login login = new Login();
                        login.Show();
                        this.Hide();
                    }
                }
            }
        }

        // the hash function i will apply to the username is the one
from the aqa a level textbook for unit 1
        // since each username has to be unique, the hash function will
be applied to the username and not the password because more than one
user can choose the same password

        // create new method to get the hash value of the password
chosen using a hash function
        // the hash function gets the ascii character for each character
in the password and squares it
        // the modulus function is then applied to the value above to
get the remainder when divided by 523
        // this is because the hash values are in the range 0, 522 so
the value is modded with the prime number 523
        public string hashUsername(string username)
        {
            Hashtable hashtable = new Hashtable();
            int sum = 0;
            string hash = string.Empty;
            StringBuilder sb = new StringBuilder();
            byte[] ascii = Encoding.ASCII.GetBytes(username);
            foreach (byte b in ascii)
            {
                sb.Append(b);
                hash = sb.ToString();
            }
```

```
            int i;

            for (i = 0; i < sb.Length; i++)
            {
                sum += hash[i] * hash[i];
            }

            usernameHash = sum % 523;

            hashtable.Add(usernameHash, username);
            return Convert.ToString(usernameHash);
        }

        public void deleteAccount()
        {
            // delete the account stored on user profile if the user
specific details are not entered
            SqlCommand delete = new SqlCommand("Delete from UserProfile
where Username = '" + hashUsername(usernameTextboxRegister.Text) + "'",
connection);
            delete.ExecuteNonQuery();
            MessageBox.Show("Not registered");
            goForward = true;
        }

        private void goBackButton_Click(object sender, EventArgs e)
        {
            // open new login form
            Login goback = new Login();
            goback.Show();
            this.Hide();
        }

        private void roleComboBox_SelectionChangeCommitted(object
sender, EventArgs e)
        {
            // disable each group box containing the text box for the
other type of user
            // only one grioup box can be seen at one time (as the user
can only be either a student, parent or teacher)
            // all group boxes will be disabled from the begininning and
the selected one will be enabled while the remaining two are disabled
            if(roleComboBox.SelectedItem.ToString().Contains("Student"))
            {
                studentGroupBox.Enabled = true;
                parentGroupBox.Enabled = false;
                teacherGroupBox.Enabled = false;
            }

            if(roleComboBox.SelectedItem.ToString().Contains("Parent"))
            {
                parentGroupBox.Enabled = true;
```

```
                studentGroupBox.Enabled = false;
                teacherGroupBox.Enabled = false;
            }

            if(roleComboBox.SelectedItem.ToString().Contains("Teacher"))
            {
                teacherGroupBox.Enabled = true;
                studentGroupBox.Enabled = false;
                parentGroupBox.Enabled = false;
            }
        }

        private void addClassButton_Click(object sender, EventArgs e)
        {
            // this allows the student to add extra subjects after they
have registerd
            SqlCommand insertSubject = new SqlCommand(@"INSERT INTO
[dbo].[StudentSubject]
            ([StudentID]
            ,[SubjectID]
            )
            VALUES
            ((SELECT StudentID FROM [dbo].[Student] WHERE profileID =
(SELECT profileID FROM [dbo].[UserProfile] WHERE Username = '" +
hashUsername(usernameTextboxRegister.Text) + "')), (SELECT SubjectID
FROM [dbo].[Subject] WHERE SubjectTaken = '" + subjectComboBox.Text +
"'))", connection);
            insertSubject.ExecuteNonQuery();

            // link tables student and teacher
            // this is done after the student is registered
            // this shows which students are in which class and taught
by which teacher
            SqlCommand linkTables = new SqlCommand(@"INSERT INTO
[dbo].[StudentTeacher]
            ([StudentID]
            ,[TeacherID]
            ,[SubjectID]
            )
            VALUES
            ((SELECT StudentID FROM [dbo].[Student] WHERE profileID =
(SELECT profileID FROM [dbo].[UserProfile] WHERE Username = '" +
hashUsername(usernameTextboxRegister.Text) + "')), (SELECT TeacherID
FROM [dbo].[Teacher] WHERE TeacherID = (Select TeacherID from
[dbo].[Teacher] where ClassCode = '" + classcodetextBoxStudent.Text +
"')), (SELECT SubjectID FROM [dbo].[Subject] WHERE SubjectTaken = '" +
subjectComboBox.Text + "'))", connection);
            linkTables.ExecuteNonQuery();

            MessageBox.Show("New class added");
            addClass();
        }
```

```
        private void registerTeacher()
        {
            if (subjectTextBoxTeacher.Text == "" ||
classCodeTextBoxteacher.Text == "" || departmentTextBox.Text == "")
            {
                MessageBox.Show("Enter details");
            }

            else
            {
                try
                {
                    // store the int value in the insert statement into
teacher table and do the same for the other tables
                    SqlCommand insertProfile = new SqlCommand(@"INSERT
INTO [dbo].[Teacher]
                                ([profileID]
                                ,[Department]
                                ,[ClassCode]
                                )
                            VALUES
                            ('"+profileIDValue+"', '" +
departmentTextBox.Text + "', '" + classCodeTextBoxteacher.Text + "')",
connection);
                    insertProfile.ExecuteNonQuery();

                    // link tables subject and teacher, this is to show
which teacher teaches which subject
                    SqlCommand teacherSubject = new SqlCommand(@"INSERT
INTO [dbo].[Subjectteacher_FK]
                                ([TeacherID]
                                ,[SubjectID])
                            VALUES
                            ((SELECT TeacherID FROM [dbo].[Teacher]
WHERE profileID = '"+profileIDValue+"'), (SELECT SubjectID FROM
[dbo].[Subject] WHERE SubjectTaken = '" + subjectTextBoxTeacher.Text +
"'))", connection);
                    teacherSubject.ExecuteNonQuery();
                    MessageBox.Show("Registered successfully");
                    accountCreate = true;
                }

                catch (SqlException)
                {
                    MessageBox.Show("Enter details");
                }
            }
        }

        private void registerStudent()
        {
            if (levelComboBox.SelectedIndex == -1 ||
subjectComboBox.SelectedIndex == -1 || parentnametextBox.Text == "" ||
```

```
classcodetextBoxStudent.Text == "")
            {
                MessageBox.Show("Enter details");
            }

            else
            {
                try
                {
                    // insert profile id, level studying at and parent
name to student table
                    SqlCommand insertStudent = new SqlCommand(@"INSERT
INTO [dbo].[Student]
                                ([profileID]
                                ,[LevelID]
                                ,[ParentName]
                                    )
                            VALUES
                            ('"+profileIDValue+"', (SELECT LevelID FROM
[dbo].[Level] WHERE subjectLevel = '" + levelComboBox.Text + "'), '" +
parentnametextBox.Text + "')", connection);
                    insertStudent.ExecuteNonQuery();

                    // link tables student and subject in many-to-many
relationship
                    // select student id of student for which the
profile id has just been created
                    // select subject id where the subject is equal to
the one specified
                    // this shows how many students study which subject
                    SqlCommand insertSubject = new SqlCommand(@"INSERT
INTO [dbo].[StudentSubject]
                                ([StudentID]
                                ,[SubjectID]
                                )
                            VALUES
                            ((SELECT StudentID FROM [dbo].[Student]
WHERE profileID = '"+profileIDValue+"'), (SELECT SubjectID FROM
[dbo].[Subject] WHERE SubjectTaken = '" + subjectComboBox.Text + "'))",
connection);
                    insertSubject.ExecuteNonQuery();

                    // link tables student and teacher
                    // this is done after the student is registered
                    // this shows which students are in which class and
taught by which teacher
                    SqlCommand linkTables = new SqlCommand(@"INSERT INTO
[dbo].[StudentTeacher]
                                ([StudentID]
                                ,[TeacherID]
                                ,[SubjectID]
                                )
                            VALUES
```

```
                            ((SELECT StudentID FROM [dbo].[Student]
WHERE profileID = '"+profileIDValue+"'), (SELECT TeacherID FROM
[dbo].[Teacher] WHERE TeacherID = (Select TeacherID from [dbo].[Teacher]
where ClassCode = '" + classcodetextBoxStudent.Text + "')), (SELECT
SubjectID FROM [dbo].[Subject] WHERE SubjectTaken = '" +
subjectComboBox.Text + "'))", connection);
                    linkTables.ExecuteNonQuery();

                    MessageBox.Show("Registered successfully");
                    accountCreate = true;

                    // call private method to empty the class code
textbox and subject so the student can add a new class code
                    addClass();
                }

                catch (Exception)
                {
                    MessageBox.Show("Details entered incorrectly");
                }
            }
        }

        private void registerParent()
        {
            if (nameChildTextBox.Text == "" ||
studentIDTextBoxParent.Text == "" || addressTextBox.Text == "" ||
postcodeTextBox.Text == "")
            {
                MessageBox.Show("Enter details");
            }

            else
            {
                try
                {
                    // insert parent details to database
                    SqlCommand insertParent = new SqlCommand(@"INSERT
INTO [dbo].[Parent]
                                ([profileID]
                                ,[Address]
                                ,[NumberofKids]
                                ,[NameofChild])
                            VALUES
                            ('" + profileIDValue + "', '" +
addressTextBox.Text + "', '" + studentIDTextBoxParent.Text + "', '" +
nameChildTextBox.Text + "')", connection);
                    insertParent.ExecuteNonQuery();

                    // link tables student and parent, this is done
after the parent has registered as a parent does not necessarily have to
be registered before a student registers
                    // this shows which parent has which child
```

```
                    SqlCommand studentParent = new SqlCommand(@"INSERT
INTO [dbo].[ParentStudent_FK]
                               ([StudentID]
                               ,[ParentID])
                               VALUES
                               ((SELECT StudentID FROM [dbo].[Student]
WHERE StudentID = '" + studentIDTextBoxParent.Text + "'), (SELECT
ParentID FROM [dbo].[Parent] WHERE profileID = '"+profileIDValue+"'))",
connection);
                    studentParent.ExecuteNonQuery();
                    MessageBox.Show("Registered successfully");
                    accountCreate = true;
                }

                catch(SqlException ex)
                {
                    MessageBox.Show("Details entered incorrectly");
                    MessageBox.Show(ex.Message);
                }
            }
        }

        private void addClass()
        {
            // ask if the student needs to enter more than one class
code for their subjects
            DialogResult dialog = MessageBox.Show("Do you want to add
another subject?", "Confirm", MessageBoxButtons.YesNo);
            if (dialog == DialogResult.Yes)
            {
                addClassButton.Show();
                enterDetailsButton.Enabled = false;

                usernameTextboxRegister.Text = "";
                passwordTextboxRegister.Text = "";
                nameTextboxRegister.Text = "";
                surnameTextboxRegister.Text = "";
                mailAddressTextbox.Text = "";
                telephoneTextboxRegister.Text = "";
                roleComboBox.Text = "";
                // empty values of text box for the user to enter new
values
                // disable username group box so the system does not
display message 'username already exists'
                registerGroupBox.Enabled = false;

                subjectComboBox.Text = "";
                classcodetextBoxStudent.Text = "";
                goForward = true;
            }

            else if (dialog == DialogResult.No)
            {
```

```
                MessageBox.Show("Thank you");
                goForward = false;
                Login login = new Login();
                login.Show();
                this.Close();
            }
        }

        private void label12_Click(object sender, EventArgs e)
        {
            // leave empty
        }
        private void comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            // leave empty
        }
    }
}
```

Third form (student page):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Project
{
    public partial class StudentPage : Form
    {
        private int studentID;
        public StudentPage(string profileValue, int ID)
        {
            InitializeComponent();
            // set the label equal to the value of their profile id
            welcomelabel.Text = profileValue;
            studentIDLabel.Text = Convert.ToString(ID);
            studentID = ID;
        }
        private void StudentPage_Load(object sender, EventArgs e)
        {
            // leave empty
        }
```

```
        private void answerQuestionsButton_Click(object sender,
EventArgs e)
        {
            try
            {
                if(chooseSubjectComboBox.SelectedIndex == -1)
                {
                    MessageBox.Show("Choose a subject");
                }

                else if (chooseSubjectComboBox.SelectedItem.ToString()
== "Latin")
                {
                    answerQuestions questions = new
answerQuestions(studentID);
                    questions.Show();
                }
            }

            catch(Exception)
            {
                MessageBox.Show("Choose a subject");
            }
        }

        private void viewMessagesButton_Click(object sender, EventArgs
e)
        {
            ViewMessages view = new ViewMessages();
            view.Show();
        }

        private void sendFeedbackButton_Click(object sender, EventArgs
e)
        {
            /*
            string r = "Review";
            Comment comment = new Comment(r);
            comment.Show();*/
        }

        private void viewPreviousScoresButton_Click(object sender,
EventArgs e)
        {
            ViewScores scores = new ViewScores(studentID);
            scores.Show();
        }

        private void viewOtherScoresButton_Click(object sender,
EventArgs e)
        {
            ViewScores scores = new ViewScores(studentID);
            scores.Show();
```

```
        }

        private void teacherEnquiryButton_Click(object sender, EventArgs
e)
        {
            /*
            string type = "teacherComment";
            Comment comment = new Comment(type);
            comment.Show();*/
        }
        private void label9_Click(object sender, EventArgs e)
        {

        }
        private void label8_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Fourth form (answer questions):

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Net.Mail;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Project
{
    public partial class answerQuestions : Form
    {
        SqlConnection sqlConnection = new SqlConnection("Data Source =
LAPTOP-CKS3DTGK; Initial Catalog=LatinRevision; Integrated
Security=True");
        private ArrayList questionAsked = new ArrayList { };
        private ArrayList questionAnswered = new ArrayList { };
        private int index = 0, id;
        public answerQuestions(int studentID)
        {
            InitializeComponent();
            id = studentID;
        }
```

```
        private void answerQuestions_Load(object sender, EventArgs e)
        {
            goBackButton.Enabled = false;
            SubmitButton.Enabled = false;
        }

        private void doneButton_Click(object sender, EventArgs e)
        {
            sqlConnection.Open();
            SubmitButton.Enabled = true;
            if (TopicComboBox.SelectedIndex == -1 ||
LevelComboBox.SelectedIndex == -1)
            {
                MessageBox.Show("Choose topic and level");
            }

            else
            {
                try
                {
                    SqlCommand selectQuestions = new SqlCommand("Select
QuestionName, NumberofMarks, QuestionDifficulty from Question where
(TopicID = (Select TopicID from Topic where TopicName = '" +
TopicComboBox.SelectedItem.ToString() + "') and LevelID = (Select
LevelID from Level where subjectLevel = '" +
LevelComboBox.SelectedItem.ToString() + "'))", sqlConnection);
                    SqlDataReader questionReader =
selectQuestions.ExecuteReader();
                    int marks = 0;
                    string question = string.Empty;

                    while (questionReader.Read())
                    {
                        question =
questionReader["QuestionName"].ToString();

                        marks =
Convert.ToInt32(questionReader["NumberofMarks"]);

                        questionAsked.Add(question);
                        questionAnswered.Add("");
                    }
                    questionLabel.Text = questionAsked[0].ToString();
                    questionReader.Close();
                }

                catch (NullReferenceException)
                {
                    MessageBox.Show("Enter topic and level");
                }
            }
        }
```

```
        private void SubmitButton_Click(object sender, EventArgs e)
        {
            goBackButton.Enabled = true;
            questionAnswered[index] = questionTextBox.Text;

            string x = questionTextBox.Text.ToString();
            try
            {
                index++;
                questionLabel.Text = questionAsked[index].ToString();
                // questionTextBox.Text =
questionAnswered[index].ToString();
                // the attrribute marks awarded will be set when the
teacher has marked the question, plus the teacher ID
                SqlCommand storeQuestions = new SqlCommand(@"INSERT INTO
[dbo].[Answer]
           ([QuestionID]
           ,[StudentID]
           ,[AnswerGiven]
           )
       VALUES
           ((Select QuestionID from Question where QuestionName = '" +
questionAsked[index].ToString() + "'), '" + id + "', '" + x + "')",
sqlConnection);
                storeQuestions.ExecuteNonQuery();
            }

            catch(Exception)
            {
                MessageBox.Show("Finished");
                sqlConnection.Close();
                this.Close();
            }

            questionTextBox.Text = "";
        }
        private void seeClueButton_Click(object sender, EventArgs e)
        {
            if(questionLabel.Text.Contains("bam"))
            {
                MessageBox.Show("bam: perfect tense past first",
"Clue");
            }
        }
        private void goBackButton_Click(object sender, EventArgs e)
        {
            index = index - 1 ;
            questionLabel.Text = questionAsked[index].ToString();
            questionTextBox.Text = questionAnswered[index].ToString();
        }

        private void groupBox2_Enter(object sender, EventArgs e)
        {
```

```
            // leave empty
        }

        private void topicLable_Click(object sender, EventArgs e)
        {

        }
        private void LevelComboBox_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }
        private void chooseChapterchecbox_SelectedIndexChanged(object
sender, EventArgs e)
        {

        }

        private void questionsTimer_Tick(object sender, EventArgs e)
        {

        }

        private void nextQuestionButton_Click(object sender, EventArgs
e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Fifth form (teacher page):

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Project
```

```
{
    public partial class teacherLoggedin : Form
    {
        SqlConnection sqlConnection = new SqlConnection("Data Source =
LAPTOP-CKS3DTGK; Initial Catalog=LatinRevision; Integrated
Security=True");

        public int profileID;
        public teacherLoggedin(string teacherName, int ID)
        {
            InitializeComponent();
            sqlConnection.Open();
            nonelabel.Text = teacherName;
            profileID = ID;
        }
        private void teacherLoggedin_Load(object sender, EventArgs e)
        {

        }
        private void addQuestionsButton_Click(object sender, EventArgs
e)
        {
            addQuestion newQuestion = new addQuestion();
            newQuestion.Show();
        }

        private void viewMessagesButton_Click(object sender, EventArgs
e)
        {
            ViewMessages messages = new ViewMessages();
            messages.Show();
        }

        private void viewProgressButton_Click(object sender, EventArgs
e)
        {
            SqlCommand selectID = new SqlCommand("Select TeacherID from
Teacher where profileID = '" + profileID + "'", sqlConnection);
            SqlDataReader profileReader = selectID.ExecuteReader();
            while(profileReader.Read())
            {
                profileID = Int32.Parse((String.Format("{0}",
profileReader["TeacherID"])));
            }
            profileReader.Close();
            ViewScores viewScores = new ViewScores(profileID);
            viewScores.Show();
        }

        private void commentStudentButton_Click(object sender, EventArgs
e)
        {
            SqlCommand receiverID = new SqlCommand("Select
```

```
UserProfile.profileID from UserProfile inner join Student on
Student.profileID = UserProfile.profileID inner join StudentTeacher on
StudentTeacher.StudentID = Student.StudentID and
StudentTeacher.TeacherID = '40'", sqlConnection);
            SqlDataReader receiver = receiverID.ExecuteReader();
            ArrayList receiverIDS = new ArrayList { };
            while(receiver.Read())
            {
                int receiverIDNumber = Int32.Parse((String.Format("{0}",
receiver["profileID"])));
                receiverIDS.Add(receiverIDNumber);
            }
            receiver.Close();
            string q = "StudentProgress";
            Comment comment = new Comment(q, profileID, receiverIDS);
            comment.Show();
        }

        private void communicateParentButton_Click(object sender,
EventArgs e)
        {
            /*
            string a = "ParentCommunicate";
            Comment commentParent = new Comment(a);
            commentParent.Show();*/
        }

        private void MarkWorkButton_Click(object sender, EventArgs e)
        {
            Mark_Work work = new Mark_Work(profileID);
            work.Show();
        }
    }
}
```

Sixth form (add questions):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Project
{
```

```
    public partial class addQuestion : Form
    {
        public bool next = true;
        public addQuestion()
        {
            InitializeComponent();
        }

        private void addQuestion_Load(object sender, EventArgs e)
        {

        }

        public void saveQuestionButton_Click(object sender, EventArgs e)
        {
            if (next == true)
            {
                try
                {
                    SqlConnection question = new SqlConnection("Data
Source = LAPTOP-CKS3DTGK; Initial Catalog=LatinRevision; Integrated
Security=True");
                    question.Open();
                    // insert new question to the database
                    SqlCommand command = new SqlCommand(@"INSERT INTO
[dbo].[Question]
           ([QuestionName]
           ,[QuestionAnswer]
           ,[NumberofMarks]
           ,[SubjectID]
           ,[LevelID]
           ,[TopicID]
           ,[QuestionDifficulty])
     VALUES
           ('" + questionTextBox.Text + "', '" + answerTextBox.Text +
"', '" + marksTextBox.Text + "', (SELECT SubjectID FROM [dbo].[Subject]
WHERE SubjectTaken = '"+subjectComboBox.Text+"'), (SELECT LevelID FROM
[dbo].[Level] WHERE subjectLevel = '"+levelComboBox.Text+"'), (SELECT
TopicID FROM [dbo].[Topic] WHERE TopicName =
'"+topicComboBox.Text+"'),'"+gradeTextBox.Text+"')", question);
                    command.ExecuteNonQuery();
                    MessageBox.Show("Question saved");
                    topicComboBox.Text = "";
                    gradeTextBox.Text = "";
                    questionTextBox.Text = "";
                    answerTextBox.Text = "";
                    marksTextBox.Text = "";
                    DialogResult result = MessageBox.Show("Are you
done?", "Exit", MessageBoxButtons.YesNo);
                    if (result == DialogResult.Yes)
                    {
                        // in case the user wishes to add more question,
the system will ask if they want to add another one
```

```
                        next = false;
                        this.Hide();
                    }

                    else if (result == DialogResult.No)
                    {
                        MessageBox.Show("Enter the next question");
                    }
                }

                catch(SqlException)
                {
                    MessageBox.Show("Enter all fields");
                }
            }
        }
        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

Seventh form (view scores):

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.ComponentModel;
using System.Collections;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Project
{
    public partial class ViewScores : Form
    {
        SqlConnection sqlConnection = new SqlConnection("Data Source =
LAPTOP-CKS3DTGK; Initial Catalog=LatinRevision; Integrated
Security=True");

        public int markResult, studentIDNumber, progressTopic,
progressAverage, scoreTotal, IDNumber, index, y;
        private ArrayList scores = new ArrayList { };
```

```
        private ArrayList Zscores = new ArrayList { };
        public static List<int> sDeviation = new List<int> { };
        public ViewScores(int IDName)
        {
            InitializeComponent();
            IDNumber = IDName;
        }
        private void ViewScores_Load(object sender, EventArgs e)
        {
            sqlConnection.Open();
        }

        private void enterButton_Click(object sender, EventArgs e)
        {
            try
            {
                SqlCommand insertProgress = new SqlCommand(@"INSERT INTO
[dbo].[Progress]
           ([TargetGrade]
           ,[AverageProgress]
           ,[TopicProgress]
           ,[StudentID]
           ,[SubjectID]
           ,[LevelID]
            )
     VALUES
           ('" + targetGradeTextBox.Text + "', '" + progressAverage +
"', '" + progressTopic + "','" + studentIDNumber + "', (Select SubjectID
from Subject where SubjectTaken = '" + subjectTextBox.Text + "'),
(Select LevelID from Student where StudentID = '" + studentIDNumber +
"'))", sqlConnection);
                insertProgress.ExecuteNonQuery();

                insertScores();
            }

            catch(SqlException ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void viewButton_Click(object sender, EventArgs e)
        {
            viewButton.Hide();
            if (topicComboBox.SelectedIndex != -1)
            {
                try
                {
                    // the scores and progress are inserted into the sql
tables when the teacher or student or parent wishes to view progress
                    // this is to avoid inserting the fields into sql
from the form MarkWork which is not necessary and would be less
```

```
efficient

                    // select number of students per class for the
teacher form and their names
                    // select the name of each student to label the x
axis
                    SqlCommand selectStudents = new SqlCommand("Select
UserProfile.Name from ((UserProfile inner join Student on
UserProfile.profileID = Student.profileID) inner join StudentTeacher on
Student.StudentID = StudentTeacher.StudentID and
StudentTeacher.TeacherID = '" + IDTextBox.Text + "')", sqlConnection);
                    SqlDataReader sqlData =
selectStudents.ExecuteReader();
                    string numStudents;
                    List<string> students = new List<string> { };
                    while (sqlData.Read())
                    {
                        numStudents = sqlData["Name"].ToString();
                        students.Add(numStudents.ToString());
                    }
                    sqlData.Close();

                    for (int i = 0; i < students.Count; i++)
                    {
                        // this is to avoid writing the query into the
other ones, instead it can be separate so it becomes just a single value
and is not repeated
                        // select the student id for the student in the
class with the index of the queue in the for loop
                        SqlCommand studentID = new SqlCommand("Select
StudentID from Student where profileID = (Select profileID from
UserProfile where Name = '" + students[i] + "')", sqlConnection);
                        SqlDataReader dataReader =
studentID.ExecuteReader();
                        while (dataReader.Read())
                        {
                            studentIDNumber =
Int32.Parse((String.Format("{0}", dataReader["StudentID"])));
                        }
                        dataReader.Close();

                        // select the sum of the marks awarded for a
given student for a given topic
                        // this will only include questions that they
have answered, if the student has not yet answered any questions, the
result will be zero
                        SqlCommand scores = new SqlCommand("Select
Sum(MarksAwarded) as 'Marks' from Answer inner join Question on
Answer.QuestionID = Question.QuestionID and Question.TopicID = (Select
TopicID from Topic where TopicName = '" +
topicComboBox.SelectedItem.ToString() + "') and Answer.StudentID = '" +
studentIDNumber + "'", sqlConnection);
                        SqlDataReader scoresReader =
```

```
scores.ExecuteReader();
                        string scoreX;
                        while (scoresReader.Read())
                        {
                            scoreX = scoresReader["Marks"].ToString();
                            topicProgressChart.Series["Scores"].Points.A
ddXY(students[i], scoreX);
                        }
                        scoresReader.Close();

                        SqlCommand totalScore = new SqlCommand("Select
Sum(MarksAwarded) as 'TotalMarks' from Answer inner join Question on
Question.QuestionID = Answer.QuestionID and Question.LevelID = '2'",
sqlConnection);
                        SqlDataReader sqlDataReader =
totalScore.ExecuteReader();
                        scoreTotal = 0;
                        while (sqlDataReader.Read())
                        {
                            scoreTotal =
Int32.Parse((String.Format("{0}", sqlDataReader["TotalMarks"])));
                            sDeviation.Add(scoreTotal);
                        }
                        sqlDataReader.Close();

                        SqlCommand studentScore = new SqlCommand("Select
Avg(MarksAwarded) as 'Avg' from Answer where StudentID = '" +
studentIDNumber + "'", sqlConnection);
                        SqlDataReader scoreReader =
studentScore.ExecuteReader();
                        while (scoreReader.Read())
                        {
                            y = Int32.Parse((String.Format("{0}",
scoreReader["Avg"])));
                            averageChart.Series["Average"].Points.AddXY(
students[i], y);
                        }
                        scoreReader.Close();

                        NormalDistribution();
                    }
                }

                catch (SqlException)
                {
                    MessageBox.Show("Error");
                }
            }

            else
            {
                MessageBox.Show("Choose topic");
            }
```

```
        }

        private double NormalDistribution()
        {
            // get the list of values of marks awarded for a given topic
by all students (not one student in particular) which is already stored
in a global variable (scoreTotal)
            // standard deviation is defined as the square root of
variance
            // variance is defined as Sxx = Σx^2 - n*mean^2

            // the sum of the total scores is already a value retrieved
by an aggregate sql function
            List<int> values = new List<int> { };
            SqlCommand sigmaX = new SqlCommand("Select MarksAwarded from
Answer inner join Question on Question.QuestionID = Answer.QuestionID
and Question.TopicID = (Select TopicID from Topic where TopicName = '" +
topicComboBox.SelectedItem.ToString() + "' and Answer.isMarked =
'True')", sqlConnection);
            SqlDataReader data = sigmaX.ExecuteReader();
            while (data.Read())
            {
                int value = Int32.Parse((String.Format("{0}",
data["MarksAwarded"])));
                values.Add(value);
            }
            data.Close();

            float numberMarks = values.Count;
            float totalSum = 0;
            for (int i = 0; i < numberMarks; i++)
            {
                totalSum += values[i];
            }

            float mean = totalSum / numberMarks;

            double sumX = 0;
            for (int y = 0; y < values.Count; y++)
            {
                sumX += Math.Pow(values[y], 2);
            }

            double valueStandard = Math.Sqrt((sumX / values.Count) -
Math.Pow(mean, 2));
            standardDeviationLabel.Text = valueStandard.ToString();

            meanLabel.Text = mean.ToString();

            index = 0;
            double probability = 0, zScore = 0;
            for (int a = 0; a < values.Count; a++)
            {
```

```
                // this says how spread out the data is from the mean
                int xValue = values[a];
                probability = (1 / (valueStandard * Math.Sqrt(2 *
Math.PI))) * Math.Exp(-0.5 * Math.Pow(((xValue - mean) / valueStandard),
2));
                scores.Add(probability.ToString());

                // store z scores in array
                zScore = (xValue - mean) / valueStandard;
                Zscores.Add(zScore.ToString());
            }

            distributionLabel.Text = scores[index].ToString();
            zScoreLabel.Text = zScore.ToString();

            // variance is the standard deviation squared
            double variance = Math.Pow(valueStandard, 2);
            varianceLabel.Text = variance.ToString();

            return probability;
        }

        private void topicComboBox_SelectedIndexChanged(object sender,
EventArgs e)
        {
            NormalDistribution();
        }

        private void nextButton_Click(object sender, EventArgs e)
        {
            try
            {
                index++;
                distributionLabel.Text = scores[index].ToString();
                // update z score label
                zScoreLabel.Text = Zscores[index].ToString();
            }

            catch (ArgumentOutOfRangeException)
            {
                // this means that the scores have ended, the index
value is equal to the size of the array containing the students' scores
                MessageBox.Show("End");
                nextButton.Hide();
            }
        }

        private void insertScores()
        {
            for(int j = 0; j<sDeviation.Count; j++)
            {
                // insert details regarding student scores into specific
sql table
```

```
                SqlCommand scoresUpdate = new SqlCommand(@"INSERT INTO
[dbo].[Score]
           ([TopicScore]
           ,[AverageScore]
           ,[StudentID]
           ,[TopicID]
           ,[TeacherID]
           ,[LevelID]
           ,[SubjectID])
     VALUES
           ('"+sDeviation[j].ToString()+"', '"+y+"', '"+studentIDNumber+
"', (Select TopicID from Topic where TopicName = '" +
topicComboBox.SelectedItem.ToString() + "'), '"+IDTextBox.Text+"', '2',
(Select SubjectID from Subject where SubjectTaken =
'"+subjectTextBox.Text+"'))", sqlConnection);
                scoresUpdate.ExecuteNonQuery();
            }

            MessageBox.Show("Finished");
        }
        private void viewScores_Closing(object sender, EventArgs e)
        {
            // insert scores after the teacher has reviewed them
            insertScores();
            MergeSort();
        }
        private void MergeSort()
        {
            int[] questions = new int[]
{3,2,2,3,4,5,64,3,3,3,55,3,3,2,4,3,4,5};
            int i = questions[0];
            int j = questions[questions.Length / 2];
            int indexNumber = i;
            int[] B = new int[] { };
            while (indexNumber < questions[questions.Length -1])
            {
                if(j >= questions[questions.Length - 1] || (i < j &&
questions[i] < questions[j]))
                {
                    B[indexNumber] = questions[i];
                    i++;
                }

                else
                {
                    B[indexNumber] = questions[j];
                    j++;
                }
                indexNumber++;

                MessageBox.Show(B.ToString());
            }
        }
```

```
        private void label4_Click(object sender, EventArgs e)
        {

        }
        private void chart2_Click(object sender, EventArgs e)
        {

        }
        private void subjectTextBox_TextChanged(object sender, EventArgs
e)
        {

