COMP4 Project

Contents

Analysis

Identification

Particularly at A-level, revision for physics within a large science department is very time consuming for students and teachers, requiring manual intervention such as photocopying, creating and printing of worksheets, and marking tests and other questions. Much time is also spent by teachers in ensuring that students have an understanding of all topics before taking final examinations.

A teacher of A-level Physics, discussed with me the concept of using an electronic system for student revision of key topics in Physics, which could be used individually outside of lessons, so that students could revise more efficiently, with less dependency on use of questions within lessons. This would reduce time taken in lessons, and allow more time for students to go over and ensure understanding of topics, using some lesson time to cover certain topics in more detail if necessary. A common way of testing students understand a topic is using exam questions; however, it can be time consuming to mark all papers and find where problems exist.

Description of Current System

Students are provided lists of tasks to complete between lessons, such as questions from a book – students may require photocopies of the questions, they may also be given photocopies of past exam questions to complete. This system means that students complete the work, before it is marked by teachers. Teachers also usually record student progress for all questions and tasks through a logging system, this can be time consuming to update or to find and retrieve information from.

Time management is very important in order to support a large number of students, time spent producing revision material could be reduced, allowing time for students to develop understanding and revise on their own more effectively, and teachers to prepare better teaching materials.

Worksheets for each topic or lesson are created and multiple copies printed for students to use, creating these worksheets - particularly if every lesson - is very time consuming. When using past exam papers, the relevant questions to the topic must be searched for, in some cases, questions for a particular topic may not be found on all past papers.

Users & needs

Teachers are administrators of the system, they are able to modify, add, and remove questions – they are also able to check student progress and create/delete students – when students are deleted, data (for example question answers) associated with them is also deleted; similarly, when a question is deleted, all its data and references to it will be deleted as well – these steps are taken in order to comply with data protection. Students are able to log into the system, in order to view and answer questions shown as images. Both students and teachers use Windows 7 within school, so are familiar with it; the interface will be designed to follow most applications designed for Windows. The system should be available to access through all school computers and at home on student computers. The questions will be shown in their original form (image) and can be added from a variety of different topics as needed, and all answers will be numerical.

Questionnaire

NAME:	
▪ What is the current system used?	Task sheets and photocopies of exam questions are given to students, when tasks are done they are checked off by students - answers are checked and tasks/questions are marked by teachers Worksheets for a particular lesson or topic are created or reused by teachers and many copies are printed for all students
▪ What problems exist with the current system?	It is difficult and time consuming to gather questions – answers must marked by teachers during lesson time Creation of work sheets - in some cases every lesson - is also very time consuming As not all students work at the same pace, some may be restricted by time in the lesson Answers may be more complex than necessary depending on the type of revision
▪ What is most time-consuming or difficult with the current system?	Questions must be gathered manually from various sources, such as textbooks, and printed or photocopied Large parts of lesson time are spent working on questions Finding the appropriate questions/topics from past exam papers, sometimes questions for a topic are not on a particular exam Marking long, often detailed exam questions Tracking progress and finding specific problems for each student
▪ Do you know of any systems that already exist for this purpose? What do you like about them, and/or what could be improved?	Online tests provide easy/quick revision However, they are restricted to certain topics, are not real exam questions, and are not always the same questions, so solutions cannot always be explained as easily Not all websites are consistent in type and format of questions
▪ Which groups or individuals will most commonly use this system?	A-level students studying physics, outside of lesson times Teachers may add questions for students

▪ What benefits can an electronic/automated system provide?	Conserves lesson time Encourage students to work independently Allows questions to be updated easily, and more added if necessary, or if students want to do extra questions Students would be able to work at their own pace Reduce time spent in lessons, allowing students to go over topics in their own time, meaning that lesson time can be focused on covering more detail of certain topics – allows students to revise more efficiently Less paper is used
▪ Where most commonly will this system be used, in what locations/facilities?	Within school, using school computers On student computers outside of school time
▪ When and how will this system be used, in what use cases/time periods?	Primarily as practice or preparation for exams To aid with revision
▪ What is the available hardware and software?	Students are able to access computers at school, many also use have access to computers at home Computers in the school use Windows 7, the majority of students have access to a computer running Windows outside of school
▪ What security is required, what groups of people need to access the system?	Students and teachers should be able to use the system Questions should be able to be added by users
▪ How often is the system expected to be used?	In between lessons At least once every 2 weeks, more often closer to the exam period

Potential Solutions

Scan/email solution:

Scans of questions can be made by teachers and emailed to students to complete outside of lessons; mark schemes can be included with the questions. This solutions means that less paper is used, and that actual exam questions can be used, with only the appropriate questions set to be completed. Problems with this solution include that it cannot easily be tracked, it must still be checked manually by teachers during the lessons, and students could use the answers included with the questions.

Web based solution:

Existing solutions exist on the internet, such as revision & question websites. Problems with these solutions are that information cannot always be verified, and questions may be incorrect. Content of questions may not always be relevant to work set. Websites are restricted to their own questions, and do not use real exam questions; so they lack in preparation for exams, and that the progress cannot be easily tracked in a centralised place; teachers are unable to track the progress of the students, unless students check off the websites & questions that they have completed as set by the teacher. Websites also pose the risk of being infected; potentially exposing a security risk to students or the school, especially if used on school computers.

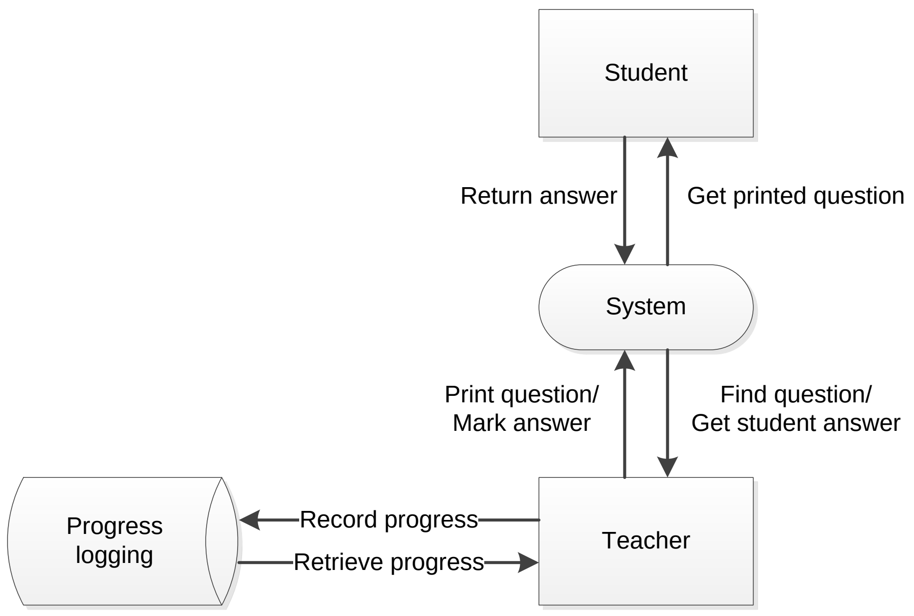
Proposed Solution

An electronic system can work in a similar way to the existing paper-based system, it means that little to no paper is used, as questions and answers can be taken from digital copies through a teacher's computer. It also allows for progress to be tracked easily – in a centralised location, and requires little to no input from teachers outside of cases where work can be set by teachers; allowing them more time to prepare lessons. This solution means that real exam questions can be combined with questions from other sources, or generated randomly in order to provide quick practice of simple questions. The system will be programmed in Visual C#; which features a variety of built-in graphical tools, such as to display images, text, tables and buttons – this aids in making the final program easier to use and understand, also making it possible to produce a suitable layout of elements quickly and easily. My experience from the first year and working on other projects in C# means that I have an understanding of how it is used and can be applied to this solution. The system will also use a Microsoft SQL Server database, meaning that data such as questions and student progress can be accessed through any computer running the program in any location.

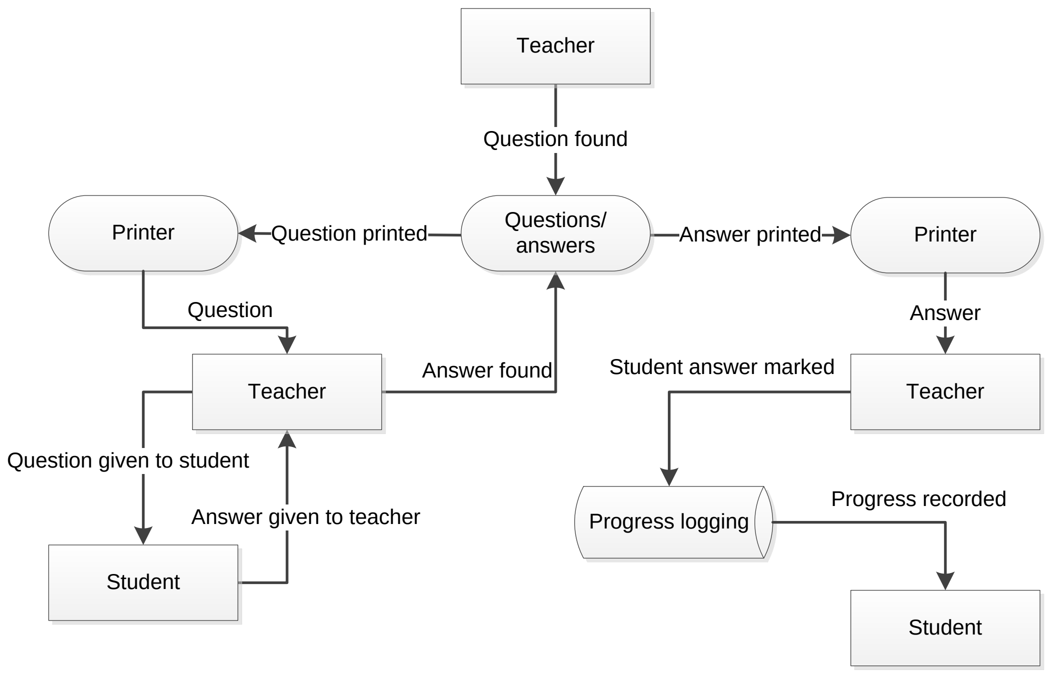
Data Flow Diagrams

Current System

Level 0

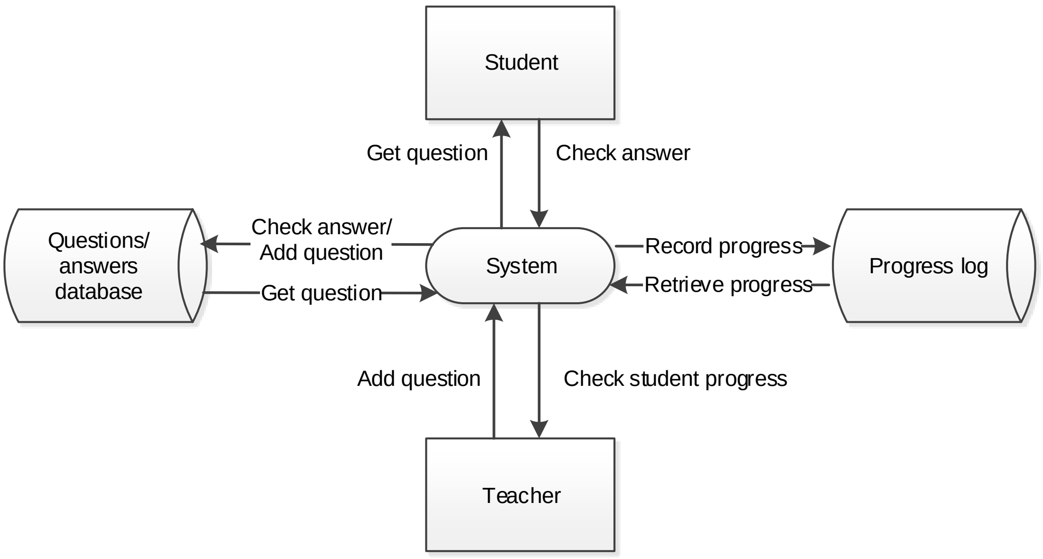


Level 1

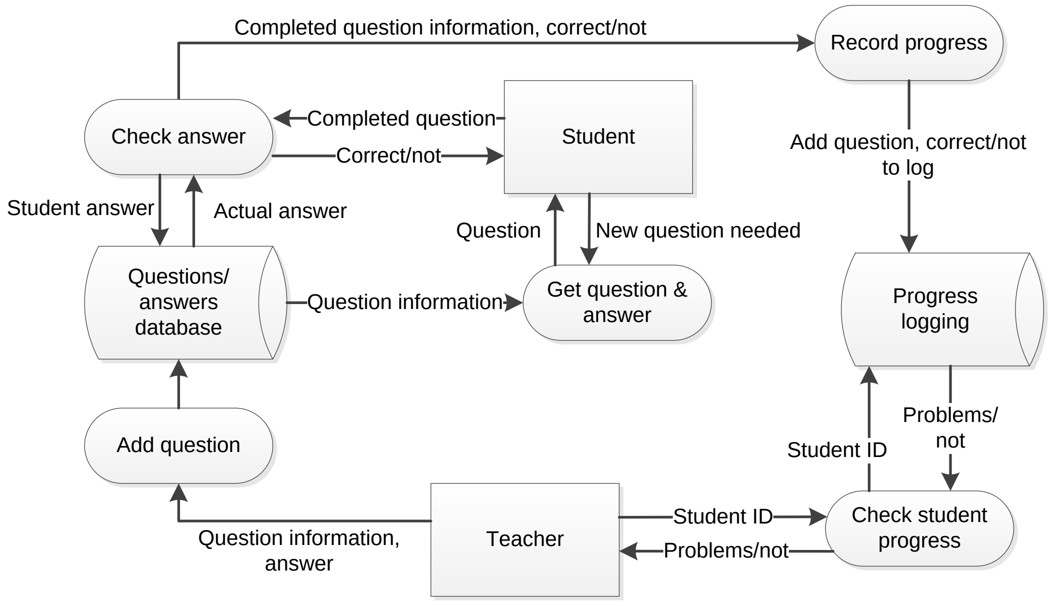


Proposed System

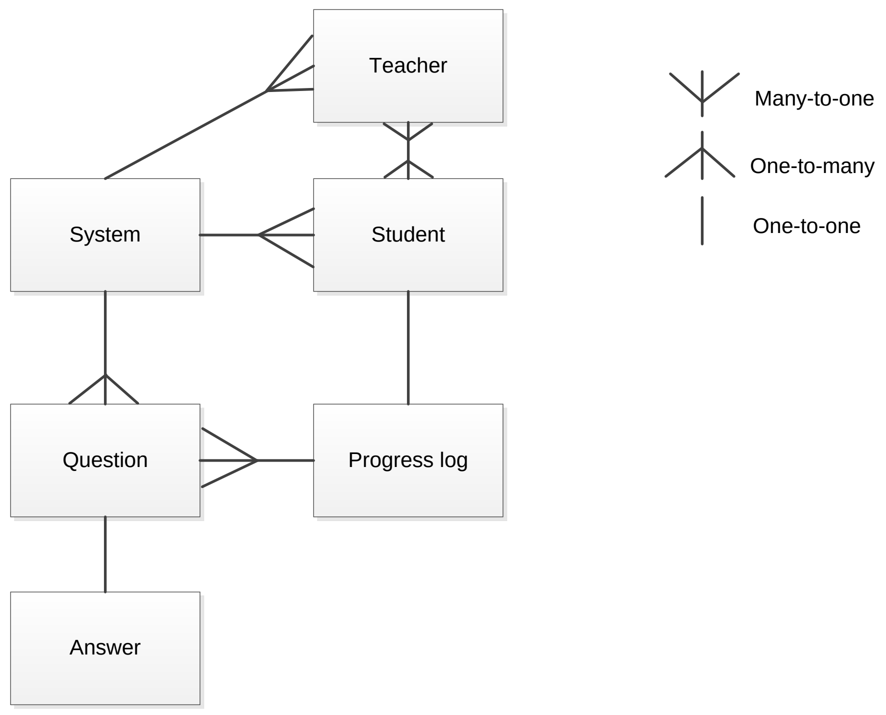
Level 0



Level 1



Entity Relationship Diagram



Object Analysis Diagram

N/A

Data Sources & Destinations

Current System

Data	Source	Destination
Question/answer	Textbook, past paper	Paper copy to student
Student progress	Completed tasks list	Logging book

Proposed System

Data	Source	Destination
Question/answer	Textbook, past paper	Questions/answers database table
Student details	Teachers	Users database table
Student progress	System, after question completed	Student progress database table

Data Dictionary

Name	Purpose	Type	Size	Example data	Validation
Username	Stores user identification	String	50	101-auser	Not empty
Password	Stores user password	String	50	Abc123	Not empty
Year group	Stores year group of user	Integer	2	13	Between 12-13 or 0 (teacher)
Completed questions	Stores number of completed questions	Integer	3	42	Must be a number
Correct questions	Stores number of correct questions	Integer	3	37	Must be a number
Question image	Stores image for question	Binary	2000	Image	Minimum size 50x50px
Question name	Stores name for question	String	50	4.a.ii.2	Not empty
Answer	Stores answer for question	Floating point number	50	427.35	Must be a number
Number of marks	Stores number of marks for question	Integer	2	3	Must be a number

Data Volumes

The system will store the details of up to 70 users (teachers/students); each user will have their information stored, which will be about 70*10KB = 700KB total user information data. Students will also have progress records stored; the progress log for each student can contain up to 100 questions, so each log can be up to 100*1KB = 100KB each; around 60 of the users are students, meaning around 60*100KB=6MB maximum total data for student progress logs.

The system will also be able to store up to 100 combined questions/answers at a time; if the maximum size of a question image is 2MB, 100*2MB=200MB, plus answers of up to 1KB each, is 100*2KB=100KB means 200.1MB maximum total question/answer data.