        }
        private void meanLabel_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Eighth form (comment):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data.SqlClient;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Collections;

namespace Project
{
    public partial class Comment : Form
    {
        SqlConnection Connection = new SqlConnection("Data Source =
LAPTOP-CKS3DTGK; Initial Catalog=LatinRevision; Integrated
Security=True");

        public string typeofComment;
        public int senderProfileID;
        // queue
        public Queue<int> receivers = new Queue<int> { };
        public Comment(string type , int senderID, ArrayList receiver)
        {
```

```
            InitializeComponent();
            typeofComment = type;
            senderProfileID = senderID;
            for (int h = 0; h < receiver.Count; h++)
            {
                receivers.Enqueue(Convert.ToInt32(receiver[h]));
            }
            typeofCommentLabel.Text = typeofComment;
            Connection.Open();
        }

        private void Comment_Load(object sender, EventArgs e)
        {
            /*
if(typeofComment == "ParentCommunicate")
{
    SqlCommand insertComment3 = new SqlCommand(@"INSERT INTO
[dbo].[Comment]
([commentText]
,[SenderID]
,[CommentType])
VALUES
('" + commentTextBox.Text + "', '"+senderProfileID+"'
,'"+typeofComment+"')", Connection);
    insertComment3.ExecuteNonQuery();
}
            if (typeofComment == "CommunicateTeacher")
            {
                SqlCommand insertComment = new SqlCommand(@"INSERT INTO
[dbo].[Comment]
           ([commentText]
           ,[ReceiverID]
           ,[SenderID]
           ,[CommentType])
     VALUES
           ('" + commentTextBox.Text+"', '"+senderProfileID+"'
,'"+typeofComment+"')", Connection);
                insertComment.ExecuteNonQuery();
            }


            if(typeofComment == "Review")
            {
                SqlCommand insertComment1 = new SqlCommand(@"INSERT INTO
[dbo].[Comment]
           ([commentText]
           ,[SenderID]
           ,[CommentType])
     VALUES
           ('" + commentTextBox.Text + "', '"+senderProfileID+"'
,'"+typeofComment+"')", Connection);
                insertComment1.ExecuteNonQuery();
            }*/
```

```
        }

        private void commentTextBox_TextChanged(object sender, EventArgs
e)
        {

        }

        private void enterButton_Click(object sender, EventArgs e)
        {
            if (typeofComment == "StudentProgress")
            {
                try
                {
                    for (int f = 0; f < receivers.Count; f++)
                    {
                        SqlCommand insertComment2 = new
SqlCommand(@"INSERT INTO [dbo].[Comment]
           ([commentText]
           ,[SenderID]
           ,[ReceiverID]
           ,[CommentType])
     VALUES
           ('" + commentTextBox.Text + "', '" + senderProfileID + "' ,
'" + receivers.Dequeue() + "' ,'" + typeofComment + "')", Connection);
                        insertComment2.ExecuteNonQuery();
                    }
                    MessageBox.Show("Comment sent");
                    this.Close();
                }

                catch (ArgumentNullException)
                {
                    MessageBox.Show("Error");
                }
            }
        }
    }
}
```

Ninth form (mark students' work):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Data.SqlClient;
using System.Linq;
```

```
using System.Text;
using System.Collections;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Project
{
    public partial class Mark_Work : Form
    {
        SqlConnection sqlConnection = new SqlConnection("Data Source =
LAPTOP-CKS3DTGK; Initial Catalog=LatinRevision; Integrated
Security=True");

        public int profileConstructor, studentIDValue;
        public string studentAns;

        public Mark_Work(int profileID)
        {
            InitializeComponent();
            // profile id of the teacher from the constructor argument
which is to be used in subsequent queries
            profileConstructor = profileID;
        }
        private void Mark_Work_Load(object sender, EventArgs e)
        {
            sqlConnection.Open();
        }
        private void studentCombobox_SelectedIndexChanged(object sender,
EventArgs e)
        {
            studentNameLabel.Text =
studentComboBox.SelectedItem.ToString();

            // select student's student id to be used in other queries
            SqlCommand selectStudentID = new SqlCommand("Select
StudentID from Student where profileID = (Select profileID from
UserProfile where Name = '" + studentComboBox.SelectedItem.ToString() +
"')", sqlConnection);
            SqlDataReader IDReader = selectStudentID.ExecuteReader();
            while(IDReader.Read())
            {
                studentIDValue = Int32.Parse((String.Format("{0}",
IDReader["StudentID"])));
            }
            IDReader.Close();

            SqlCommand selectQuestions = new SqlCommand("Select
Question.QuestionName, Question.QuestionAnswer, Question.NumberofMarks,
Answer.AnswerGiven from Question inner join Answer on
Question.QuestionID = Answer.QuestionID and Answer.isMarked = 'False'
and Answer.StudentID = '" + studentIDValue + "'", sqlConnection);
            SqlDataReader questionReader =
selectQuestions.ExecuteReader();
```

```
            while(questionReader.Read())
            {
                string x = questionReader["QuestionName"].ToString();
                questionsComboBox.Items.Add(x);
            }
            questionReader.Close();
        }

        private void loadNamesButton_Click(object sender, EventArgs e)
        {
            // select every student name using link table student
teacher
            // this means that by having a corresponding teacher id, the
students are in the same class
            // SqlCommand selectName = new SqlCommand("Select Name from
UserProfile, StudentTeacher, Student where StudentTeacher.StudentID =
UserProfile.profileID and StudentTeacher.TeacherID =
'"+profileConstructor+"' and Student.profileID =
StudentTeacher.StudentID", sqlConnection);
            SqlCommand selectName = new SqlCommand("Select
UserProfile.Name from ((UserProfile inner join Student on
UserProfile.profileID = Student.profileID) inner join StudentTeacher on
Student.StudentID = StudentTeacher.StudentID and
StudentTeacher.TeacherID = '"+profileConstructor+"')", sqlConnection);
            SqlDataReader nameReader = selectName.ExecuteReader();
            while (nameReader.Read())
            {
                string k = nameReader["Name"].ToString();
                studentComboBox.Items.Add(k);
            }
            nameReader.Close();

            loadNamesButton.Enabled = false;
        }

        private void nextButton_Click(object sender, EventArgs e)
        {
            // exception handling
            if (numberMarksTextBox.Text != "")
            {
                try
                {
                    // update the answer table with the input of number
of marks set by the teacher where student id corresponds to the student
who answered the question
                    SqlCommand updateMarks = new SqlCommand("Update
Answer set MarksAwarded = '" + numberMarksTextBox.Text + "', isMarked =
'True' where AnswerID = (Select AnswerID from Answer where AnswerGiven =
'" + studentAns + "' and StudentID = '" + studentIDValue + "')",
sqlConnection);
                    updateMarks.ExecuteNonQuery();
                    MessageBox.Show("Question marked succesffully");
                }
```

```
                catch (SqlException)
                {
                    MessageBox.Show("Marks entered incorrectly");
                }
            }

            else
            {
                MessageBox.Show("Input number of marks");
            }
        }

        private void doneQuestionButton_Click_1(object sender, EventArgs
e)
        {
            try
            {
                questionLabel.Text =
questionsComboBox.SelectedItem.ToString();

                SqlCommand selectQAnswer = new SqlCommand("Select
Question.QuestionAnswer, Question.NumberofMarks, Answer.AnswerGiven from
Question inner join Answer on Question.QuestionID = Answer.QuestionID
and Question.QuestionName =
'"+questionsComboBox.SelectedItem.ToString()+"' and Answer.StudentID =
'"+studentIDValue+"'", sqlConnection);
                SqlDataReader resultReader =
selectQAnswer.ExecuteReader();
                while(resultReader.Read())
                {
                    string h =
resultReader["QuestionAnswer"].ToString();
                    originalAnswerLabel.Text = h;

                    studentAns = resultReader["AnswerGiven"].ToString();
                    answerLabel.Text = studentAns;

                    string i = resultReader["NumberofMarks"].ToString();
                    numberMarksLabel.Text = i;
                }
                resultReader.Close();
            }

            catch(NullReferenceException)
            {
                MessageBox.Show("No question selected");
            }
        }
        private void questionsComboBox_SelectedIndexChanged(object
sender, EventArgs e)
        {
            // leave empty
```

```
        }

        private void label8_Click(object sender, EventArgs e)
        {
            // leave empty
        }
    }
}
```

Tenth form (parent login):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Project
{
    public partial class ParentLoggedin : Form
    {
        SqlConnection Connection = new SqlConnection("Data Source =
LAPTOP-CKS3DTGK; Initial Catalog=LatinRevision; Integrated
Security=True");

        private int IDProfile;
        public ParentLoggedin(string parentName, int profileID)
        {
            InitializeComponent();
            nonelabel.Text = parentName;
            IDProfile = profileID;
        }
        private void ParentLoggedin_Load(object sender, EventArgs e)
        {
            Connection.Open();
            SqlCommand nameParent = new SqlCommand( "Select count (*)
Name from UserProfile", Connection);
            nameParent.ExecuteNonQuery();
        }
        private void viewProgressButton_Click(object sender, EventArgs
e)
        {
            ViewScores viewScores = new ViewScores(IDProfile);
            viewScores.Show();
        }
```

```
        private void communicateTeacherButton_Click(object sender,
EventArgs e)
        {
            /*
            string c = "CommunicateTeacher";
            Comment comment = new Comment(c);
            comment.Show();*/
        }

        private void viewMessagesButton_Click(object sender, EventArgs
e)
        {
            ViewMessages viewMessages = new ViewMessages();
            viewMessages.Show();
        }
    }
}
```

Eleventh form (view messages):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Project
{
    public partial class ViewMessages : Form
    {
        public ViewMessages()
        {
            InitializeComponent();
        }

        private void ViewMessages_Load(object sender, EventArgs e)
        {
            // this is not finished due to a lack of time, however, this
would have been done similarly to the method that was used to display
questions (using an index to change the text of a label to a value from
sql stored in a list)
        }
    }
}
```

This algorithm is to interpret numerical values of various elasticities of demand and supply for A Level Economics A which is easier than having to look for tables to interpret them, it is also made faster by the choice to continue to input values after the system has displayed a response, thus it is recursive.

Algorithm that uses recursion with the base case of the boolean variable being false which is set to false when the user inputs that they are done (user defined/ recursive algorithm):

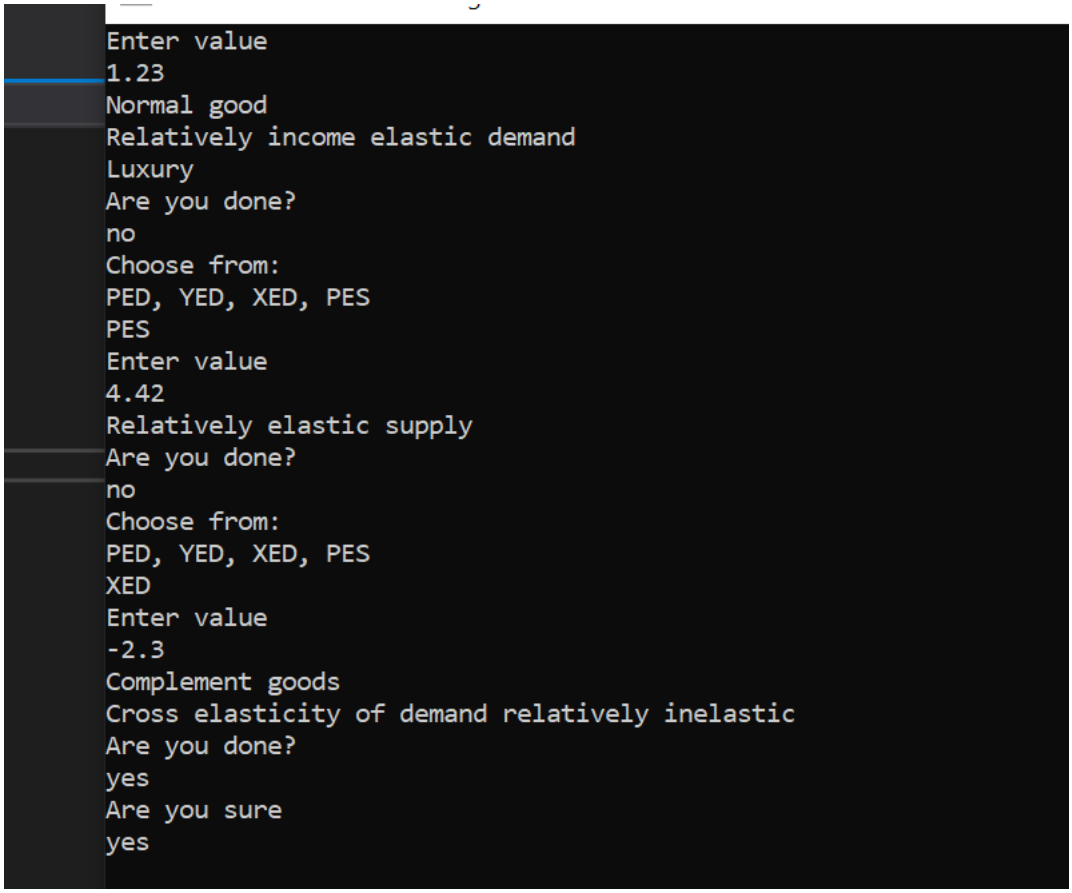
```
using System;
using System.Collections.Generic;
using System.Text;

namespace Elasticity
{
    class Hangman
    {
        static void Main(string[] args)
        {
            bool game = true;
            while (game == true)
            {
                Console.WriteLine("Choose from:");
                Console.WriteLine("PED, YED, XED, PES");
                string elasticity = Console.ReadLine();
                Console.WriteLine("Enter value");
                decimal value = Convert.ToDecimal(Console.ReadLine());
                if (elasticity == "PED")
                {
                    if (value < -1)
                    {
                        Console.WriteLine("Relatively elastic demand");
                    }
                    else if (value == 1)
                    {
                        Console.WriteLine("Unitary elasticity");
                    }
                    else if (value > -1)
                    {
                        Console.WriteLine("Relatively inelastic
demand");
                    }
                }
                if (elasticity == "YED")
                {
                    if (value < 0)
                    {
                        Console.WriteLine("Inferior good");
                        if (value > -1)
                        {
                            Console.WriteLine("Relatively income
inelastic demand");
```

```
                        }
                        else if (value < -1)
                        {
                            Console.WriteLine("Relatively income elastic
demand");
                        }
                    }
                    if (value > 0)
                    {
                        Console.WriteLine("Normal good");
                        if (value < 1)
                        {
                            Console.WriteLine("Relatively income
inelastic demand");
                        }
                        else if (value > 1)
                        {
                            Console.WriteLine("Relatively income elastic
demand");
                            if (value > 2)
                            {
                                Console.WriteLine("Luxury");
                            }
                        }
                    }
                }
                if (elasticity == "XED")
                {
                    if (value == 0)
                    {
                        Console.WriteLine("Unrelated goods");
                    }
                    else if (value < 0)
                    {
                        Console.WriteLine("Complement goods");
                        if (value > -1 || value < 1)
                        {
                            Console.WriteLine("Cross elasticity of
demand relatively inelastic");
                        }
                        else if (value < -1)
                        {
                            Console.WriteLine("Cross elasticity of
demand relatively elastic");
                        }
                        else if (value > 0)
                        {
                            Console.WriteLine("Substitute goods");
                            if (value > 1)
                            {
                                Console.WriteLine("Cross elasticity of
demand relatively elastic");
                            }
```

```
                        }
                    }
                }
                if (elasticity == "PES")
                {
                    if (value < 0)
                    {
                        Console.WriteLine("Price elasticity of supply is
positive");
                    }
                    else if (value == 0)
                    {
                        Console.WriteLine("Perfectly price inelastic
supply");
                    }
                    else if (value < 1)
                    {
                        Console.WriteLine("Relatively inealstic
supply");
                    }
                    else if (value > 1)
                    {
                        Console.WriteLine("Relatively elastic supply");
                    }
                }
                Console.WriteLine("Are you done?");
                string choice1 = Console.ReadLine();
                if (choice1 == "Yes" || choice1 == "yes")
                {
                    Console.WriteLine("Are you sure");
                    string ans = Console.ReadLine();
                    if (ans == "Yes" || ans == "yes")
                    {
                        game = false;
                    }
                }
                else if (choice1 == "No" || choice1 == "no")
                {
                    game = true;
                }
            }
        }
    }
}
```

Evidence that it works (testing):



Testing:

The values used in the testing section are different from the ones in the database testing because the testing section is based on user input and the former on data inserted manually.

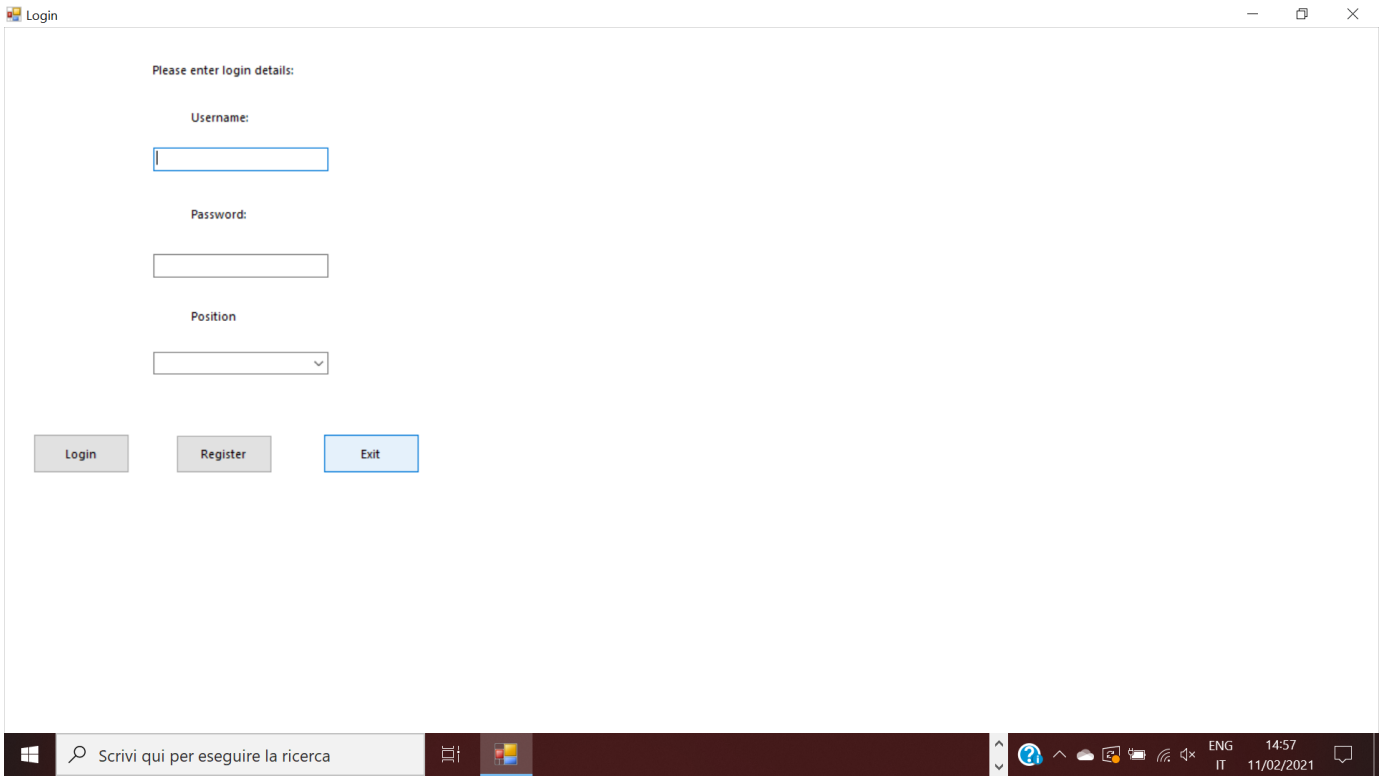
Test ID	Description	Expected result	Pass/ fail
1	Register as a student	Register successfully storing details in database (tables: user profile and student + link tables)	Pass
2	Register as a teacher	Register successfully storing details in database (tables: user profile and teacher + link tables)	Pass
3	Register as a parent	Register successfully storing details in database (tables: user profile and parent + link tables)	Pass
4	Login with student credentials	Load student menu page	Pass
5	Login with teacher credentials	Load teacher menu page	Pass

6	Login with parent credentials	Load parent menu page	Pass
7	Change selected role (choose different types of user at the same time)	Only be able to access one grouped box of information for a type of user at a time	Pass
8	Erroneous data when trying to register: Entering a username that already exists	Not being able to register, error message	Pass
9	Leaving text boxes empty in register form	Error message	Pass
10	Entering a username and password but not user specific information for each type of user	Error message as account should not be created	Pass
11	Answer questions as a student	Be able to move onto the next question which belongs to the chosen topic and enter an answer for each question	Pass
12	Leaving login credentials empty and entering wrong details	User not able to login, error message	Pass
13	Mark students' work from a teacher account	View the original question that was answered with its corresponding answer compareed to the one submitted by the student and the mark allocation and be able to input a number of marks which updates the sql table, answer where the question was originally answered	Pass
14	View students' scores as a graph of their individual marks against other students' for one topic	Display the bar plot of marks awarded for a given topic for the students in one class and the averages as a pie chart	Pass
15	Communicate with other users	Send a message to other users through forms and view messages within the application	Part fail (the receiver and sender are successfully selected and passed as an argument but there wasn't time to display the comment (also because this would

			have been similar to previous forms)
16	Database connection	Check that the insert statements to sql work successfully updating the relevant information where required including relationship tabels	Pass
17	Inserting user input data into sql	Every piece of data prompted from the user updated successfully into the database	Pass
18	Display questions retrieved from the database into windows forms using a label and array list	Display questions so the user can answer each one in a user-friendly way and store each answer into the corresponding field of the database	Pass
19	Insert hash value of username using hashing algorithm from the A Level textbook into the database and login with the original username selected	Avoid inserting actual username into the database but still be able to login with real username	Pass (functionality though there are security issues discussed earlier)

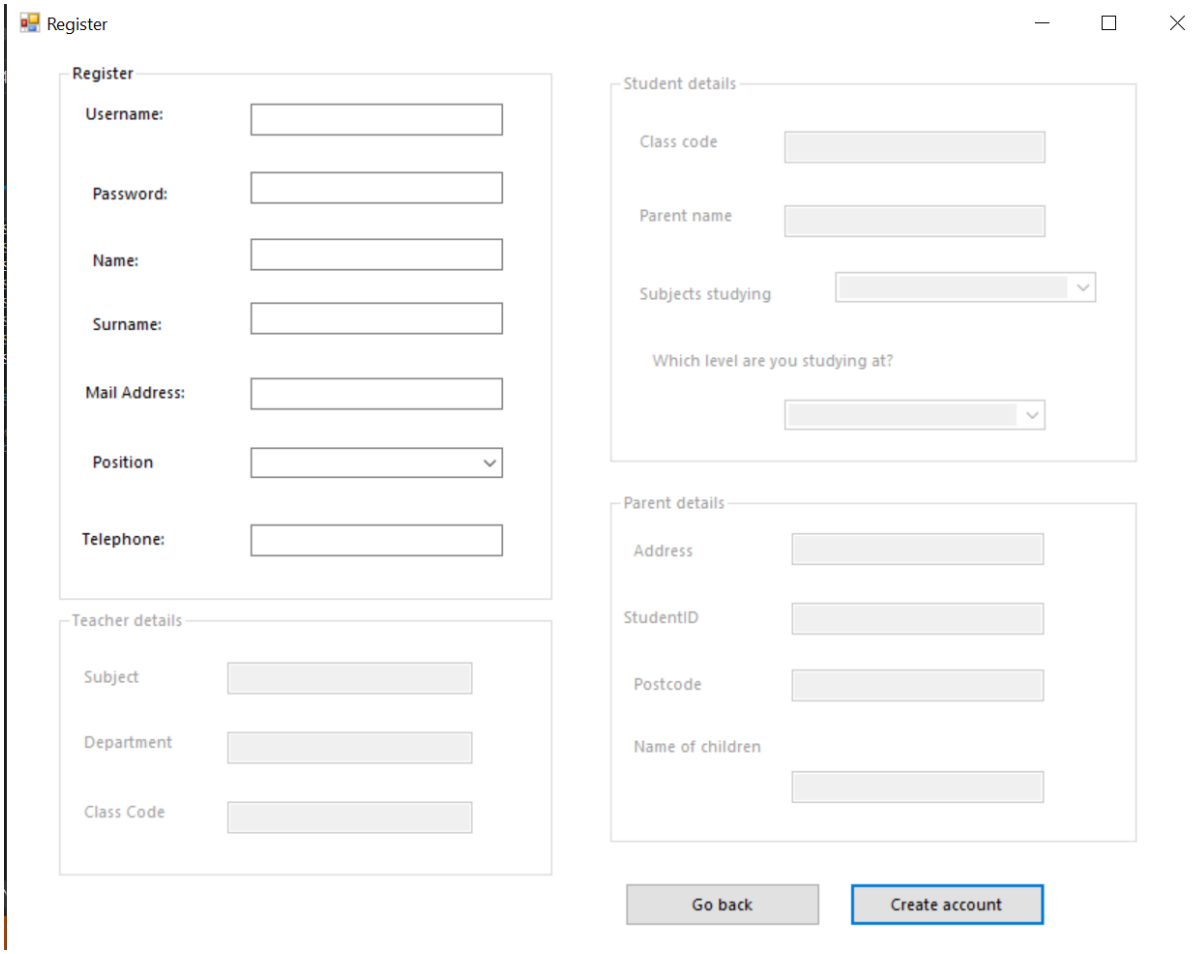
Each screenshot corresponds to the equivalent test id number.

The first form that is opened when the project is run:

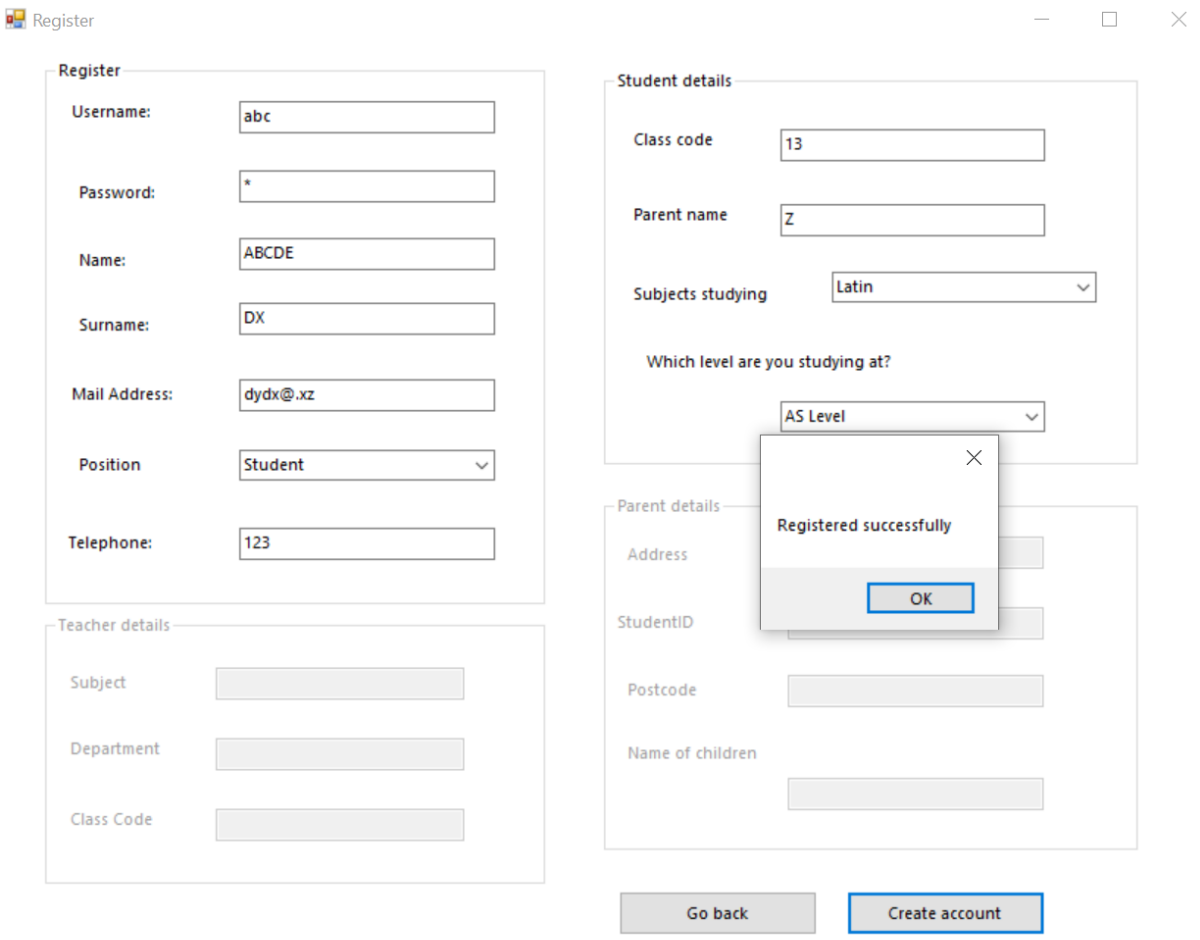


After clicking the exit button, it exits the application and it doesn't appear on the application bar:

Initial load of the register form with all specific user registration disabled:



Test screenshot 1:



UserProfile (only because the subsequent information has been validated):

profileID	Username	Password	Name	Surname	Mail	Telephone	Role
376	329 ...	a ...	ABCDE ...	DX ...	dydx@.xz ...	123 ...	Student

Student:

StudentID	profileID	QuestionsRight	LevelID	ParentName	classCode
141	376	*NULL*	2	abc ...	13

StudentTeacher:

StudentID	TeacherID	SubjectID
141	39	10

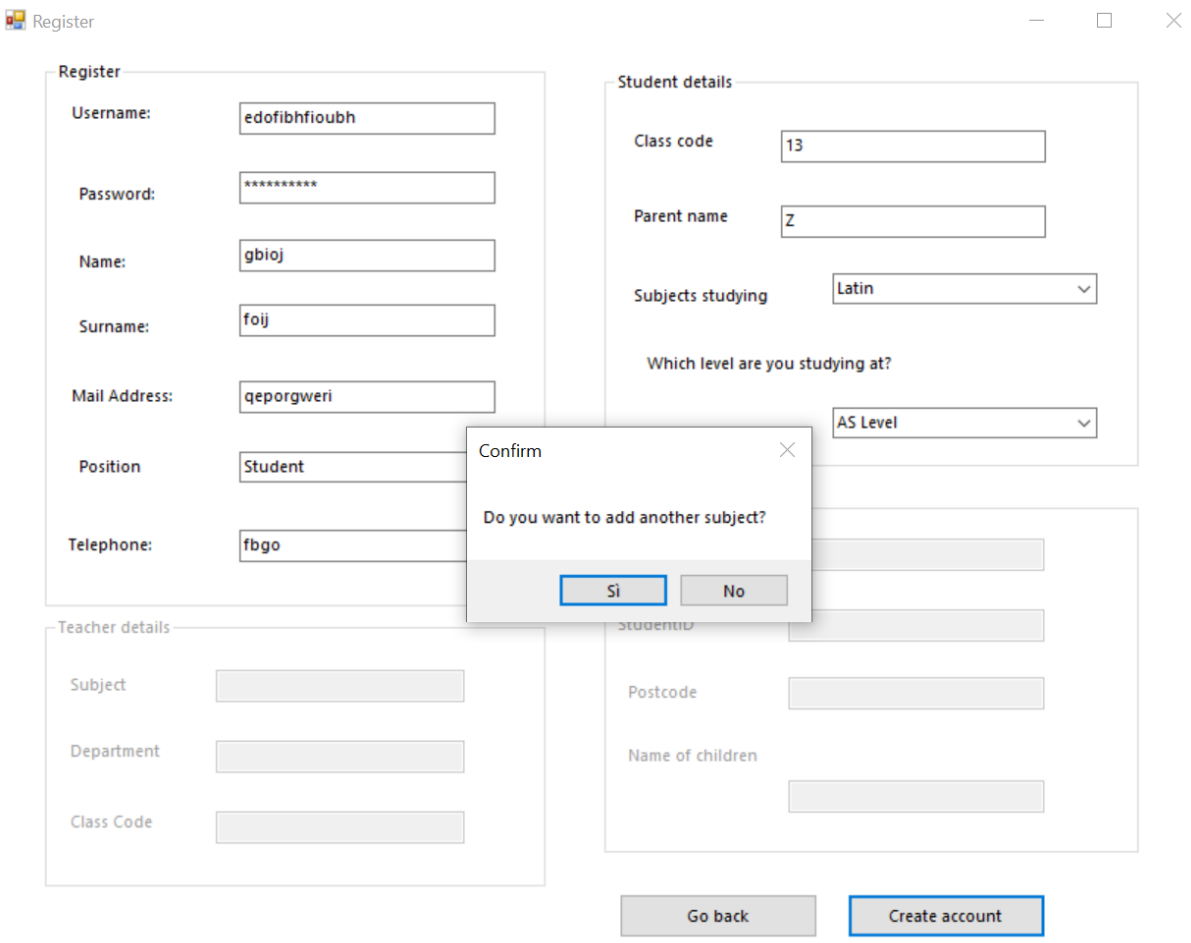
Subject:

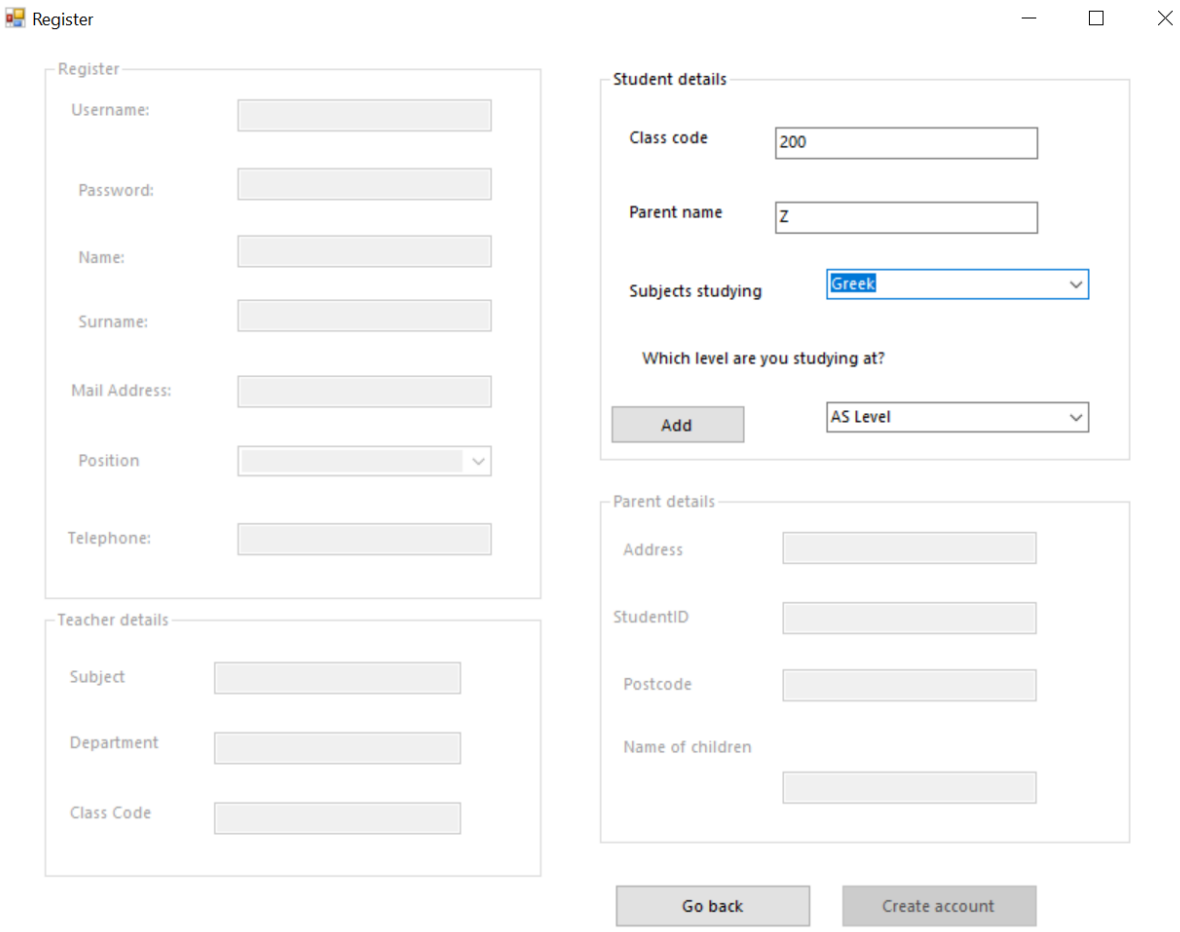
SubjectID	SubjectTaken
9	Ancient History...
10	Latin ...
11	Greek ...
12	Classical Civiliza...