Objectives

1. Students are able to input answers with 2 decimal places or as integers as required, all answers are converted to 2 decimal places, complete by October 11th
2. The main interface will have a menu bar, complete by October 14th
3. Multiple windows can be opened simultaneously within the main interface, complete by October 15th
4. Questions and quizzes can be printed from the same menu bar; the item currently in focus determines which is printed, complete by October 22nd
5. Users can be created for individual students and teachers, student users can be created by teachers, complete by January 2nd
6. Deleted users will have their associated data also deleted, complete by November 4th
7. Deleted questions will have their data and references to them deleted, complete by November 4th
8. Users are able to login to the system, complete by November 6th
9. Administrative users (teachers) have permissions to add and remove questions, complete by January 9th
10. Input must be validated to only allow numerical answers, complete by November 15th
11. Images will be used for questions, as they will be shown in the original form; such as a screenshot of an exam question/scan of textbook, complete by November 24th
12. A working space is available for notes, it can be drawn on using a virtual pen, complete by December 14th
13. The working space will have multiple selectable foreground/background colours, complete by December 16th
14. Quizzes can be created automatically by randomly selecting variables and generating values to create a set number of questions, as set by the user, complete by February 12th
15. The database will be accessible in under 10 seconds, completed by November 1st
16. The main interface will completed by the end of October
17. Database will be created and linked by the end of November
18. Working space/drawing functionality will be completed by the end of December
19. User login/creation and student progress logging functionality will be completed by the end of January
20. Random quiz and printing functionality will be completed by the end of February

Limitations

1. The system cannot be too complex using the available resources
2. All answers must be in number form, it is too complex to add text answers as answers such as definitions can vary significantly
3. Randomly generated quizzes created can only be completed within the application or printed as a hard copy – adding these question types to the database is too complex
4. Users are created manually, adding the ability to import many users simultaneously is too complex

5. There is only limited access to hardware in and outside of school for students and teachers, the system will use Windows as it is available in school and for most students
6. Passwords will not be encrypted as this is too complex

Design

Record Structure

N/A

Overall System Design

Login Form

Inputs	Processes	Storage	Outputs
Username Password	Check username exists Check password is correct		Valid username/not Correct password/not

Quiz Form

Inputs	Processes	Storage	Outputs
User information, username, password, year group	Check student answers Add question Get question	See database diagram	Question images Student progress Random questions Printed copy output

Add Exam Question Form

Inputs	Processes	Storage	Outputs
Image location	Convert image to binary Generate question name		Generated name for question being added

Definition of Data Requirements (Design Data Dictionary)

Name	Description	Type	Size	Validation
username	User identification	String	50	Length check – username is >=5 and <=12 characters
password	User password	String	50	Length check - password is >=8 and <=50 characters
yearGroup	Year group of user	Integer	2	Range check - year group is 12, 13 or 0 (teacher)
quizDateTime	Date/time of quiz being created	DateTime	10	None, used programmatically
questionImage	Image for question	Binary	2000	Datatype check – must be valid image
questionName	Name for question	String	50	Presence check - question name is entered when adding question

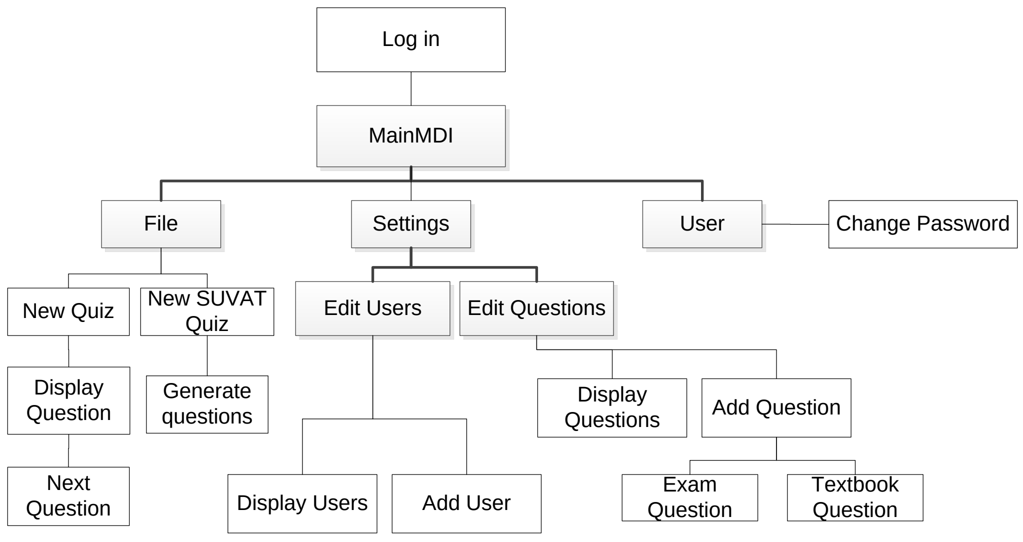
questionAnswer	Answer for question	Floating point number	50	Datatype check - only a number is inserted into answer
questionMarks	Number of marks for question	Integer	2	Datatype check – must be an integer
userAnswer	User's answer to question	Floating point number	50	Datatype check - only a number is inserted into answer
questionMonth	Month of question being added, used to create questionName	String	3	List check - month of questionName is 'Jan' or 'Jun', uses dropdown
numberOfRandomQuestions	Number of random questions to be generated	Integer	2	Range check – number between 1 and 20
s,u,v,a,t	Used to generate random questions	Integer	3	None, used programmatically

Modular Structure

The components of the system will be separated through use of different forms with separate code. For example, the form used to add exam questions to the database will use its own code which is completely separate from the rest of the system. The system will use an MDI (Multiple Document Interface), allowing easy management of windows and navigation to different forms.

All forms are accessible through the MenuBar at the top of the MDI container, which contains the following items:

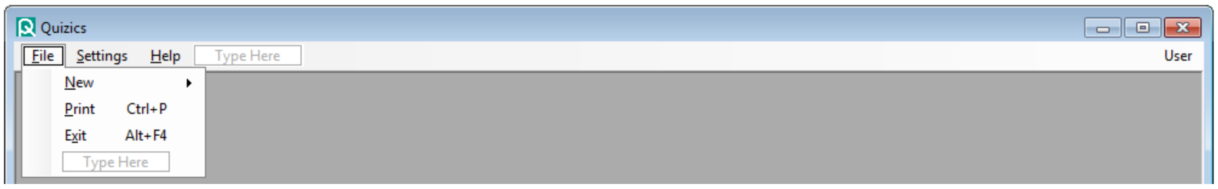
- File
  - New
    - Quiz
      - 1 question
      - 5 questions
      - 10 questions
      - 15 questions
      - 20 questions
    - SUVAT Quiz
  - Print
  - Exit
- Settings
  - Edit Users
  - Edit Questions
- Help
- User
  - Change Password



User Interface Design

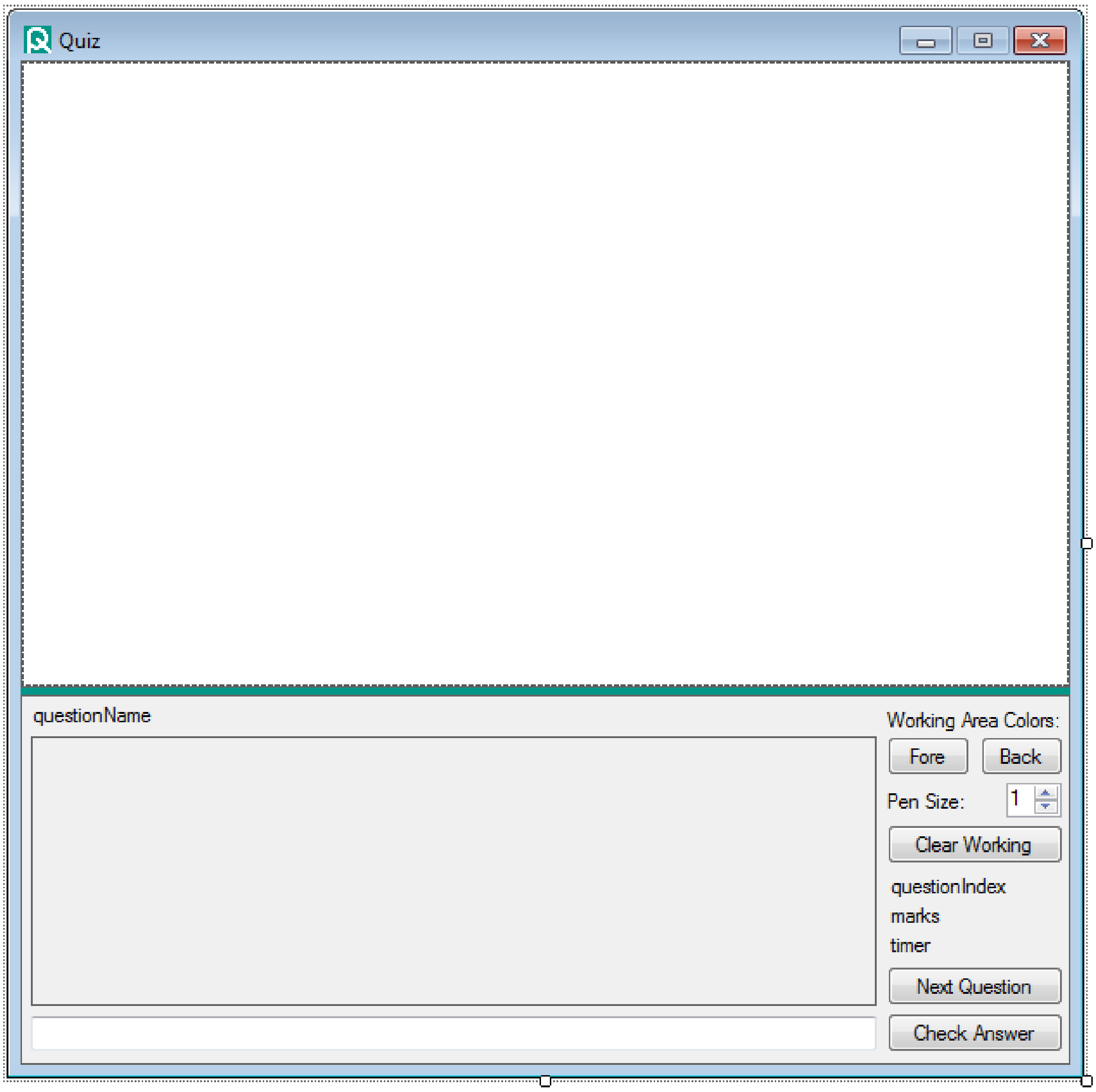
The GUI will be designed to follow the convention of Windows applications; including the use of a menu bar and standard keyboard shortcuts, for example Ctrl-P for Print, F1 for Help, and Alt-F4 for Exit. All forms will also follow the standard colour scheme for Windows Forms, matching the conventional Windows style.

In order to allow multiple windows to be opened simultaneously, the main interface will be an MDI (Multiple Document Interface). The MenuBar at the top of the main interface will be used to open other forms and perform tasks such as printing, and at the end will display the current username and user type.



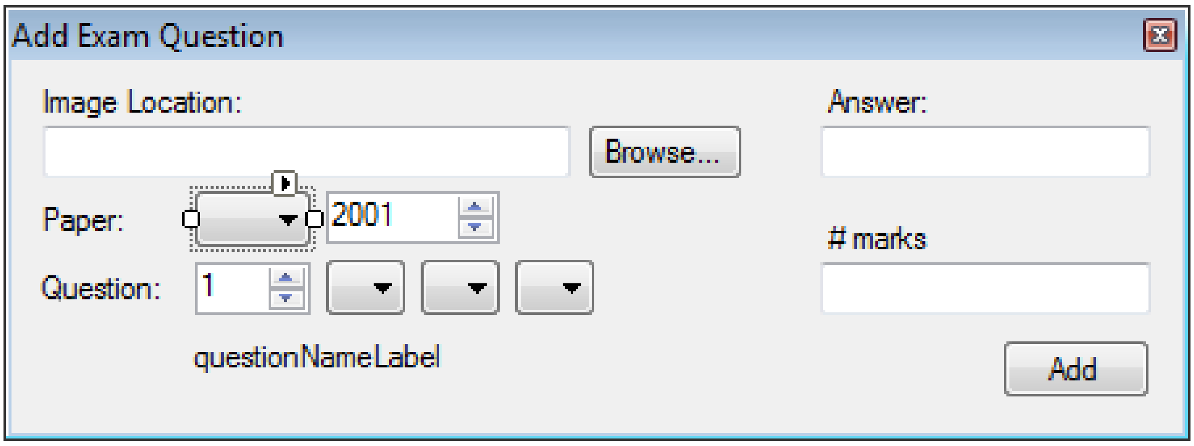
Quiz Form

The quiz interface will show questions in image form using a PictureBox, and allow users to enter their answers into a TextBox. It will feature a SplitContainer that allows the size of the image/working area to be changed. The name of the current question's name and its index will be shown to the user using Labels.



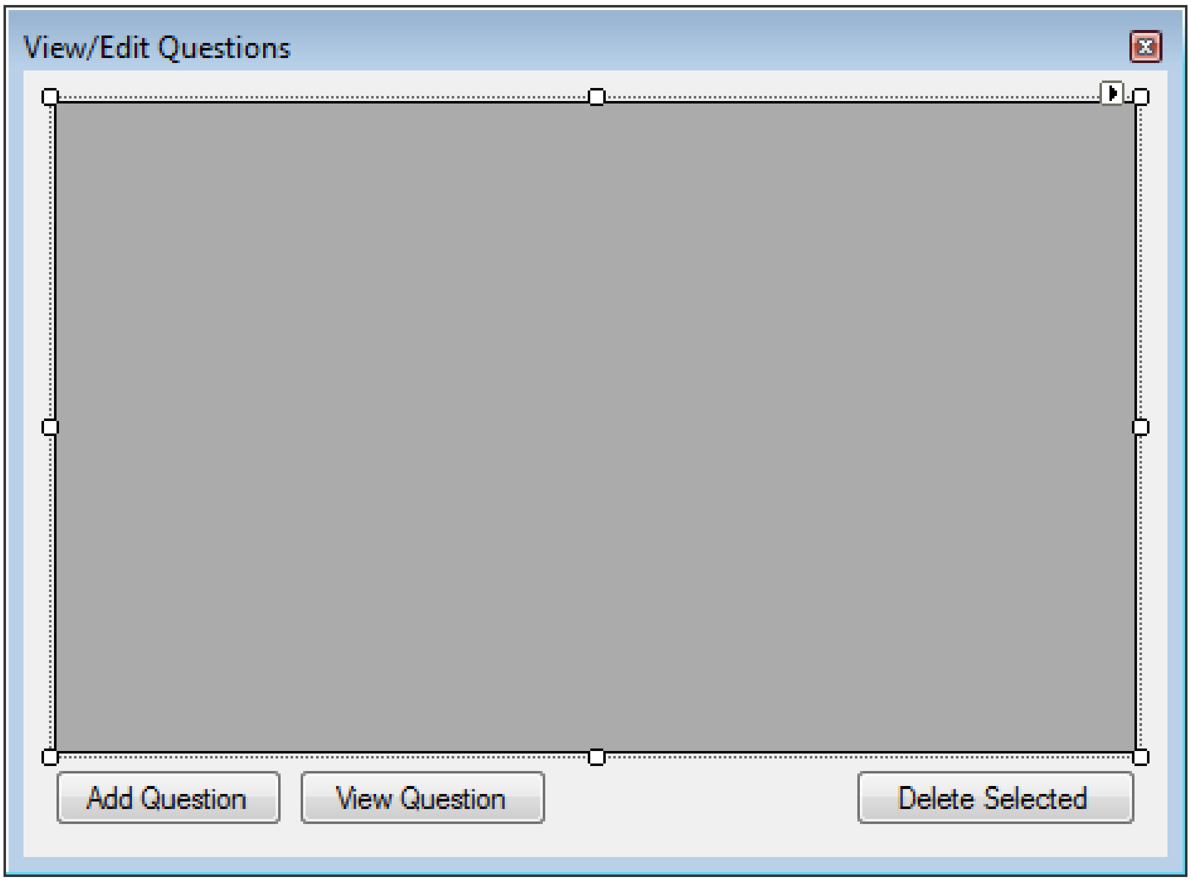
Add Exam Question Form

The form used to add exam questions will use dropdown lists, allowing only set values to be used to create the question name, for example all exam questions must be from the months January or June. A numeric up/down selector will be used for the year, as this allows only numbers to be entered, and the range to be specified. As values are changed, the question name is automatically updated and displayed using a Label to the user.



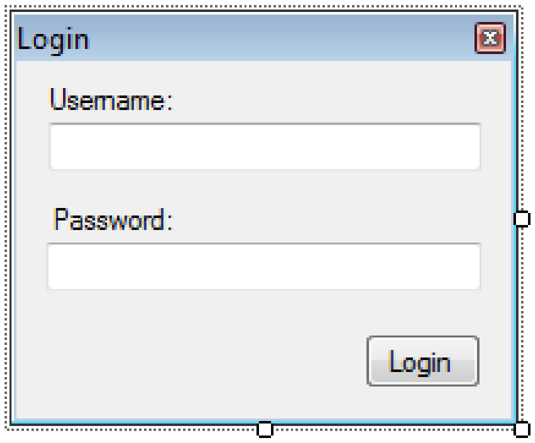
Edit Questions Form

Information about current questions can be viewed and modified by teachers; the data will be presented using a DataGridView. It will feature a button which shows a ContextMenuStrip to allow the user to select the type of question they wish to add and show the appropriate form. A Button will be used to open a form that shows the question image.



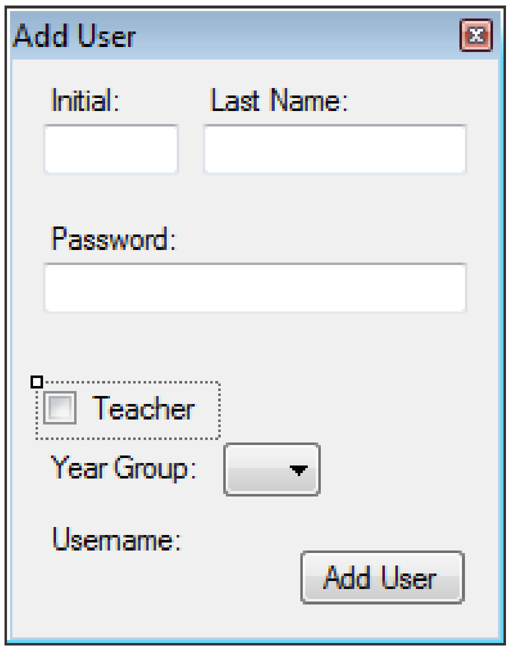
Login Form

The form used for logins will use 2 TextBoxes, one for username and one for password, it will also feature a label to show the type of user for the username inputted as it is typed (admin, student, or if no user found with that username, none). The password TextBox will use password characters '*' instead of showing the originally typed characters.



Add User Form

The form used to add users will feature 2 TextBoxes for username and password, although the password will not be hidden as it is intended to be copied or printed to give to students, there will be a checkbox which is used to mark the user as a teacher, this disables setting the year group when checked.

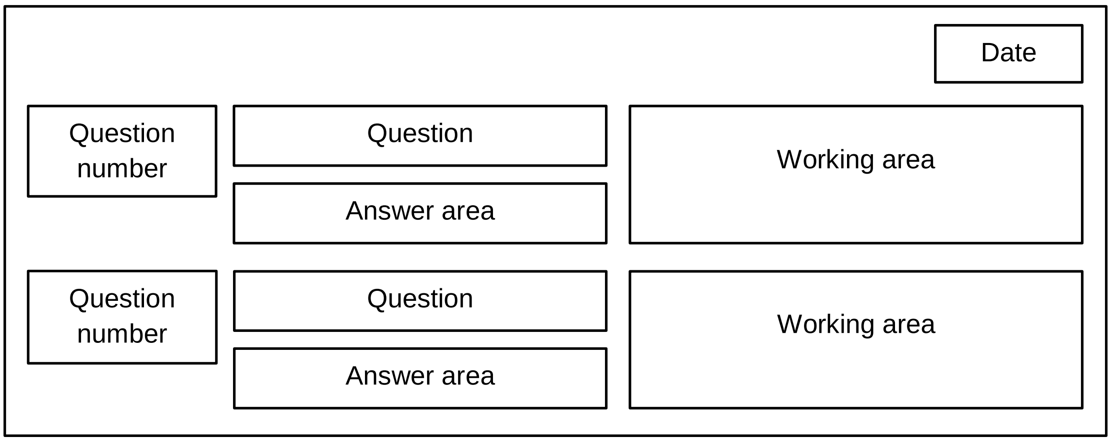


Questions Printed Output

Question images are able to be printed out to allow them to be completed on paper before the user enters their answer into the software.

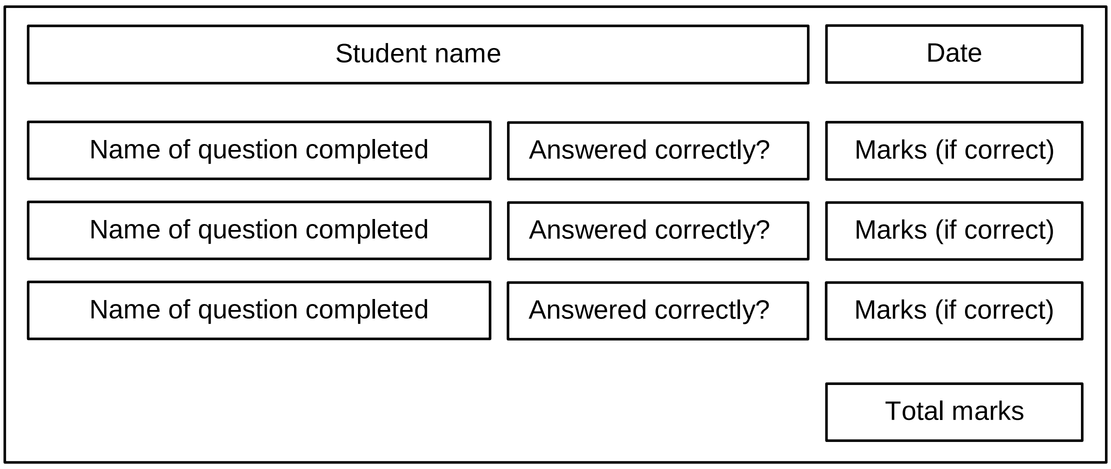
Random Questions Printed Output

After generating a number of random questions, these can be printed out by students to complete for example to practice, or by teachers to students to complete for example as homework. Each generated question will be added with its number to the printout with space for working for each and an answer area if it is to be marked manually, the current date will be printed on the page.