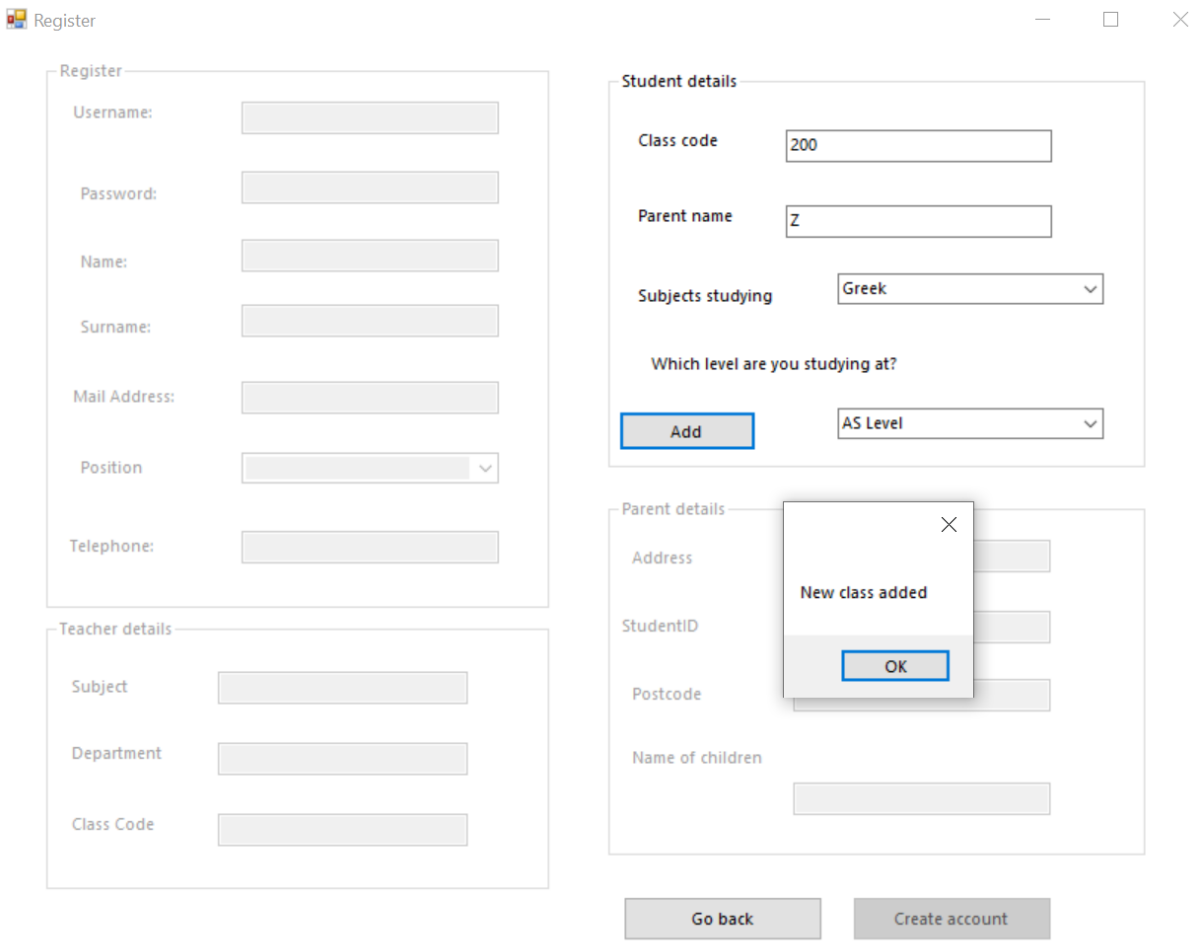
StudentSubject:

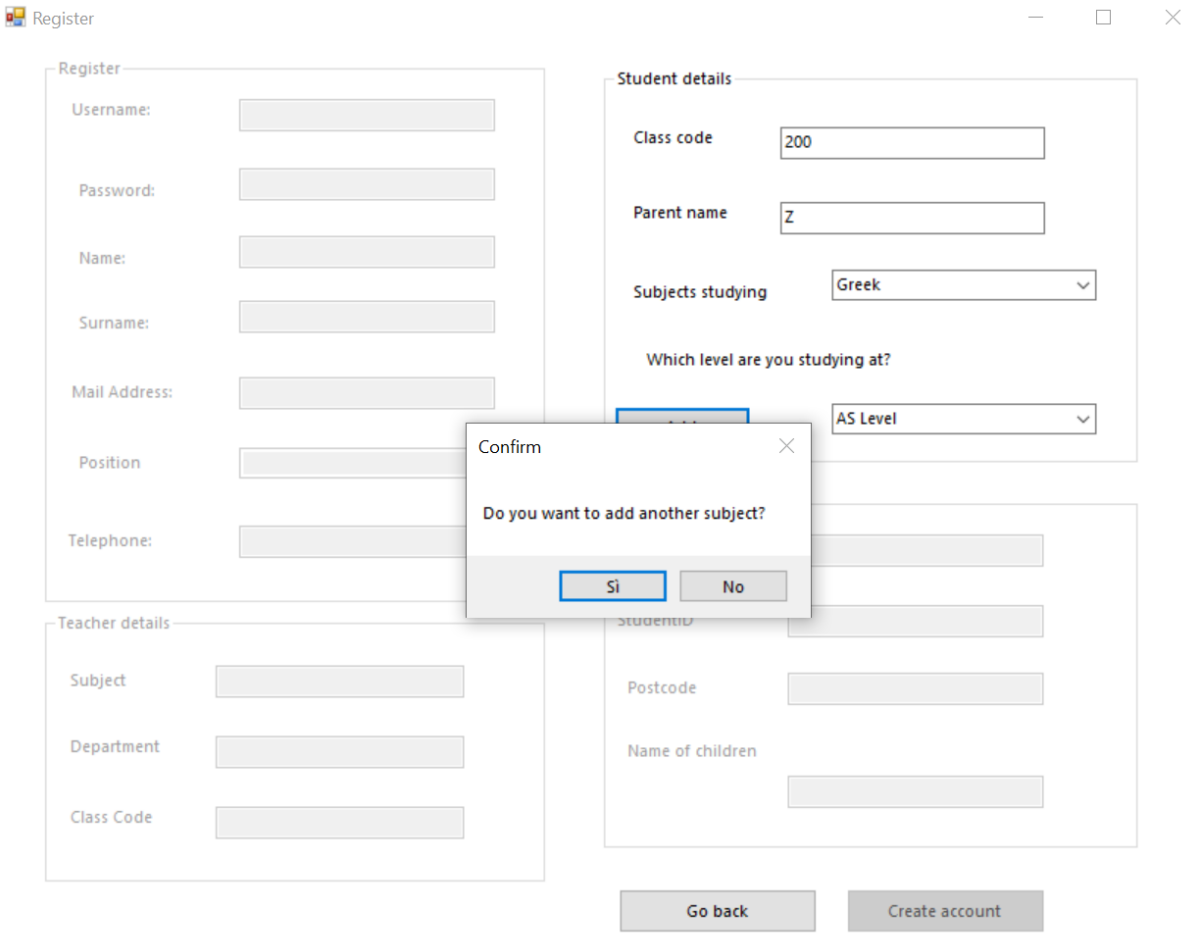
StudentID	SubjectID
376	10

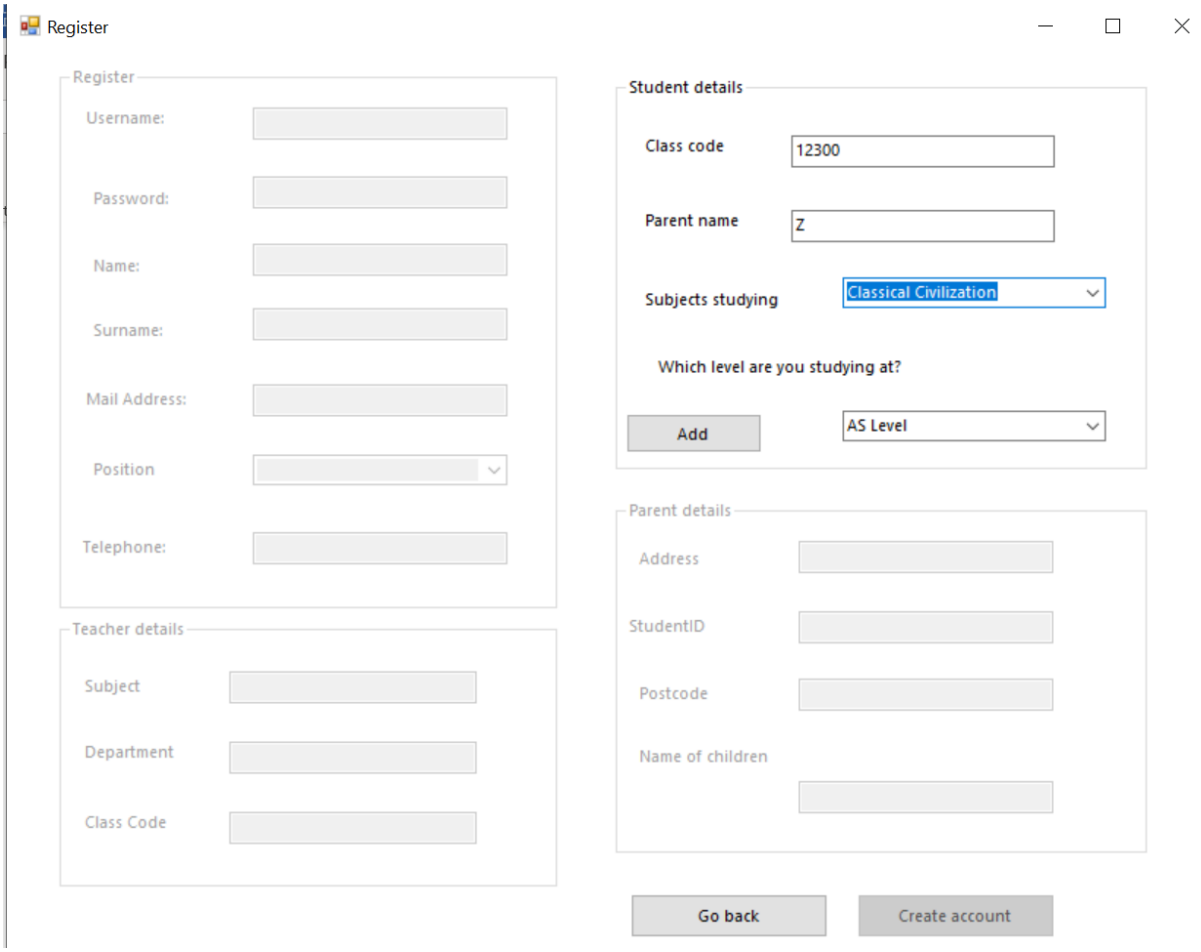
Evidence that the user can add more than one class and therefore the recursive algorithm works:




Register
Register
Username:
Password:
Name:
Surname:
Mail Address:
Position
Telephone:
Teacher details
Subject
Department
Class Code
Student details
Class code
200
Parent name
Z
Subjects studying
Greek
Which level are you studying at?
Add
AS Level
Parent details
Address
StudentID
Postcode
Name of children
Go back
Create account

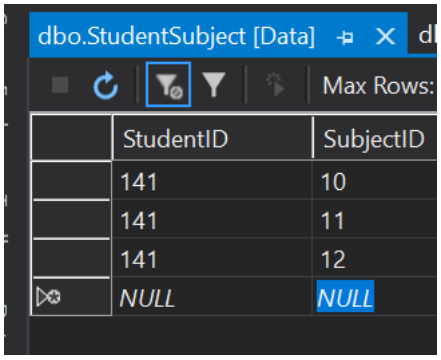

Register
Register
Username:
Password:
Name:
Surname:
Mail Address:
Position
Telephone:
Teacher details
Subject
Department
Class Code
Student details
Class code
200
Parent name
Z
Subjects studying
Greek
Which level are you studying at?
Add
AS Level
Parent details
Address
StudentID
Postcode
Name of children
New class added
OK
Go back
Create account


Register
Register
Username:
Password:
Name:
Surname:
Mail Address:
Position
Telephone:
Teacher details
Subject
Department
Class Code
Student details
Class code
200
Parent name
Z
Subjects studying
Greek
Which level are you studying at?
AS Level
Confirm
Do you want to add another subject?
Sì
No
Postcode
Name of children
Go back
Create account



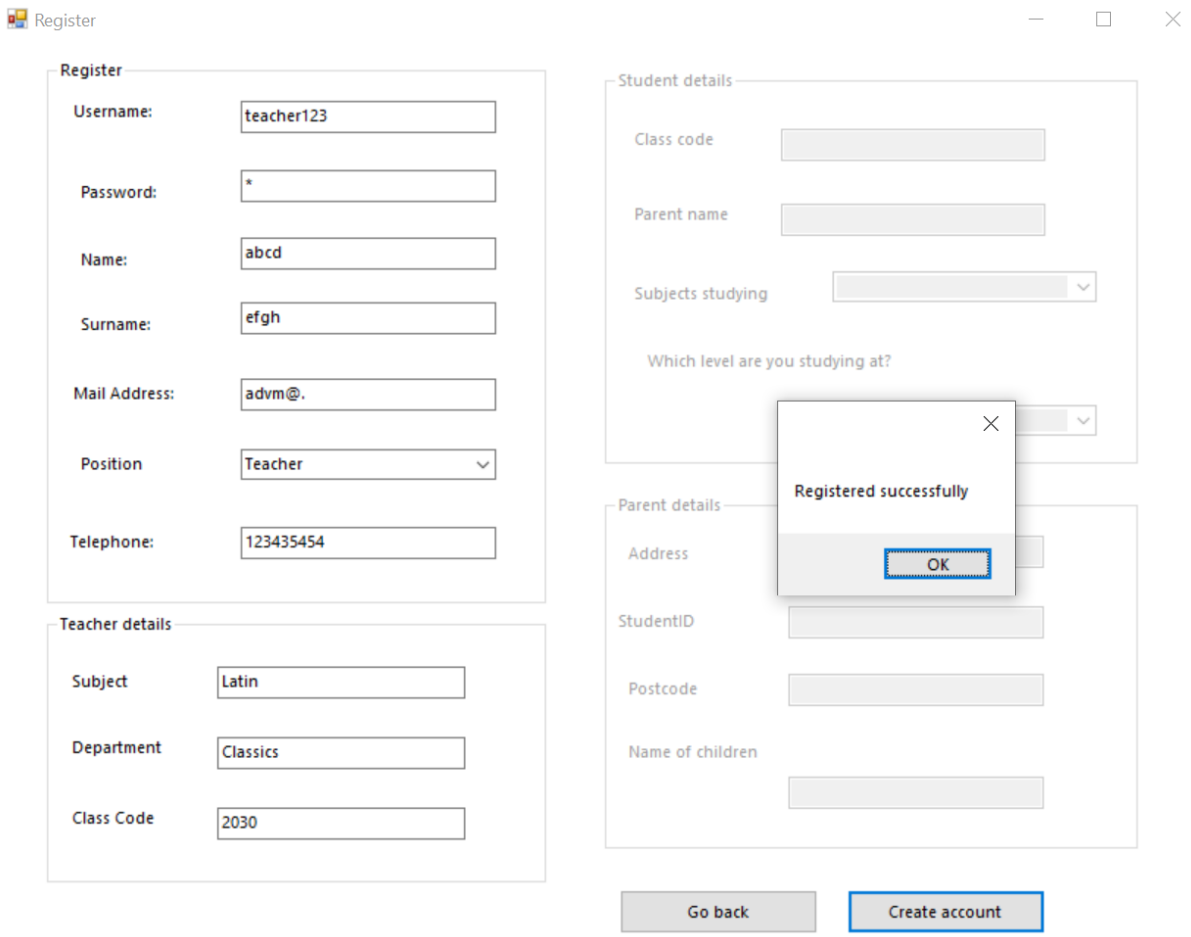
Until the user says they do not wish to add any more classes.

Evidence of database link tables updated successfully:



The table student teacher which says which student is in which class will be updated when there will be a teacher for that subject and provided a class code.

Test screenshot 2:



(The previous values have been deleted to make testing clearer):

UserProfile:

profileID	Username	Password	Name	Surname	Mail	Telephone	Role
377	518 ...	a ...	abcd ...	efgh ...	advm@. ...	123435454 ...	Teacher

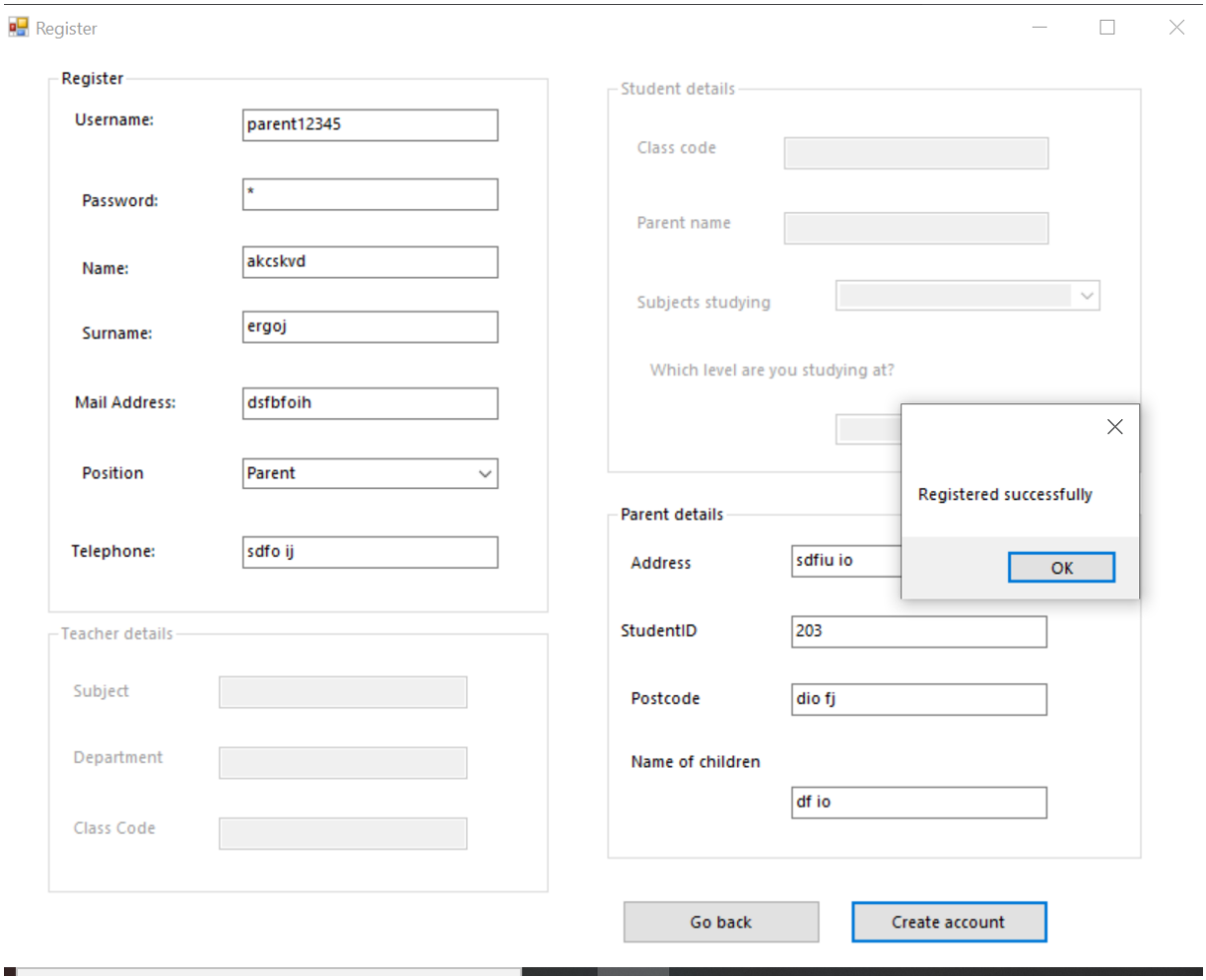
Teacher:

TeacherID	profileID	Department	ClassCode
87	377	Classics ...	2030

SubjectTeacher:

TeacherID	SubjectID
87	10

Test screenshot 3:



UserProfile:

profileID	Username	Password	Name	Surname	Mail	Telephone	Role
378	448 ...	a ...	akcskvd ...	ergoj ...	dsfbfoih ...	sdfo ij ...	Parent

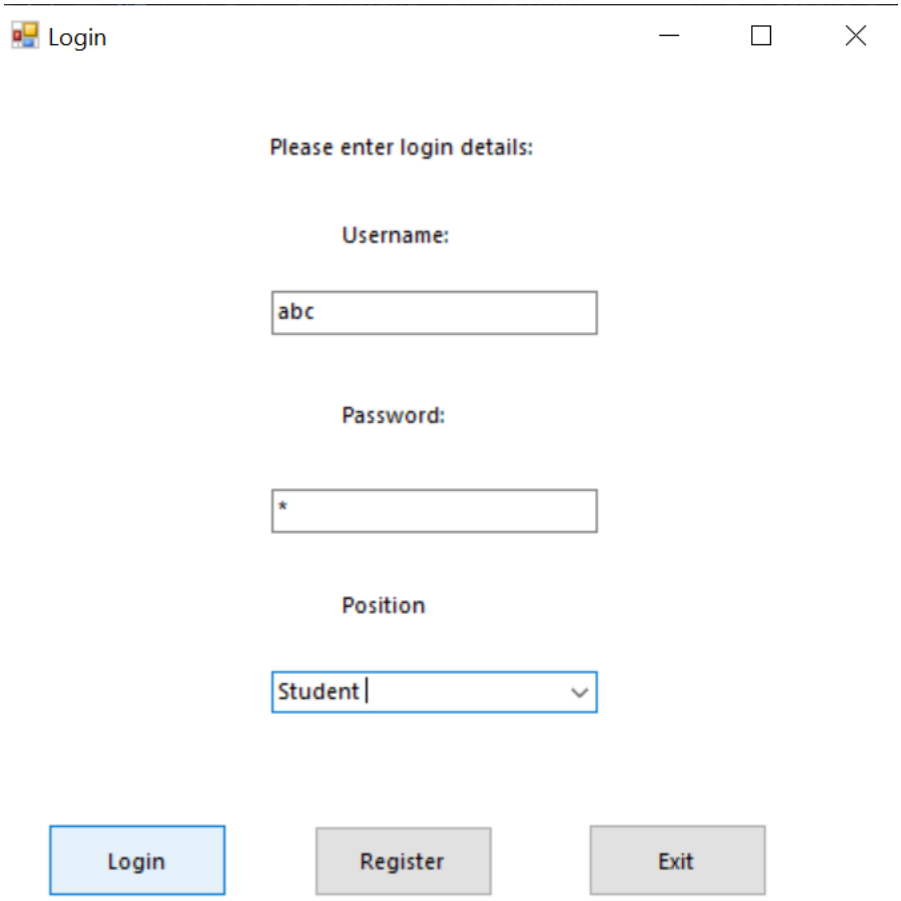
Parent:

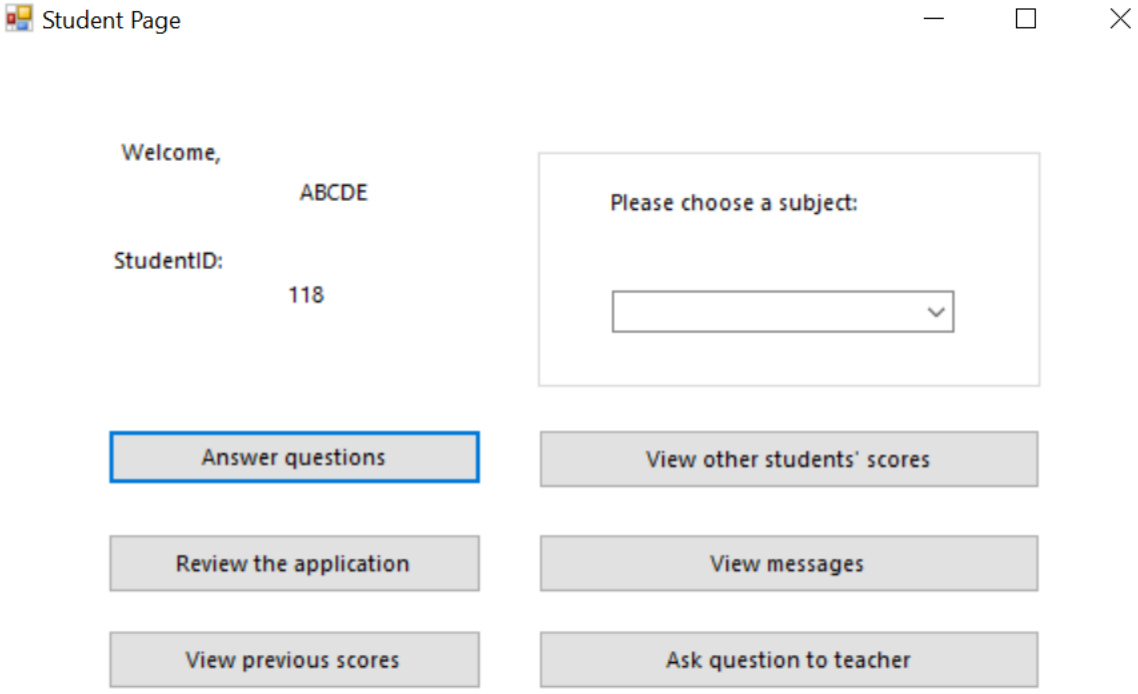
ParentID	Address	NumberofKids	NameofChild	profileID
44	sdfiu io ...	1	dio fj ...	378

ParenStudent:

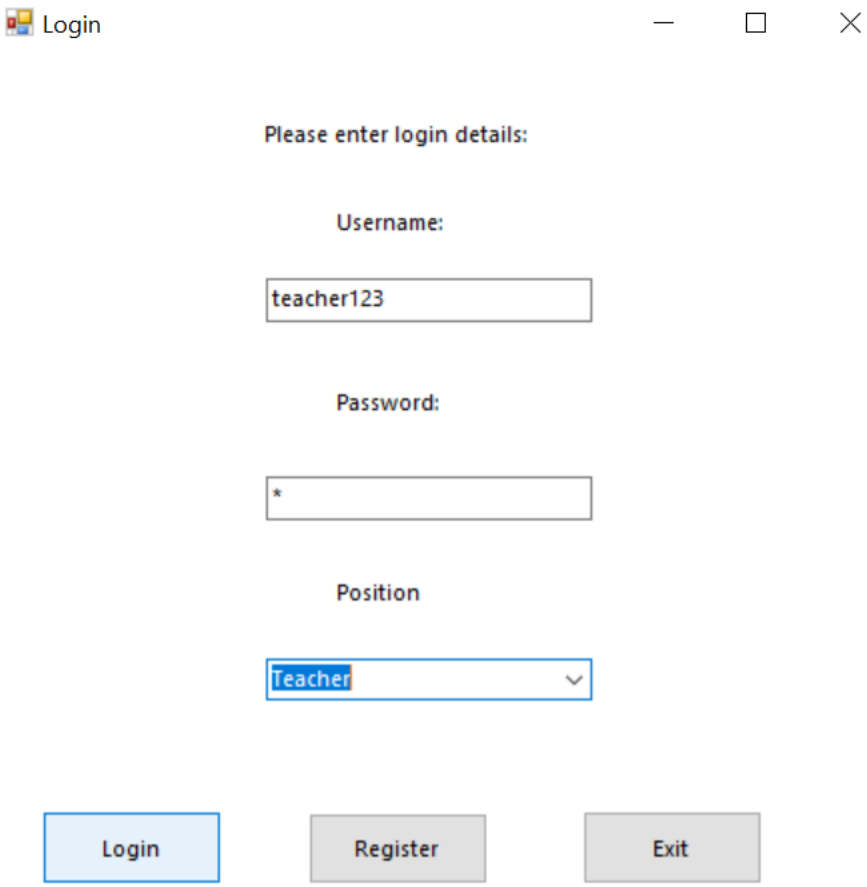
StudentID	ParentID
203	44

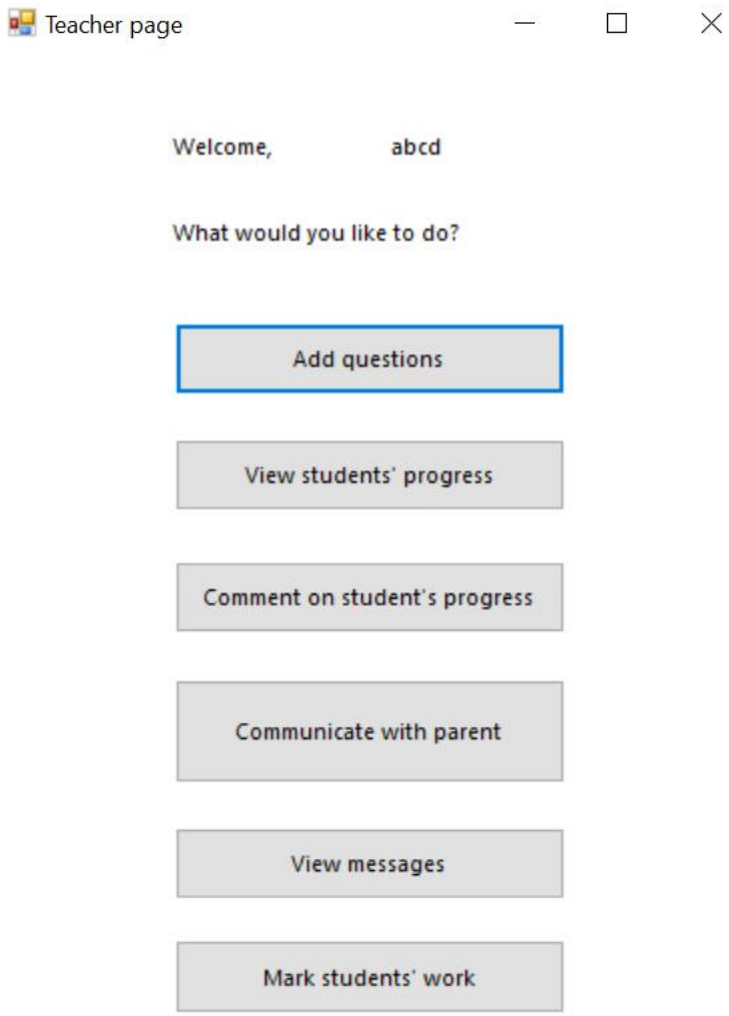
Test screenshot 4:


Login
Please enter login details:
Username:
abc
Password:
*
Position
Student
Login
Register
Exit

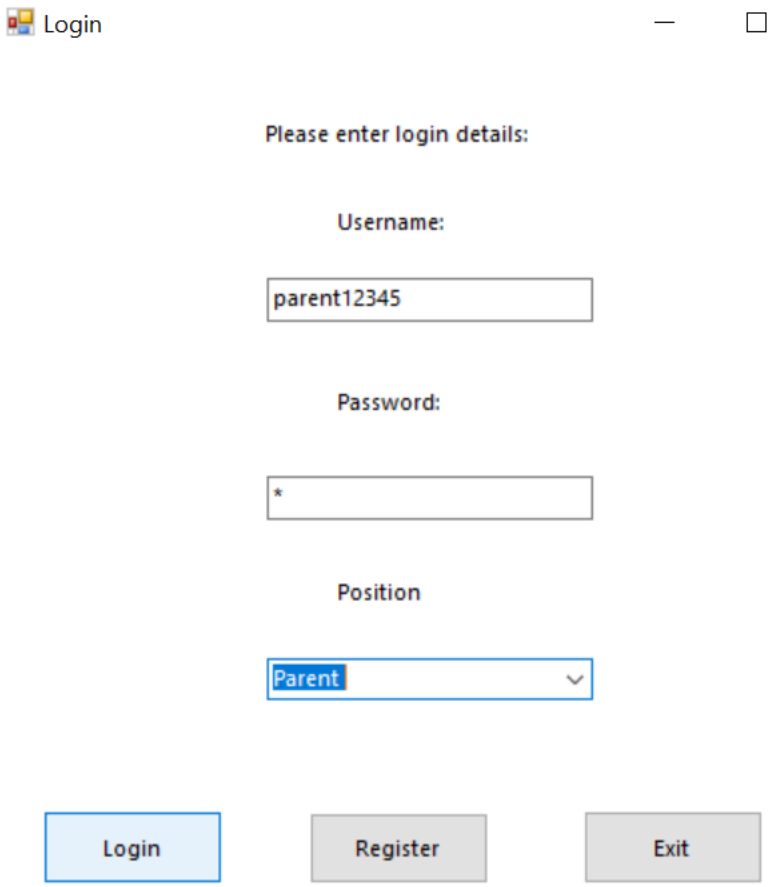

Student Page
Welcome,
ABCDE
StudentID:
118
Please choose a subject:
Answer questions
View other students' scores
Review the application
View messages
View previous scores
Ask question to teacher

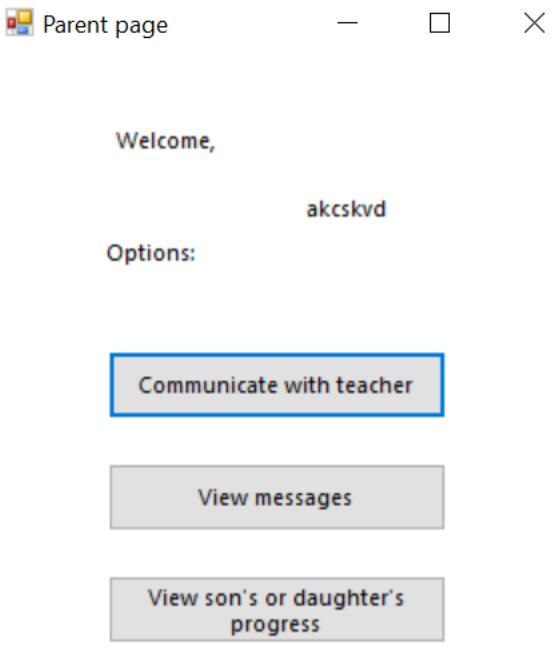
Test screenshot 5:



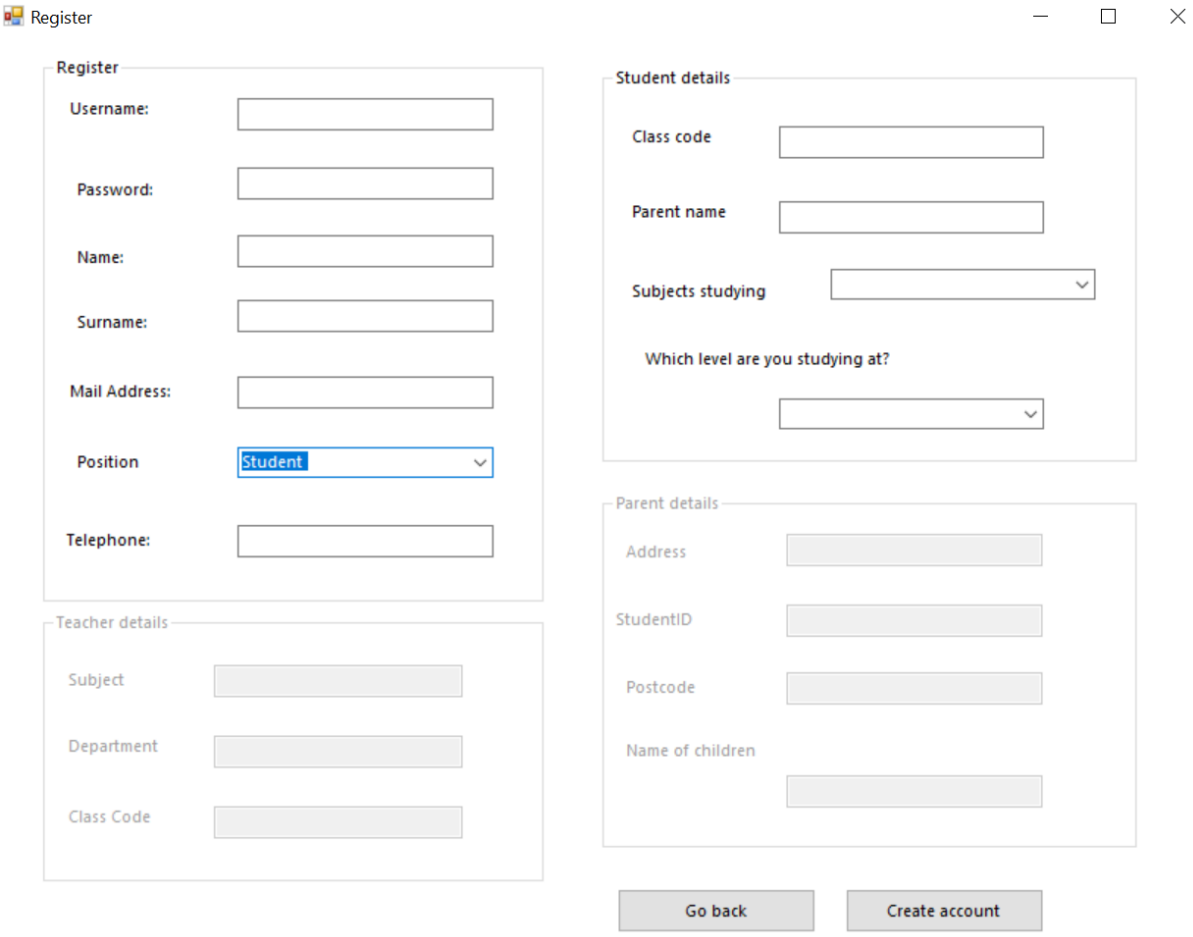


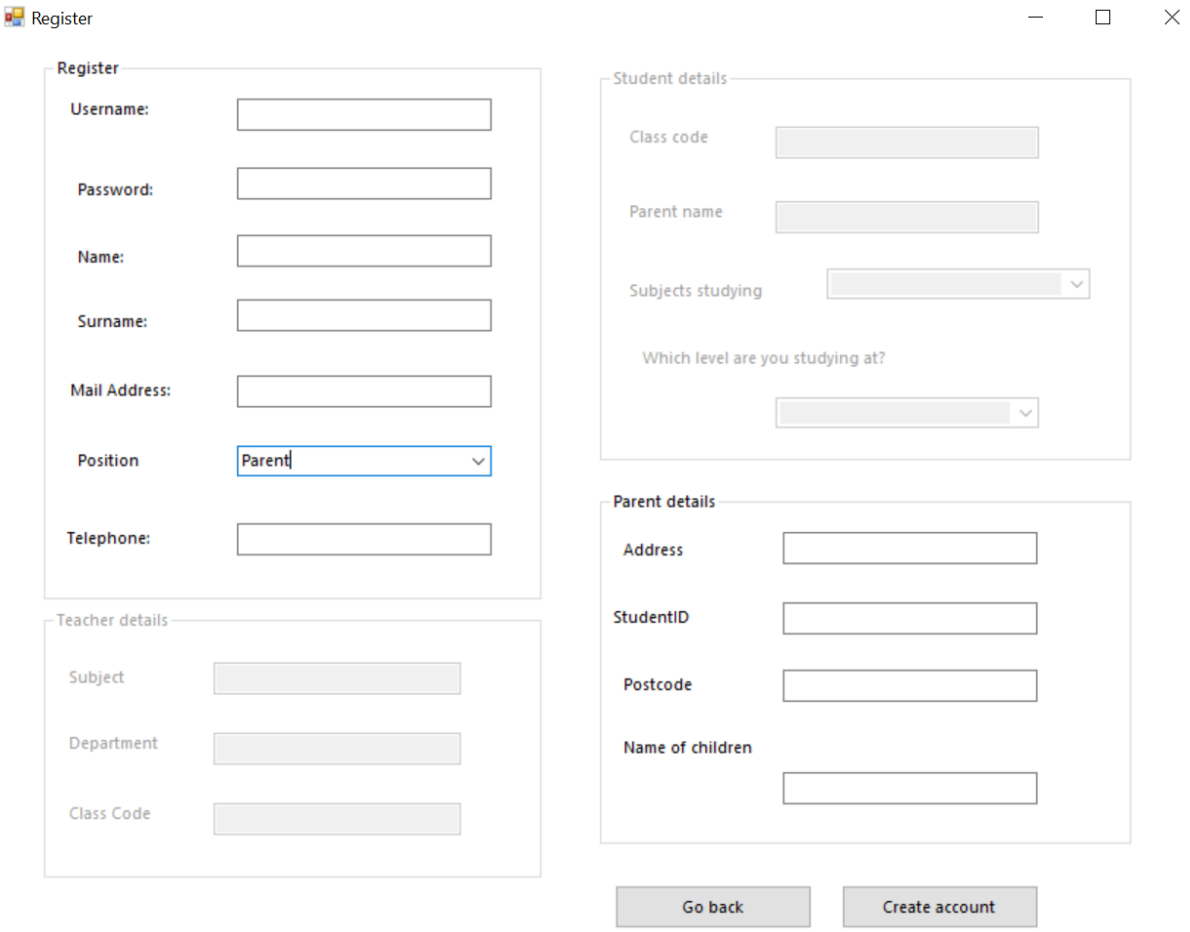
Test screenshot 6:

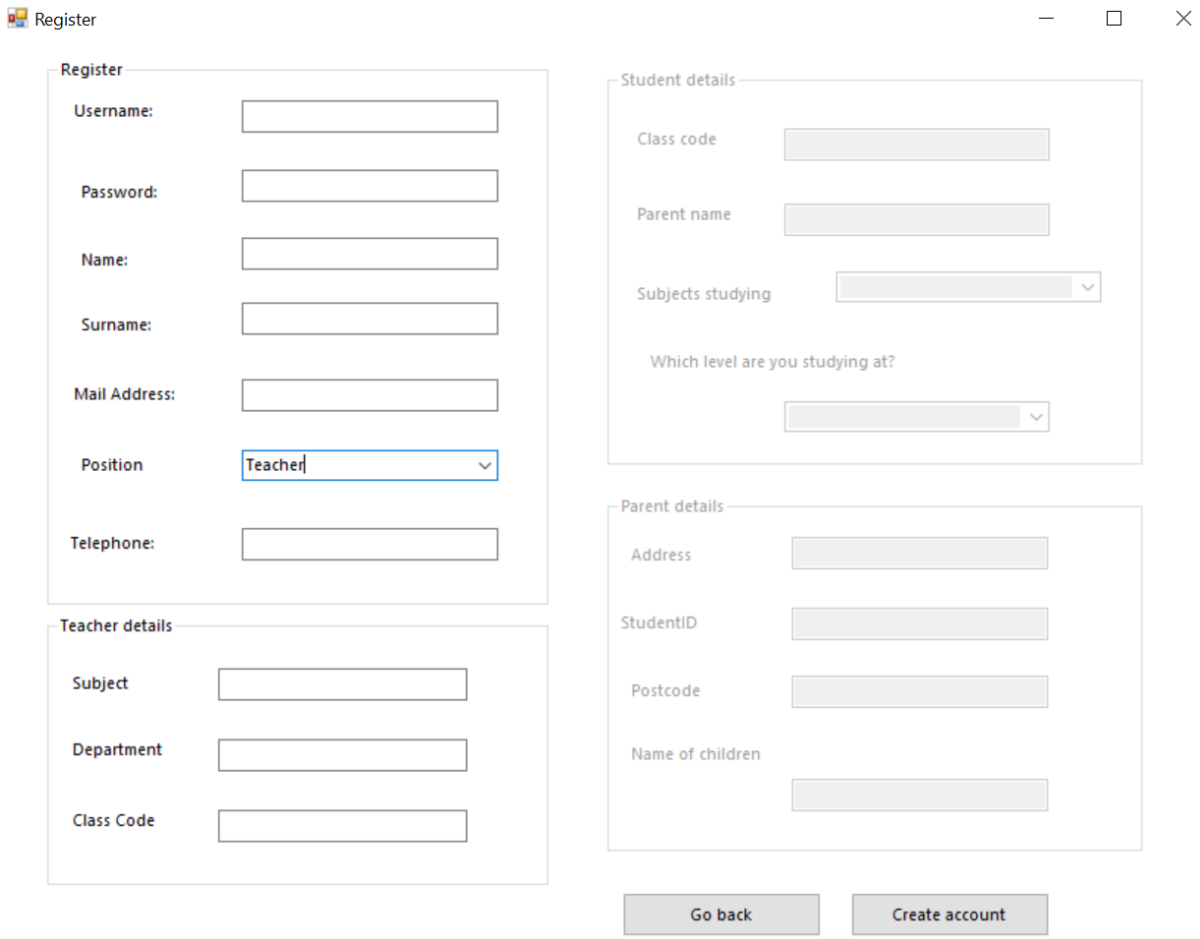




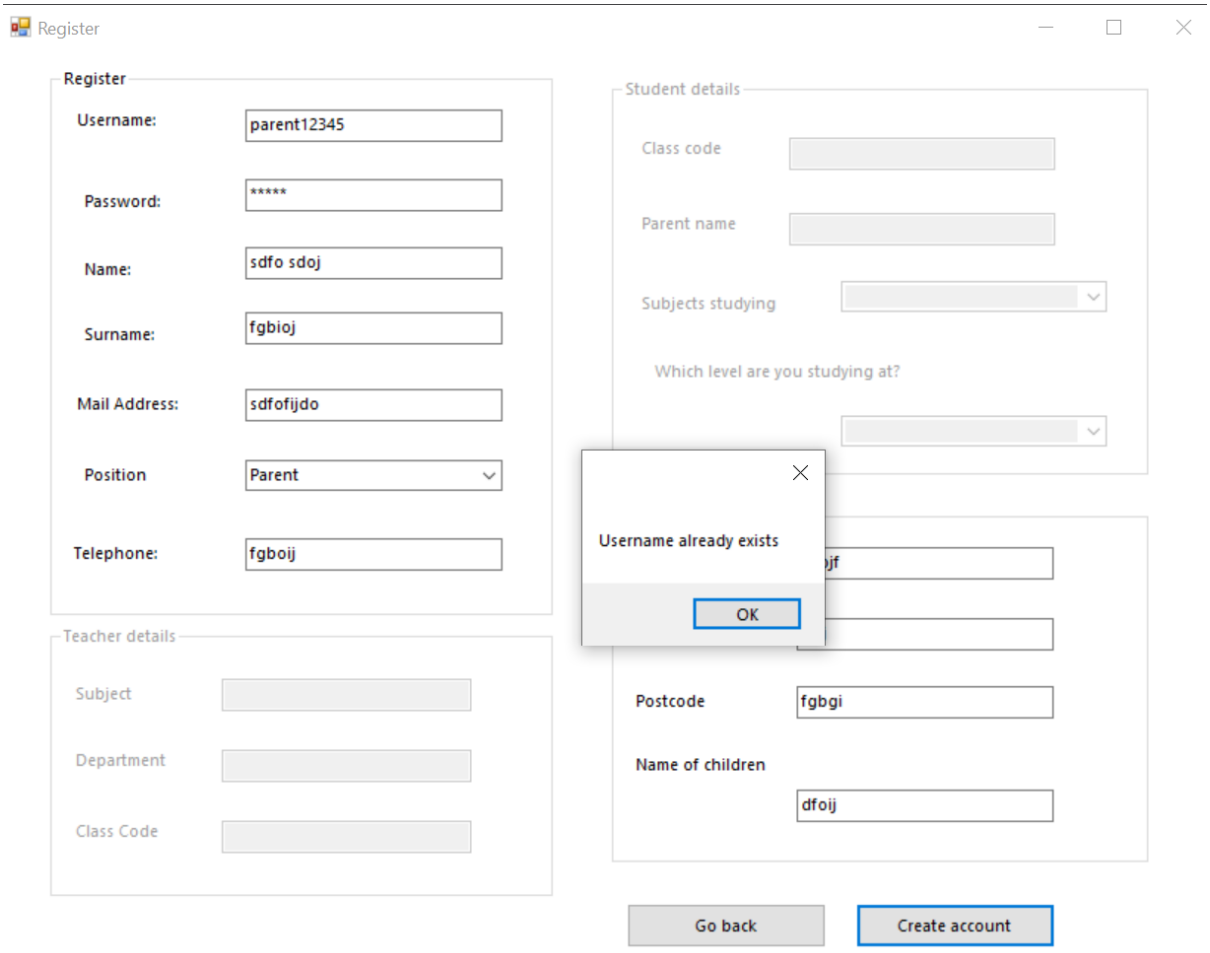
Test screenshot 7:


Register
Register
Username:
Password:
Name:
Surname:
Mail Address:
Position
Student
Telephone:
Teacher details
Subject
Department
Class Code
Student details
Class code
Parent name
Subjects studying
Which level are you studying at?
Parent details
Address
StudentID
Postcode
Name of children
Go back
Create account

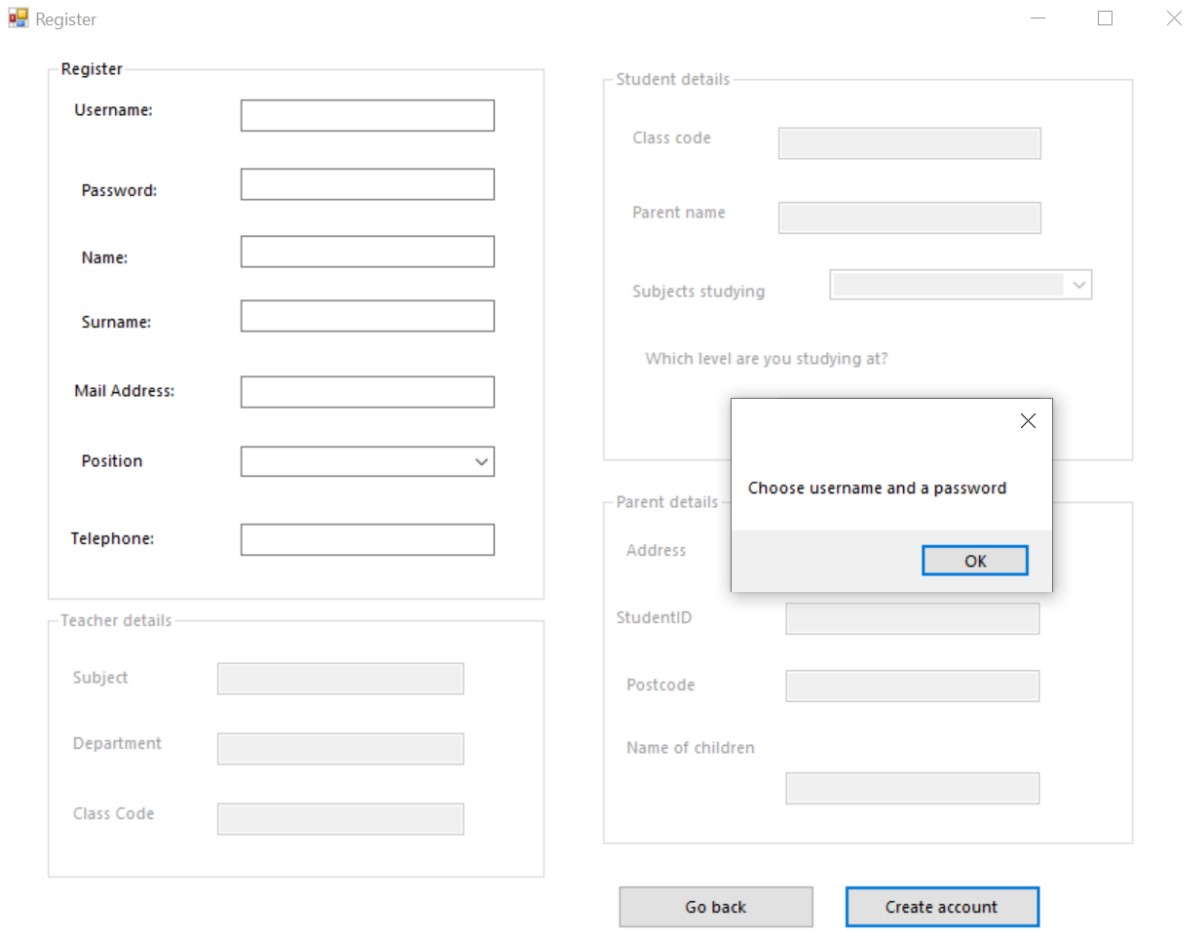

Register
Register
Username:
Password:
Name:
Surname:
Mail Address:
Position
Parent
Telephone:
Teacher details
Subject
Department
Class Code
Student details
Class code
Parent name
Subjects studying
Which level are you studying at?
Parent details
Address
StudentID
Postcode
Name of children
Go back
Create account



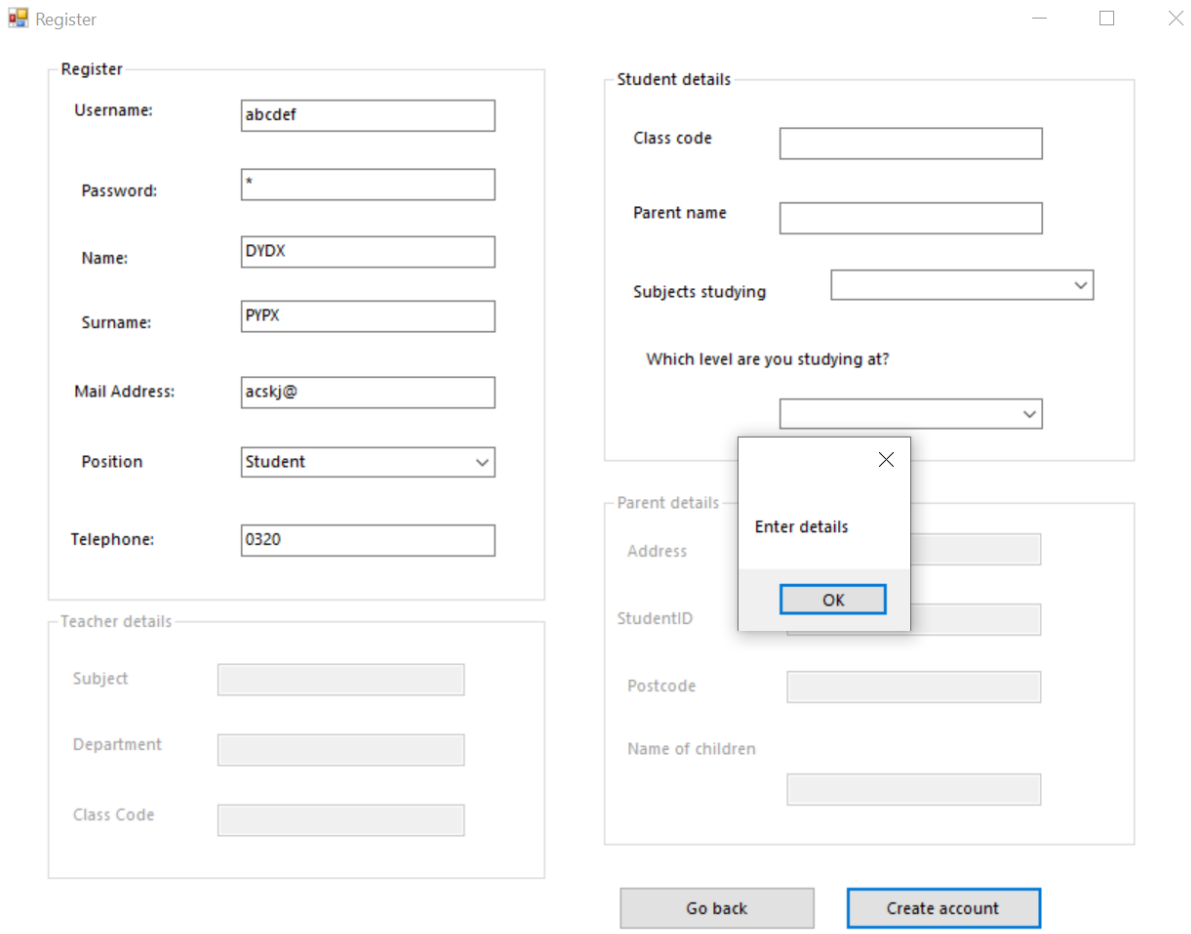
Test screenshot 8:

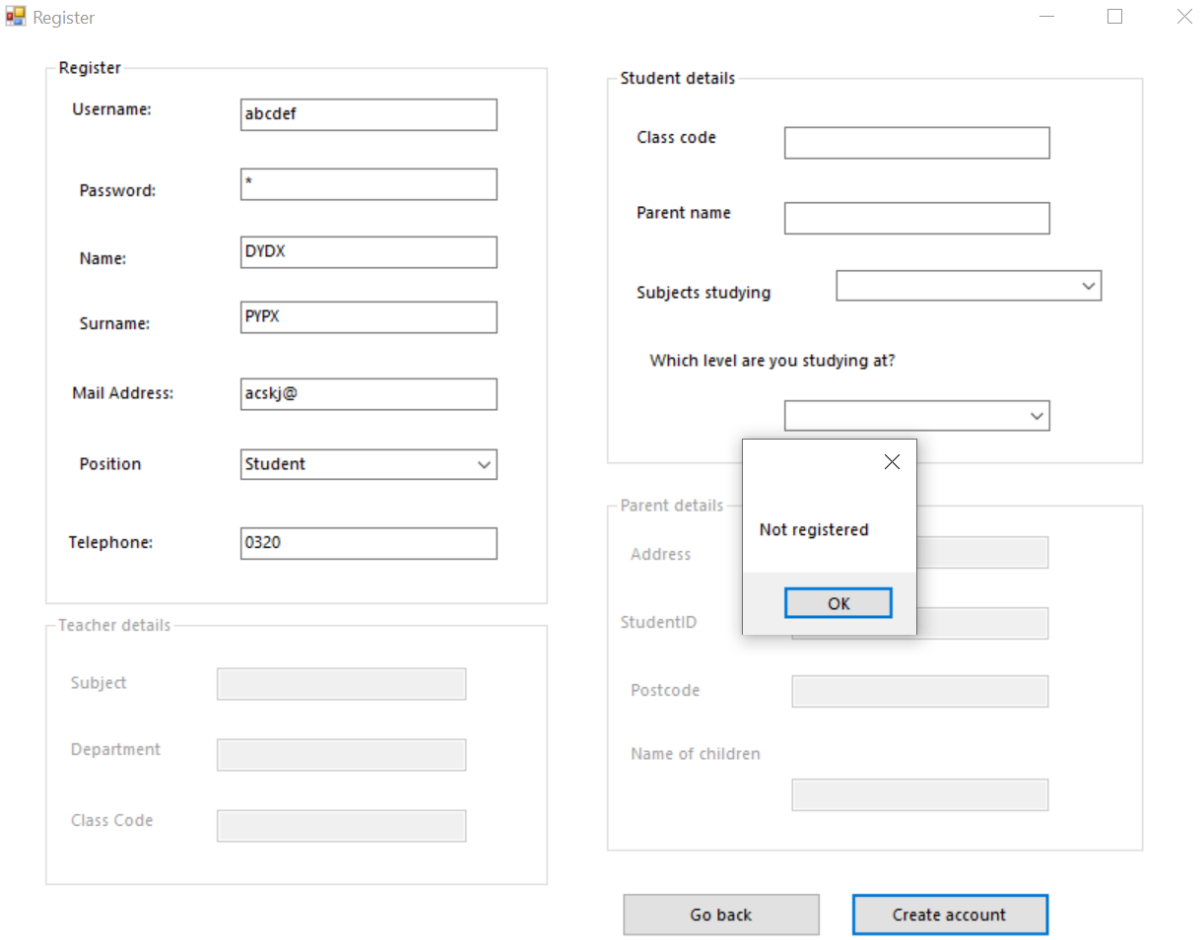


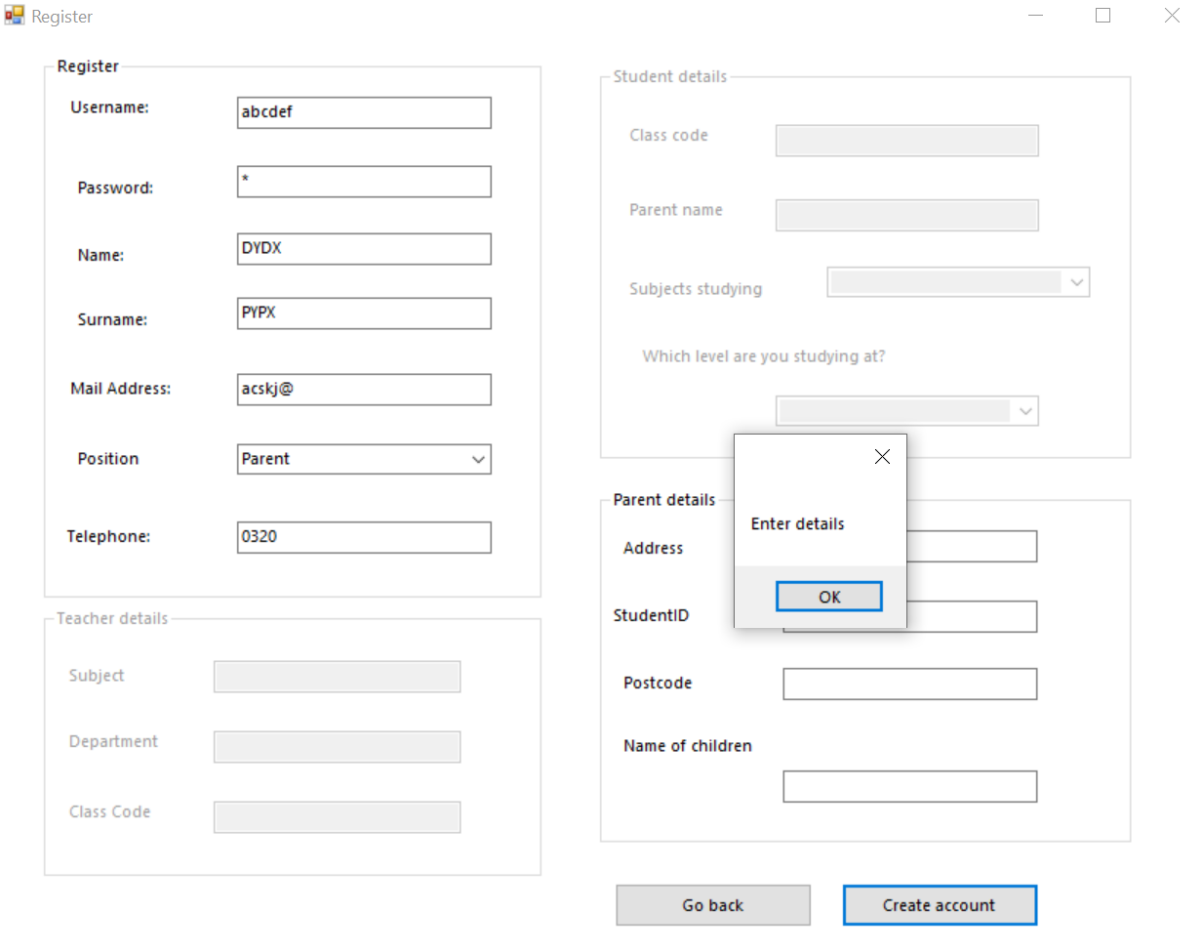
Test screenshot 9:

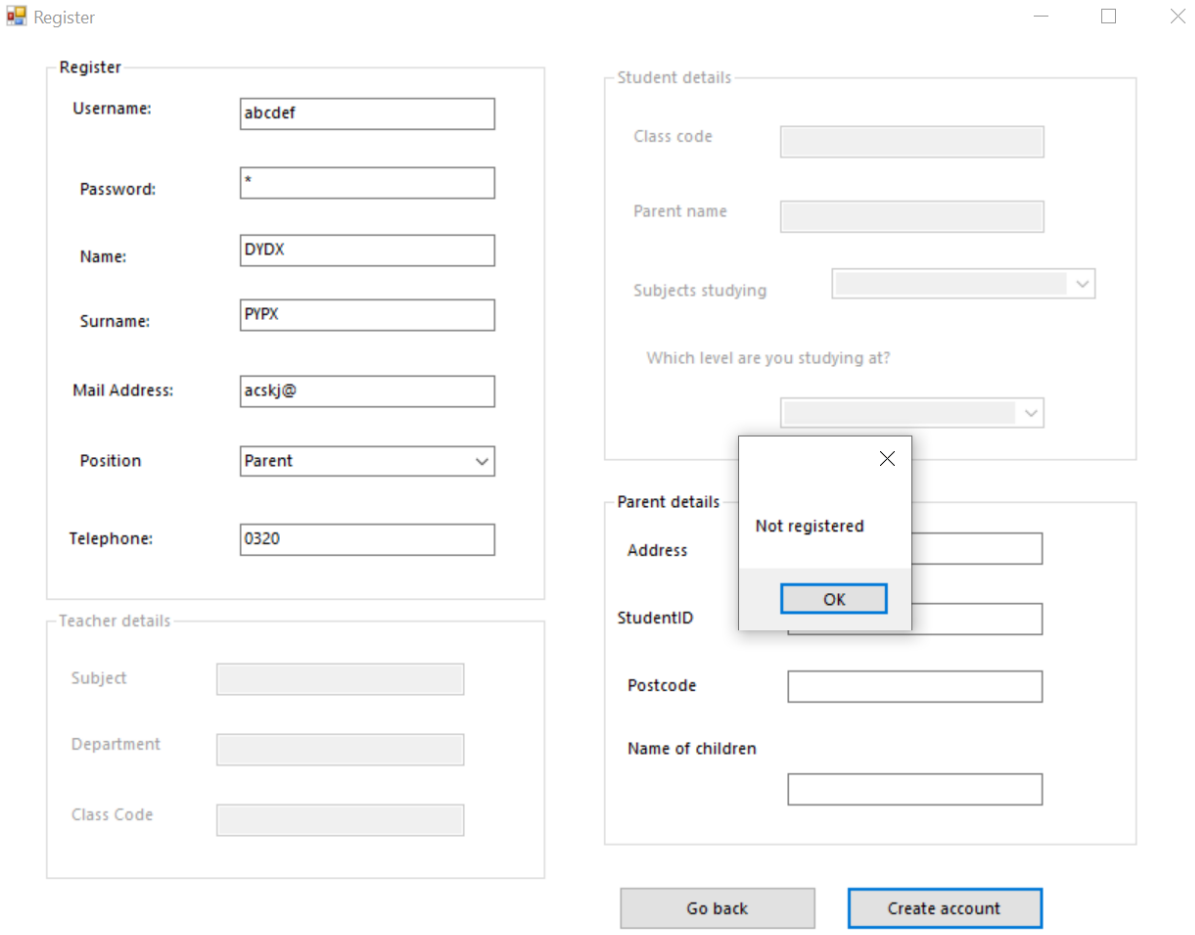


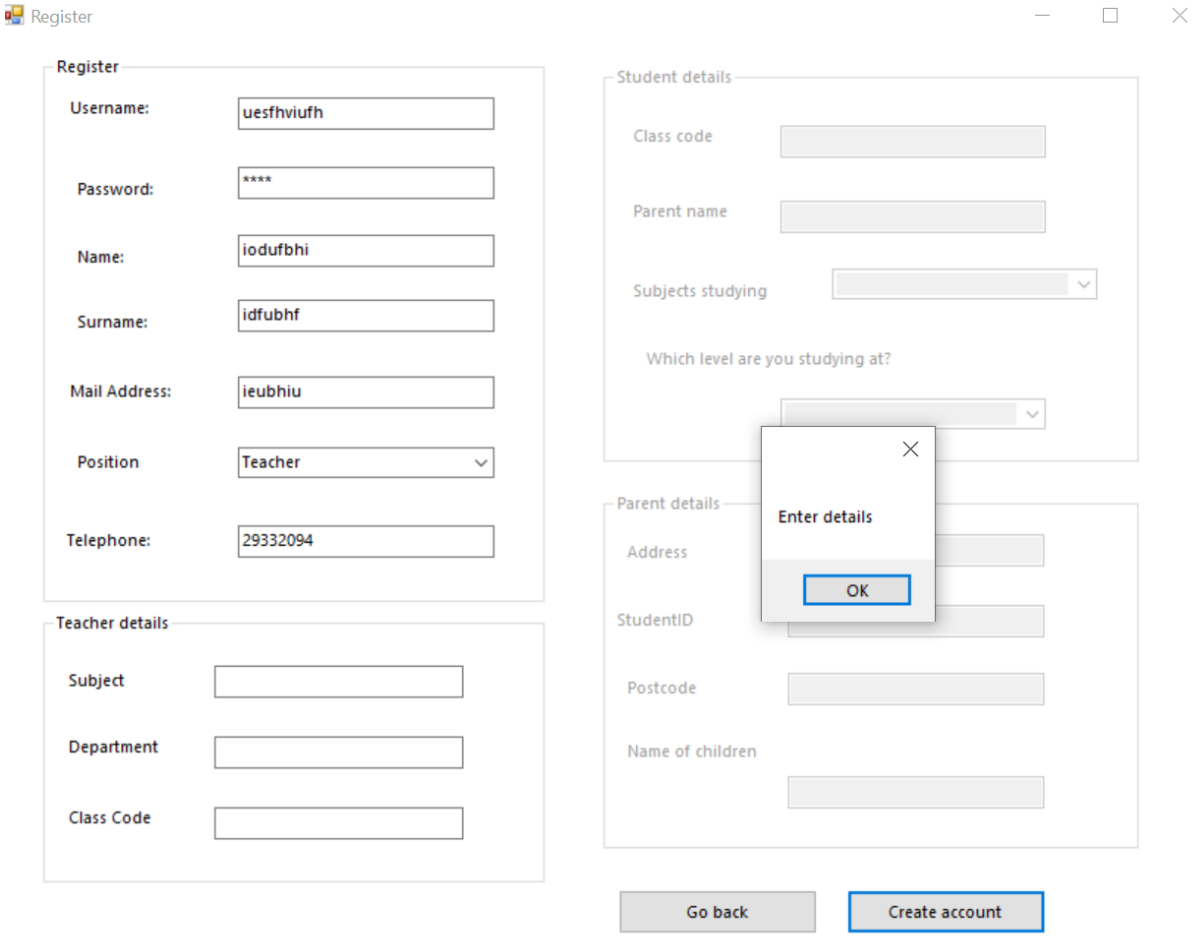
Test screenshot 10:

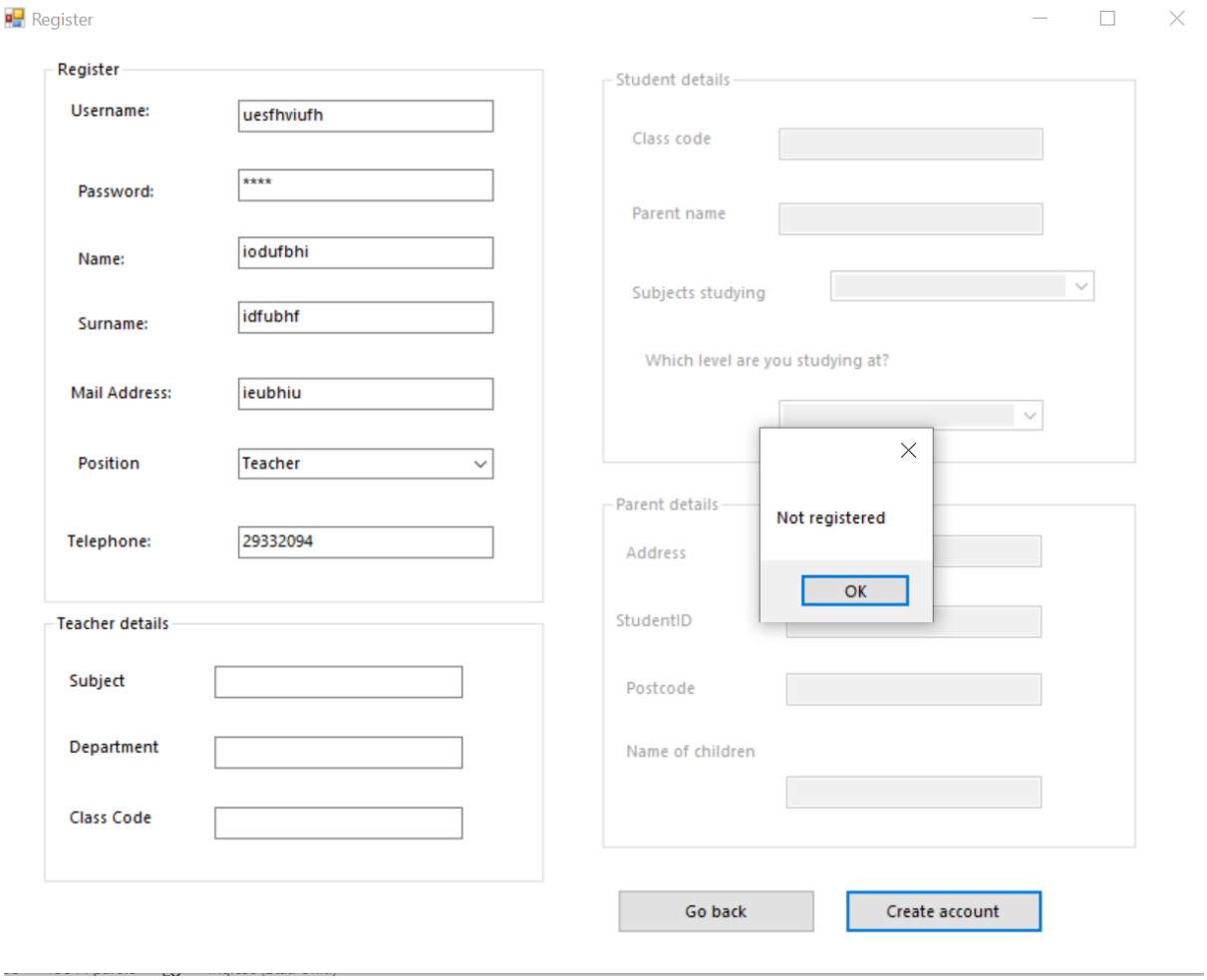

Register
Register
Username:
abcdef
Password:
*
Name:
DYDX
Surname:
PYPX
Mail Address:
acskj@
Position
Student
Telephone:
0320
Teacher details
Subject
Department
Class Code
Student details
Class code
Parent name
Subjects studying
Which level are you studying at?
Enter details
OK
Parent details
Address
StudentID
Postcode
Name of children
Go back
Create account


Register
Register
Username:
abcdef
Password:
*
Name:
DYDX
Surname:
PYPX
Mail Address:
acskj@
Position
Student
Telephone:
0320
Teacher details
Subject
Department
Class Code
Student details
Class code
Parent name
Subjects studying
Which level are you studying at?
Parent details
Address
StudentID
Postcode
Name of children
Not registered
OK
Go back
Create account

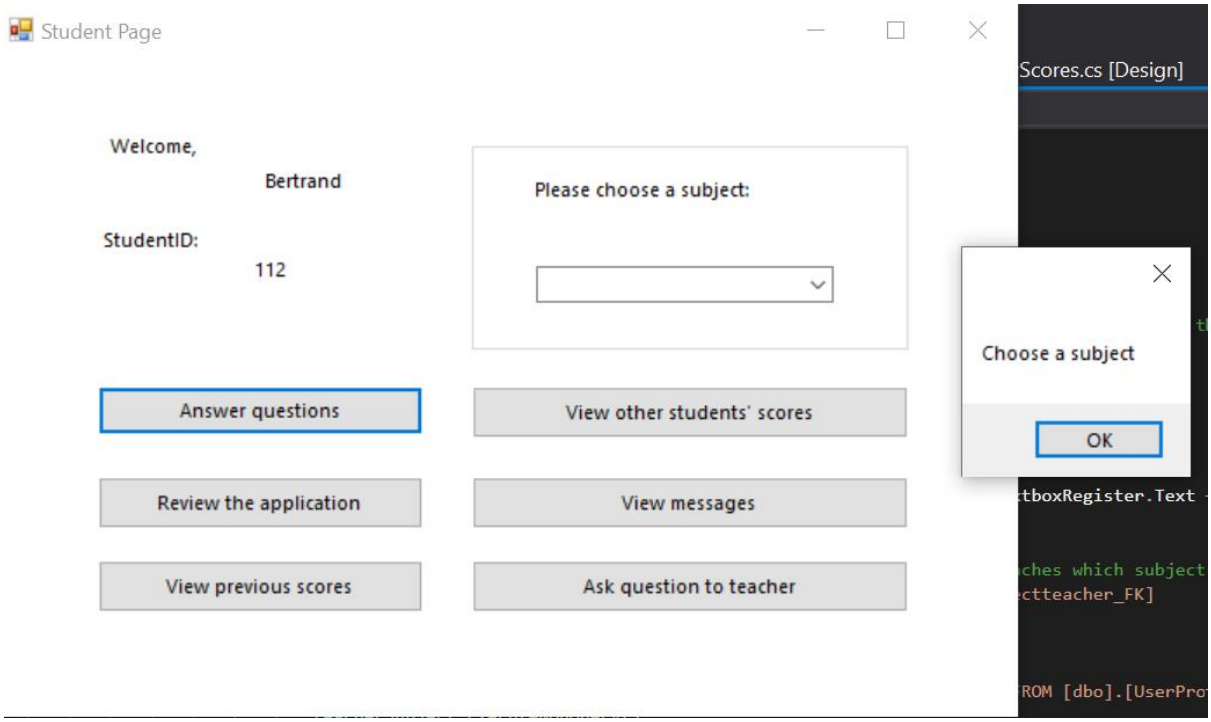

Register
Register
Username:
abcdef
Password:
*
Name:
DYDX
Surname:
PYPX
Mail Address:
acskj@
Position
Parent
Telephone:
0320
Teacher details
Subject
Department
Class Code
Student details
Class code
Parent name
Subjects studying
Which level are you studying at?
Parent details
Address
StudentID
Postcode
Name of children
Enter details
OK
Go back
Create account

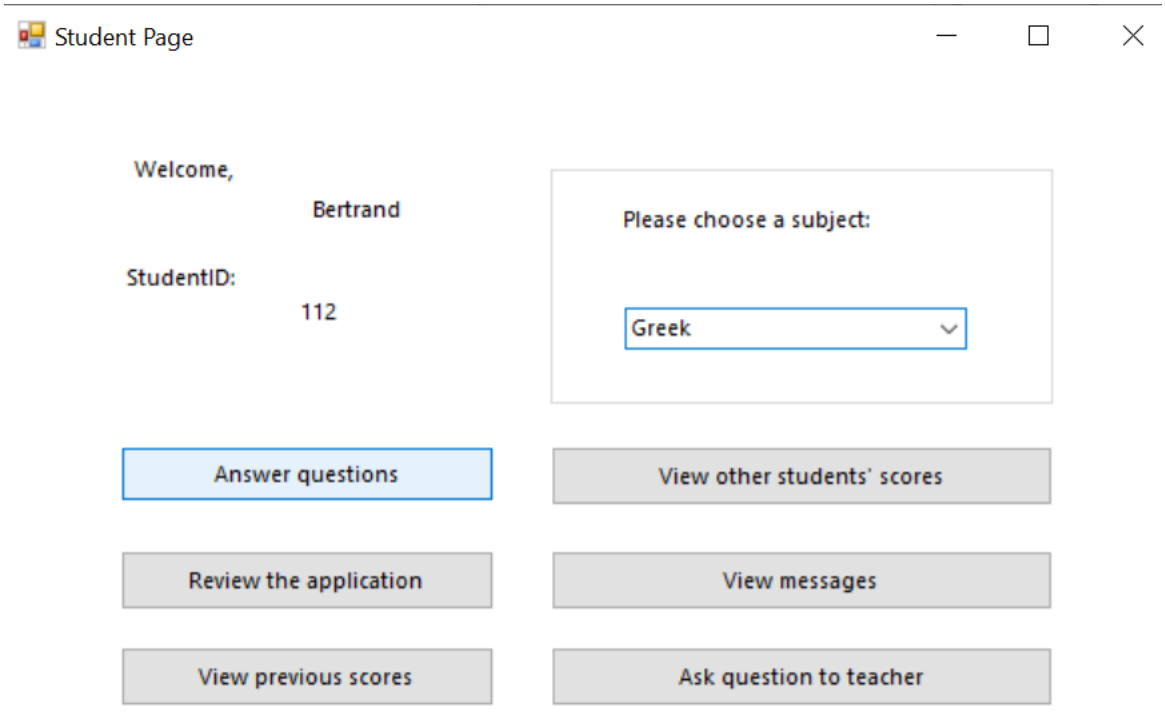

Register
Register
Username:
abcdef
Password:
*
Name:
DYDX
Surname:
PYPX
Mail Address:
acskj@
Position
Parent
Telephone:
0320
Teacher details
Subject
Department
Class Code
Student details
Class code
Parent name
Subjects studying
Which level are you studying at?
Parent details
Address
StudentID
Postcode
Name of children
Not registered
OK
Go back
Create account

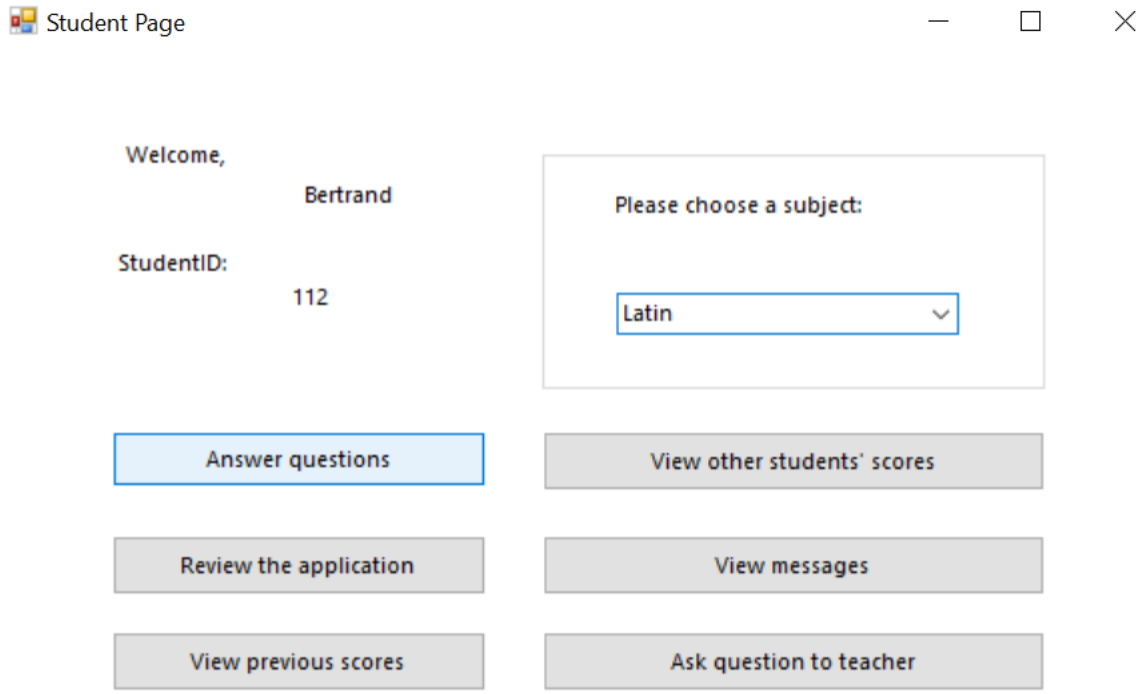

Register
Register
Username:
uesfhviufh
Password:
****
Name:
iodufbhi
Surname:
idfubhf
Mail Address:
ieubhiu
Position
Teacher
Telephone:
29332094
Teacher details
Subject
Department
Class Code
Student details
Class code
Parent name
Subjects studying
Which level are you studying at?
Parent details
Address
StudentID
Postcode
Name of children
Enter details
OK
Go back
Create account

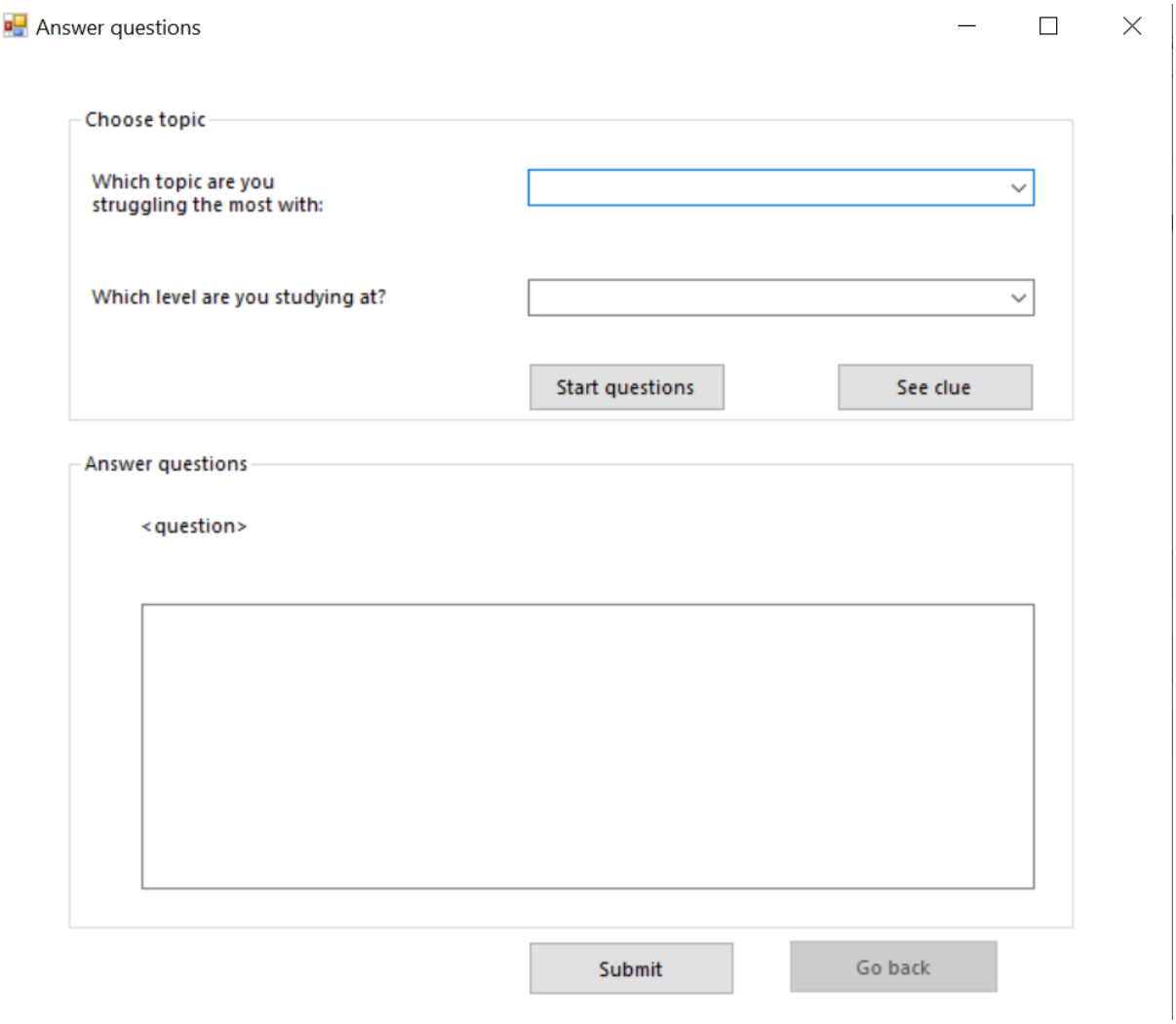


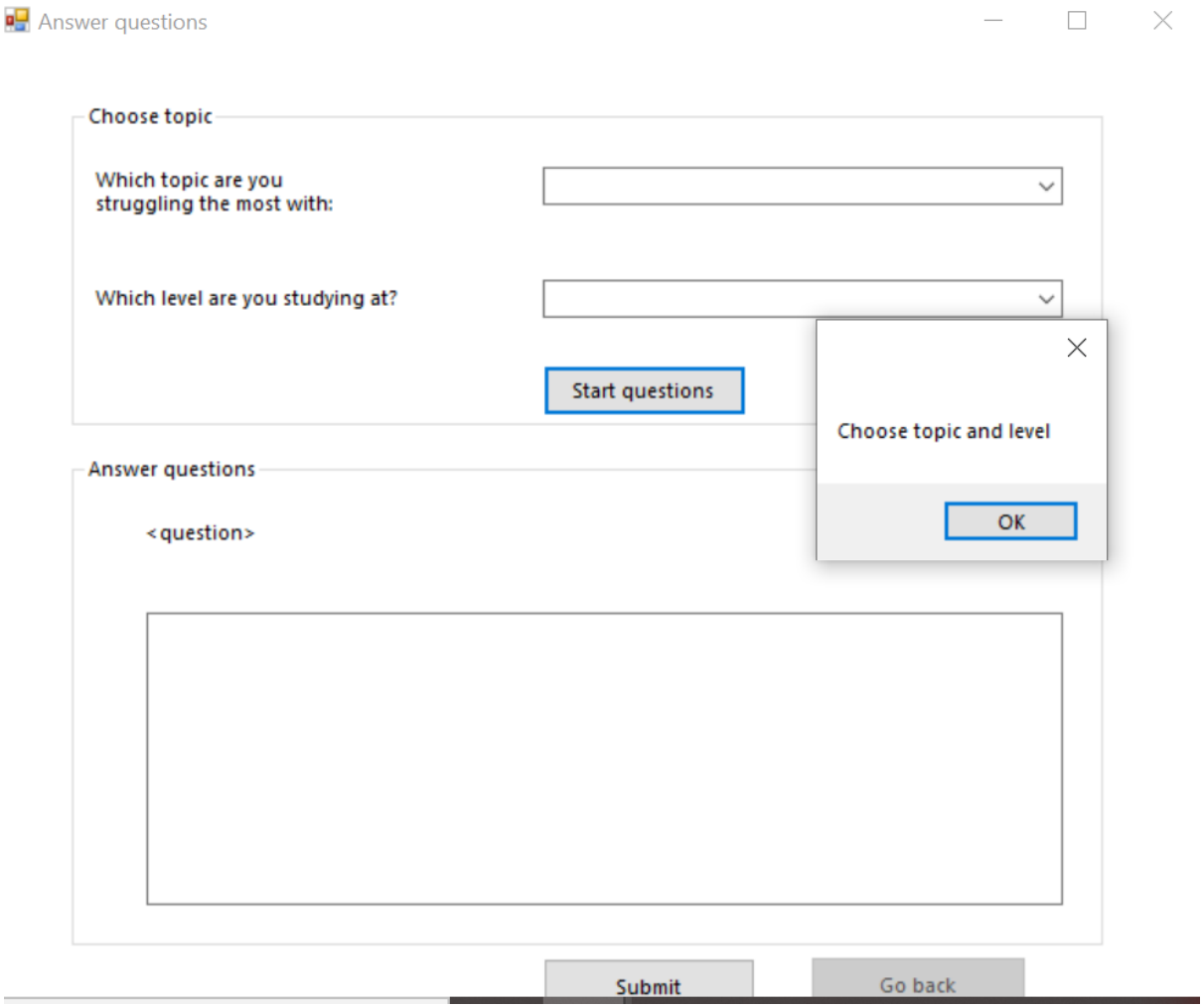
Test screenshot 11:

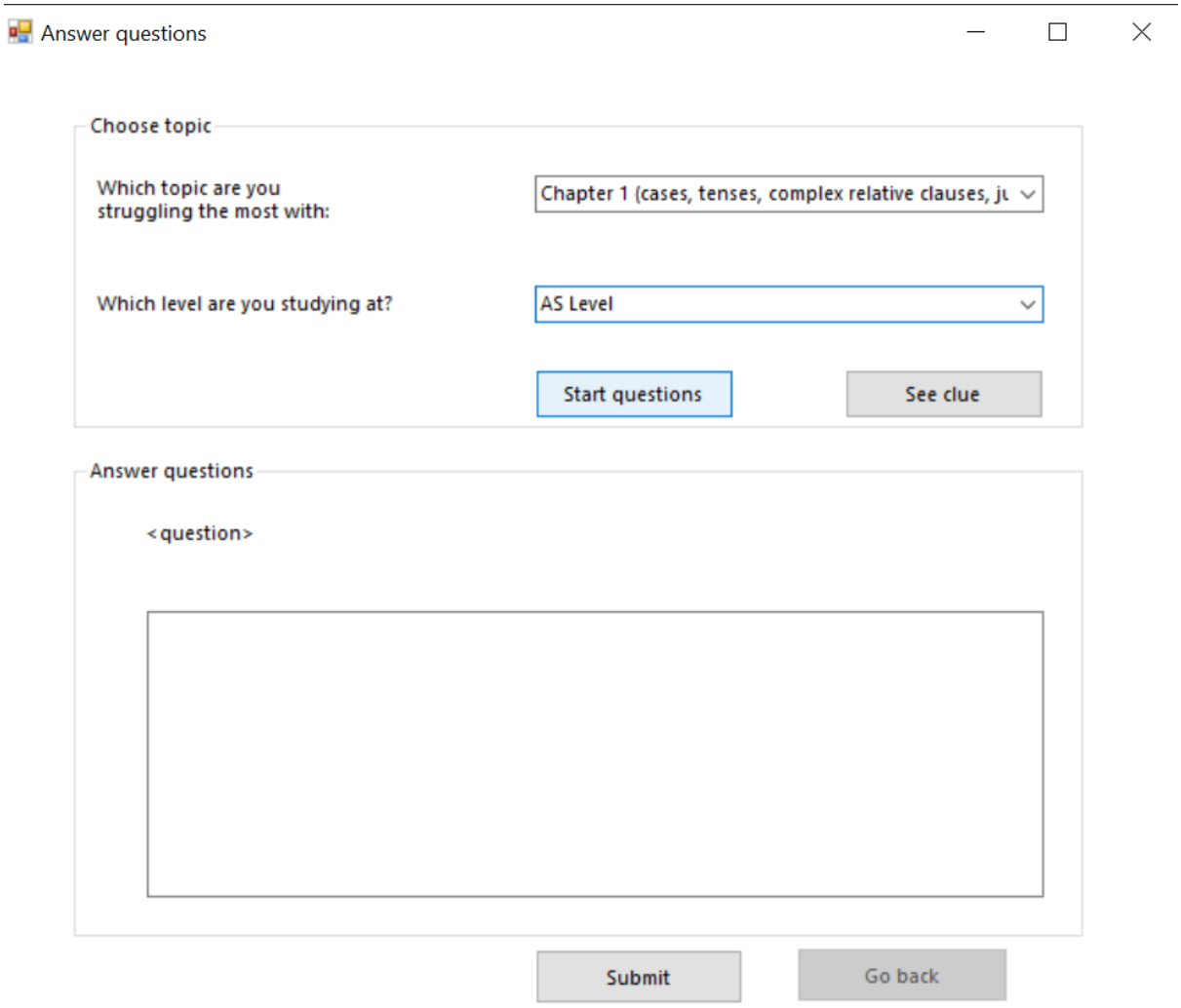

Student Page
Welcome,
Bertrand
StudentID:
112
Please choose a subject:
Answer questions
View other students' scores
Review the application
View messages
View previous scores
Ask question to teacher
Choose a subject
OK
Scores.cs [Design]

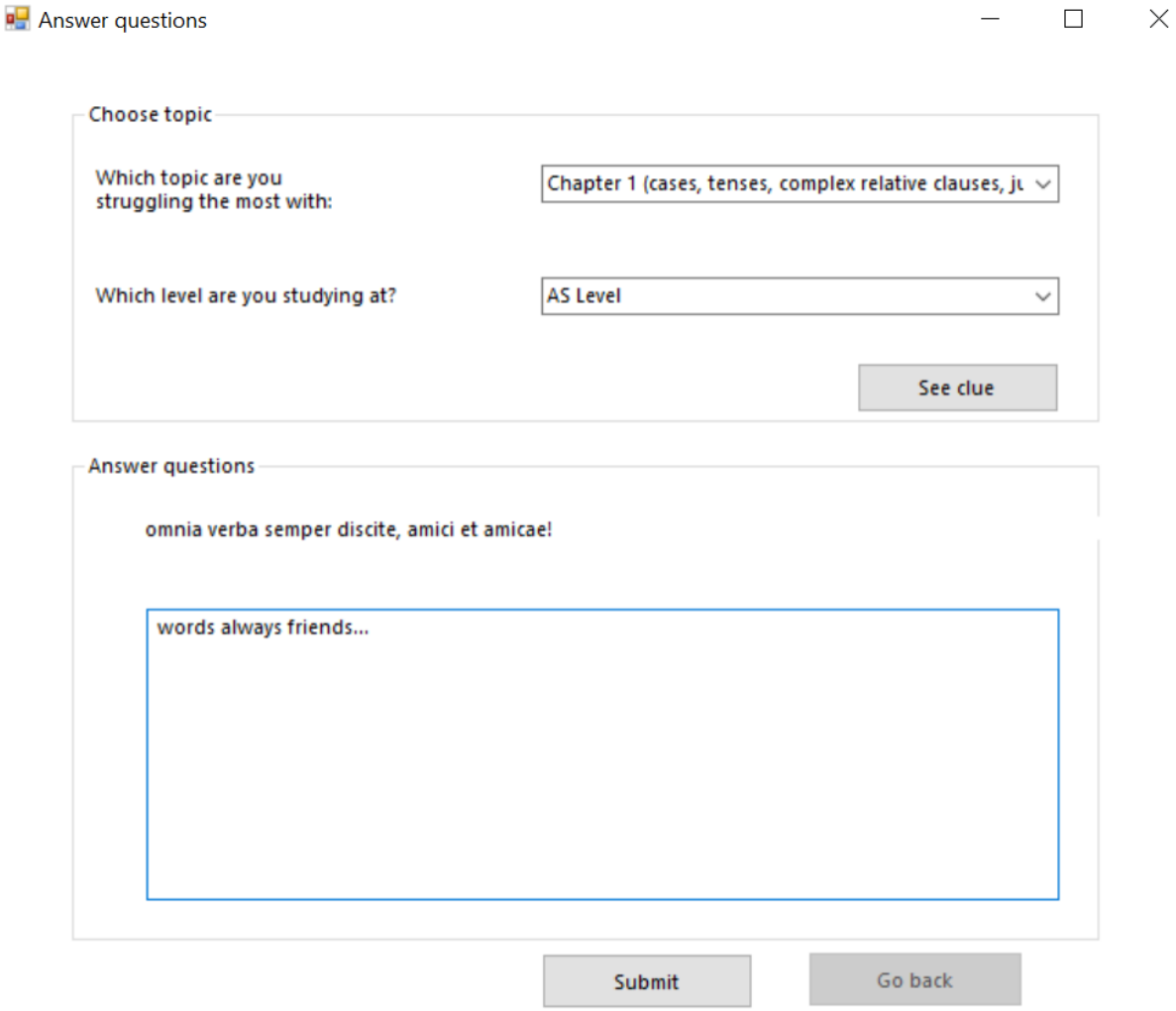

Student Page
Welcome,
Bertrand
StudentID:
112
Please choose a subject:
Greek
Answer questions
View other students' scores
Review the application
View messages
View previous scores
Ask question to teacher

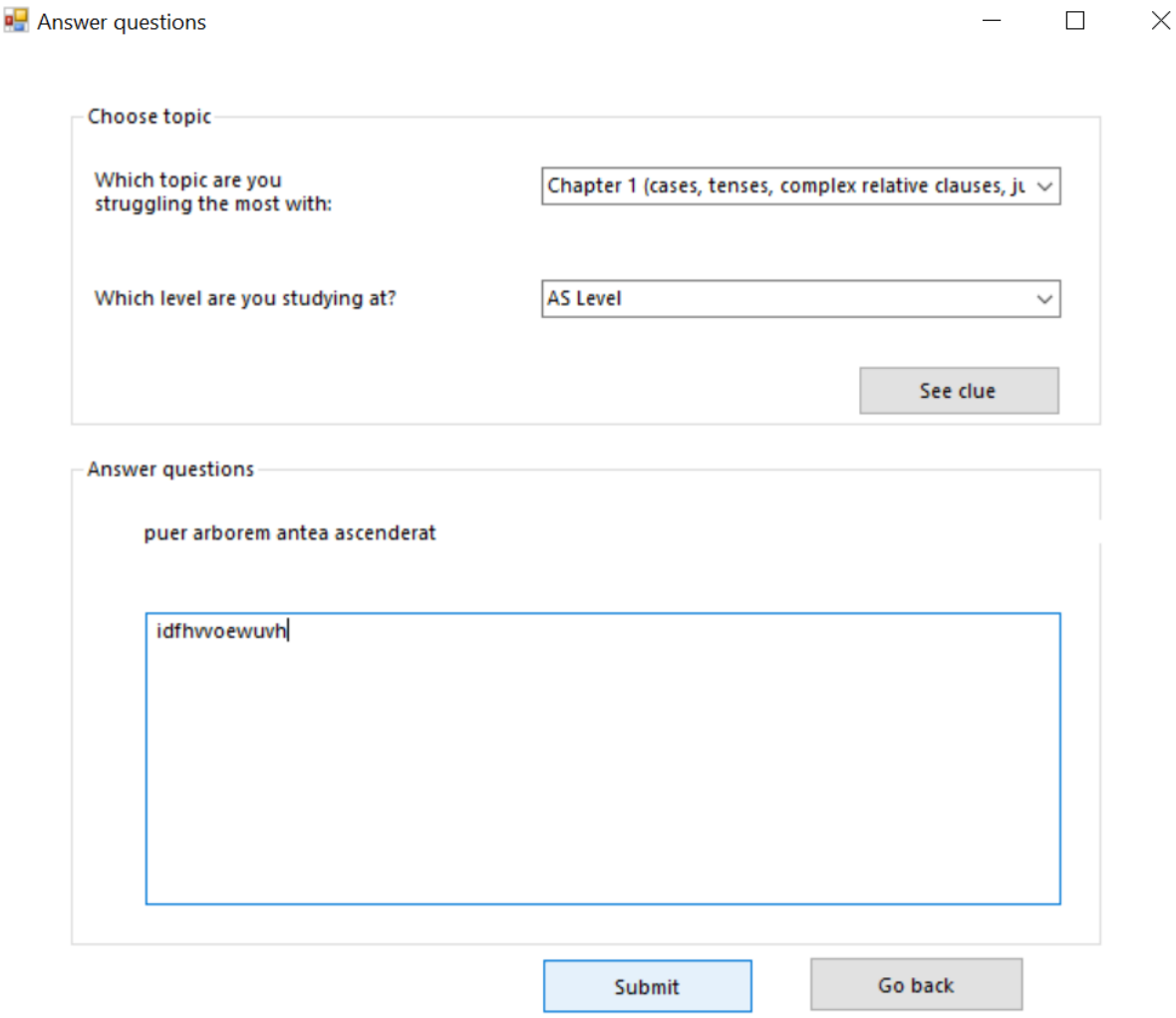

Student Page
Welcome,
Bertrand
StudentID:
112
Please choose a subject:
Latin
Answer questions
View other students' scores
Review the application
View messages
View previous scores
Ask question to teacher

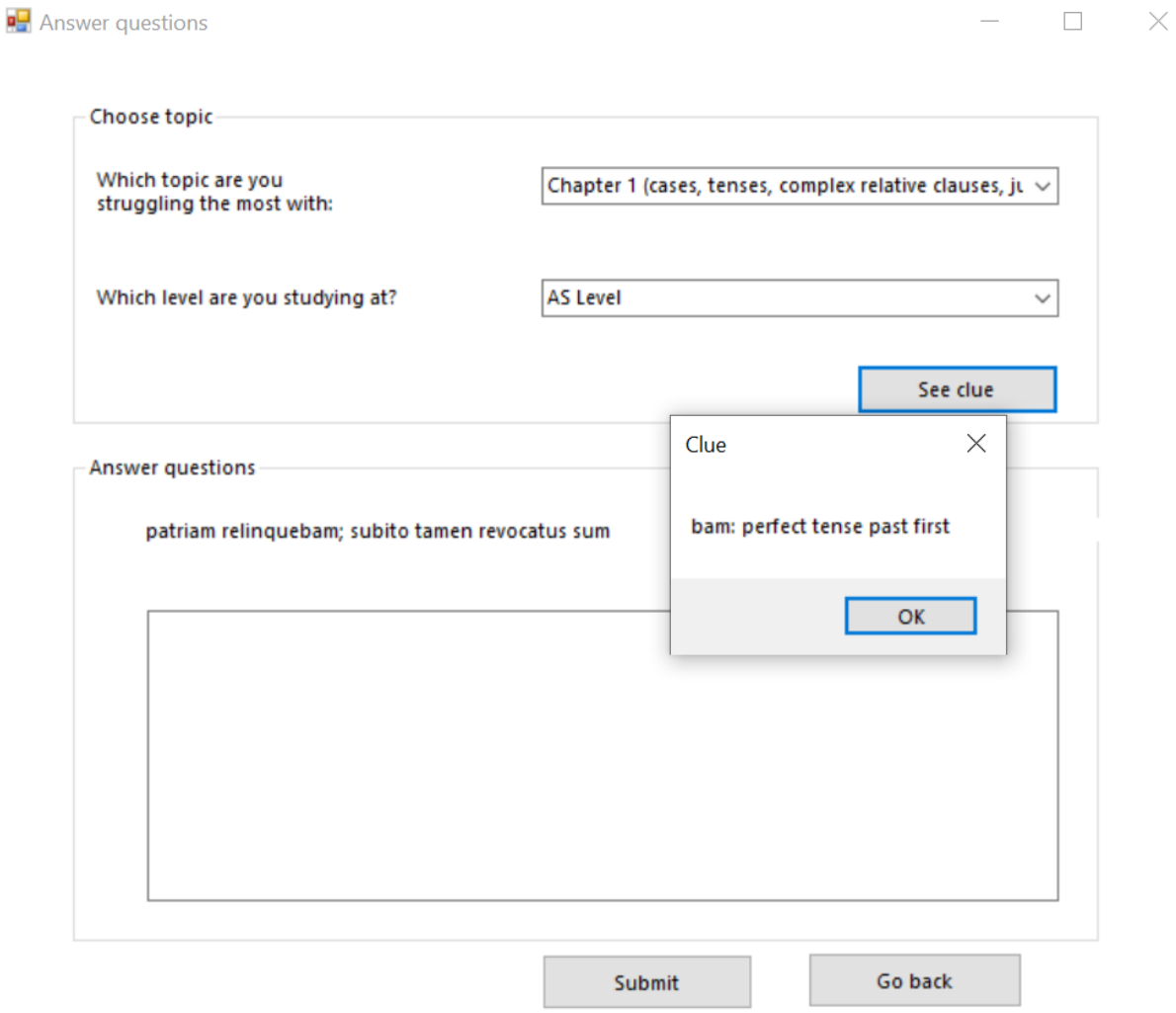

Answer questions
Choose topic
Which topic are you struggling the most with:
Which level are you studying at?
Start questions
See clue
Answer questions
<question>
Submit
Go back

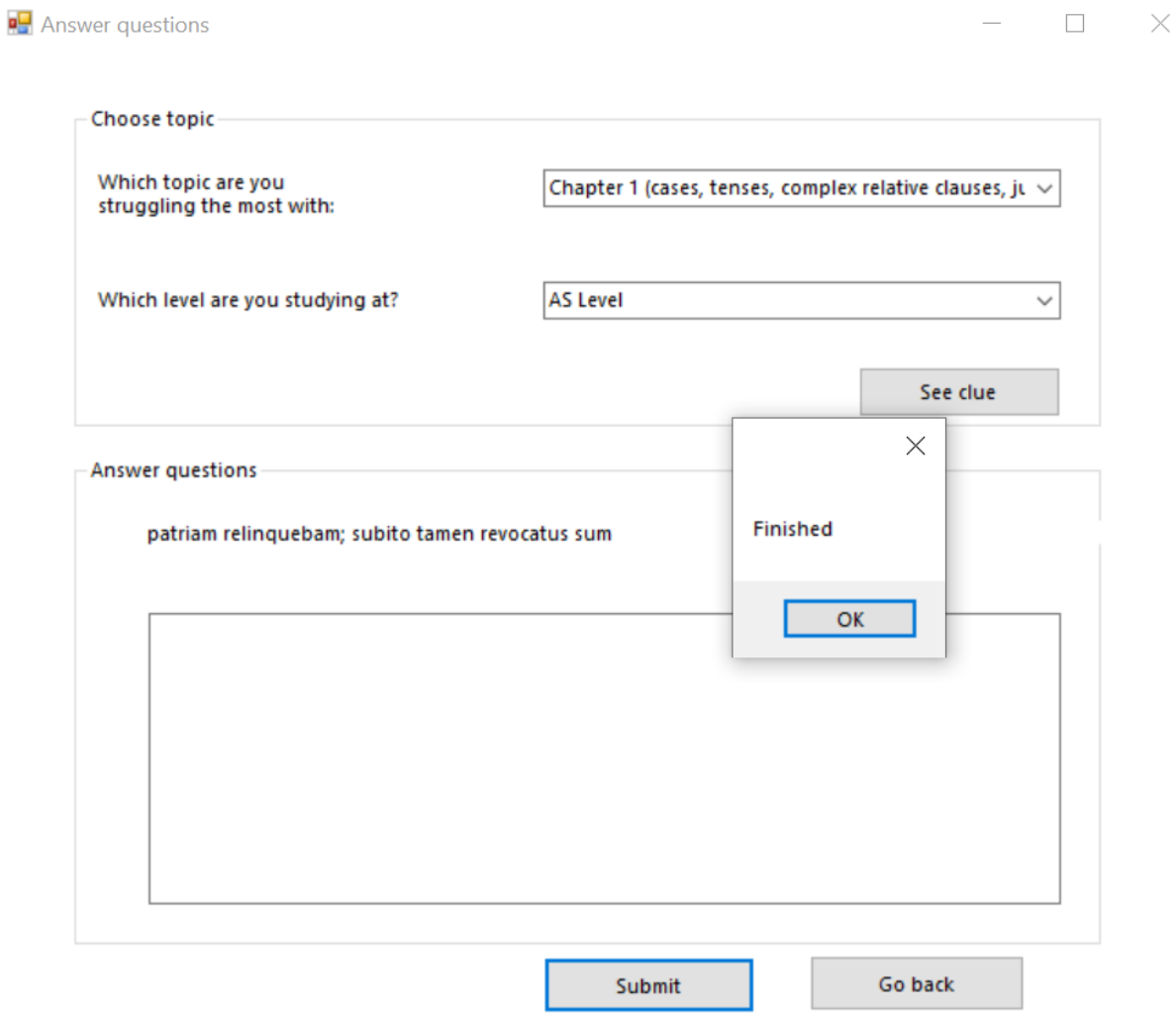

Answer questions
Choose topic
Which topic are you struggling the most with:
Which level are you studying at?
Start questions
Choose topic and level
OK
Answer questions
<question>
Submit
Go back

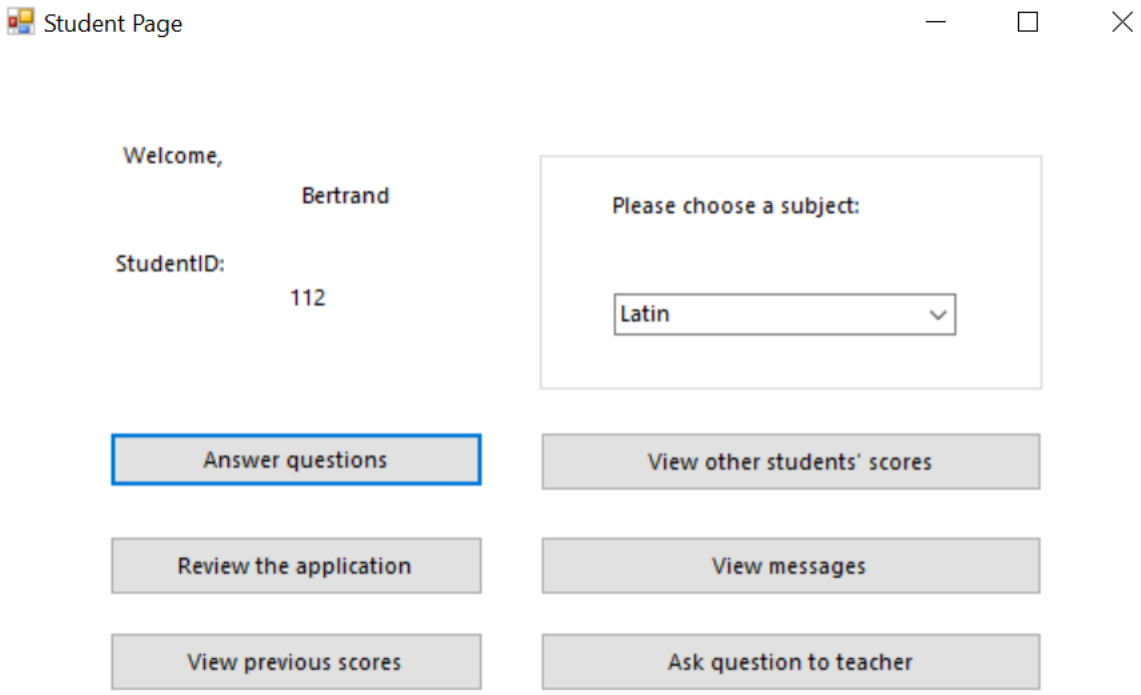

Answer questions
Choose topic
Which topic are you struggling the most with:
Chapter 1 (cases, tenses, complex relative clauses, ju
Which level are you studying at?
AS Level
Start questions
See clue
Answer questions
<question>
Submit
Go back

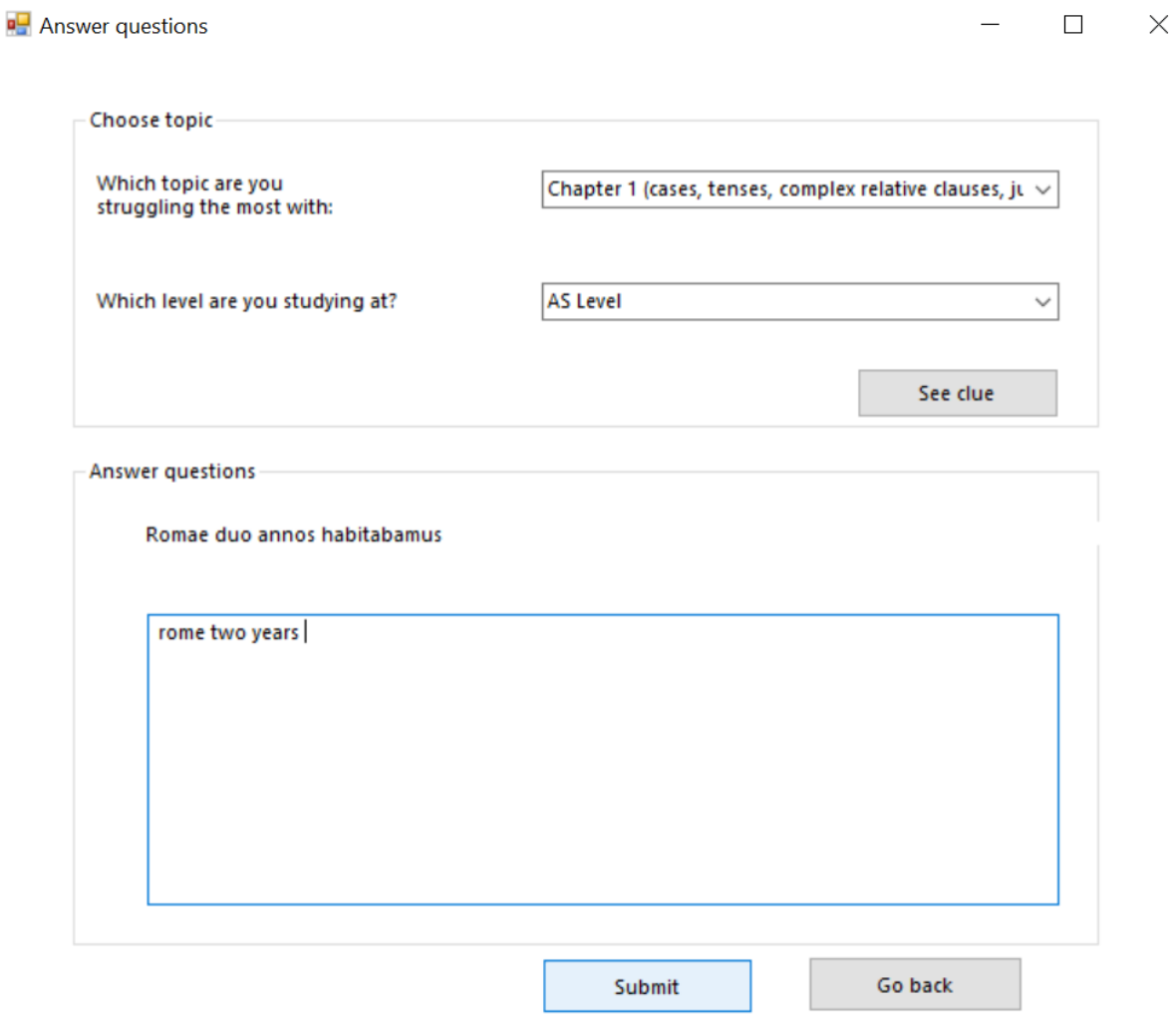

Answer questions
Choose topic
Which topic are you struggling the most with:
Chapter 1 (cases, tenses, complex relative clauses, ju
Which level are you studying at?
AS Level
See clue
Answer questions
omnia verba semper discite, amici et amicae!
words always friends...
Submit
Go back


Answer questions
Choose topic
Which topic are you struggling the most with:
Chapter 1 (cases, tenses, complex relative clauses, ju
Which level are you studying at?
AS Level
See clue
Answer questions
puer arborem antea ascenderat
idfhvvoewuvh
Submit
Go back

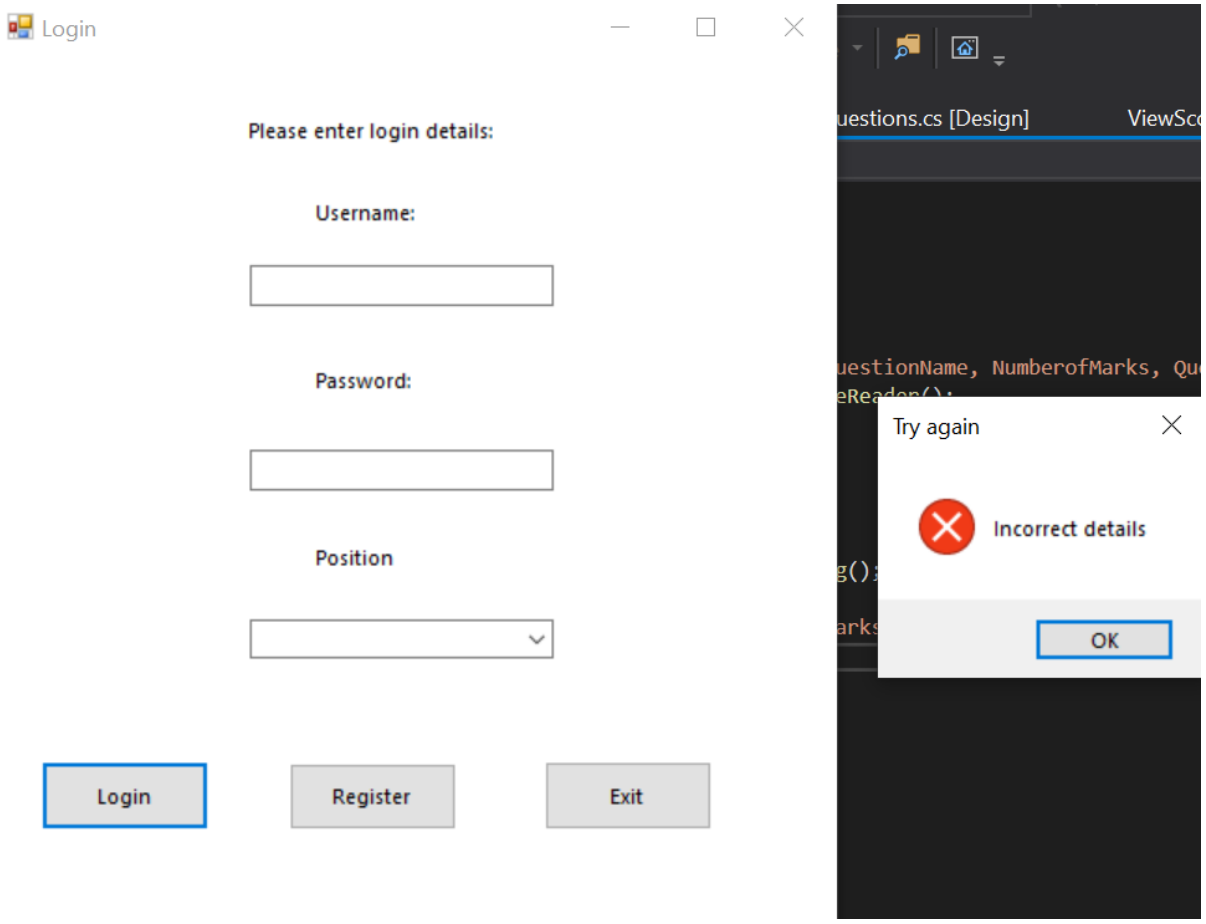

Answer questions
Choose topic
Which topic are you struggling the most with:
Chapter 1 (cases, tenses, complex relative clauses, ju
Which level are you studying at?
AS Level
See clue
Clue
bam: perfect tense past first
OK
Answer questions
patriam relinquebam; subito tamen revocatus sum
Submit
Go back