Student Progress Printed Output

Teachers are able to view the progress of each individual student through a printed output of which questions they have completed and the total number of marks earned, this will show the students name and the names of the questions that they have completed, with the number of marks earned if answered correctly, and a total number of marks for questions completed correctly. This allows teachers to find problems with specific questions for each student.



Algorithms for Data Transformation

Description: Creates a quiz with a given number of questions
Pseudocode:

```
PROGRAM GenerateQuiz: Parameter NumberOfQuestions
      QuizName←Input “Name of Quiz?”
      QuizDateTime←CurrentDateTime
      Insert QuizName,QuizDateTime into Quiz
      QuizID←Select QuizID from Quiz
      RandomIDList←Select NumberOfQuestions random QuestionIDs from
Questions
      Count←0
      WHILE Count<NumberOfQuestions:
            Insert QuizID, RandomIDList(Count), Count as QuestionIndex
            into QuizQuestions
            Count←Count+1
      ENDWHILE
END
```

Description: Checks that user is valid (exists) and password is correct to allow user to log in, checks whether the user is a teacher or not
Pseudocode:

```
PROGRAM Login:
      Username←Input “Username?”
      UserID←Select UserID from Users where username=Username
      IF UserID exists
            YearGroup←Select YearGroup from Users where userID=UserID
            IF YearGroup = 0
                  TeacherUser←true
            ELSE
                  TeacherUser←false
            ENDIF
            UserPassword←Select Password from Users where userID=UserID

            IF Input “Password?” = UserPassword
                  LoginSuccessful←true
            ELSE
                  Output “Incorrect password”
            ENDIF
      ELSE
            Output “Username not recognized”
      ENDIF
END
```

Description: Check that the users answer to a question is correct
Pseudocode:

```
PROGRAM CheckAnswer: Parameter QuestionID
      UserAnswer←Input “Answer”
      IF UserAnswer is number
            QuestionAnswer←Select QuestionAnswer from Questions where 
questionID=QuestionID
            IF UserAnswer = QuestionAnswer
                  AnswerCorrect←true
                  QuestionMarks←Select QuestionMarks from Questions where 
questionID=QuestionID
                  MarksAchieved←MarksAchieved+QuestionMarks
            ELSE
                  AnswerCorrect←False
            ENDIF
      ELSE
            Output “Not a number”
      ENDIF
END
```

Description: Adds an Exam Question to the database
Pseudocode:

```
PROGRAM AddExamQuestion:
      Month←ComboBoxMonthText
      Year←NumericUpDownYearText last 2 digits
      QuestionNumber←NumericUpDownQuestionNumberText
      QuestionPart1←ComboBoxPart1Text
      QuestionPart2←ComboBoxPart2Text
      QuestionPart3←ComboBoxPart3Text
      QuestionName←Month+Year+”_”+QuestionNumber+QuestionPart1

      IF QuestionPart2 not blank
            QuestionName←QuestionName+QuestionPart2
      ENDIF
      IF QuestionPart2 not blank
            QuestionName←QuestionName+QuestionPart3
      ENDIF

      QuestionImage←TextBoxImageLocationText to image
      QuestionAnswer←TextBoxAnswerText
      QuestionMarks←TextBoxMarksText
      Insert into Questions (questionImage, questionName, questionAnswer, 
questionMarks) values (QuestionImage, QuestionName, QuestionAnswer, 
QuestionMarks)
END
```

Description: Change a user’s password
Pseudocode:

```
PROGRAM ChangePassword: Parameter UserID
      NewPassword←TextBoxNewPasswordText
      ConfirmPassword←TextBoxConfirmPasswordText
      IF NewPassword = ConfirmPassword
            CurrentPassword←Select password from Users where
userID=UserID
            IF TextBoxCurrentPasswordText = CurrentPassword
                  Update Users set password=NewPassword where
usedID=UserID
            ELSE
                  Output “Incorrect current password”
            ENDIF
      ELSE
            Output “Passwords do not match”
      ENDIF
END
```

Sample of Planned SQL Queries

Data Manipulation Language

SELECT

Description: Selects details (ID, Image, Name, Answer, Marks) of a question

SQL: `SELECT (questionID, questionImage, questionName, questionAnswer, questionMarks) FROM Questions WHERE questionID=?`

Description: Selects details of all questions completed by a user, by joining the tables and finding which questions have been completed for each quiz the user has completed, finding the details for these questions, and sorts the results by the quiz ID in descending order

SQL: `SELECT (quizID, questionName, questionAnswer, userAnswer, questionMarks) FROM UserQuestion JOIN Questions ON UserQuestion.questionID=Questions.questionID WHERE userID=? ORDER BY quizID DESC`

Description: Selects all details for a user

SQL: `SELECT * FROM Users WHERE userID=?`

Description: Selects the number of quizzes the user has completed by the number of unique quiz IDs that are recorded for that user

SQL: `SELECT COUNT(DISTINCT quizID) FROM UserQuestion WHERE userID=?`

Description: Selects the ID for the last created quiz

SQL: `SELECT MAX(quizID) FROM Quiz`

INSERT

Description: Adds a new question with its details (Image, Name, Answer, Marks)

SQL: `INSERT INTO Questions (questionImage, questionName, questionAnswer, questionMarks) VALUES (?,?,?,?)`

Description: Adds a new user with its details (Username, Password, Year Group)

SQL: `INSERT INTO Users (username, password, yearGroup) VALUES (?,?,?)`

UPDATE

Description: Updates the Password for a user

SQL: `UPDATE Users SET password=? WHERE userID=?`

DELETE

Description: Deletes a user

SQL: `DELETE from Users WHERE userID=?`

Data Definition Language

Description: Creates Users table with Username, Password, Year Group, and ID columns

SQL:
```
CREATE TABLE Users (
      username        varchar(50)      not null,
      password        varchar(50)      not null,
      yearGroup       int              not null,
      userID          int              PRIMARY KEY
);
```

Description: Creates QuizQuestions table which links a number of questions to each quiz, it uses foreign keys for Quiz ID and Question ID, and stores a question index for the order of each question in the quiz

SQL:
```
CREATE TABLE Quiz (
      quizID            int               PRIMARY KEY,
      questionID        int               PRIMARY KEY,
      questionIndex     int               not null
);
```

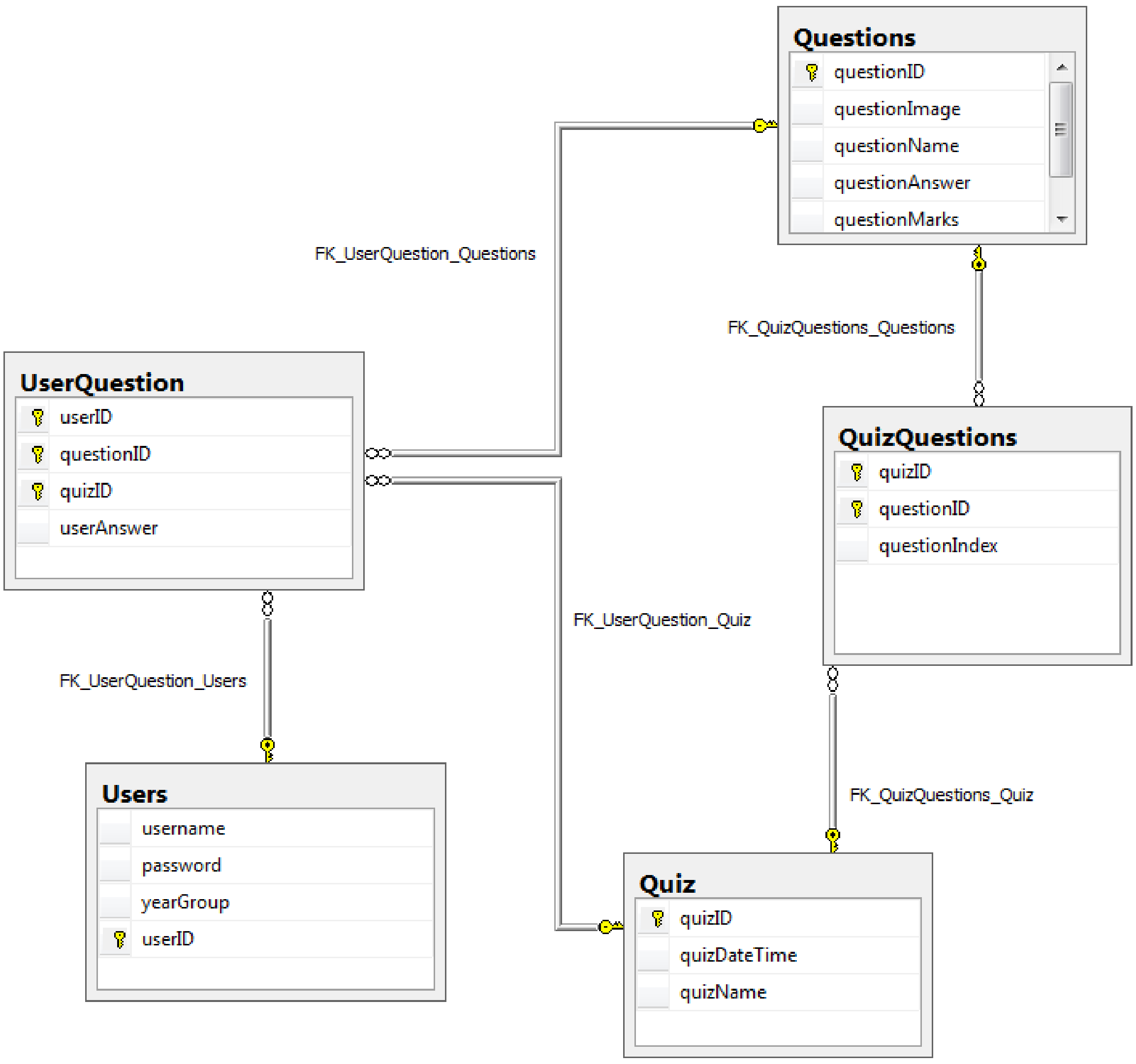
Identification of Storage Media

The program can be distributed as a single executable file with an approximate size of 1MB, question and user information will be stored in a database online, so do not need to be provided with the executable file. While the program is running, only a few variables will be used to keep track of the current index of question and quiz number, which are used with the database to for example check answers.

The software will be distributed in a way that means that students are able to keep their own copy on USB flash drives, which will be able to run on Windows computers. An alternative distribution method will be being able to be downloaded through the school website – due to the small size of the file, the download time should be very short, allowing the program to be downloaded to and run from any Windows computer with an internet connection quickly.

Database Design

The normalised design of the database and its relationships are shown in the diagram below.



Class Definitions

N/A

Security and Integrity of Data

Security

Each student will use a separate user account with a password, in order to use the program; a user must be logged in with a valid account. Student accounts are unable to access all the features of the program, such to manage users and questions, these are hidden from these users – teachers are able to access all features of the program.

Integrity

Data input will be validated when it is input to ensure that the correct data types are added to the database. The database is designed using referential integrity, using primary key and foreign key relationships, ensuring that integrity is maintained by preventing accidental deletion of related data and ensures that updates to the tables are synchronised.

System Security

SQL parameterization will be used in communication with the database, this prevents SQL injection which could expose the database to being modified or potentially deleted. The database will be stored on a password protected server. Users will be encouraged to never share their passwords, and passwords can be changed by the same user account or by teachers if necessary.

Overall Test Strategy

Black-box testing

This form of testing will often be used to check that subroutines that generate values work correctly, for example manually finding the answer to a random question using the values in the question, to ensure that the answer found manually is the same as the computer calculated value. It can also be used for example to find that a correct number of quiz questions is generated and that none are repeated.

White-box testing

Testing will be used to ensure that comparison checks work correctly, such as ensuring that a username exists, and if it does the password is correct – if it is not correct something else should happen. Another example of the use of this testing form would be to ensure that values used in loops such as while loops do not cause errors, such as checking that there are enough questions in the database for the quiz before erroneously looping infinitely when there are not enough questions found to meet the criteria.

Trace tables

Some of the more complex algorithms will be tested by manually running through them with different values and recording the results, ensuring that the result is as expected, for example finding the part of the question that needs to be solved and running through the equation used to calculate the value of a random question's answer to compare the result with the expected result.

Test Plan

1. Interface
   - 1.1. Interface flow
     - 1.1.1. MenuBar
       - 1.1.1.1. Opens correct forms
       - 1.1.1.2. Prints correct data from correct form
       - 1.1.1.3. Displays correct username, user type and year group (if applicable)
       - 1.1.1.4. Allows correct functionality based on user type
   - 1.2. Information layout
     - 1.2.1. Buttons navigate correctly
     - 1.2.2. Items move/scale correctly with window
     - 1.2.3. Images shown within PictureBox correctly
     - 1.2.4. Printed output
       - 1.2.4.1. Question images scale correctly
       - 1.2.4.2. Correct layout of items
       - 1.2.4.3. Displays message if nothing to print
       - 1.2.4.4. Multiple pages printed if necessary
2. Unit testing
   - 2.1. Black-box
     - 2.1.1. Message for maximum number of existing questions to use
     - 2.1.2. Question information displayed correctly
     - 2.1.3. Correct output if answer is correct/incorrect
   - 2.2. White-box
     - 2.2.1. Current question
       - 2.2.1.1. Correct index for current question in quiz
       - 2.2.1.2. Correct question shown for current index in quiz
     - 2.2.2. SQL executed correctly
       - 2.2.2.1. Only checked items deleted from table when requested
       - 2.2.2.2. SQL parameters handled correctly
       - 2.2.2.3. SELECT returns correct data
     - 2.2.3. SUVAT questions generated correctly
       - 2.2.3.1. Question is valid
       - 2.2.3.2. Answer calculated correctly
3. Error handling
   - 3.1. Invalid input
     - 3.1.1. Numerical only answers
     - 3.1.2. NumericUpDown minimum and maximum
     - 3.1.3. Only allow predefined ComboBox items
     - 3.1.4. Password minimum length
     - 3.1.5. Username maximum length
   - 3.2. SQL connection
     - 3.2.1. Displays correct message if cannot connect

Technical Solution

See appendix

System Testing

Test Table

Test Number	Description	Data Type	Expected result	Pass/Fail
1.1.1.1	The items within the menu bar open the correct forms	typical	The correct forms are opened	Pass
1.1.1.2	If currently focused form can be printed from, the menu item will print from the current form, otherwise an error will be shown	typical	When printing from the 'Quiz' form the question image will be printed	Pass
1.1.1.3	The correct username, type and year group is shown for the current user in the top right of the MenuBar	typical	Username for 'student' is shown with type student, year group 12	Pass
1.1.1.4	The correct functionality is accessible through the MenuBar based on the type of user	typical	When logged in as 'teacher', settings is shown in MenuBar	Pass
1.2.1	Buttons within the forms will trigger the correct event	typical	When the 'Login' button the user will be logged in if the details are correct	Pass
1.2.2	When the size of the window is changed the form controls adjust automatically	typical	The window cannot be made too small, so there is enough room for controls	Pass
1.2.3	Images automatically fit the PictureBox and are scaled with it	typical	If the form is made wider the image width stays the same, if taller the image height stays the same	Pass
1.2.4.1	The question image is scaled correctly to fit on the printed page	typical	There will be a fixed width for all questions regardless of their size, height will adjust automatically	Pass
1.2.4.2	Items within the printout are in the correct places	extreme	With (maximum) 20 random questions, they all fit onto the page in the correct places, and date is in correct place	Pass

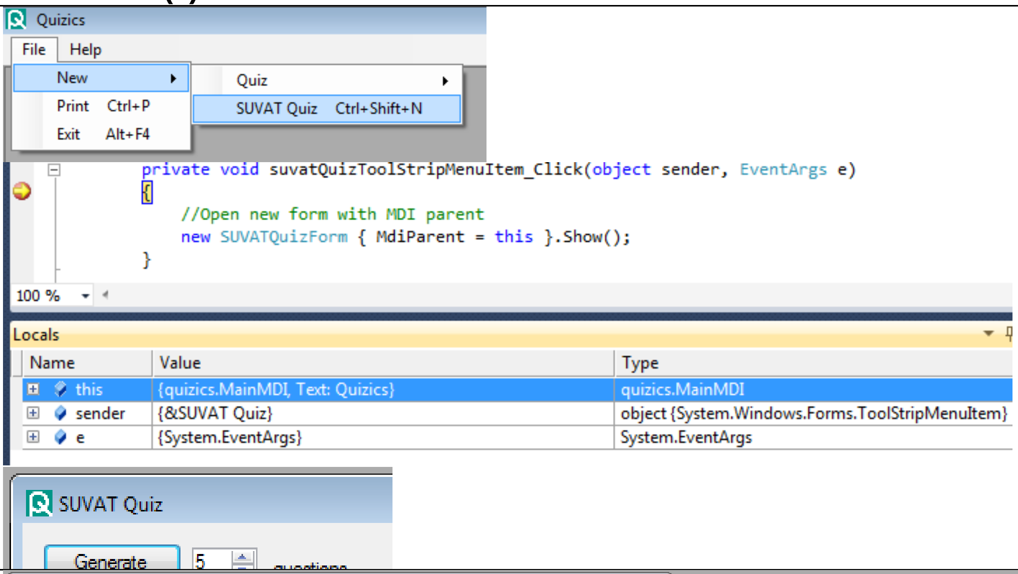
Test Number	Description	Data Type	Expected result	Pass/Fail
1.2.4.3	If no data exists to be printed (e.g. no questions/progress), a message is shown	extreme	A message is shown when 0 random questions are attempted to be printed	Pass
1.2.4.4	If there is more data to be printed than can fit on one page, multiple pages are used automatically	extreme	When there is a large amount of user progress data, multiple pages are used to print automatically, otherwise 1 page is used	Pass
2.1.1	If less questions exist in the database than the number of questions requested for the quiz, the number is reduced and a message shows this change	extreme	If a quiz with 20 questions is requested, and only 10 questions exist in the database, the quiz is reduced to 10 questions and the message states this	Pass
2.1.2	The information about a question is displayed correctly and in the correct places, including its image and name	typical	During a quiz, question 1 should show as question 1, with its name below the image and its image in the PictureBox	Pass
2.1.3	If a question is answered correctly or incorrectly, the appropriate outcome is shown (i.e. red for incorrect, green for correct)	typical	For question ‘Jan13_3a’, the answer is 3, so when a value other than 3 is entered it will show as incorrect (red), and when correct, green.	Pass
2.2.1.1	The question index increments correctly when the question number changes (i.e. when going to next question)	typical	When on question 3 (index 2), the next question increments the index to 3 and shows the updated question number (3+1) in the label	Pass

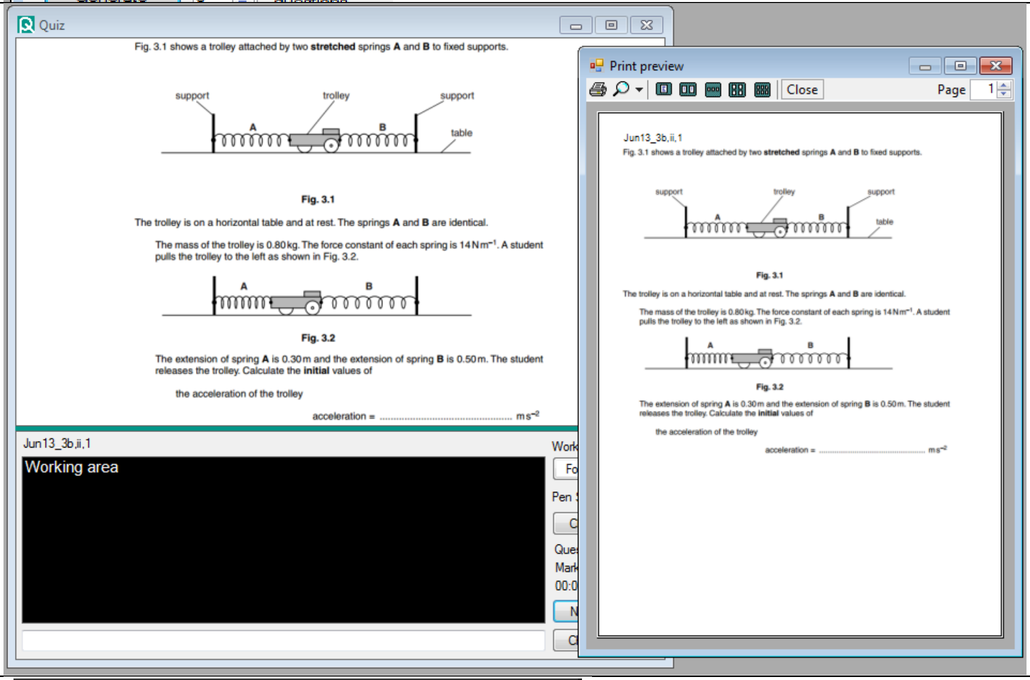
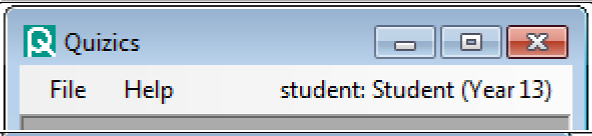
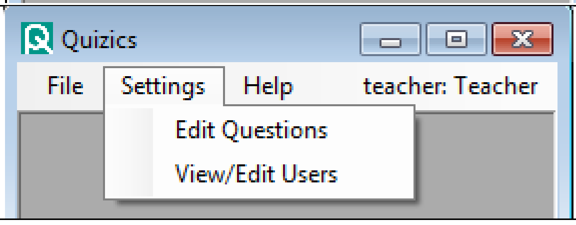
Test Number	Description	Data Type	Expected result	Pass/Fail
2.2.1.2	The appropriate question is shown for the current index in the quiz, (i.e. displayed in correct order)	typical	Question 2 (index 1) in quiz 279 should display with the details of question index 1 (‘Jan13_2b,1’)	Pass
2.2.2.1	When a delete request is made for questions or users, only the items which are checked are deleted from the database	typical	When only users ‘aaaaaaaaa’ and ‘ccccccccc’ are selected for deletion, no other users are deleted	Pass
2.2.2.2	SQL statements use parameters correctly, and the parameters are assigned from the correct data	typical	When creating a user, the username, password and year group are used as parameters with the correct data for each for the SQL statement	Pass
2.2.2.3	When getting data from the database, the correct data is returned	typical	When the user ID is selected from the database for the username ‘student’, 28 is returned	Pass
2.2.3.1	The SUVAT question that is generated must be valid and possible to calculate	erroneous	When solving for t, the question must contain (v, u, & a) or (s, u, & v), if none of these conditions are met (e.g. question contains (s, u, & a) it cannot be solved for, so answer is returned as 0	Pass

Test Number	Description	Data Type	Expected result	Pass/Fail
2.2.3.2	The correct formula is used for when calculating the answer to a question based on which part is unknown and which part is being solved for, and the answer calculated is correct for this formula	typical	When calculating an answer with parts (u, v, & t), to find s, the equation (s=((u+v)/2)*t) is used and the answer is calculated correctly e.g. ((88+80)/2)*19= 1596	Pass
3.1.1	Answers are checked to ensure that they are numerical, if not, the correct message is shown	erroneous	When entering text as an answer, a message is shown	Pass
3.1.2	A minimum and maximum value is set for NumericUpDown controls so that they only allow values in the intended range	extreme	If a value of 100 is entered for generating a random quiz, the value is reset to the maximum (20) automatically	Pass
3.1.3	ComboBoxes should use a DropDownList style to ensure that only items pre-programmed can be used	typical	In the add exam questions form, the last part of the question only allows blank, 1, 2, 3, or 4, the user cannot type their own value	Pass
3.1.4	Passwords must be longer than 8 characters	erroneous	When a password of 7 characters is given a message is shown	Pass
3.1.5	Usernames must be shorter than 12 characters	erroneous	A username of >12 characters cannot be entered due to the maximum length of the textbox	Pass
3.2.1	Whenever connecting to the database, if a connection error occurs it is handled and the error message is shown	erroneous	If a user attempts to log in while not connected to the internet the correct error message is shown	Pass

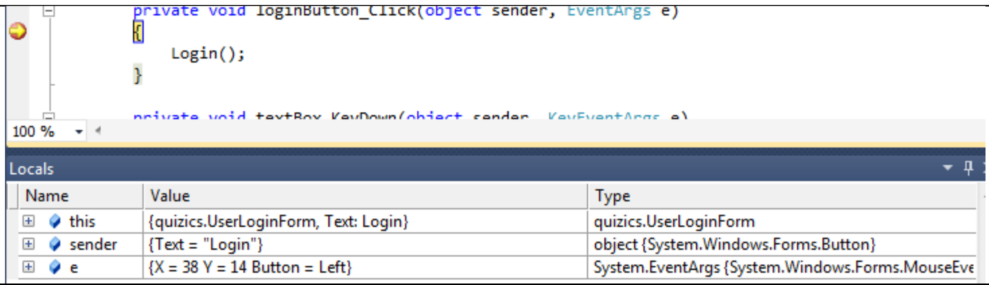
Screenshots

Test	Screenshot(s)

1.1.1.1	 Quizics
	File Help
	New ▸ / Quiz ▸ / SUVAT Quiz Ctrl+Shift+N
	Print Ctrl+P
	Exit Alt+F4
	`private void suvatQuizToolStripMenuItem_Click(object sender, EventArgs e)`
	`{`
	`    //Open new form with MDI parent`
	`    new SUVATQuizForm { MdiParent = this }.Show();`
	`}`
	100 %
	Locals
	Name | Value | Type
	this | {quizics.MainMDI, Text: Quizics} | quizics.MainMDI
	sender | {&SUVAT Quiz} | object {System.Windows.Forms.ToolStripMenuItem}
	e | {System.EventArgs} | System.EventArgs
	SUVAT Quiz
	Generate 5 questions

1.1.1.2	 Quiz Fig. 3.1 shows a trolley attached by two **stretched** springs **A** and **B** to fixed supports. **Fig. 3.1** The trolley is on a horizontal table and at rest. The springs **A** and **B** are identical. The mass of the trolley is 0.80 kg. The force constant of each spring is 14 N m^{-1}. A student pulls the trolley to the left as shown in Fig. 3.2. **Fig. 3.2** The extension of spring **A** is 0.30 m and the extension of spring **B** is 0.50 m. The student releases the trolley. Calculate the **initial** values of the acceleration of the trolley acceleration = ... m s^{-2} Jun13_3b,ii,1 Working area Print preview Close Page 1 Jun13_3b,ii,1
1.1.1.3	 Quizics File Help student: Student (Year 13)
1.1.1.4	 Quizics File Settings Help teacher: Teacher Edit Questions View/Edit Users

1.2.1



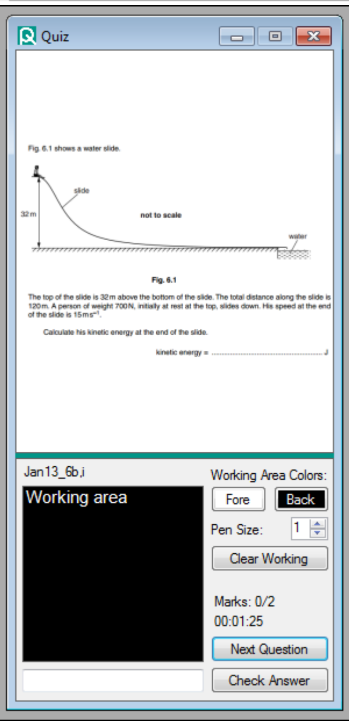
```
private void loginButton_Click(object sender, EventArgs e)
{
    Login();
}

private void textBox_KeyDown(object sender, KeyEventArgs e)
```

100 %

Locals

Name	Value	Type
this	{quizics.UserLoginForm, Text: Login}	quizics.UserLoginForm
sender	{Text = "Login"}	object {System.Windows.Forms.Button}
e	{X = 38 Y = 14 Button = Left}	System.EventArgs {System.Windows.Forms.MouseEve

1.2.2



Quiz

Fig. 6.1 shows a water slide.

Fig. 6.1

The top of the slide is 32m above the bottom of the slide. The total distance along the slide is 120m. A person of weight 700N, initially at rest at the top, slides down. His speed at the end of the slide is 15ms⁻¹.

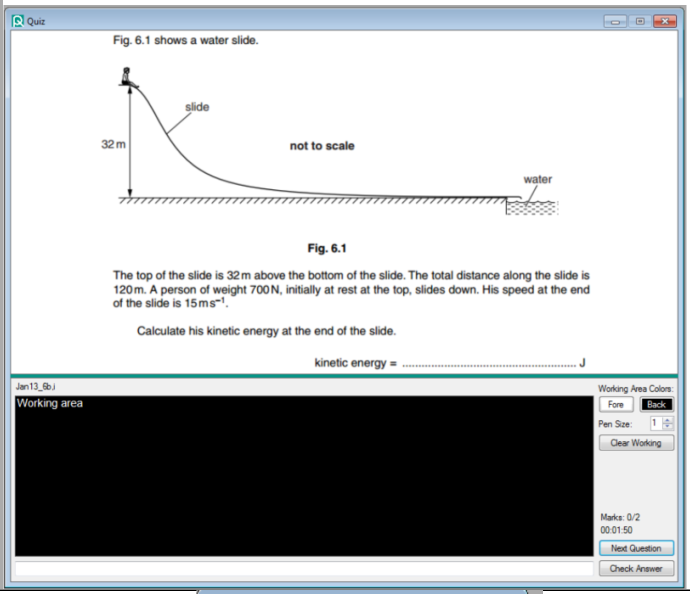
Calculate his kinetic energy at the end of the slide.

kinetic energy = ... J

Jan13_6b.i

Marks: 0/2

00:01:25



Quiz

Fig. 6.1 shows a water slide.

Fig. 6.1

The top of the slide is 32 m above the bottom of the slide. The total distance along the slide is 120 m. A person of weight 700 N, initially at rest at the top, slides down. His speed at the end of the slide is 15 m s⁻¹.

Calculate his kinetic energy at the end of the slide.

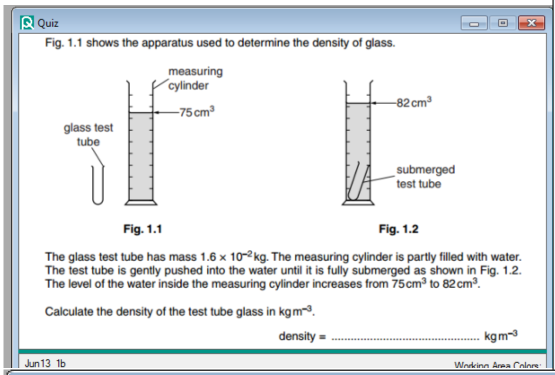
kinetic energy = ... J

Jan13_6b.i

Marks: 0/2

00:01:50

1.2.3



Quiz

Fig. 1.1 shows the apparatus used to determine the density of glass.

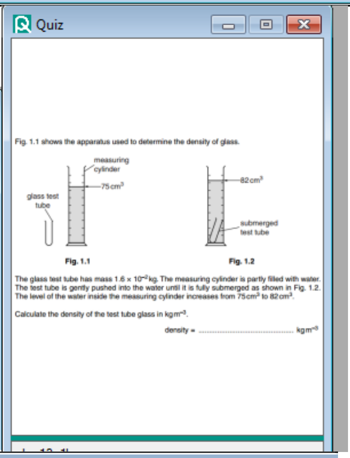
Fig. 1.1 Fig. 1.2

The glass test tube has mass 1.6×10^{-2} kg. The measuring cylinder is partly filled with water. The test tube is gently pushed into the water until it is fully submerged as shown in Fig. 1.2. The level of the water inside the measuring cylinder increases from 75 cm³ to 82 cm³.

Calculate the density of the test tube glass in kg m⁻³.

density = ... kg m⁻³

Jun13_1b



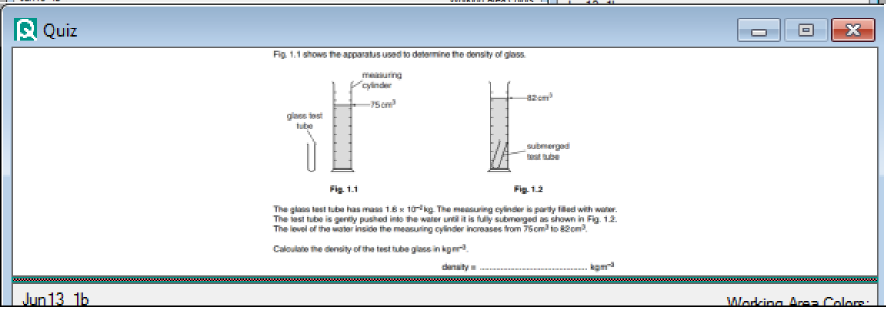
Quiz

Fig. 1.1 shows the apparatus used to determine the density of glass.

The glass test tube has mass 1.6×10^{-2} kg. The measuring cylinder is partly filled with water. The test tube is gently pushed into the water until it is fully submerged as shown in Fig. 1.2. The level of the water inside the measuring cylinder increases from 75 cm³ to 82 cm³.

Calculate the density of the test tube glass in kg m⁻³.

density = ... kg m⁻³



Quiz

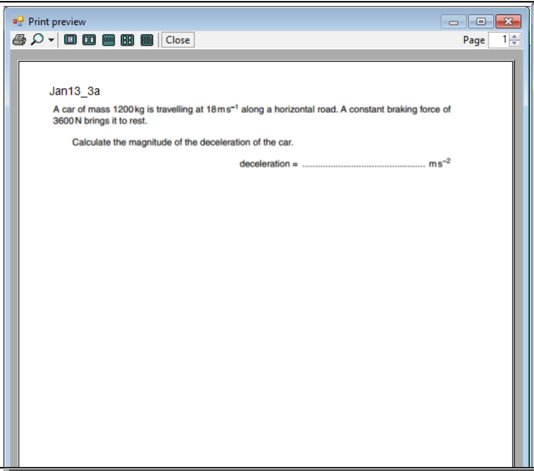
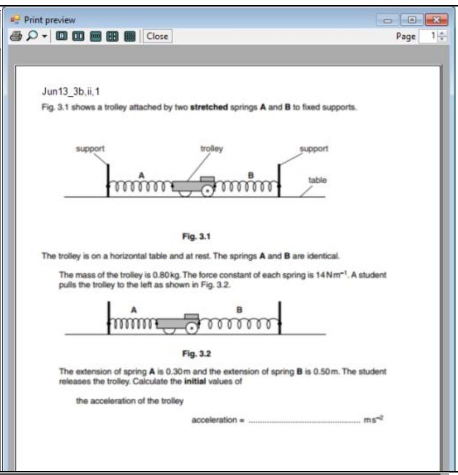
Fig. 1.1 shows the apparatus used to determine the density of glass.

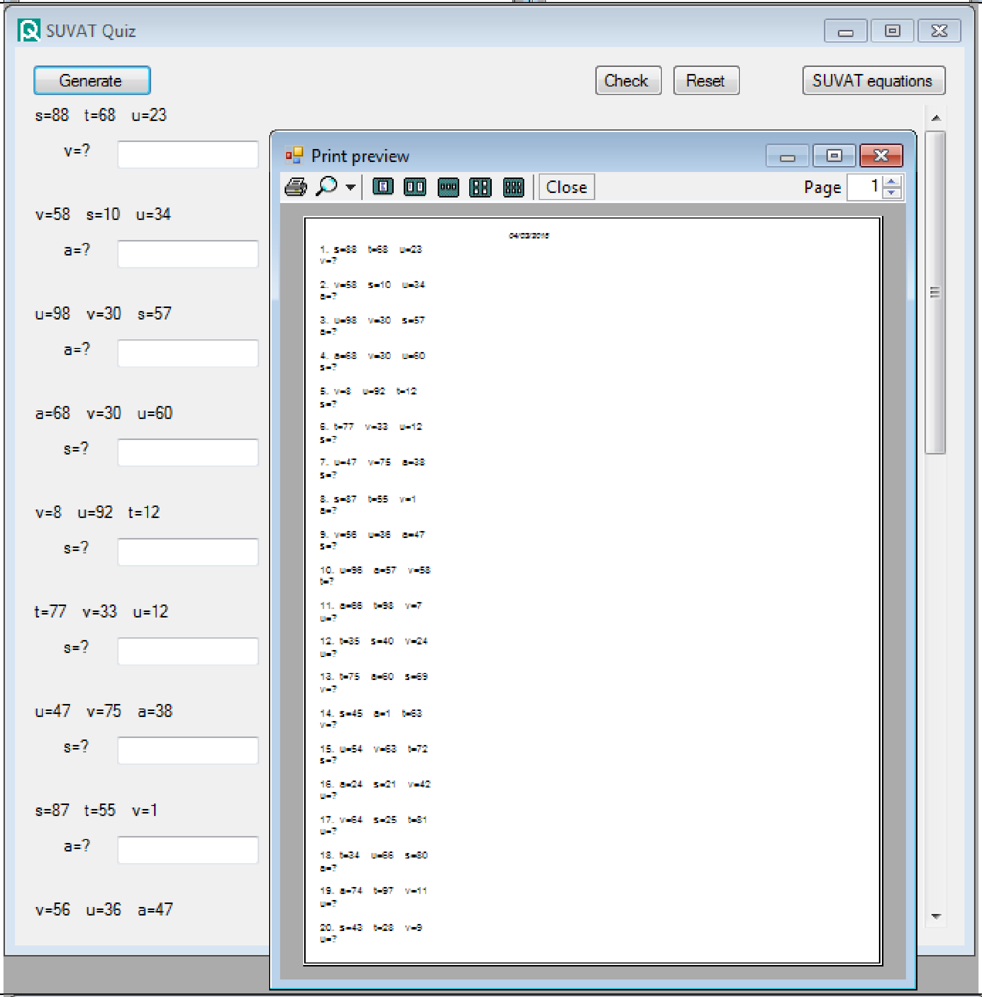
The glass test tube has mass 1.6×10^{-2} kg. The measuring cylinder is partly filled with water. The test tube is gently pushed into the water until it is fully submerged as shown in Fig. 1.2. The level of the water inside the measuring cylinder increases from 75 cm³ to 82 cm³.