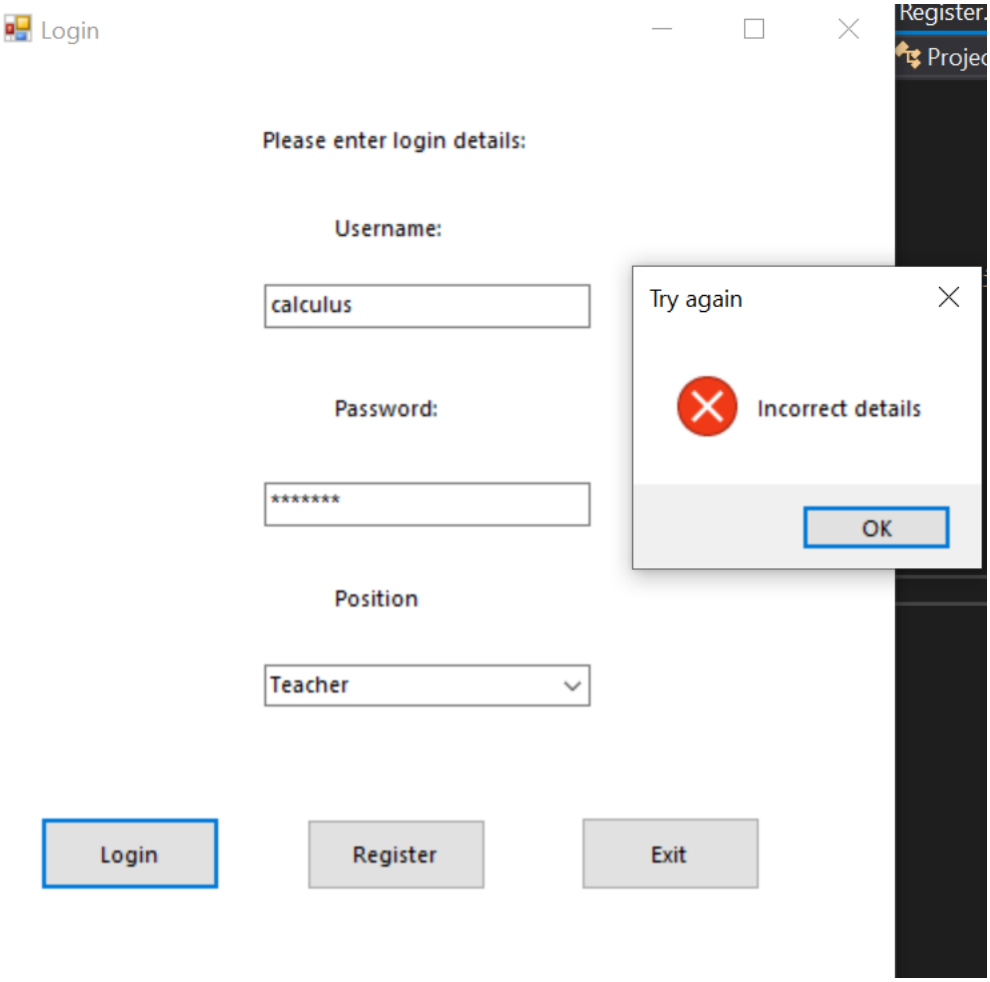

Answer questions
Choose topic
Which topic are you struggling the most with:
Chapter 1 (cases, tenses, complex relative clauses, ju
Which level are you studying at?
AS Level
See clue
Answer questions
patriam relinquebam; subito tamen revocatus sum
Finished
OK
Submit
Go back

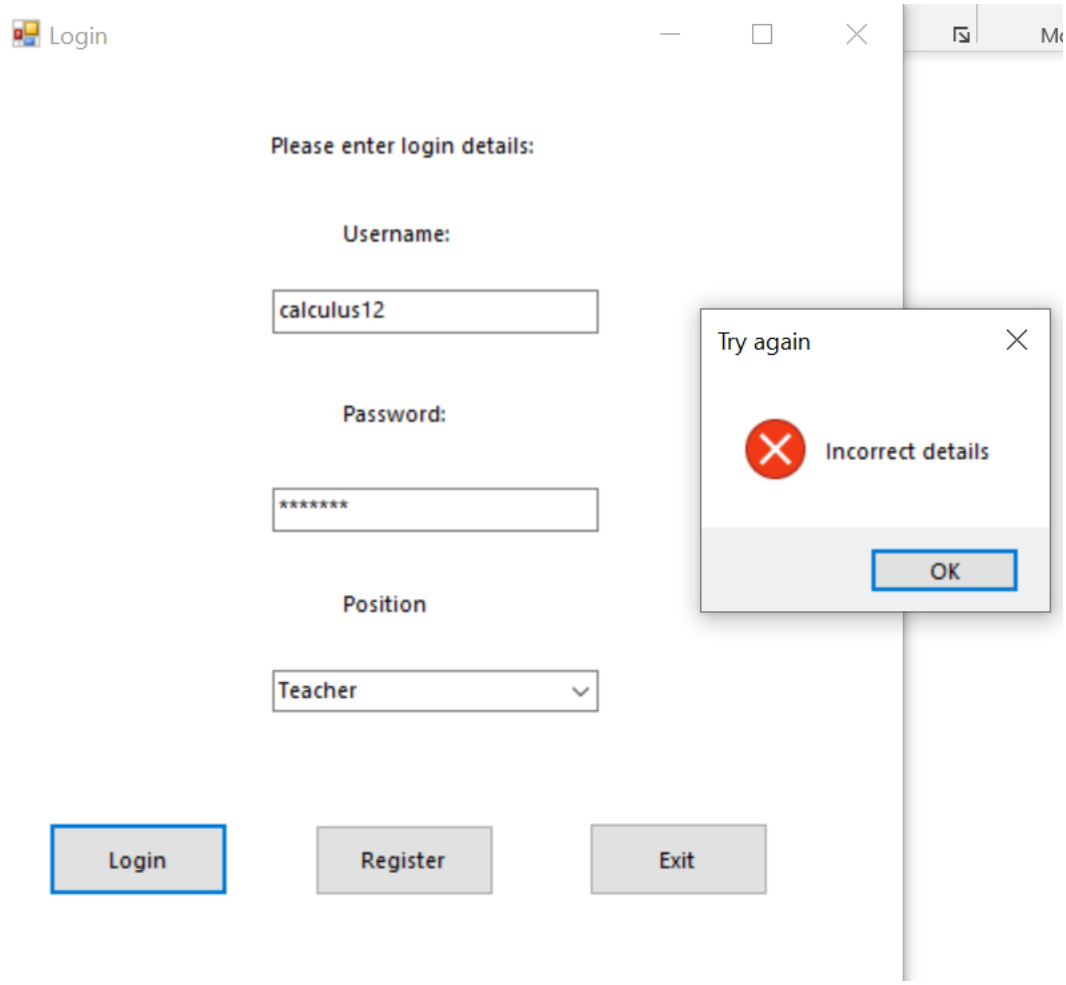

Student Page
Welcome,
Bertrand
StudentID:
112
Please choose a subject:
Latin
Answer questions
View other students' scores
Review the application
View messages
View previous scores
Ask question to teacher



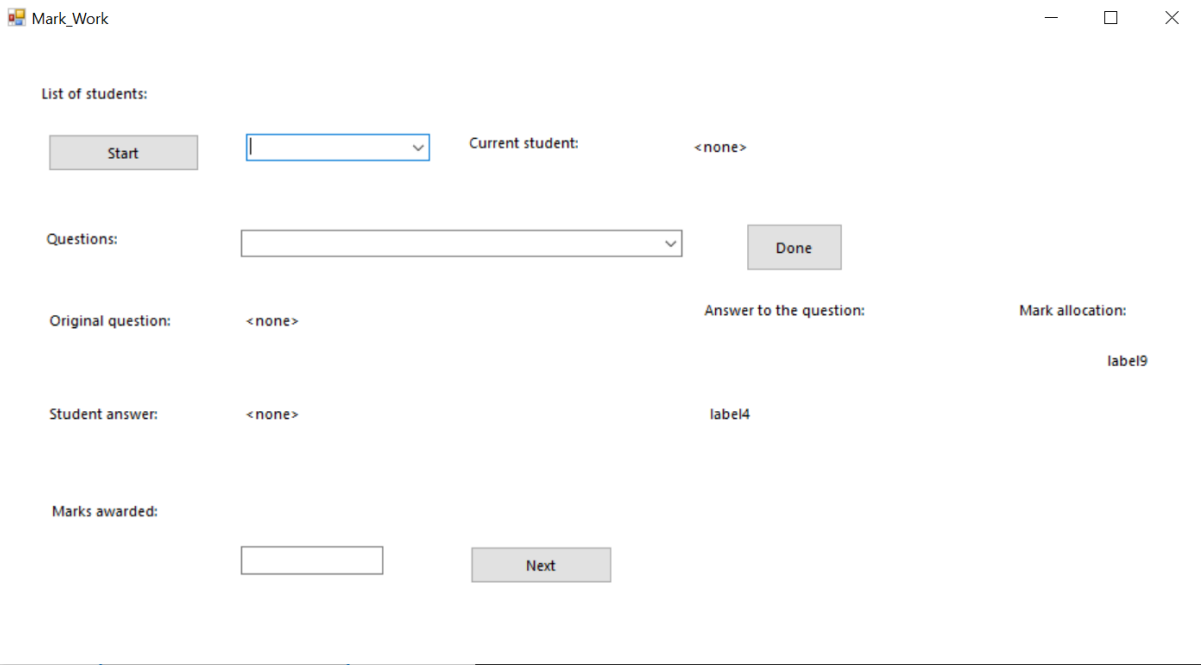
Test screenshot 12:

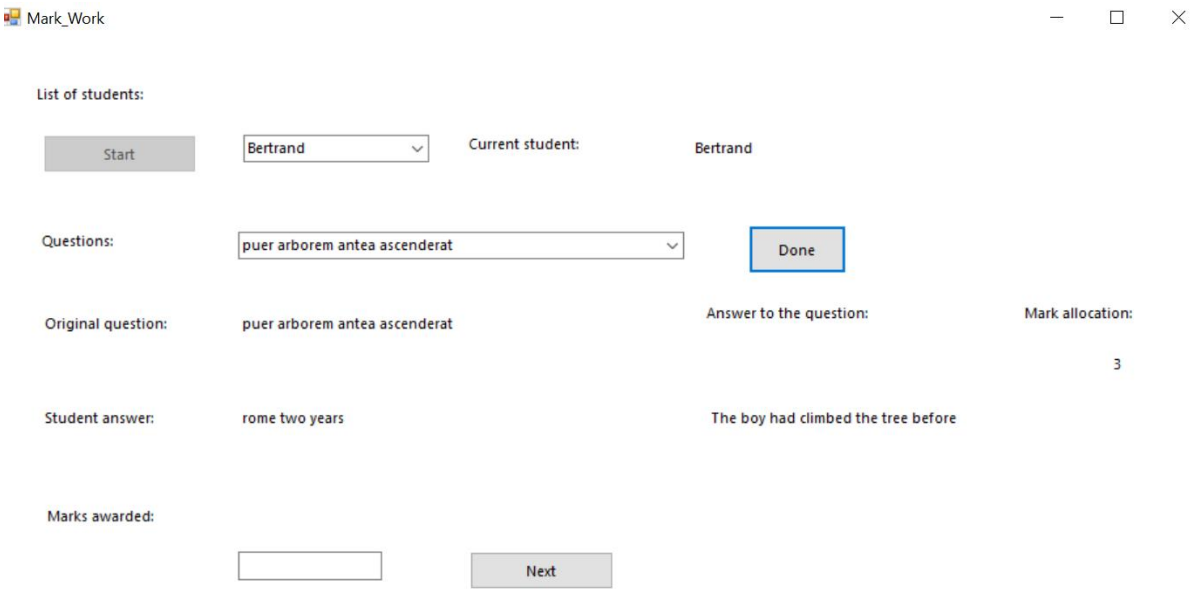

Login
Please enter login details:
Username:
Password:
Position
Login
Register
Exit
uestions.cs [Design]
ViewSc
uestionName, NumberofMarks, Qu
Try again
Incorrect details
OK

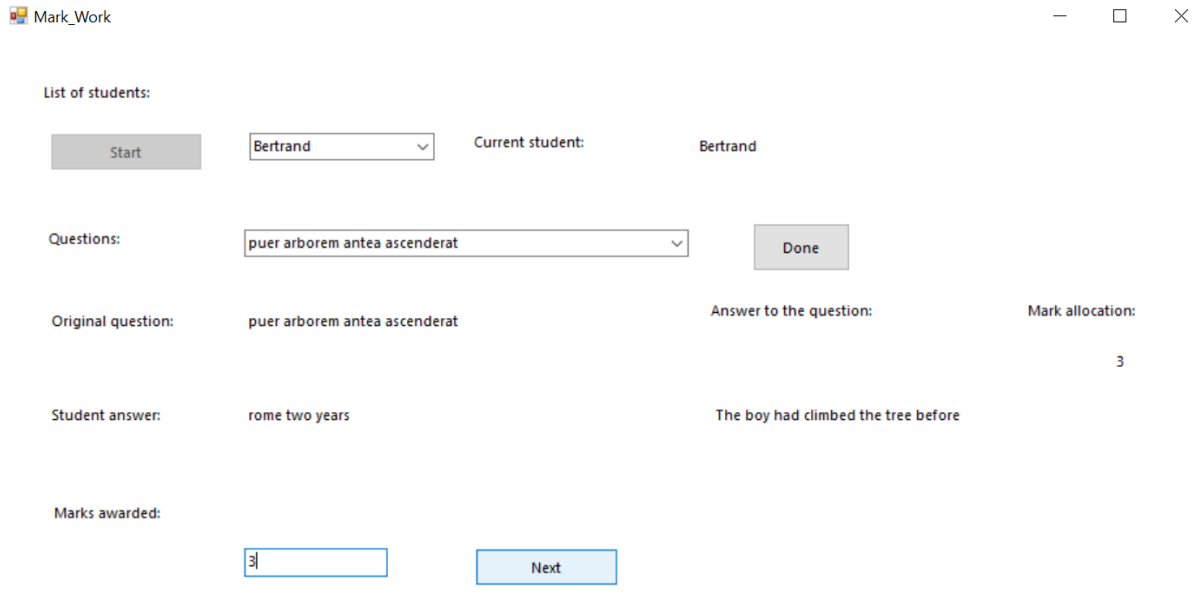

Login
Please enter login details:
Username:
calculus
Password:
****
Position
Teacher
Login
Register
Exit
Try again
Incorrect details
OK



Test screenshot 13:







Proof that these are the answers given by the student (these values are different from the user input above because their purpose is just to show that the answers correspond to the ones given by the students in the class for a given teacher id):

UserProfile:

profileID	Username	Password	Name	Surname	Mail	Telephone	Role
371	456	a	Betrand	abc	wefgerg	drth	Student
372	3543	a	Student1	abcdef	asmsnirog	abd k	Student
373	2343	a	Student2	acdegb	efgneg	dfbd	Student
374	2324	a	Teacher1	adbeefgh	efgf	ab	Teacher

Student:

StudentID	profileID	QuestionsRight	LevelID	ParentName	classCode
112	371	*NULL*	2	*NULL*	15
113	372	*NULL*	2	*NULL*	15
135	373	*NULL*	2	*NULL*	15

Teacher:

TeacherID	profileID	Department	ClassCode
39	374	Classics ...	15

StudentTeacher:

StudentID	TeacherID	SubjectID
112	39	10
113	39	10
135	39	10

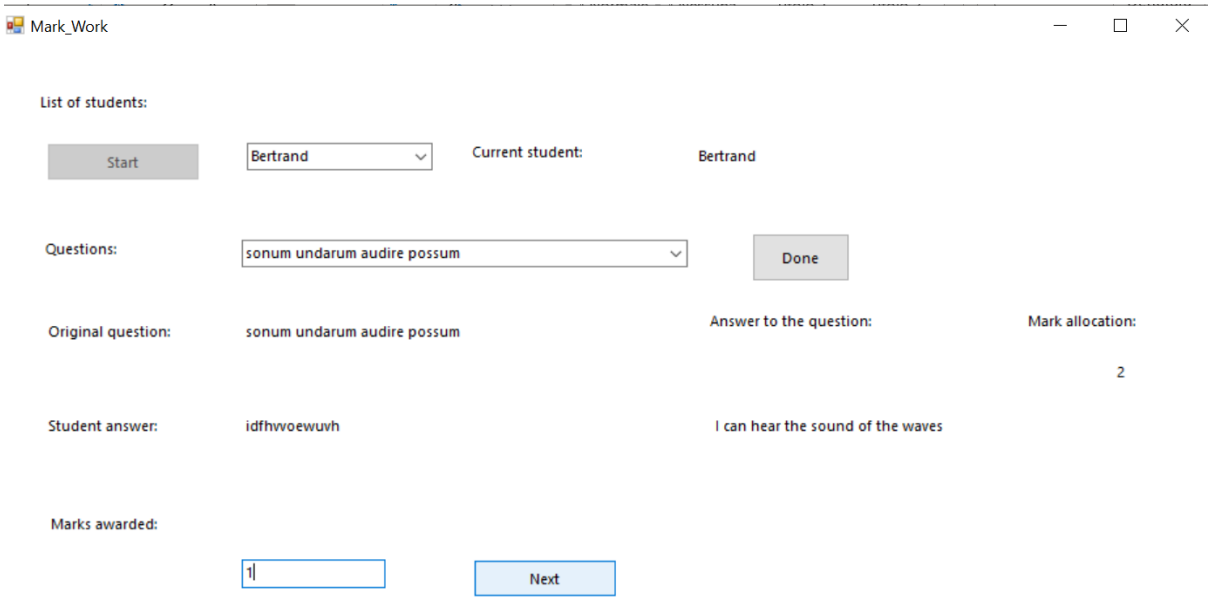
Question:

QuestionID	QuestionName	QuestionAnswer	NumberofMarks	SubjectID	LevelID	TopicID	QuestionDiffic...
18	omnia verba se...	Always learn all ...	3	10	2	1	A
19	Romae duo an...	We lived in Ro...	3	10	2	1	B
20	puer arborem a...	The boy had cli...	3	10	2	1	B
21	sonum undaru...	I can hear the s...	2	10	2	1	A
22	haec femina me...	This woman tau...	1	10	2	1	A
23	senator, vir ma...	The senator, a ...	1	10	2	1	C
24	rex amicus est ...	The king is my f...	1	10	2	1	B
25	consul plura ver...	The consul acco...	2	10	2	1	C
26	et nos et servi ...	Both we and th...	4	10	2	1	A
27	ille miles dux fa...	That soldier wa...	1	10	2	1	A
28	hunc montem c...	We shall climb t...	2	10	2	1	B
29	quinque dies c...	We were marchi...	3	10	2	1	A
30	puellam quam ...	I greeted the gi...	4	10	2	1	C
31	opus diligenter ...	I am doing the ...	2	10	2	1	B
32	omnis cibus ia...	All the food ha...	2	10	2	1	A
33	si navem consp...	If you catch sig...	3	10	2	1	A
34	patriam relinqu...	I was leaving m...	4	10	2	1	C

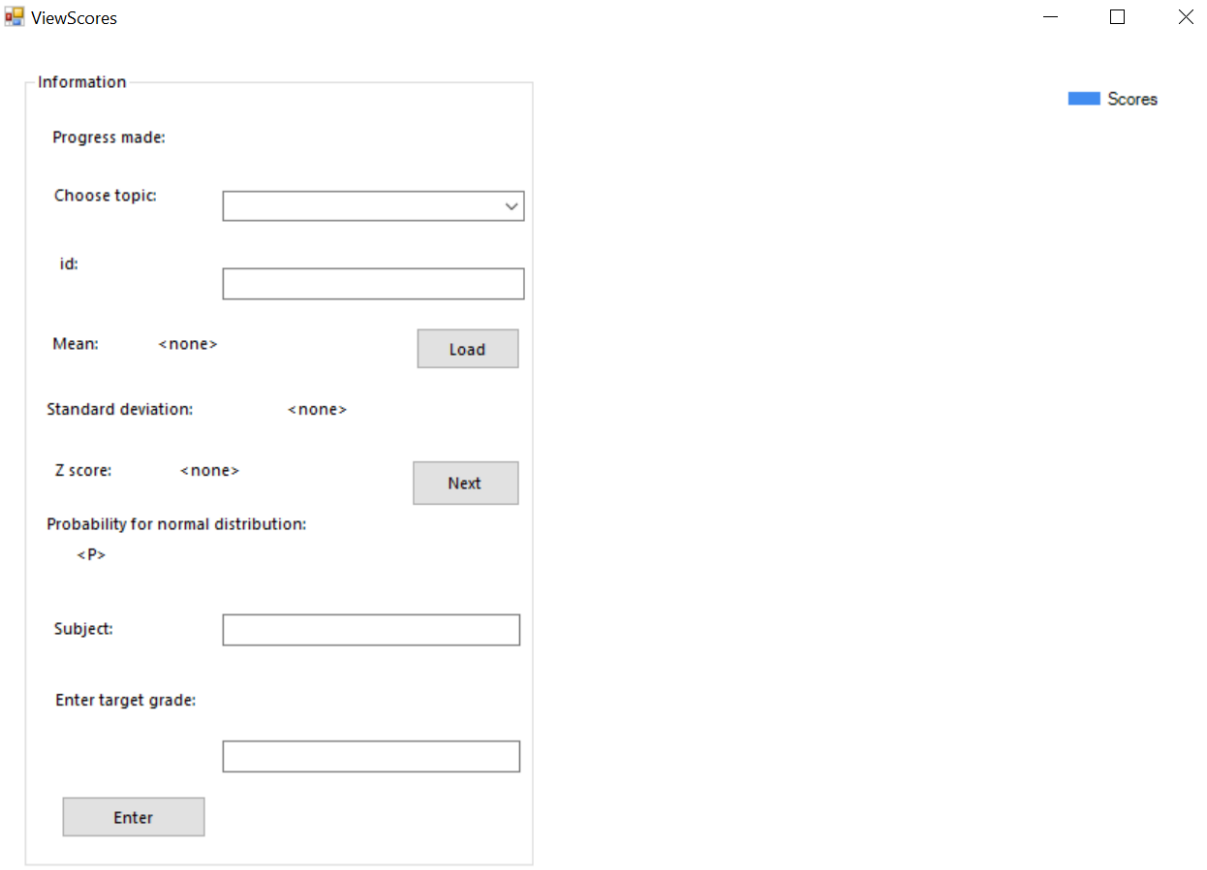
Answer:

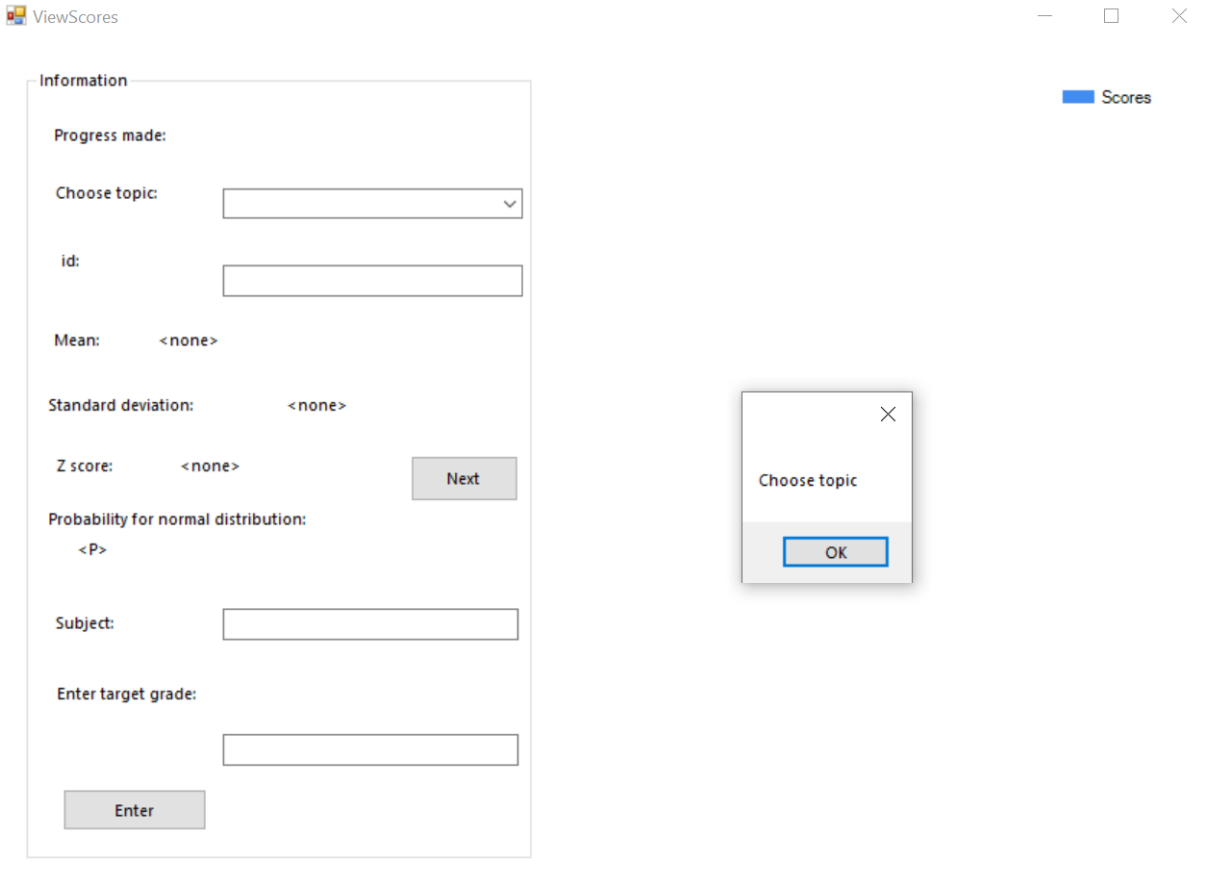
AnswerID	QuestionID	StudentID	AnswerGiven	MarksAwarded	isMarked
280	19	112	t6y4554y5y ...	3	True
281	20	112	trrtthyt ...	2	True
282	20	112	sdbfbgfbg ...	2	True
283	21	112	sdfgs ...	3	True
284	22	112	thrtyhtr ...	4	True
407	22	113	avcde ...	5	True
408	19	112	words always friends... ...	64	True
409	20	112	rome two years ...	3	True
410	21	112	idfhvvoewuvh ...	3	True
411	22	112	...	3	*NULL*
412	23	112	...	55	True
413	24	112	abcdf ...	3	True
424	19	112	iuhiuhiu ...	3	True
425	20	112	uygih ...	2	True
426	21	112	...	4	True
435	19	135	sdfdsifbnidkjfnk ...	3	True
436	20	135	sdffv ...	4	True
437	21	135	sdf ...	5	True

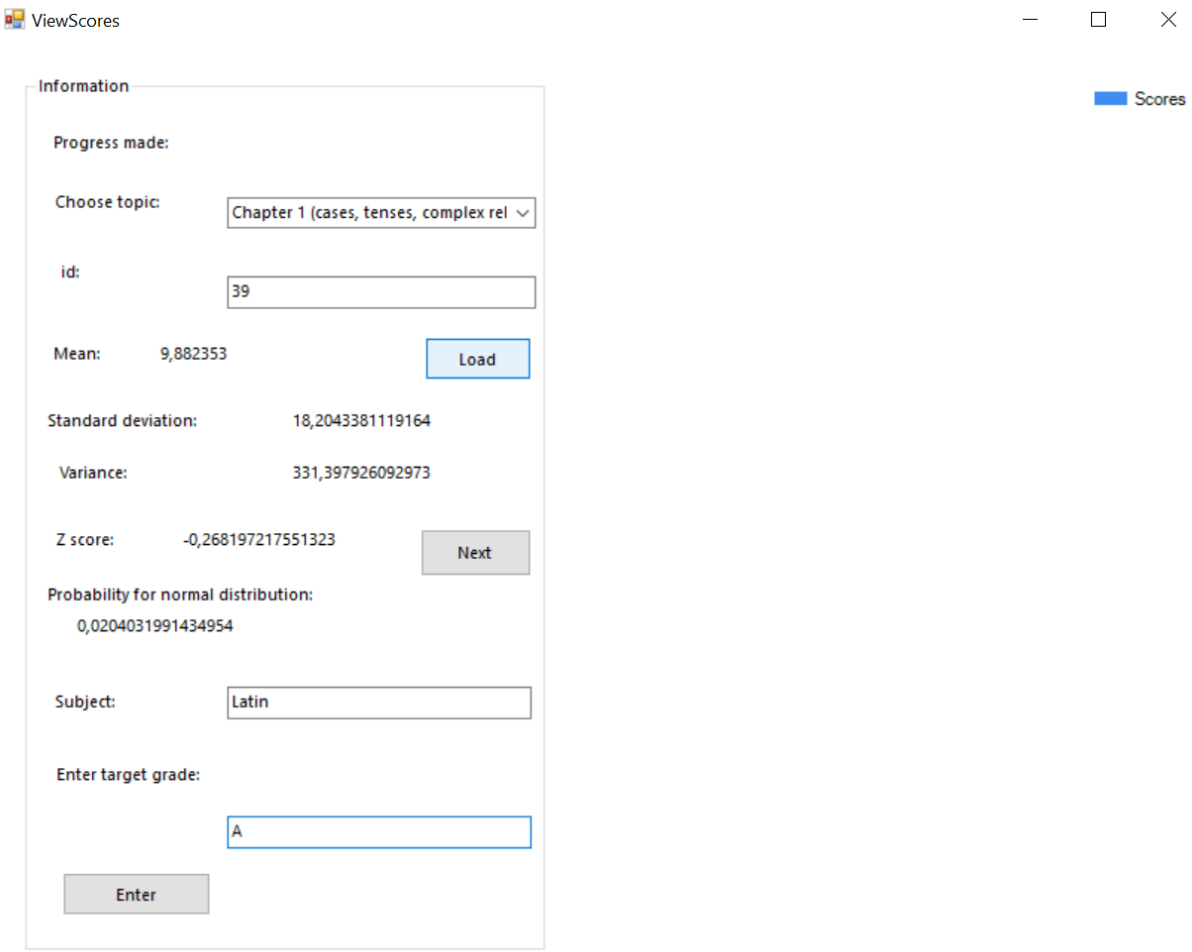
Therefore, the response submitted by the student is successfully displayed on the form.