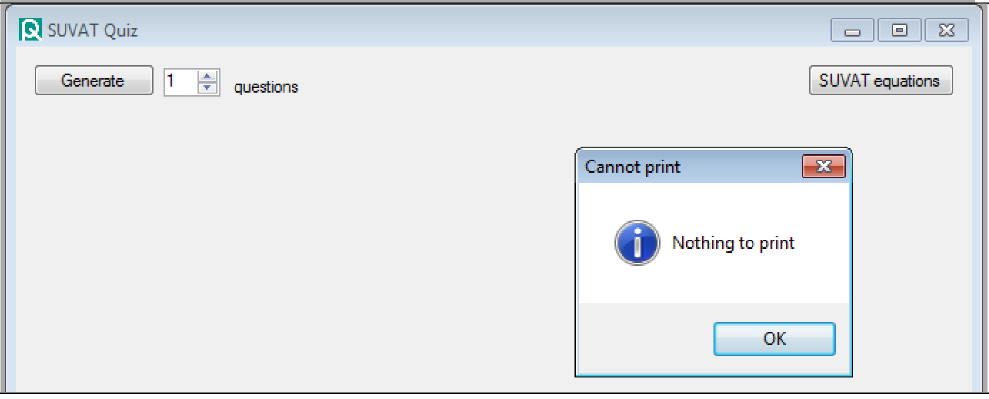
Calculate the density of the test tube glass in kg m⁻³.

density = ... kg m⁻³

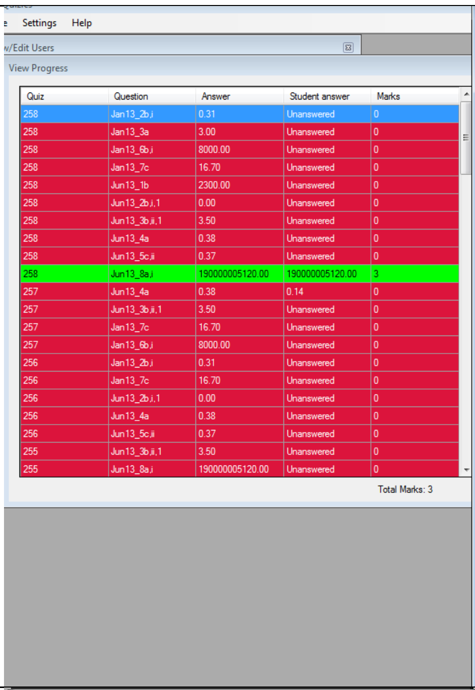
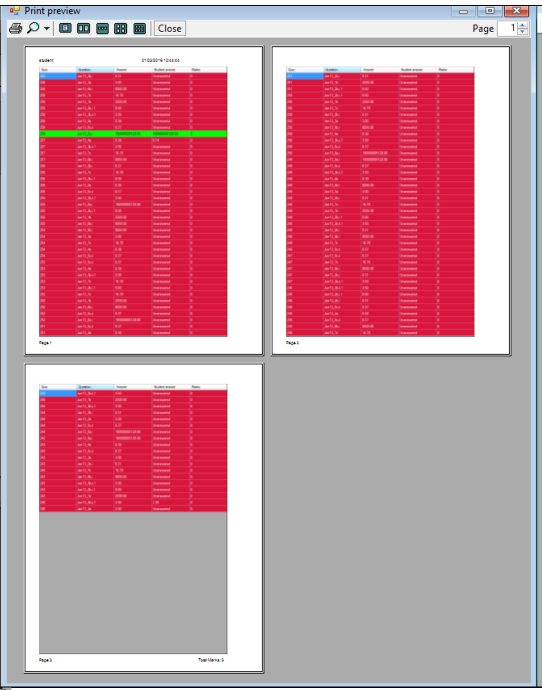
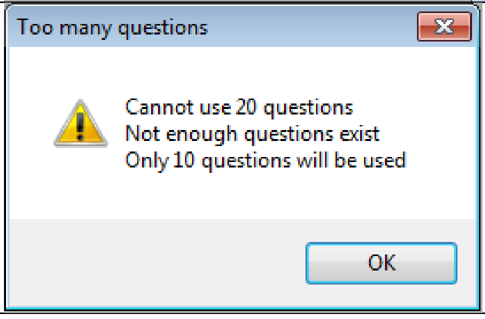
Jun13_1b

1.2.4.1	 Print preview Page 1 Jan13_3a A car of mass 1200 kg is travelling at 18 m s^{-1} along a horizontal road. A constant braking force of 3600 N brings it to rest. Calculate the magnitude of the deceleration of the car. deceleration = ... m s^{-2}  Print preview Page 1 Jun13_3b,ii,1 Fig. 3.1 shows a trolley attached by two **stretched** springs **A** and **B** to fixed supports. **Fig. 3.1** The trolley is on a horizontal table and at rest. The springs **A** and **B** are identical. The mass of the trolley is 0.80 kg. The force constant of each spring is 14 N m^{-1}. A student pulls the trolley to the left as shown in Fig. 3.2. **Fig. 3.2** The extension of spring **A** is 0.30 m and the extension of spring **B** is 0.50 m. The student releases the trolley. Calculate the **initial** values of the acceleration of the trolley acceleration = ... m s^{-2}

1.2.4.2	 SUVAT Quiz
	Generate · Check · Reset · SUVAT equations
	s=88 t=68 u=23
	v=?
	v=58 s=10 u=34
	a=?
	u=98 v=30 s=57
	a=?
	a=68 v=30 u=60
	s=?
	v=8 u=92 t=12
	s=?
	t=77 v=33 u=12
	s=?
	u=47 v=75 a=38
	s=?
	s=87 t=55 v=1
	a=?
	v=56 u=36 a=47
	Print preview
	Close · Page 1
	04/03/2016
	1. s=88 t=68 u=23
	v=?
	2. v=58 s=10 u=34
	a=?
	3. u=98 v=30 s=57
	a=?
	4. a=68 v=30 u=60
	s=?
	5. v=8 u=92 t=12
	s=?
	6. t=77 v=33 u=12
	s=?
	7. u=47 v=75 a=38
	s=?
	8. s=87 t=55 v=1
	a=?
	9. v=56 u=36 a=47
	s=?
	10. u=96 a=57 v=58
	t=?
	11. a=66 t=98 v=7
	u=?
	12. t=35 s=40 v=24
	u=?
	13. t=75 a=60 s=69
	v=?
	14. s=45 a=1 t=63
	v=?
	15. u=54 v=63 t=72
	s=?
	16. a=24 s=21 v=42
	u=?
	17. v=64 s=25 t=81
	u=?
	18. t=34 u=66 s=80
	a=?
	19. a=74 t=97 v=11
	u=?
	20. s=43 t=28 v=9
	u=?

1.2.4.3	 SUVAT Quiz Generate · 1 questions · SUVAT equations Cannot print Nothing to print OK

System Testing

1.2.4.4	 
2.1.1	

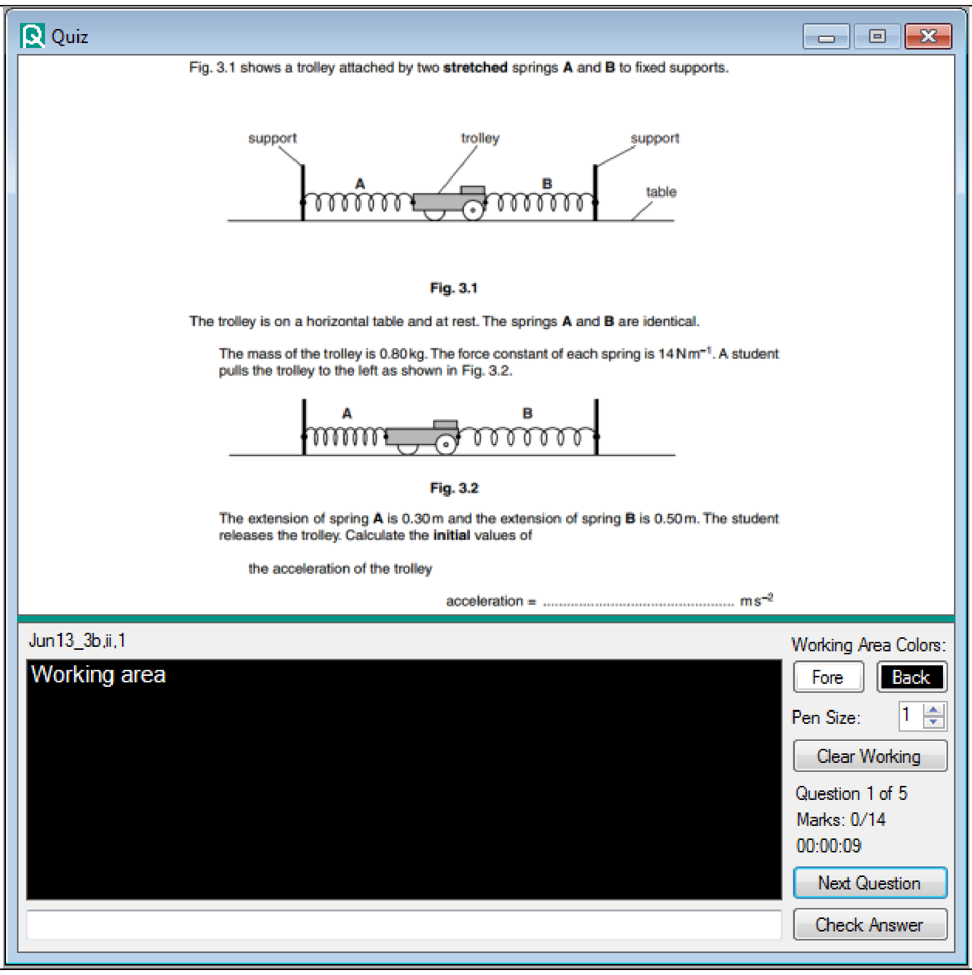
Quiz	Question	Answer	Student answer	Marks
258	Jan13_2b.i	0.31	Unanswered	0
258	Jan13_3a	3.00	Unanswered	0
258	Jan13_6b.i	8000.00	Unanswered	0
258	Jan13_7c	16.70	Unanswered	0
258	Jun13_1b	2300.00	Unanswered	0
258	Jun13_2b.i.1	0.00	Unanswered	0
258	Jun13_3b.ii.1	3.50	Unanswered	0
258	Jun13_4a	0.38	Unanswered	0
258	Jun13_5c.ii	0.37	Unanswered	0
258	Jun13_8a.i	190000005120.00	190000005120.00	3
257	Jun13_4a	0.38	0.14	0
257	Jun13_3b.ii.1	3.50	Unanswered	0
257	Jan13_7c	16.70	Unanswered	0
257	Jan13_6b.i	8000.00	Unanswered	0
256	Jan13_2b.i	0.31	Unanswered	0
256	Jan13_7c	16.70	Unanswered	0
256	Jun13_2b.i.1	0.00	Unanswered	0
256	Jun13_4a	0.38	Unanswered	0
256	Jun13_5c.ii	0.37	Unanswered	0
255	Jun13_3b.ii.1	3.50	Unanswered	0
255	Jun13_8a.i	190000005120.00	Unanswered	0

Total Marks: 3

Too many questions

Cannot use 20 questions
Not enough questions exist
Only 10 questions will be used

2.1.2



Quiz

Fig. 3.1 shows a trolley attached by two **stretched** springs **A** and **B** to fixed supports.

Fig. 3.1

The trolley is on a horizontal table and at rest. The springs **A** and **B** are identical.

The mass of the trolley is 0.80 kg. The force constant of each spring is 14 N m^{-1}. A student pulls the trolley to the left as shown in Fig. 3.2.

Fig. 3.2

The extension of spring **A** is 0.30 m and the extension of spring **B** is 0.50 m. The student releases the trolley. Calculate the **initial** values of

the acceleration of the trolley

acceleration = ... m s^{-2}

Jun13_3b,ii,1

Working area

Working Area Colors: Fore Back

Pen Size: 1

Clear Working

Question 1 of 5

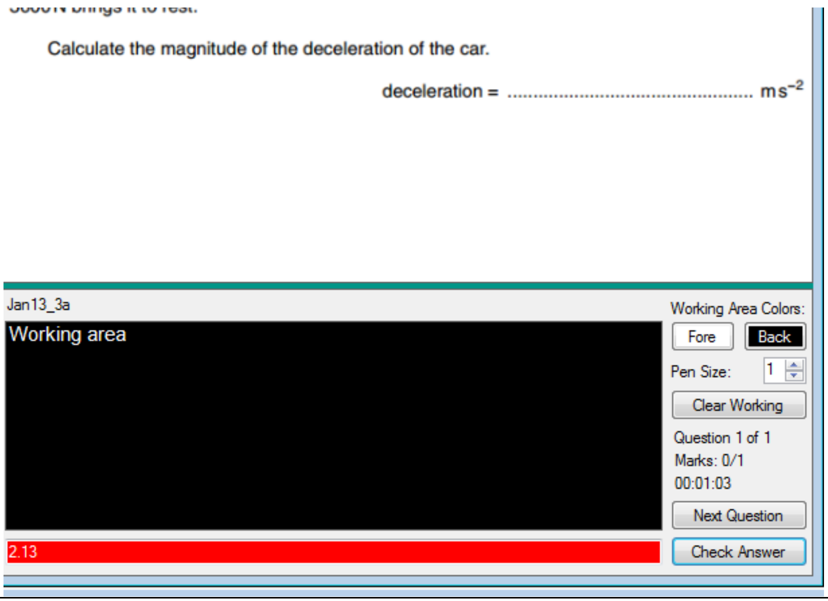
Marks: 0/14

00:00:09

Next Question

Check Answer

2.1.3



3000 N brings it to rest.

Calculate the magnitude of the deceleration of the car.

deceleration = ... m s^{-2}

Jan13_3a

Working area

2.13

Working Area Colors: Fore Back

Pen Size: 1

Clear Working

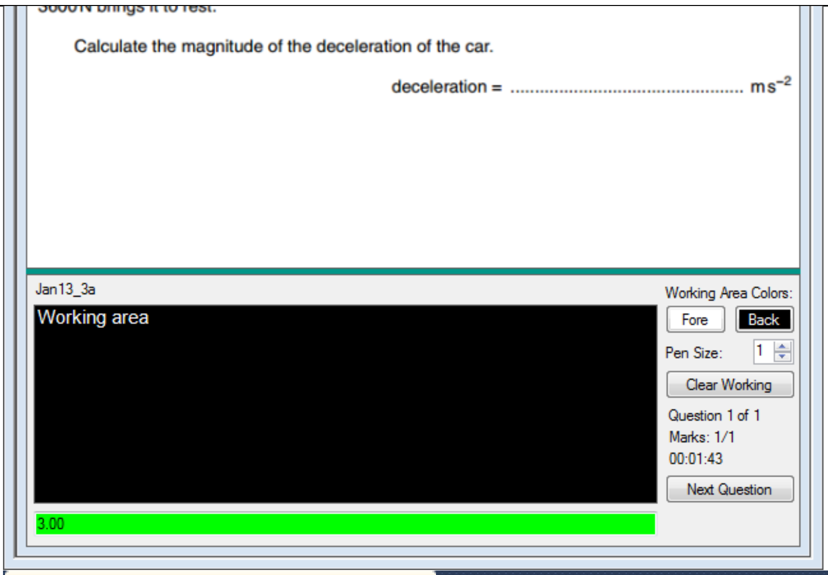
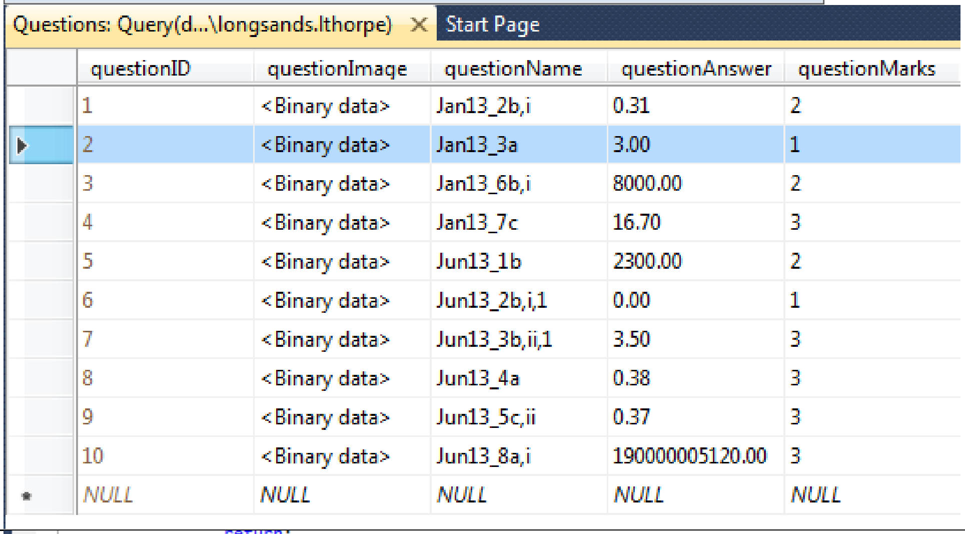
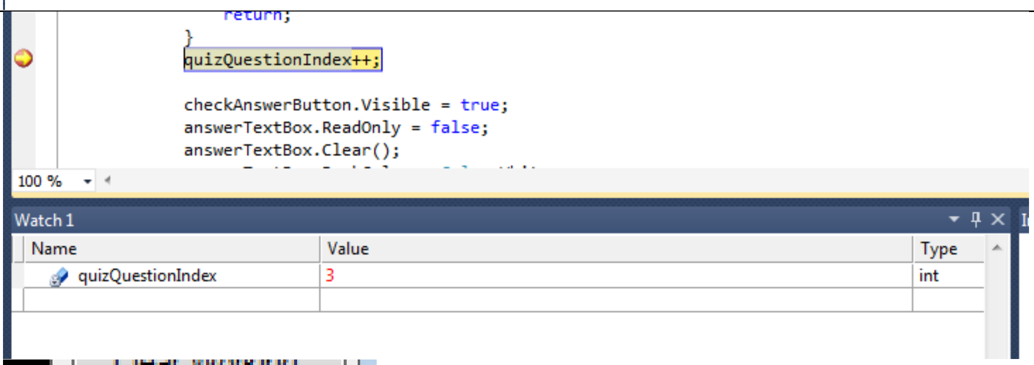
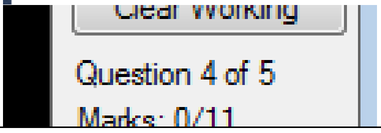
Question 1 of 1

Marks: 0/1

00:01:03

Next Question

Check Answer

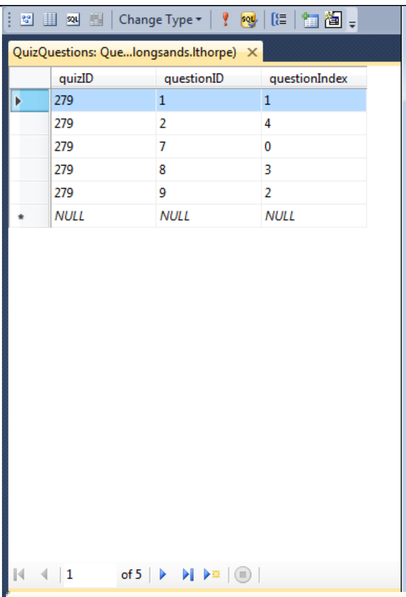
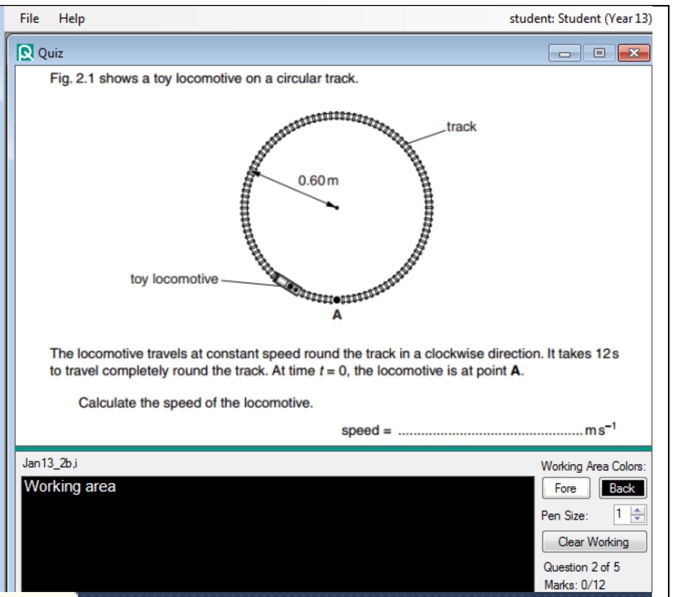
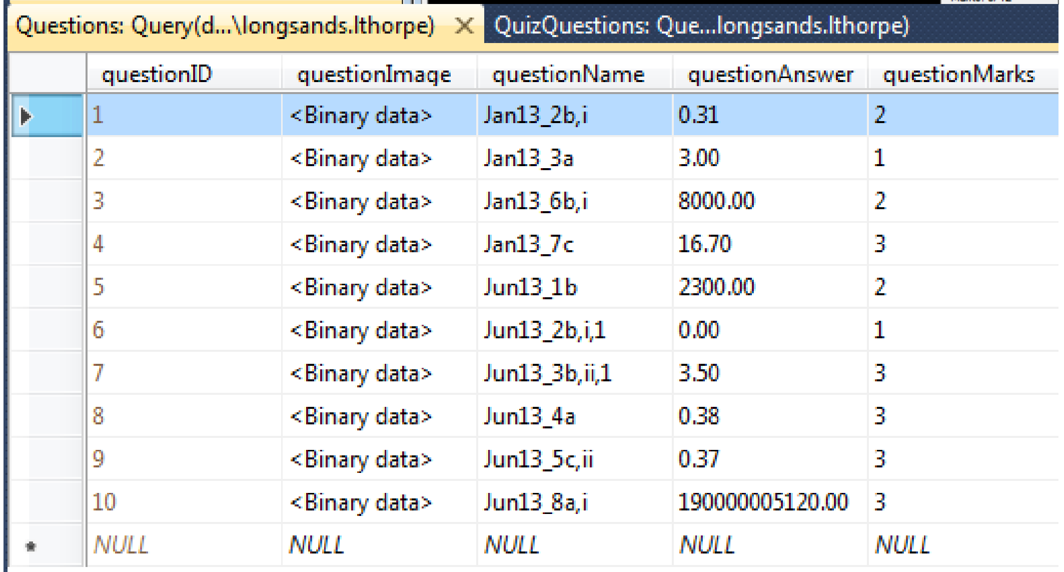
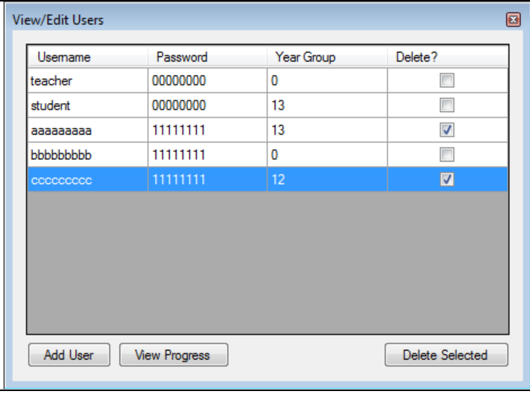
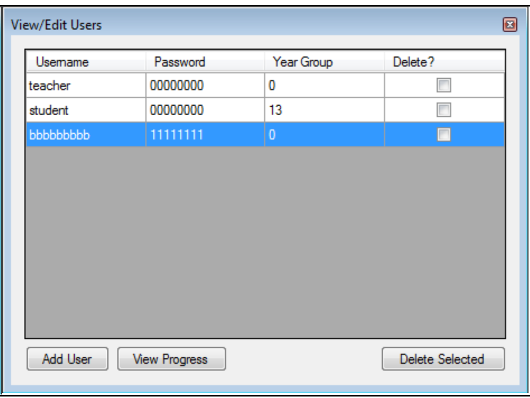
	 Calculate the magnitude of the deceleration of the car. deceleration = ... ms^{-2} Jan13_3a Working area Working Area Colors: Fore Back Pen Size: 1 Clear Working Question 1 of 1 Marks: 1/1 00:01:43 Next Question 3.00  Questions: Query(d...\longsands.lthorpe) Start Page
2.2.1.1	  Question 4 of 5 Marks: 0/11

questionID	questionImage	questionName	questionAnswer	questionMarks
1	<Binary data>	Jan13_2b,i	0.31	2
2	<Binary data>	Jan13_3a	3.00	1
3	<Binary data>	Jan13_6b,i	8000.00	2
4	<Binary data>	Jan13_7c	16.70	3
5	<Binary data>	Jun13_1b	2300.00	2
6	<Binary data>	Jun13_2b,i,1	0.00	1
7	<Binary data>	Jun13_3b,ii,1	3.50	3
8	<Binary data>	Jun13_4a	0.38	3
9	<Binary data>	Jun13_5c,ii	0.37	3
10	<Binary data>	Jun13_8a,i	190000005120.00	3
NULL	NULL	NULL	NULL	NULL

```
                    return;
                }
                quizQuestionIndex++;

                checkAnswerButton.Visible = true;
                answerTextBox.ReadOnly = false;
                answerTextBox.Clear();
```

Watch 1

Name	Value	Type
quizQuestionIndex	3	int

2.2.1.2	  
2.2.2.1	 

Change Type

QuizQuestions: Que...longsands.lthorpe)

quizID	questionID	questionIndex
279	1	1
279	2	4
279	7	0
279	8	3
279	9	2
NULL	NULL	NULL

1 of 5

File Help student: Student (Year 13)

Quiz

Fig. 2.1 shows a toy locomotive on a circular track.

The locomotive travels at constant speed round the track in a clockwise direction. It takes 12 s to travel completely round the track. At time $t = 0$, the locomotive is at point **A**.

Calculate the speed of the locomotive.

speed = ... ms^{-1}

Jan13_2b,i

Working area

Working Area Colors: Fore Back

Pen Size: 1

Clear Working

Question 2 of 5

Marks: 0/12

Questions: Query(d...\longsands.lthorpe) QuizQuestions: Que...longsands.lthorpe)

questionID	questionImage	questionName	questionAnswer	questionMarks
1	<Binary data>	Jan13_2b,i	0.31	2
2	<Binary data>	Jan13_3a	3.00	1
3	<Binary data>	Jan13_6b,i	8000.00	2
4	<Binary data>	Jan13_7c	16.70	3
5	<Binary data>	Jun13_1b	2300.00	2
6	<Binary data>	Jun13_2b,i,1	0.00	1
7	<Binary data>	Jun13_3b,ii,1	3.50	3
8	<Binary data>	Jun13_4a	0.38	3
9	<Binary data>	Jun13_5c,ii	0.37	3
10	<Binary data>	Jun13_8a,i	190000005120.00	3
NULL	NULL	NULL	NULL	NULL

View/Edit Users

Username	Password	Year Group	Delete?
teacher	00000000	0	☐
student	00000000	13	☐
aaaaaaaaa	11111111	13	☑
bbbbbbbbb	11111111	0	☐
ccccccccc	11111111	12	☑

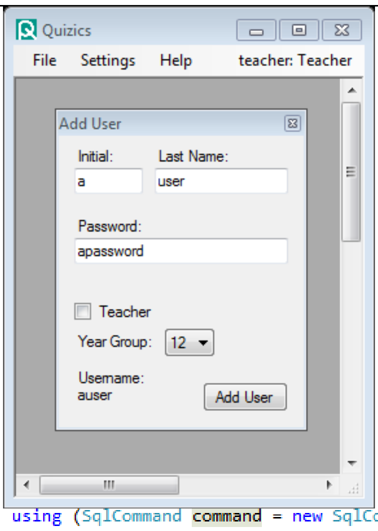
Add User View Progress Delete Selected

View/Edit Users

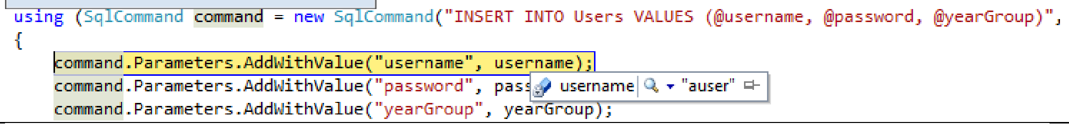
Username	Password	Year Group	Delete?
teacher	00000000	0	☐
student	00000000	13	☐
bbbbbbbbb	11111111	0	☐

Add User View Progress Delete Selected

2.2.2.2

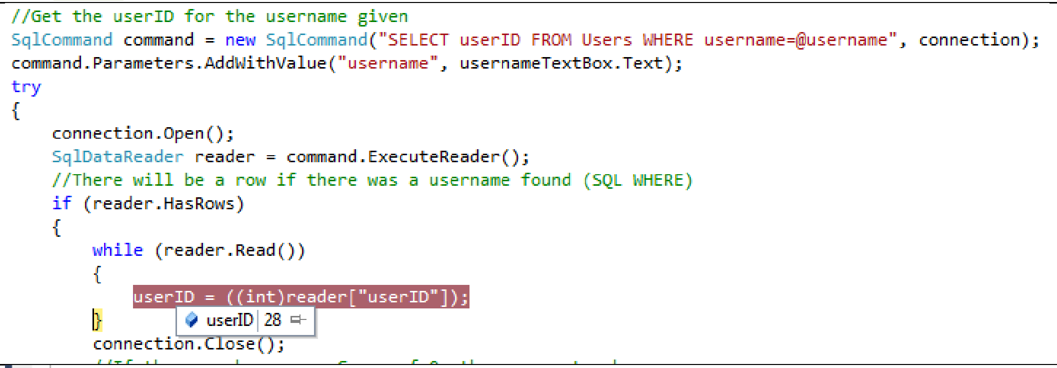


Initial: a
Last Name: user
Password: apassword
☐ Teacher
Year Group: 12
Username: auser



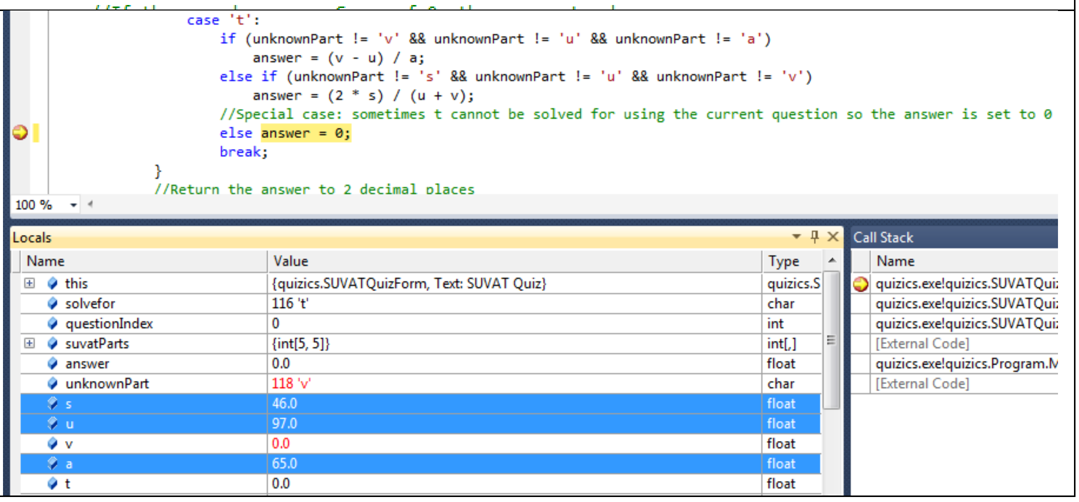
```
using (SqlCommand command = new SqlCommand("INSERT INTO Users VALUES (@username, @password, @yearGroup)",
{
    command.Parameters.AddWithValue("username", username);
    command.Parameters.AddWithValue("password", pass  username  "auser"
    command.Parameters.AddWithValue("yearGroup", yearGroup);
```

2.2.2.3



```
//Get the userID for the username given
SqlCommand command = new SqlCommand("SELECT userID FROM Users WHERE username=@username", connection);
command.Parameters.AddWithValue("username", usernameTextBox.Text);
try
{
    connection.Open();
    SqlDataReader reader = command.ExecuteReader();
    //There will be a row if there was a username found (SQL WHERE)
    if (reader.HasRows)
    {
        while (reader.Read())
        {
            userID = ((int)reader["userID"]);
        }      userID 28
    connection.Close();
```

2.2.3.1



```
                case 't':
                    if (unknownPart != 'v' && unknownPart != 'u' && unknownPart != 'a')
                        answer = (v - u) / a;
                    else if (unknownPart != 's' && unknownPart != 'u' && unknownPart != 'v')
                        answer = (2 * s) / (u + v);
                    //Special case: sometimes t cannot be solved for using the current question so the answer is set to 0
                    else answer = 0;
                    break;
            }
            //Return the answer to 2 decimal places
```

100 %

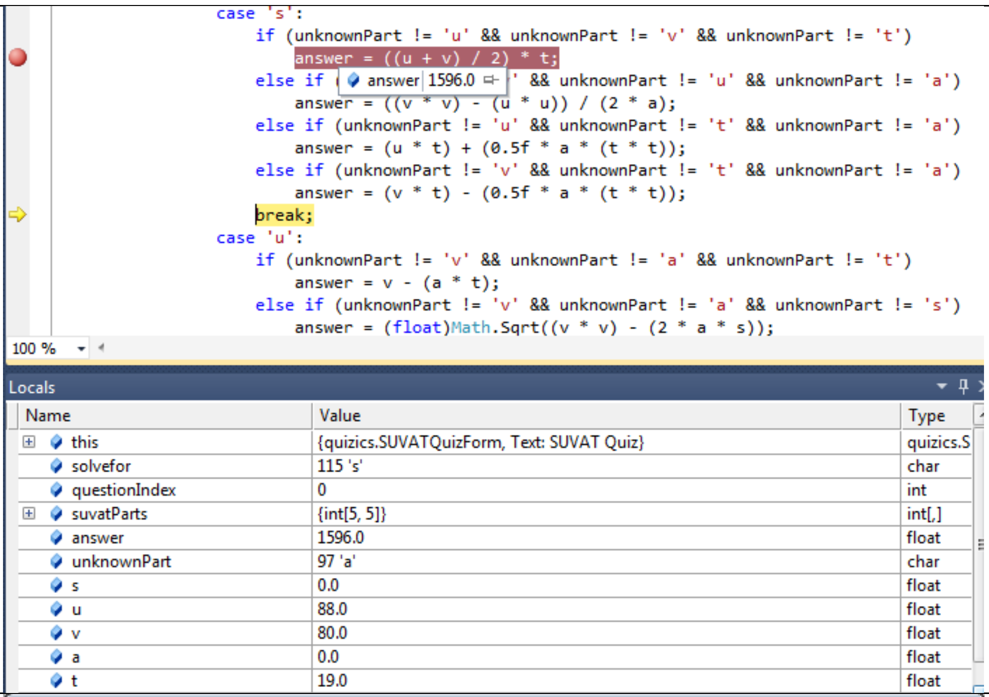
Locals

Name	Value	Type
this	{quizics.SUVATQuizForm, Text: SUVAT Quiz}	quizics.S
solvefor	116 't'	char
questionIndex	0	int
suvatParts	{int[5, 5]}	int[,]
answer	0.0	float
unknownPart	118 'v'	char
s	46.0	float
u	97.0	float
v	0.0	float
a	65.0	float
t	0.0	float

Call Stack

Name
quizics.exe!quizics.SUVATQui:
quizics.exe!quizics.SUVATQui:
quizics.exe!quizics.SUVATQui:
[External Code]
quizics.exe!quizics.Program.M
[External Code]

2.2.3.2



```
            case 's':
                if (unknownPart != 'u' && unknownPart != 'v' && unknownPart != 't')
                    answer = ((u + v) / 2) * t;
                else if ( answer 1596.0 ' && unknownPart != 'u' && unknownPart != 'a')
                    answer = ((v * v) - (u * u)) / (2 * a);
                else if (unknownPart != 'u' && unknownPart != 't' && unknownPart != 'a')
                    answer = (u * t) + (0.5f * a * (t * t));
                else if (unknownPart != 'v' && unknownPart != 't' && unknownPart != 'a')
                    answer = (v * t) - (0.5f * a * (t * t));
                break;
            case 'u':
                if (unknownPart != 'v' && unknownPart != 'a' && unknownPart != 't')
                    answer = v - (a * t);
                else if (unknownPart != 'v' && unknownPart != 'a' && unknownPart != 's')
                    answer = (float)Math.Sqrt((v * v) - (2 * a * s));
```

100 %

Locals

Name	Value	Type
this	{quizics.SUVATQuizForm, Text: SUVAT Quiz}	quizics.S
solvefor	115 's'	char
questionIndex	0	int
suvatParts	{int[5, 5]}	int[,]
answer	1596.0	float
unknownPart	97 'a'	char
s	0.0	float
u	88.0	float
v	80.0	float
a	0.0	float
t	19.0	float

3.1.1

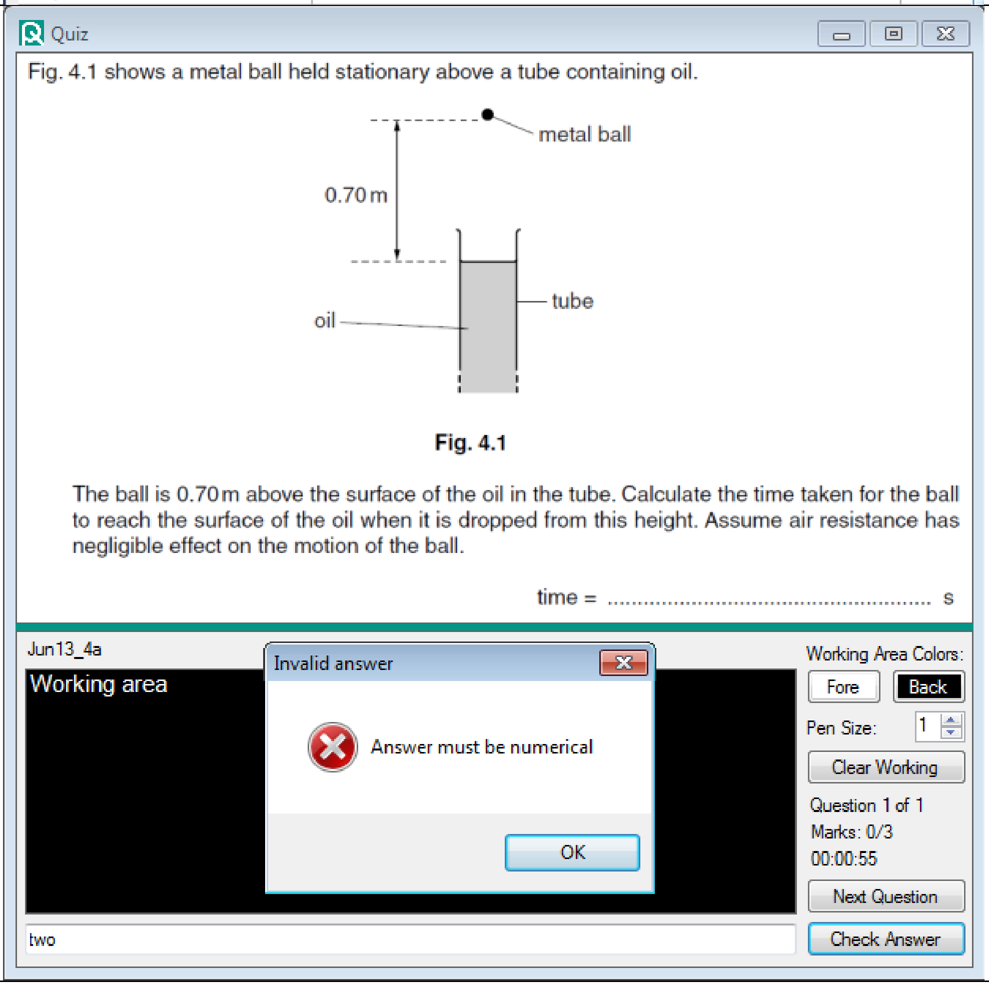


Fig. 4.1 shows a metal ball held stationary above a tube containing oil.