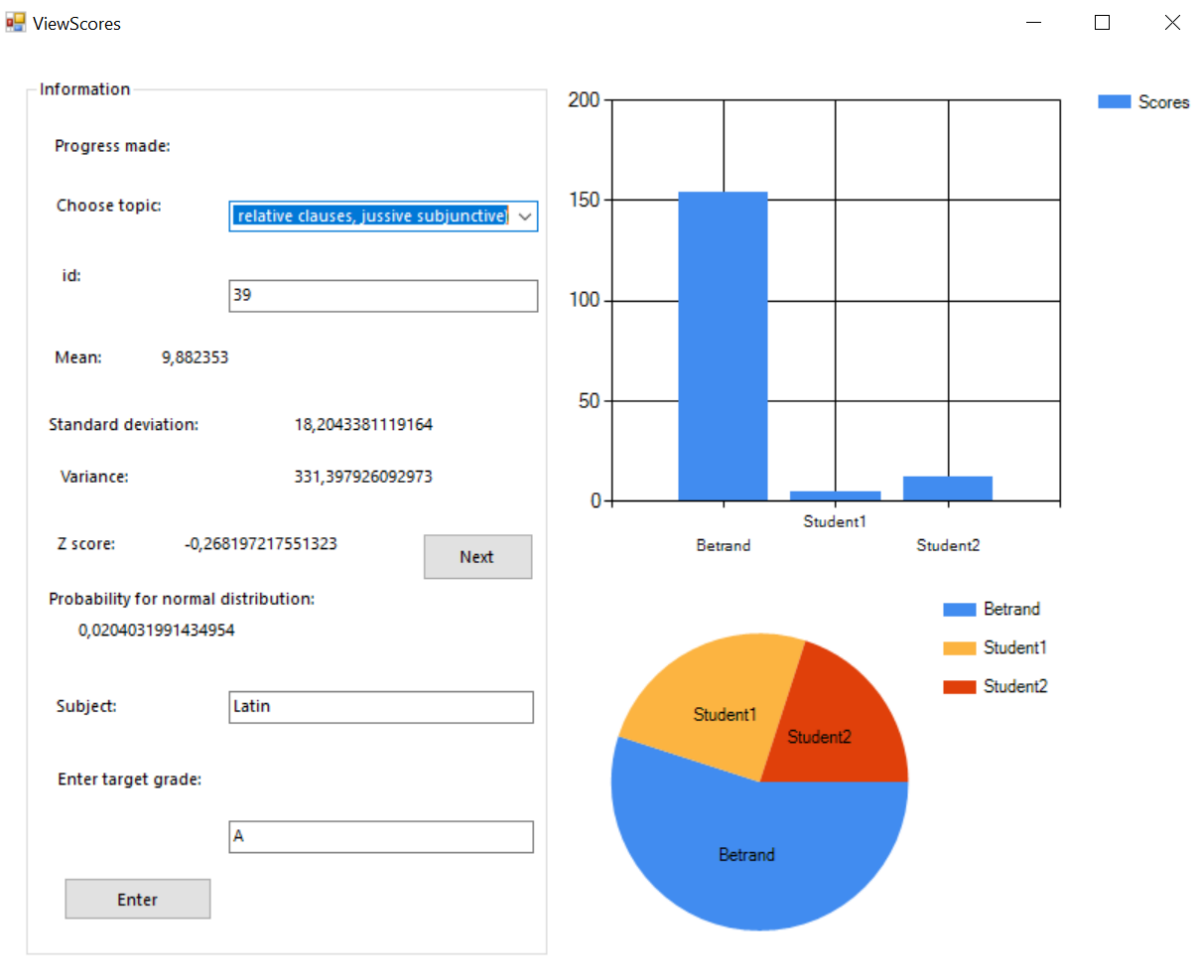


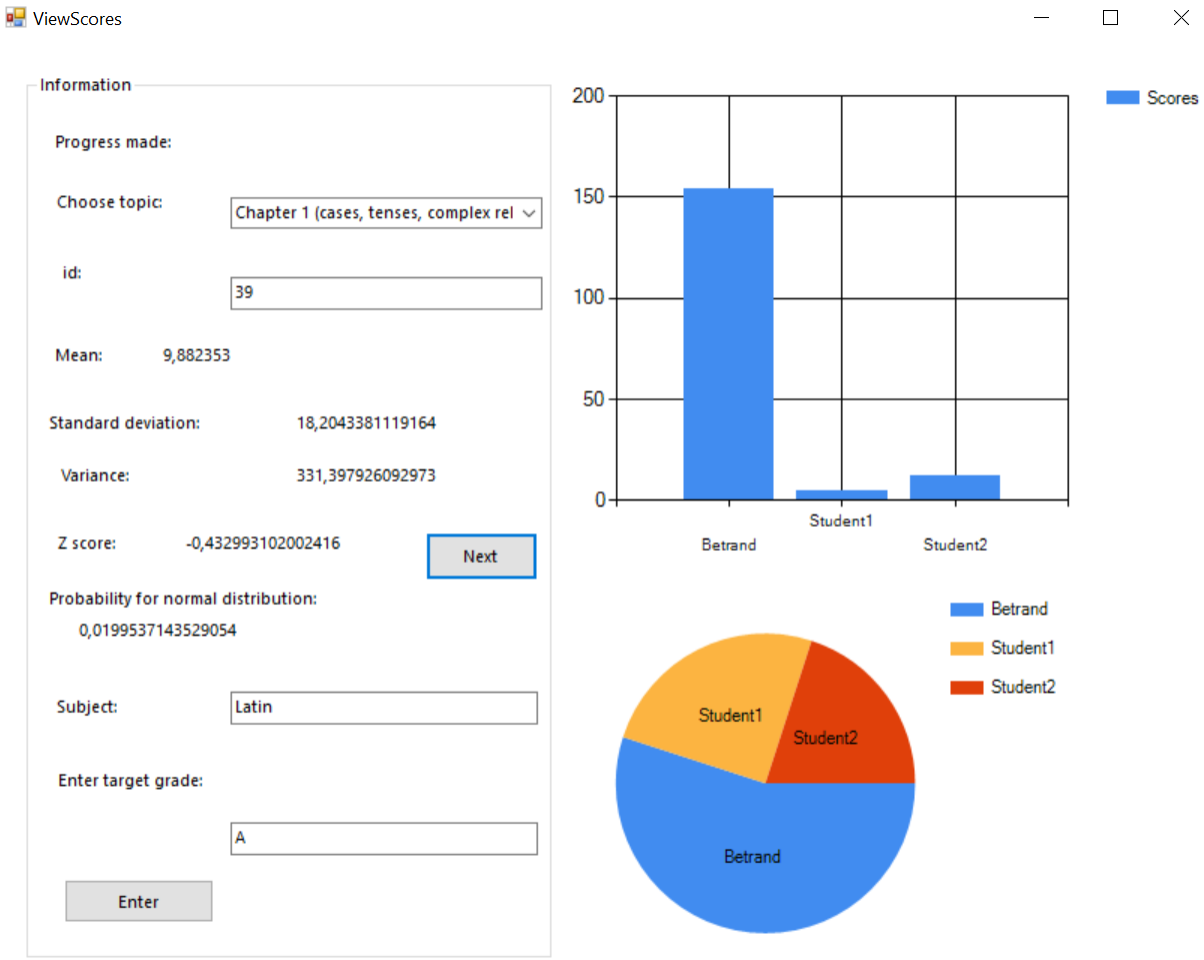
Test screenshot 14:

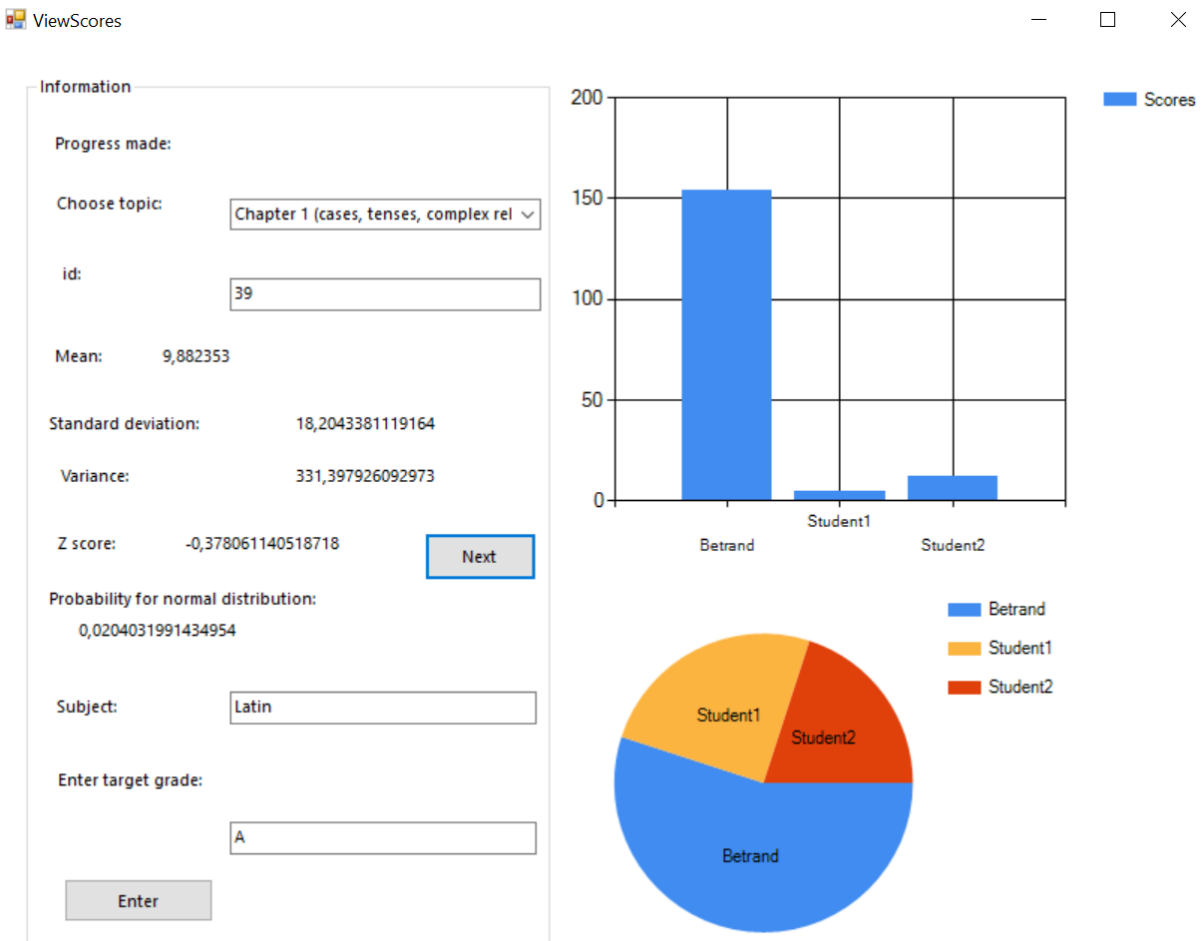

ViewScores
Information
Progress made:
Choose topic:
id:
Mean: <none>
Load
Standard deviation: <none>
Z score: <none>
Next
Probability for normal distribution:
<P>
Subject:
Enter target grade:
Enter
Scores

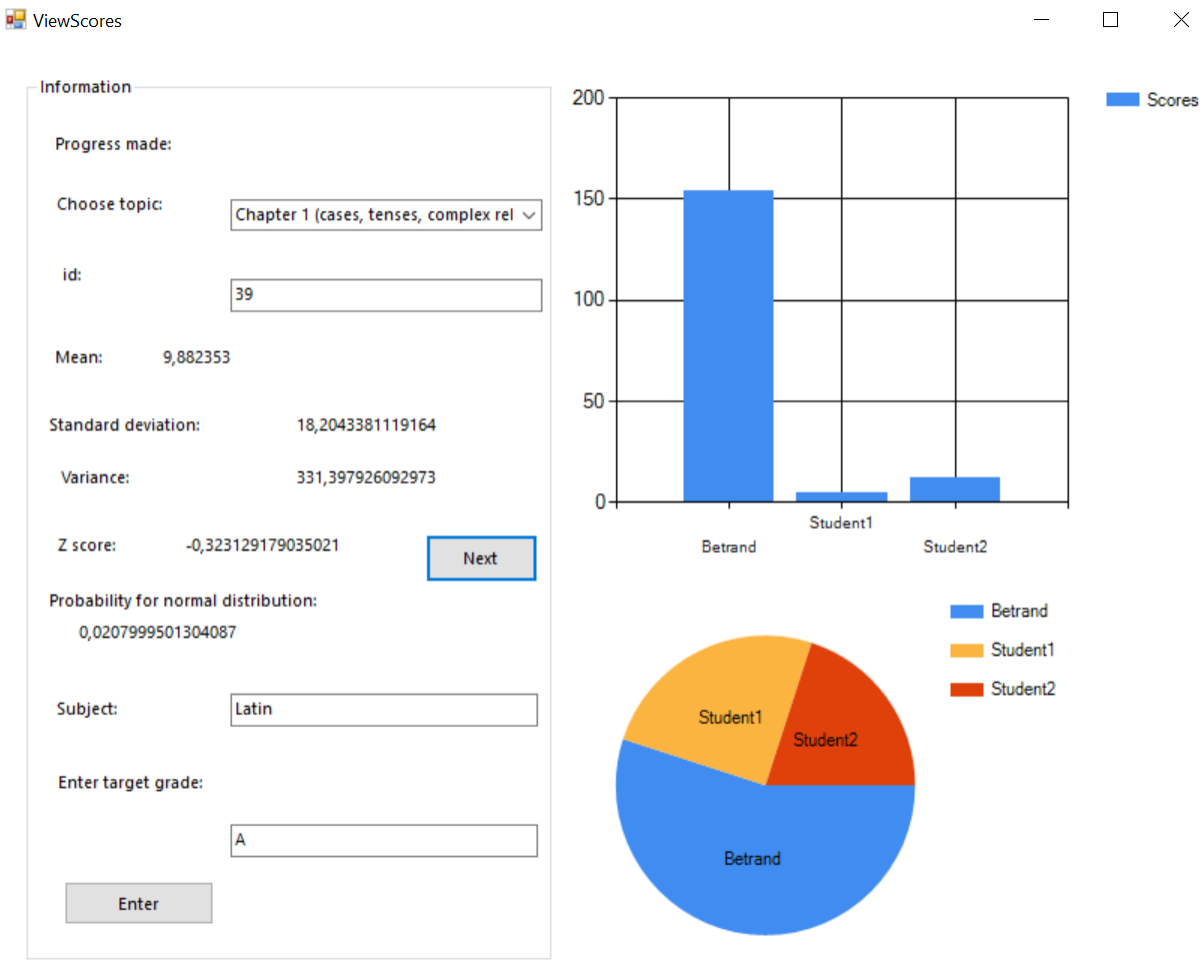

ViewScores
Information
Progress made:
Choose topic:
id:
Mean: <none>
Standard deviation: <none>
Z score: <none>
Next
Probability for normal distribution:
<P>
Subject:
Enter target grade:
Enter
Scores
Choose topic
OK


ViewScores
Information
Progress made:
Choose topic:
Chapter 1 (cases, tenses, complex rel
id:
39
Mean:
9,882353
Load
Standard deviation:
18,2043381119164
Variance:
331,397926092973
Z score:
-0,268197217551323
Next
Probability for normal distribution:
0,0204031991434954
Subject:
Latin
Enter target grade:
A
Enter
Scores

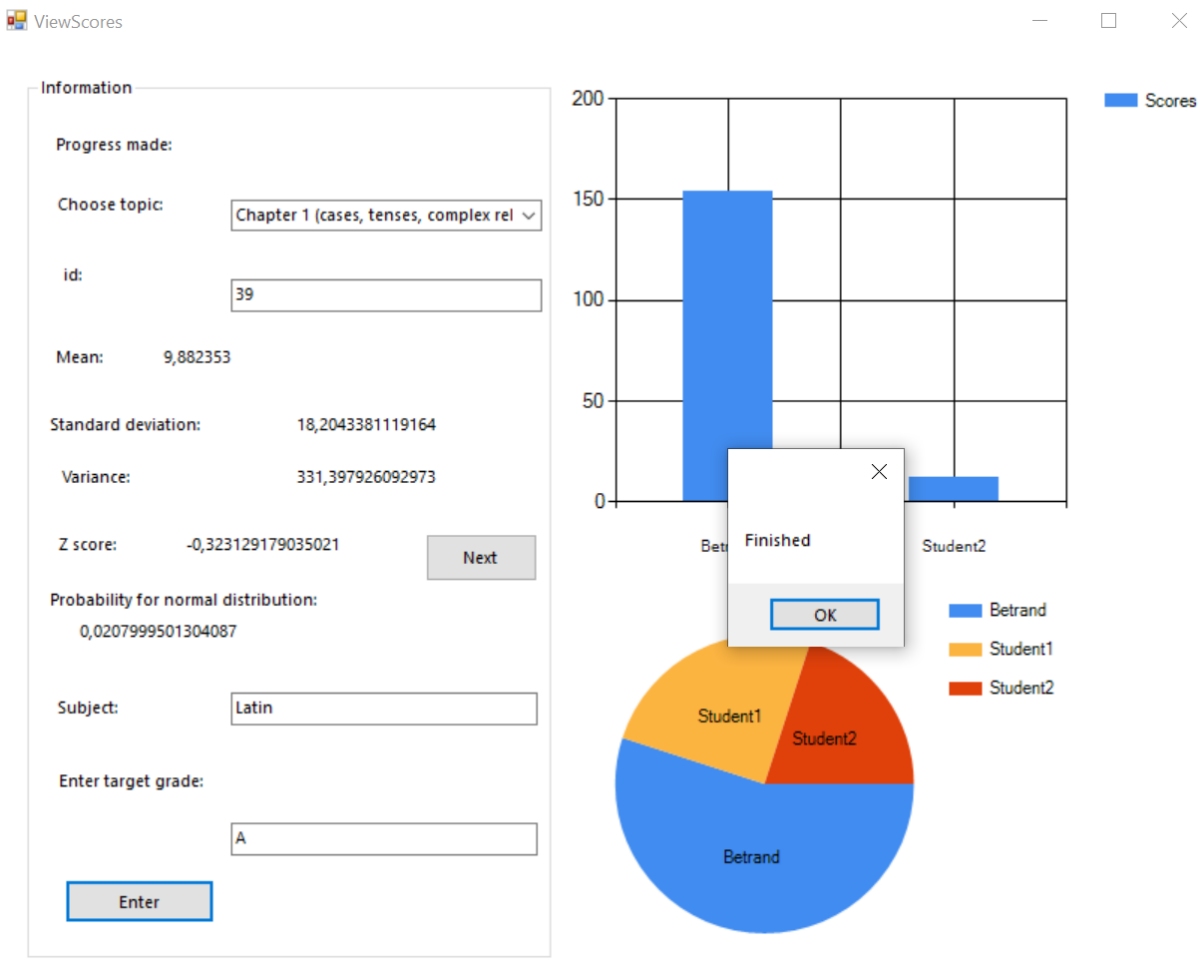

ViewScores
Information
Progress made:
Choose topic:
relative clauses, jussive subjunctive
id:
39
Mean:
9,882353
Standard deviation:
18,2043381119164
Variance:
331,397926092973
Z score:
-0,268197217551323
Next
Probability for normal distribution:
0,0204031991434954
Subject:
Latin
Enter target grade:
A
Enter
Scores
200
150
100
50
0
Betrand
Student1
Student2
Betrand
Student1
Student2
Student1
Student2
Betrand


ViewScores
Information
Progress made:
Choose topic:
Chapter 1 (cases, tenses, complex rel
id:
39
Mean:
9,882353
Standard deviation:
18,2043381119164
Variance:
331,397926092973
Z score:
-0,432993102002416
Next
Probability for normal distribution:
0,0199537143529054
Subject:
Latin
Enter target grade:
A
Enter
200
150
100
50
0
Betrand
Student1
Student2
Scores
Betrand
Student1
Student2
Student1
Student2
Betrand

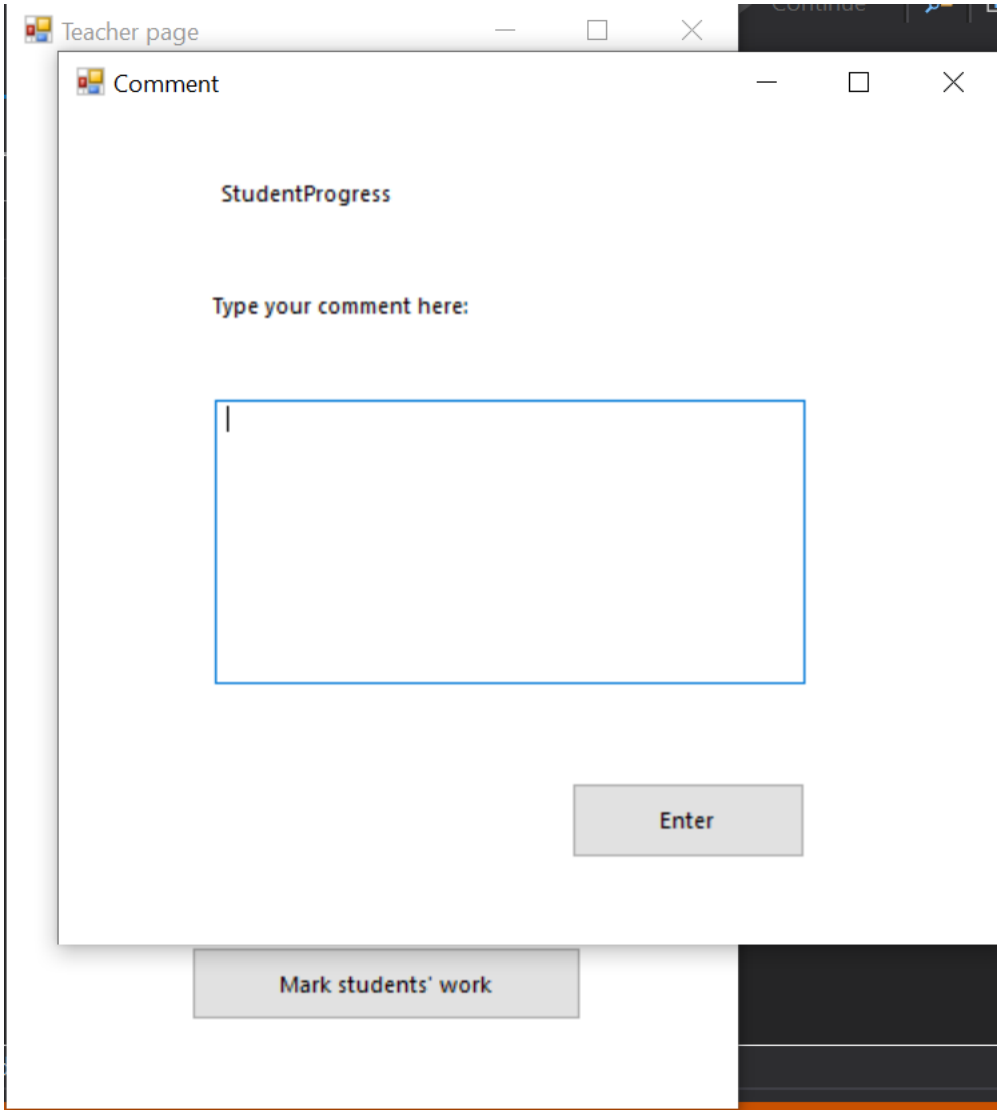

ViewScores
Information
Progress made:
Choose topic:
Chapter 1 (cases, tenses, complex rel
id:
39
Mean:
9,882353
Standard deviation:
18,2043381119164
Variance:
331,397926092973
Z score:
-0,378061140518718
Next
Probability for normal distribution:
0,0204031991434954
Subject:
Latin
Enter target grade:
A
Enter
200
150
100
50
0
Scores
Student1
Betrand
Student2
Betrand
Student1
Student2
Student1
Student2
Betrand

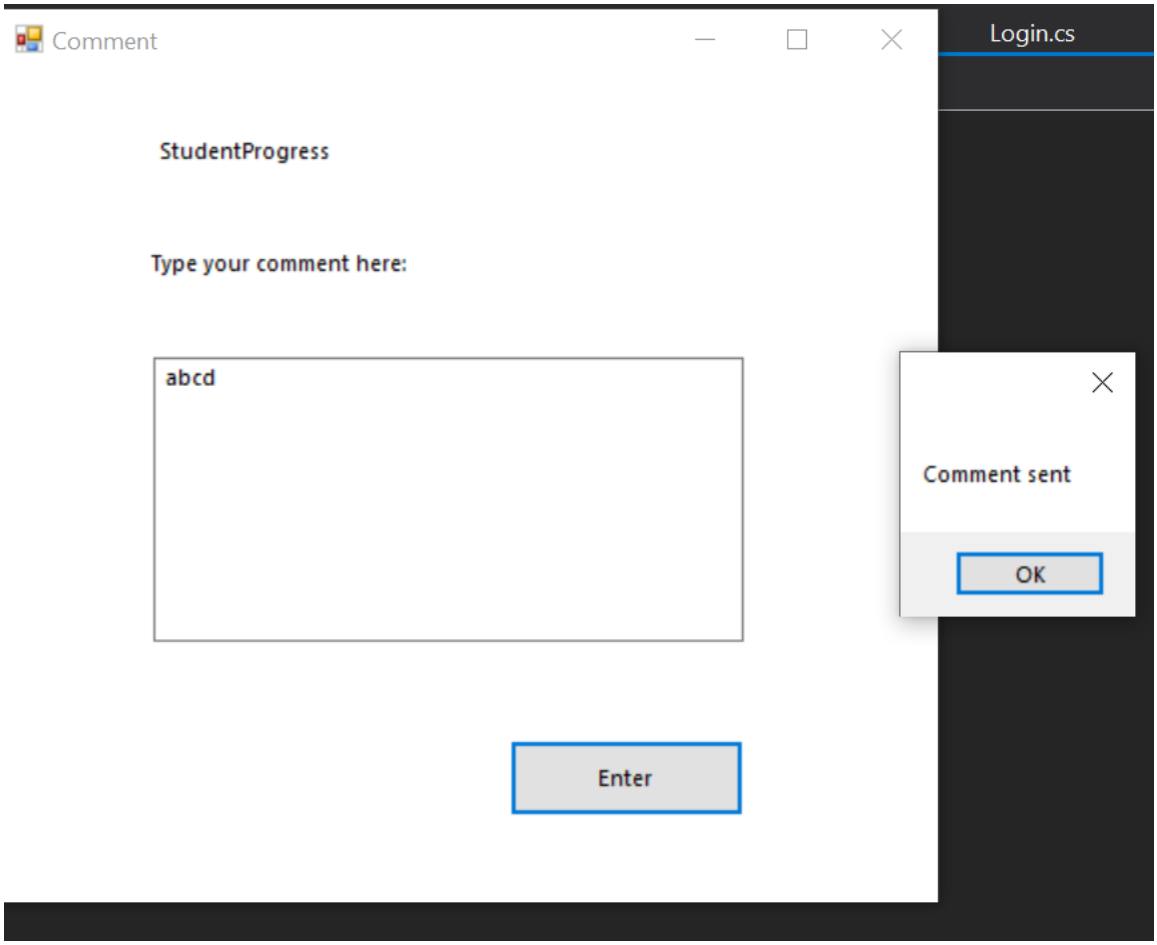


The label for the normal distribution changes successfully.



Test screenshot 15:


Teacher page
Comment
StudentProgress
Type your comment here:
Enter
Mark students' work



Comment:

CommentID	commentText	SenderID	ReceiverID	CommentType
12	abc ...	39	*NULL*	StudentProgres...

Evaluation:

Original list of objectives:

Objective	Met	Comment
1) Have multiple logins for the same application → more than one user can register and login on the same platform using different credentials	Yes	More than one type of user is able to login and register and if they enter details incorrectly, they are not stored in the database so they are not able to login
2) Be able to prevent a user from registering if they have not entered the information required for the specific type of user even if they have chosen a username and password	Yes	If a user chooses a username and password but not the other information required to register, the field is deleted from the database which means the user cannot then login with those credentials

		as they have not registered successfully (this is done using a boolean variable)
3) Prompt the student user for the topic they are finding the hardest and select questions from the database based on the topic chosen	Yes	The student selects a topic from the list and the label on the form changes accordingly, changing to the next question every time the user clicks 'submit' (done using index of array)
4) Be able to type the answer to each question in windows forms by changing the label every time the student has finished answering the question while inserting the answer given to the question which corresponds to it	Yes	Every time the student types the answer to a question, it is inserted to the database holding the student id and for which answer they have submitted every response
5) The teacher can mark the answers given by the student by inserting the number of marks and is able to see the original answer	Yes	The teacher can set the marks awarded for each student in their class and is able to see the answer given by the student and this is udpated successfully in the relevant field of the sql table
6) Draw graphs on windows forms based on the scores (marks awarded) given by the teacher for each student in one class given teacher id	Yes	The graphs of the average and individual scores for a chosen topic for the students in one given class are successfully displayed on graphs
7) Store relevant information about users such as foreign keys in relevant link tables in sql based on user input (eg. data should be stored in link table student subject after a student has selected the subject)	Yes	Every link table is updated with the relevant foreign keys after a user inputs information or has registered in the correct order
8) Use a recursive algorithm to ask the student if they wish to add another subject and keep adding the new information to the database without affecting the previous details	Yes	After the student has registered, they are asked if they wish to add another class code in which case the register group box in the form is disabled and every new class is inserted into the relevant link tables correctly
9) Use statistics such as probability density function for the normal distribution given the parameter of the marks a student has achieved	Yes	The function for the normal distribution is applied to the list of marks awarded by the teacher to the students in their particular class
10) Use aggregate sql functions to display results on windows forms graphs	Yes	Aggregate functions are used successfully with expected results
11) Connect the program to a database with which the program will interact and display information from	Yes	The visual studio solution is connected to the database and works
12) Store results from sql queries in local and minimal global variables to be used in the program	Yes	There are as few global variables as necessary to make the program work

		and send all relevant information to other forms by the constructor
13) Use a hashing algorithm with the parameter of the username selected by the user and store it in the database instead of the actual username value for added security	Yes	The hash function from A Level is applied and returns the expected value for a given username and the user is the able to login with the original username chosen even if the database table contains the ciphertext
14) Use good exception handling for when the user inputs wrong information	Yes	The system displays an error message when the user inputs invalid information or leaves the text box blank

User feedback:

Student:

I found the app easy to work with and could answer each question with no problem. Modern method to learn an ancient language. It is also different from using a trxtbook as I could easily select a topic and the questions would come up immediately which saves time. This makes it individual and can be customised to specific user data as a development.

Teacher:

I can see the graph for students in my class and mark each question. I can have an overview of the class and monitor the results and it is user friendly. And it is digital so there is less paperwork. Easy to mark questions submitted without having to search for them by hand which is more efficient.

Parent:

It's good but a possible extension could be an option to work out the determinant and eigenvalue of a 2x2 matrix and an exponential function calculator.

Possible extensions:

If I had more time or I had to start again, I would apply the central limit theorem and find a way to draw a bell curve for the normal distribution by having the z standard on the horizontal axis and the probability function on the y axis. I would also use more algorithms to find patterns in the data of students scores and make suggestions on topics. Another extension could be to write a program to work out the eigenvalue of a matrix and use sorting algorithms.

Response to user feedback:

I could incorporate the change suggested to include the determinant of a 2x2 matrix if I had more time and I would do this by getting the input values and checking for errorrs and then

send them to another form thorugh the constructor which will will the matrix with user input and store it in an array and use indexing to perform calculations on the matrix.

Conclusion:

Although the solution is not entirely completed with more complex algorithms, it works as a system with exception handling and inserting user input and foreign keys into the relevant sql tables plus the chart control in windows forms. Before opening visual studio, the plan was to write a program that learns from the pattern of students' scores. However, the first step was to display the questions on a platform which is different to using a simple console application where the questions would have been easier to display. Another aim was to draw a normal distribution curve, and even if there is a website that does this using bitmaps, this was not essential for the project as an identical solution already exists. Therefore, the skeleton (structure) of the program works in a way that is accessible to users.

Link to the video of the full working solution including database tables:

Important:

https://youtu.be/sAiNSzonlG0

Websites used (plus textbooks):

http://news.bbc.co.uk/1/hi/education/7172077.stm
https://www.youtube.com/watch?v=zid-MVo7M-E
https://app.lucidchart.com/documents/edit/91180499-3fce-4458-a35c-42a64de61cac/YGcM5DNywbTK
https://www.youtube.com/watch?v=i3dg99MWLZU
https://www.tutorialspoint.com/design_pattern/mvc_pattern.htmhttps://www.codeproject.com/Articles/383153/The-Model-View-Controller-MVC-Pattern-with-Csharp
https://www.codeproject.com/Articles/383153/The-Model-View-Controller-MVC-Pattern-with-Csharp
https://www.youtube.com/watch?v=wfWxdh-_k_4&t=1753s
https://www.aspsnippets.com/Articles/Create-Bar-Chart-in-Windows-Forms-Application-using-C-and-VBNet.aspx
https://www.aspsnippets.com/Articles/Create-Bar-Chart-in-Windows-Forms-Application-using-C-and-VBNet.aspx
https://ilnumerics.net/bar-plots.html
https://www.youtube.com/watch?v=QmT1oec1YBc
https://www.youtube.com/watch?v=VT76ruswmHo
https://www.bing.com/videos/search?q=how+to+link+combobox+database+and+show+values+in+textbox&&view=detail&mid=B6730BB7B9D0BA386B1FB6730BB7B9D0BA386B1F&&FORM=VRDGAR&ru=%2Fvideos%2Fsearch%3Fq%3Dhow%2520to%2520link%2520combobox%2520database%2520and%2520show%2520values%2520in%2520textbox%26qs%3Dn%26form%3DQBVR%26sp%3D-

1%26pq%3Dhow%2520to%2520link%2520combobox%2520database%2520and%2520show%2520values%2520in%2520textbox%26sc%3D0-56%26sk%3D%26cvid%3D7CE709205D374EA0B6144BFF327EF358
https://stackoverflow.com/questions/24098263/check-if-username-exists-in-database
https://www.bing.com/videos/search?q=login+windows+forms+c+sharp&&view=detail&mid=44A73D7156121A9A0EB744A73D7156121A9A0EB7&&FORM=VRDGAR&ru=%2Fvideos%2Fsearch%3Fq%3Dlogin%2Bwindows%2Bforms%2Bc%2Bsharp%26FORM%3DHDRSC4
https://stackoverflow.com/questions/25739788/select-query-to-get-data-from-sql-server
https://stackoverflow.com/questions/15412000/how-to-replace-label-text-with-text-in-the-column-c-sharp
https://stackoverflow.com/questions/40864484/priority-queue-in-sql-server
(18) What is the Normal Distribution? - YouTube
(18) The Normal Distribution (1 of 3: Introductory definition) - YouTube
(18) Probability Density Function of the Normal Distribution - YouTube
Selecting a Simple Random Sample from a SQL Server Database (mssqltips.com)
Untitled document - Google Docs