Fig. 4.1

The ball is 0.70 m above the surface of the oil in the tube. Calculate the time taken for the ball to reach the surface of the oil when it is dropped from this height. Assume air resistance has negligible effect on the motion of the ball.

time = ... s

Jun13_4a

Working area

Invalid answer

Answer must be numerical

Working Area Colors:

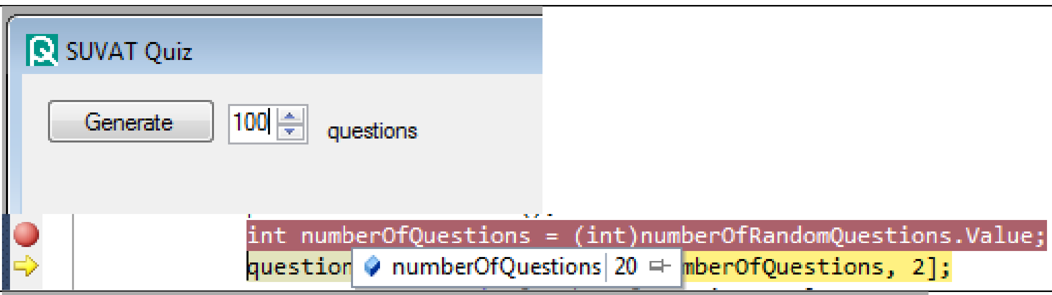
Pen Size: 1

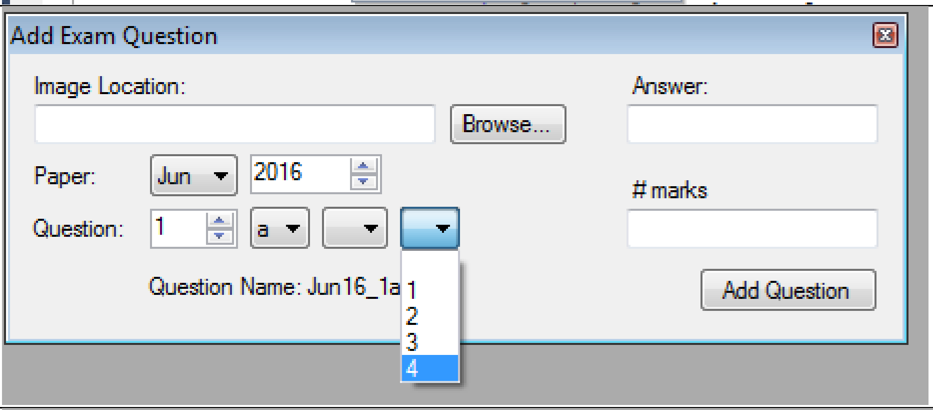
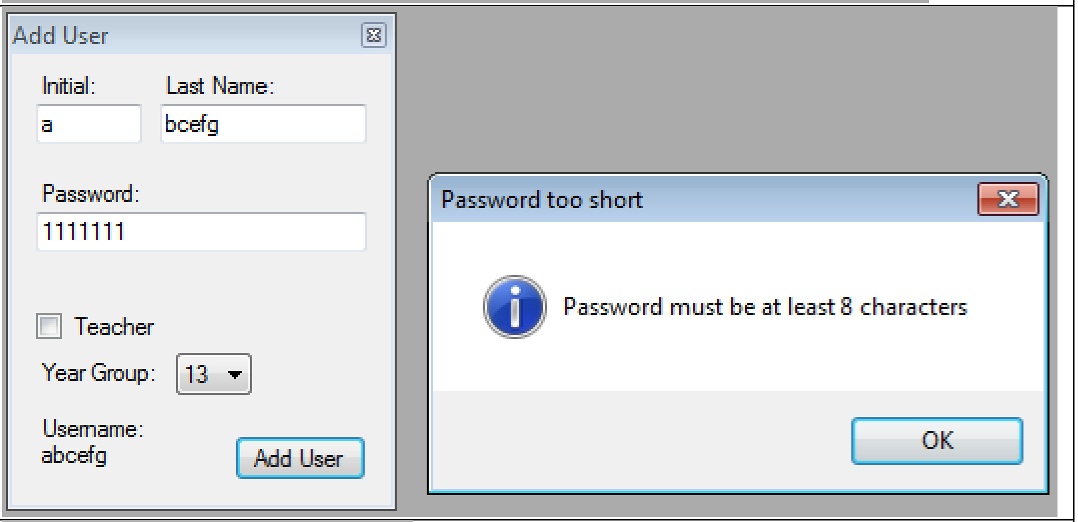
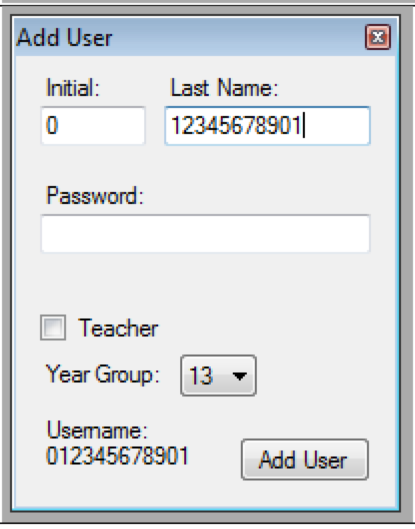
Question 1 of 1

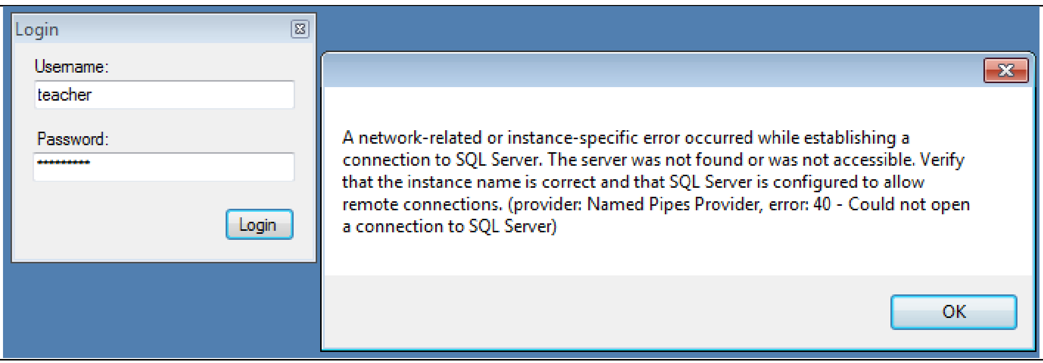
Marks: 0/3

00:00:55

two

3.1.2	 SUVAT Quiz
	Generate 100 questions
	`int numberOfQuestions = (int)numberOfRandomQuestions.Value;`
	`questior` numberOfQuestions 20 `mberOfQuestions, 2];`

3.1.3	 Add Exam Question Image Location: Browse... Answer: Paper: Jun 2016 # marks Question: 1 a 1 2 3 4 Question Name: Jun16_1a Add Question
3.1.4	 Add User Initial: a Last Name: bcefg Password: 1111111 Teacher Year Group: 13 Username: abcefg Add User Password too short Password must be at least 8 characters OK
3.1.5	 Add User Initial: 0 Last Name: 12345678901 Password: Teacher Year Group: 13 Username: 012345678901 Add User

3.2.1	 Login Username: teacher Password: •••••••••• Login A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: Named Pipes Provider, error: 40 - Could not open a connection to SQL Server) OK

User Manual

See appendix

System Maintenance

System Overview

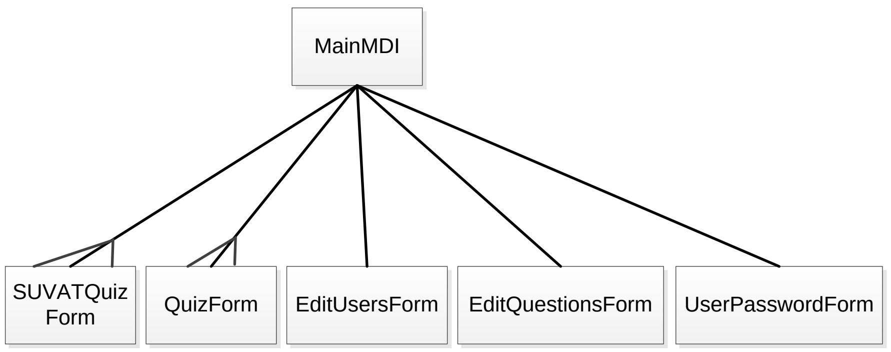
The functionality of the program is split between different forms, the ‘MainMDI’ form contains a MenuBar which is used to open forms and perform tasks such as printing. This modularity means that functionality can be changed; added or removed by modifying forms and changing the appropriate MenuBar items as necessary.

Forms

Form	Description
AddExamQuestionForm	Used to add exam questions to use with QuizForm, allows image to be uploaded along with question information: answer, marks, name.
AddTextbookQuestionForm	Used to add textbox question, works similarly to adding exam questions but name is comprised of different components.
AddUserForm	Used to add users to login and track progress for, username is initial, lastname, password must be 8 characters minimum, when a user is generated a MessageBox is shown to allow the details for the created user to be printed.
EditQuestionForm	Used to access Add(Exam|Textbook)QuestionForm, or to edit existing questions, data is pulled from database into DataGridView, questionID column hidden and image not requested from database until View Question button is clicked.
EditUsersForm	Same format as EditQuestionForm, used to access AddUserForm or to view existing users, userID column hidden. The View Progress button opens the UserProgressForm for the selected user.
MainMDI	(Multi Document Interface), used as the main window of the program, container for all other forms created and allows them to be moved together within one window, MenuBar at top used to open other forms or to print, exit. Opens minimised and starts by opening the login form – once logged in the userID of the user is stored in the variable userID of MainMDI to be accessed by other forms. When the user logged in is a teacher the Settings MenuBar item is visible, otherwise it is not visible.
QuizForm	The main quiz of the program, used with questions added by Add(Exam|Textbook)QuestionForm, or edited by EditQuestionForm, displays images for questions. Provides ‘working area’ which allows user to draw; this code is contained within a separate region. Opened with MenuBar in MainMDI which states how many (n) questions will be used, then requests (n) random questions if there are enough, otherwise shows MessageBox that there are not enough questions and reduces number automatically. When printing from the MenuBar the question image is scaled by width.

SUVATQuizForm	Different form of quiz, generates random SUVAT questions and displays them, stores the answer for each which allows user to complete the quiz interactively using the provided textbox, or alternatively the quiz can be printed using the print option in the MenuBar, which formats the quiz into a way that fits better on paper with the date at the top of the page.
UserLoginForm	Opened automatically when the program is run, used to enter the username and password for logging in, checks that username and password are valid and displays appropriate MessageBoxes if they are not, if the username and password are valid the userID is stored in the MainMDI userID variable.
UserPasswordForm	Used to edit the password of the currently logged in user, current password must be typed in correctly and new password must be typed in twice to ensure that there are no mistakes. The change makes the appropriate update in the database for the userID logged in.
UserProgressForm	Shows progress in the main quiz for a user, displays which questions they have completed on which quizIDs, the user's answer and the real answer for comparison, and if the user was correct – if the user was correct the row is highlighted green, otherwise it is highlighted red. Total marks is calculated and displayed in a label below the DataGridView. Data is displayed by descending quizID. Data is selected from UserQuestion joined with Questions where userID is the current logged in user When printing, the DataGridView is split into parts which are split across pages if there is more data than can fit on one page, the current date and username is added to the top, page numbers at the bottom left, and at the end of all pages the total marks is displayed.

Entity Relationship Diagram



The static Tools class contains code that is reused across other classes; it consists of the following functions:

Function	Parameters	Returns	Description
AlreadyOpenMessage	None	Void	Displays a MessageBox stating that a form already exists (for forms where only one should be open at a time)
ByteArrayToImage	Byte array	Image	Converts a byte array to an image and returns it
GetQuestionData	Int (questionID)	Dictionary	Returns a dictionary containing the data for the given questionID
GetUserData	Int (userID)	Dictionary	Returns dictionary containing user data for userID
ImageToByteArray	Image	Byte array	Converts an image to a byte array and returns it
Invert	Color	Color	Inverts a colour and returns it
IsFloat	String	Bool	Checks if string can be converted to float, returns true/false
IsInt	String	Bool	Returns true if can be converted to int
NonNumericAnswerMessage	None	None	Displays MessageBox stating that answer is not a number
NothingToPrintMessage	None	None	Displays MessageBox stating there is nothing to print
To2DP	Float	String	Rounds a float to 2 decimal places, converts to string and returns it
To2DP	String	String	Alternative method to convert string to 2 decimal places by converting to float first before rounding

Detailed Algorithm Design

Generate a random quiz, QuizForm

Pseudocode

```
PROGRAM GenerateQuiz: Parameter NumberOfQuestions
      QuizName←Input “Name of Quiz?”
      QuizDateTime←CurrentDateTime
      Insert QuizName,QuizDateTime into Quiz
      QuizID←Select QuizID from Quiz
      RandomIDList←Select NumberOfQuestions random QuestionIDs from
Questions
      Count←0
      WHILE Count<NumberOfQuestions:
            Insert QuizID, RandomIDList(Count), Count as QuestionIndex
            into QuizQuestions
            Count←Count+1
      ENDWHILE
END
```

Real Code

```
using (SqlConnection connection = new
SqlConnection(Tools.connectionString))
            {
                string quizName = "";
                try
                {
                    quizName =
(string)Tools.GetUserData(MainMDI.userID)["username"];
                }
                catch { return; }
                //Select the number of unique quizIDs for the user, to get
a count of number of quizzes completed
                using (SqlCommand command = new SqlCommand("SELECT
COUNT(DISTINCT quizID) FROM UserQuestion WHERE userID=@userID",
connection))
                {
                    command.Parameters.AddWithValue("userID",
MainMDI.userID);
                    try
                    {
                        connection.Open();
                        //The next quiz will the the (n+1) quiz so add 1
and add this number to the quiz name
                        quizName += (int)command.ExecuteScalar() + 1;
                        connection.Close();
                    }
                    catch (Exception ex) { MessageBox.Show(ex.Message); }
                    finally { connection.Close(); }
                }
                using (SqlCommand command = new SqlCommand("INSERT INTO
Quiz VALUES (@quizDateTime, @quizName)", connection))
                {
                    command.Parameters.AddWithValue("quizDateTime",
DateTime.Now);
                    command.Parameters.AddWithValue("quizName", quizName);
                    try
                    {
                        connection.Open();
                        command.ExecuteNonQuery();
                        connection.Close();
                    }
                    catch (Exception ex) { MessageBox.Show(ex.Message); }
                    finally { connection.Close(); }
                }
                //Get the last added quizID
                using (SqlCommand command = new SqlCommand("SELECT
MAX(quizID) FROM Quiz", connection))
                {
                    try
                    {
                        connection.Open();
                        quizID = (int)command.ExecuteScalar();
                        connection.Close();
                    }
                    catch (Exception ex) { MessageBox.Show(ex.Message); }
                    finally { connection.Close(); }
                }
                int questionIDCount = 0;
                //Get number of question IDs in database
                using (SqlCommand command = new SqlCommand("SELECT
COUNT(questionID) FROM Questions", connection))
                {
                    try
                    {
                        connection.Open();
                        questionIDCount = (int)command.ExecuteScalar();
                        connection.Close();
                    }
                    catch (Exception ex) { MessageBox.Show(ex.Message); }
                    finally { connection.Close(); }
                }
                //Make sure that it does not try to choose more questions
than exists in the database
```

```
                if (numberOfQuestions > questionIDCount)
                {
                    MessageBox.Show("Cannot use " + numberOfQuestions + "
questions\r\nNot enough questions exist\r\nOnly " + questionIDCount + "
questions will be used",
                        "Too many questions", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    //If less questions in the database than user
requested, reduce number of questions to number of existing questions
                    numberOfQuestions = questionIDCount;
                }
                //Set the number of questions in the quiz to the number of
questions whether changed or not
                quizQuestionCount = numberOfQuestions;
                //List used to temporarily store questionIDs to check
which are used
                List<int> randomQuestionIDList = new List<int>();
                //Select a random list of questions from the database
                using (SqlCommand command = new SqlCommand("SELECT TOP
(@numberOfQuestions) questionID FROM Questions ORDER BY (NEWID())",
connection))
                {
                    command.Parameters.AddWithValue("numberOfQuestions",
numberOfQuestions);
                    try
                    {
                        connection.Open();
                        using (SqlDataReader reader =
command.ExecuteReader())
                        {
                            while (reader.Read())
                            {

randomQuestionIDList.Add(reader.GetInt32(0));
                            }
                        }
                        connection.Close();
                    }
                    catch (Exception ex) { MessageBox.Show(ex.Message); }
                    finally { connection.Close(); }
                }
                //Add all the random IDs and indexes to the QuizQuestions
table with the current quiz ID
                int questionIndex = 0;
                foreach (int randomQuestionID in randomQuestionIDList)
                {
                    using (SqlCommand command = new SqlCommand("INSERT
INTO QuizQuestions VALUES "
                        + "(@quizID, @questionID, @questionIndex)",
connection))
                    {
                        command.Parameters.AddWithValue("quizID", quizID);
                        command.Parameters.AddWithValue("questionID",
randomQuestionID);
                        command.Parameters.AddWithValue("questionIndex",
questionIndex);
                        try
                        {
                            connection.Open();
                            command.ExecuteNonQuery();
                            connection.Close();
                        }
                        catch (Exception ex) {
MessageBox.Show(ex.Message); }
                        finally { connection.Close(); }
                        //Sum marks for each question to set maximum marks
available for quiz
                        try
                        {
                            quizMaximumMarks +=
(int)Tools.GetQuestionData(quizID, questionIndex)["questionMarks"];
                        }
                        catch { return; }
                        questionIndex++;
                    }
                }
            }
```

Check answers are correct and get total marks, UserProgressForm

Pseudocode

```
TotalMarks ← 0
FOR EACH ROW
      UserAnswer ← Cell[UserAnswer]
      QuestionAnswer ← Cell[QuestionAnswer]
      IF (UserAnswer = QuestionAnswer)
            CellColour ← Green
            TotalMarks ← TotalMarks + 1
      ELSE
            Cell[QuestionMarks] = 0
            CellColour ← Red
      TotalMarksLabel ← TotalMarks
NEXT ROW
```

Real Code

```
            int totalMarks = 0;
            foreach (DataGridViewRow row in userProgressDataGridView.Rows)
            {
                //Check if user's answer is correct
                if ((string)row.Cells["userAnswer"].Value ==
(string)row.Cells["questionAnswer"].Value)
                {
                //If correct, add the number of marks earned to the total
                    row.DefaultCellStyle.BackColor = Color.Lime;
                    totalMarks += (int)row.Cells["questionMarks"].Value;
                }
                else
                {
                    //Otherwise, show 0 marks earned
                    row.Cells["questionMarks"].Value = 0;
                    row.DefaultCellStyle.BackColor = Color.Crimson;
                    row.DefaultCellStyle.ForeColor = Color.White;
                }
            }
            //Set the label text to show the total marks
            totalMarksLabel.Text += totalMarks;
```

Update name of question as it is being created, AddQuestionForm

Pseudocode

```
Month ← MonthComboBox
Year ← YearNumericUpDown[Last2Digits]
QuestionNumber ← QuestionNumberNumericUpDown
QuestionPart1 ← QuestionPart1ComboBox
IF QuestionPart2ComboBox BLANK
      QuestionPart2 ← EMPTYSTRING
ELSE
      QuestionPart2 ← QuestionPart2ComboBox
IF QuestionPart3ComboBox BLANK
      QuestionPart3 ← EMPTYSTRING
ELSE
      QuestionPart3 ← QuestionPart3ComboBox
Question ← Month + Year + "_" + QuestionNumber + QuestionPart1 +
QuestionPart2 + QuestionPart3
```

Real Code

```
            questionName = (
                monthComboBox.Text
                + yearNumericUpDown.Value.ToString().Substring(2) //Take
last 2 digits of year
```

```
                + "_"
                + questionNumericUpDown.Value.ToString()
                + questionComboBox1.Text
                //If parts are specified, add comma and part to question
name
                + (string.IsNullOrWhiteSpace(questionComboBox2.Text) ? ""
: "," + questionComboBox2.Text)
                + (string.IsNullOrWhiteSpace(questionComboBox3.Text) ? ""
: "," + questionComboBox3.Text)
                );
            questionNameLabelText = questionName;
```

Check QuizForm answer is correct

Pseudocode

```
UserAnswer ← AnswerTextBox
IF UserAnswer IS float
      QuestionAnswer ← 0
      TRY
            QuestionAnswer = GET Answer FOR quizID, quizQuestionIndex
      CATCH RETURN False
      IF (QuestionAnswer TO 2 decimal places = UserAnswer TO 2 decimal
places)
            AnswerCorrect ← True
            TRY
                  Marks ← GET Marks FOR quizID, quizQuestionIndex
                  MarksAchieved ← MarksAchieved + Marks
            CATCH RETURN False
            CheckAnswerButtonVisible ← False
            AnswerTextBoxReadOnly ← True
            AnswerTextBoxColor ← Green
            STOP Timer
            MarksLabelText ← MarksAchieved
      ELSE
            AnswerCorrect ← False
            AnswerTextBoxColor ← Red
ELSE IF NOT UserAnswer BLANK
      Message(NonNumericAnswer)
RETURN False
```

Real Code

```
            bool answerCorrect;
            if (Tools.IsFloat(answerTextBox.Text))
            {
                string questionAnswer = "0";
                try
                {
                    questionAnswer = (string)Tools.GetQuestionData(quizID,
quizQuestionIndex)["questionAnswer"];
                }
                catch { return false; }
                if (Tools.To2DP(questionAnswer)
                    == Tools.To2DP(answerTextBox.Text))
                {
                    answerCorrect = true;
                    //Add marks for question
                    try
                    {
                        marksAchieved +=
(int)Tools.GetQuestionData(quizID, quizQuestionIndex)["questionMarks"];
                    }
                    catch { return false; }
                    checkAnswerButton.Visible = false;
                    answerTextBox.ReadOnly = true;
                    answerTextBox.BackColor = Color.Lime;
                    answerTextBox.ForeColor = Color.Black;
                    timer.Stop();
                    //Update number of marks achieved displayed to user
                    marksLabelText = marksAchieved.ToString();
                }
                else
                {
                    answerCorrect = false;
                    answerTextBox.BackColor = Color.Red;
                    answerTextBox.ForeColor = Color.White;
                }
                return answerCorrect;
            }
            else if (!string.IsNullOrWhiteSpace(answerTextBox.Text))
            {
                Tools.NonNumericAnswerMessage();
            }
            return false;
```

Print QuizForm question image

Pseudocode

```
QuestionImage ← QuestionImageDisplayImage
IF QuestionImage NOT NULL
      PrintDocument ← NEW PrintDocument
      PrintDocumentPrintPageEvent ← NEW EVENT
            TRY
```

```
                  QuestionName ← GET questionName FOR quizID,
quizQuestionIndex
                  DRAW QuestionName ON PAGE
            CATCH RETURN
            ImageScale = 700/QuestionImage[Width]
            DRAW QuestionImage ON PAGE USING ImageScale
      SHOW NEW PrintPreviewDialog WITH PrintDocument
ELSE
      Message(NothingToPrint)
```

Real Code

```
            if (questionImageDisplay.Image != null)
            {
                PrintPreviewDialog PrintPreviewDialog = new
PrintPreviewDialog();
                PrintDocument printDocument = new PrintDocument();
                //Declare anonymous method that triggers when the document
is printed
                printDocument.PrintPage += delegate (object sender,
PrintPageEventArgs e)
                {
                    try
                    {

e.Graphics.DrawString((string)Tools.GetQuestionData(quizID,
quizQuestionIndex)["questionName"],
                            new Font("Arial", 14), Brushes.Black, 50, 40);
                    }
                    catch { return; }
                    Image questionImage = questionImageDisplay.Image;
                    float imageScale = 700f / questionImage.Width;
                    e.Graphics.DrawImage(questionImage, 50, 70, 700,
questionImage.Height * imageScale);
                };
                PrintPreviewDialog.Document = printDocument;
                PrintPreviewDialog.ShowDialog();
            }
            else Tools.NothingToPrintMessage();
```

Check login details are correct and get userID, UserLoginForm

Pseudocode

```
TRY
      Username = UsernameTextBox
      Command ← SELECT userID FROM Users WHERE username=Username
      OPEN CONNECTION
      Reader ← RESULTS OF Command
      IF READER HAS ROWS
            WHILE READING
                  UserID ← Reader[userID]
            CLOSE CONNECTION
            TRY
                  YearGroup ← GET yearGroup FOR userID
                  IF (YEARGROUP = 0)
                        TeacherUser ← True
                  ELSE TeacherUser ← False
            CATCH RETURN
            IF PASSWORD CORRECT
                  LOGIN SUCCESSFUL
                  Close
      ELSE
            Message(InvalidUsername)
            RETURN
CATCH Message
FINALLY CLOSE CONNECTION
```

Real Code

```
using (SqlConnection connection = new
SqlConnection(Tools.connectionString))
            {
                //Get the userID for the username given
                SqlCommand command = new SqlCommand("SELECT userID FROM
Users WHERE username=@username", connection);
                command.Parameters.AddWithValue("username",
usernameTextBox.Text);
                try
                {
                    connection.Open();
                    SqlDataReader reader = command.ExecuteReader();
                    //There will be a row if there was a username found
(SQL WHERE)
                    if (reader.HasRows)
                    {
                        while (reader.Read())
                        {
                            userID = ((int)reader["userID"]);
                        }
                        connection.Close();
                        //If the user has a yearGroup of 0, they are a
teacher
                        try
                        {
                            if
((int)Tools.GetUserData(userID)["yearGroup"] == 0) teacherUser = true;
                            else teacherUser = false;
                        }
                        catch { return; }
                        //Check if password is correct, if not shows
message
                        if (CheckPassword())
                        {
                            //Set result so MainMDI knows that login was
successful
                            DialogResult = DialogResult.OK;
                            //Close form when done
                            Close();
                        }
                    }
                    else
                    {
                        //If no row, username not found so show message
                        MessageBox.Show("Username was not recognized",
"Invalid username", MessageBoxButtons.OK, MessageBoxIcon.Information);
                        return;
                    }
                }
                catch (Exception ex) { MessageBox.Show(ex.Message); }
                finally { connection.Close(); }
            }
```

Change user's password, UserPasswordForm

Pseudocode

```
NewPassword ← TextBoxNewPasswordText
ConfirmPassword ← TextBoxConfirmPasswordText
IF NewPassword = ConfirmPassword
      CurrentPassword ← Select password from Users where userID=UserID
      IF TextBoxCurrentPasswordText = CurrentPassword
            Update Users set password=NewPassword where usedID=UserID
      ELSE
            Output "Incorrect current password"
ELSE
      Output "Passwords do not match"
```

Real Code

```
//Check that the user has confirmed their new password correctly
            if (confirmPasswordTextBox.Text == newPasswordTextBox.Text)
            {
                using (SqlConnection connection = new
SqlConnection(Tools.connectionString))
                {
                    //Check if user entered current password matches
actual current password
                    string password = "";
                    try
                    {
                        password =
(string)Tools.GetUserData(MainMDI.userID)["password"];
                    }
                    catch { return; }
                    if (currentPasswordTextBox.Text == password)
                    {
                        if (MessageBox.Show("Are you sure you want to
change password?", "Change password",
                        MessageBoxButtons.YesNo, MessageBoxIcon.Question)
== DialogResult.Yes)
                        {
                            using (SqlCommand command = new
SqlCommand("UPDATE Users SET password=@password WHERE userID=@userID",
connection))
                            {

command.Parameters.AddWithValue("password", newPasswordTextBox.Text);
                                command.Parameters.AddWithValue("userID",
MainMDI.userID);
                                try
                                {
                                    connection.Open();
                                    command.ExecuteNonQuery();
                                    connection.Close();
                                }
                                catch (Exception ex) {
MessageBox.Show(ex.Message); }
                                finally { connection.Close(); }
                            }
                            //Close the form when done
                            Close();
                        }
                    }
                    else
                    {
                        //Focus the password textbox so the user can
correct it quickly
                        currentPasswordTextBox.Focus();
                        MessageBox.Show("Current password was not entered 
correctly", "Incorrect password");
                    }
                }
            }
            else
            {
                //Clear both new password textboxes
                newPasswordTextBox.Clear();
                confirmPasswordTextBox.Clear();
                MessageBox.Show("Passwords do not match", "Password 
confirmation failed");
            }
```

Procedure and Variable Lists

QuizForm

Functions

Name	Parameters	Description	Output
GenerateQuiz	Int numberOfQuestions	Creates a quiz for a set number of questions, selects random questions from database	None
CheckAnswer	None	Checks if user's answer for current question is correct	Bool
NextQuestion	Bool checkingAnswer	Optionally checks answer if not done already, inserts user's answer into the database before moving to next question or ending quiz if end. Loads next question including displaying image, restarting timer, etc	None
Print	None	Scales the question image and prints it	None
DrawWorking	None	Draws on the working area with the mouse	None

Variables

Name	Type	Description
quizID	Int	Stores the ID for the current quiz
quizQuestionCount	Int	Stores number of questions for quiz
quizMaximumMarks	Int	Stores maximum number of marks for test
quizQuestionIndex	Int	Stores index of current question in quiz
timerSeconds	Int	Stores number of seconds used by timer
marksAchieved	Int	Stores number of marks achieved by user so far

SUVATQuizForm

Functions

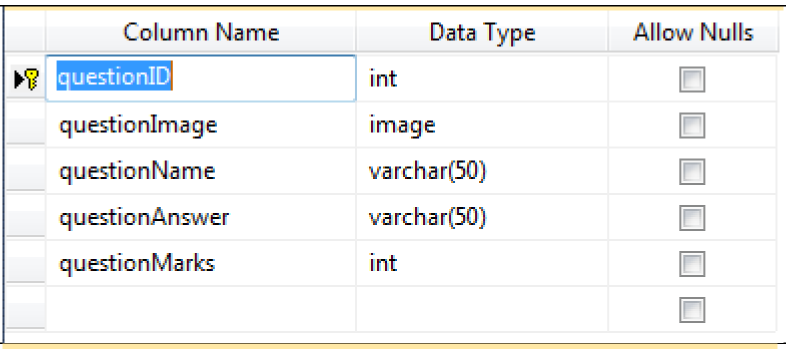
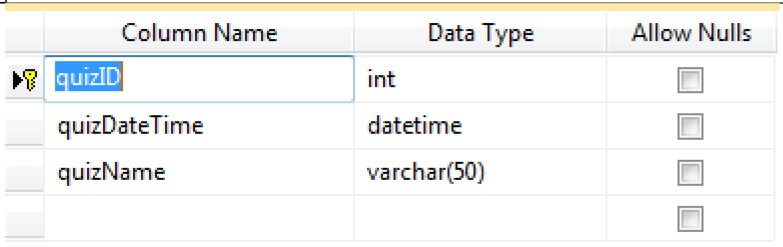
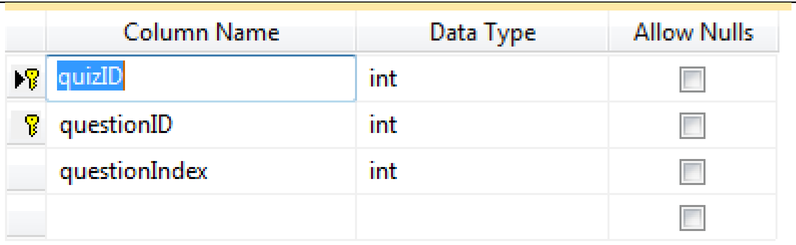
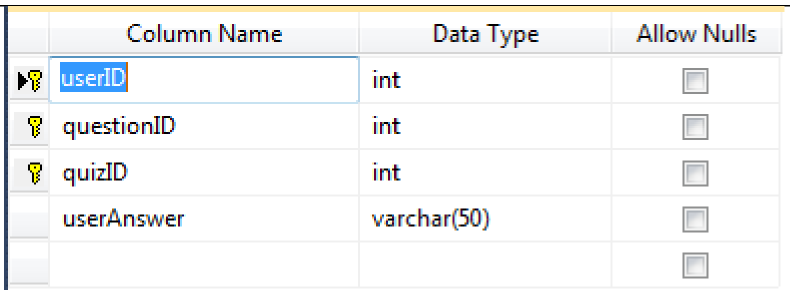
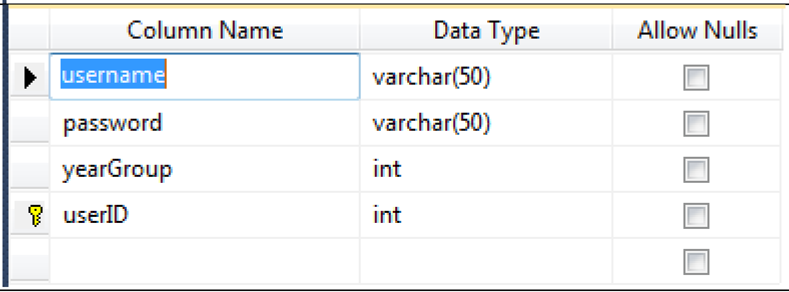
Name	Parameters	Description	Output
GenerateQuestions	None	Sets up the quiz area and creates a number of questions set by the numberOfQuestions NumericUpDown. Request random questions until it has enough and ensures that there are no repeated questions. Adds each question to the quiz area panel	None
RandomQuestion	Int questioIndex	Uses a random number generator to choose a random SUVAT part, checks if it already exits then assigns a value to it if not, the completed question is returned as a string	String

Variables

Name	Type	Description
questionsAnswers	String array	Stores the questions that are displayed and their answers
suvatParts	Int array	Stores the numerical values of each part of the SUVAT equation for each question

parts	String	Used to temporarily store the parts of the question when it is generated

Database Tables

Name	Definition		
Questions	 Column Name	Data Type	Allow Nulls
	questionID	int	☐
	questionImage	image	☐
	questionName	varchar(50)	☐
	questionAnswer	varchar(50)	☐
	questionMarks	int	☐
			☐
Quiz	 Column Name	Data Type	Allow Nulls
	quizID	int	☐
	quizDateTime	datetime	☐
	quizName	varchar(50)	☐
			☐
QuizQuestions	 Column Name	Data Type	Allow Nulls
	quizID	int	☐
	questionID	int	☐
	questionIndex	int	☐
			☐
UserQuestion	 Column Name	Data Type	Allow Nulls
	userID	int	☐
	questionID	int	☐
	quizID	int	☐
	userAnswer	varchar(50)	☐
			☐
Users	 Column Name	Data Type	Allow Nulls
	username	varchar(50)	☐
	password	varchar(50)	☐
	yearGroup	int	☐
	userID	int	☐
			☐

Program Code

See appendix

Appraisal

Objectives

Number	Met?	Objective	Notes
1	Yes	Students are able to input answers with 2 decimal places or as integers as required, all answers are converted to 2 decimal places	This can be easily implemented anywhere as required using the Tools class' To2DP function, met deadline
2	Yes	The main interface will have a menu bar	Decided to use a typical 'File' menu so that is better fits the standard rather than just using a new Menu on its own, met deadline
3	Yes	Multiple windows can be opened simultaneously within the main interface	Had to change all forms to work with the MDI container parent which took some time, met deadline
4	Yes	Questions and quizzes can be printed from the same menu bar; the item currently in focus determines which is printed	Changed print functions of each form so that they all had the same name [Print()], which made it easier to find which form is active and call its function from the MDI parent, met deadline
5	Yes	Users can be created for individual students and teachers, student users can be created by teachers	Student users must be created manually with a manually typed password, instead of for example generating a password or creating users in bulk, met deadline
6	Yes	Deleted users will have their associated data also deleted	When the delete button is clicked in EditUsersForm the data for the selected users is deleted from the UserQuestion table before deleting the user from Users, met deadline
7	Yes	Deleted questions will have their data and references to them deleted	Deleting a question through EditQuestionsForm first deletes all of the history of it being used from UserQuestion and QuizQuestion tables, before the question is deleted from Questions, met deadline
8	Yes	Users are able to login to the system	The username is checked to make sure that it exists, and the password must be correct in order to log in, met deadline
9	Yes	Administrative users (teachers) have permissions to add and remove questions	When logged in as a teacher, a settings menu item in the MenuBar is available which provides access to forms to modify questions and users, met deadline
10	Yes	Input must be validated to only allow numerical answers	Numbers can be easily checked using the IsInt or IsFloat functions in the Tools class, met deadline

11	Yes	Images will be used for questions, as they will be shown in the original form; such as a screenshot of an exam question/scan of textbook	For the main Quiz, questions can only be uploaded as images, and the forms used to add the questions are designed for exam and textbook questions. The SUVATQuizForm does not use images, met deadline
12	Yes	A working space is available for notes, it can be drawn on using a virtual pen	Took some time to learn to use graphics, works as intended but does not feel smooth like some other applications, met deadline
13	Yes	The working space will have multiple selectable foreground/background colours	The 'eraser' uses the background colour to draw over existing lines, so when changing the background colour the working area must be cleared, met deadline
14	Yes	Quizzes can be created automatically by randomly selecting variables and generating values to create a set number of questions, as set by the user	The SUVATQuizForm has a NumericUpDown to allow the user to set the number of questions, variables are picked randomly and assigned values, met deadline
15	Yes	The database will be accessible in under 10 seconds	Except where there are connection issues, all database queries are completed within 10 seconds, met deadline
16	Yes	The main interface will completed by the end of October	Met deadline
17	Yes	Database will be created and linked by November	Met deadline
18	Yes	Working space/drawing functionality will be completed by December	Took longer than expected but met deadline
19	Yes	User login/creation and student progress logging functionality will be completed by January	Met deadline
20	Yes	Random quiz and printing functionality will be completed by February	Met deadline

Feedback

Would you say the system is easy to use from a teachers' perspective?	I would say that this program is easy to use, although the user interface could be clearer and easier to use. Although, manually creating and or changing users is difficult and time consuming for larger groups of students.
Have most or all students been able to use the system without any issues?	Sometimes users forget their passwords, but they are unable to reset them themselves, so they must wait for a teacher to change it for them before they can continue to use the system. Some students also find it difficult to navigate the system and require help from teachers. A few students have reported cases where an answer is actually correct it is marked as incorrect.
Was the transition to using the system easy?	It can be used in conjunction with the existing system easily. Eventually I could see it being used as the primary system for setting work outside of lessons although to do this I think it would be necessary to allow teachers to be able to set work for individual students or groups, as well as splitting questions into topics rather than the quiz just using all of the questions.
Was it easy to set up the program to use?	The program could be easily distributed to students either by allowing them to keep their own copy on a USB flash drive or by emailing them a copy – installing the program from the installer was easy. However, manually creating the necessary users for the first time took a long time.
Do you feel that your expectations for the program were met? Does it meet the objectives set out?	Overall I feel that the program does what I wanted it to do – it works as I expected it to work. All of the functionally of the program is there but I feel that it could be presented more clearly and laid out better to be more easily access some of this functionality.
Are there any outlying issues or criticisms of the system that you have?	Error messages are not always clear – for example when there is a connection issue to the database it is not always clear what the problem is. The help menu item is not very useful.
Do you have any suggestions for improvements or extensions to the system?	The user interface could be better laid out and more colourful. An option to add a group of users quickly would be useful. The help menu item could be expanded into something more useful.
Date/Signature	

Analysis of feedback

Overall, the feedback suggests that the user interface (UI) could be better laid out to be clearer and easier to use – also suggesting use of more colours to make it more interesting, using different colours would also allow for highlighting key elements such as the menu bar or navigation buttons which would help clarity in using the program. It was also suggested in feedback to make the help button more useful.

Another point of the feedback suggested that it would be easier to maintain the system if users could make more changes on their own, such as to reset their own password or create their own login details. This would also mean students would be able to continue to use the system if for example they forget their password, without waiting for a teacher to reset it for them; teachers would also less frequently need to make changes to users manually, allowing them more time to prepare lessons or manage quiz questions for example.

A drawback of the system that may prevent it being used instead of the existing system is that the use of a quiz that uses all questions in the database limits the usefulness of it as a revision tool, as students cannot focus on one particular area – the feedback suggests that this should be done splitting questions by topic. Allowing teachers to set questions for particular students or groups would allow the quiz to work more as a replacement for a worksheet, while also having the benefits provided by the program including automatic marking.

One of the identified problems was that errors are not always clear; while some have been described others use the default exception message which does not always explain clearly what the issue is. An encountered error was where correct answers were marked incorrect; this is due to the fact that if there is no connection to the database a question is always marked as incorrect – this could be resolved by getting the answers when the questions are requested and checking locally rather than requesting the answer, or by rechecking automatically if a connection could not be made.

Extensions

Due to time constraints, the help menu item only shows a simple MessageBox with information about the program; however this could be extended to provide an interactive user guide which could for example show information about the form that is currently open.

The current system only allows only a single user to be added at a time; however a way to add a range of users at once could be to generate random passwords and assign them to usernames from a list. This would allow adding a whole group or even year group at a time quickly.

Splitting topics would allow for easier revision as students can do a quiz on the topic which they wish to revise, this could be implemented by adding a column to the Questions database table that specifies the topic, and when generating a new quiz the topic could be specified which would limit the questions to questions in the desired topic. In order to make work settable users could be assigned to groups/classes by adding a column to the Users database, and creating a new database table which links groups to quizIDs – a quiz could manually be generated instead of randomly and the ID assigned could be used for the set work which is accessible to all groups it is linked to.

With the current system, updates would have to be done manually, if there are smaller changes this may not be a big problem but if larger changes were needed to be made such as changes to the database the client copy may no longer work. A possible alternative solution could be to use a web-based application which would not need to be manually updated; and also brings the benefit of allowing the system to be accessed by different devices and operating systems, allowing it to be used portably on mobile devices.