OCR F454 Computing Coursework

Alicia Sykes 0063

Centre Name: St John's Marlborough

Centre Number: 66631

Section 1 – Definition, Investigation and Analysis

Problem Definition

St John's School, Marlborough is an 11-18 comprehensive school in Wiltshire. There are approximately 1700 students, including nearly 400 in the sixth form, and about 200 members of staff. Mrs Munt is a teacher of English at St John's, and she teaches pupils of all abilities across all of the year groups. Part of her teaching involves setting homework and testing pupils on their knowledge. Mrs Munt usually sets homework and class work in paper form. Both teacher and pupils can find it difficult to keep track of assignments, also assignments need to be marked, which can be time consuming and boring.

Mrs Munt tries to use a variety of methods to teach and test her pupils, which include using computers, although there is a limited amount of available resources in computer form. She also tries to engage her pupils using modern tools that they can relate to, such as the internet, but again the small number of resources available online tends to be very limited, and often not directly relevant to what she is trying to teach.

To overcome this problem Mrs Munt would like some sort of system that will allow her to set class work and homework to pupils in a specific format, and that could generate self-marking multiple choice quizzes based on information entered by her

I am going to arrange structured interviews and questionnaires for both Mrs Munt and her pupils in order to collect more information about the problem, and what they might want from a new electronic system.

Investigation and Analysis

In order for me to know exactly what the end user needs I will spend some time collecting information about both the current system and the requirements for the new system. This will ensure that the new system will fit its user requirements exactly, and will include all the necessary features needed for the program to fulfil its purpose.

I will conduct a series of structured interviews with my client in order to decide exactly what will be required of the new system. I will plan the questions before hand, based upon what information I will need in order to design and develop the program. I will also create flow diagrams illustrating the current system, this should make it easy to analyse and show areas in which it can be improved with the new system.

I will also need to find out the end users ability with software, so that I can design and make the program to the right complexity, and if provide the necessary training materials.

Planned Interviews

Below is a list of the interviews I plan to undertake in order to collect more information about the client, the current system and requirements for the new system.

Date	Time	Location	Name	Title	Aims	Complete
11/10/11	9:45 AM	St Johns	Mrs Munt	The client	To collect information about the client, that will help with the design of the new system.	✓
12/10/11	11:30 AM		Mrs Munt	Current System	To collect information about the current system.	✓
18/10/11	10:00 AM	Library	Mrs Munt	The new system	To collect information necessary for creating the requirements specification for the new system.	✓

Questions I plan to ask during the client interviews

The first interview, that is due to take place on Tuesday 10th of October at St John's school with Mrs Munt. The aim of this interview is to find out information about the client that will be relevant to the design of the new system. This will include asking the client about their specific job role, and what parts of this will the new system will aid, as well as asking what their computer skill are like, and how accessible are computers to them.

These are the questions I plan to ask:

1. What is your job role?
2. How many pupils do you teach?
3. What is the average age of your pupils?
4. How much homework and independent class work do you set?
5. How much of your teaching and setting work is currently done using computers?
6. How computer literate would you say you were?
7. How computer literate are your pupils?
8. Do you and your pupils have access to computers?
9. If so, are these computers connected to the internet?

The second interview will be about the current system. In this interview I hope to find out more information about the current system, and what problems there are with it, this will then highlight areas that the new system should aim to solve. The questions I plan to ask are:

1. What is the current system you use to set class work and short tests?
2. Is this system done completely manually?
3. What problems are there with this system?
4. What do your pupils think of your current system?
5. Which areas of your current system would you hope to be improved on for the new system?

The final, but probably most important interview I will carry out is the requirements for the new system. In this interview I hope to collect information about what the client would like the new system to do, and what features it must have in order to serve its purpose. The questions I plan to ask will be:

1. What roles must the new system be able to do?
2. How would you prefer the new system to run?
3. Any installation preferences?
4. What colour scheme would you like?
5. Would a detailed built in help manual be of help?
6. Are there any other features you can think of that you will need to program to have?

Results from interview 1 – The Client

1. What is your job role?

I am a specialist teacher

2. How many pupils do you teach?
   About 25

3. What is the average age of your pupils?
   They range from 11 to 18, and the average age is probably about 13

4. How much homework and independent class work do you set?
   A bit

5. How much of your teaching and setting work is currently done using computers?
   I try to use computers most lessons

6. How computer literate would you say you were?
   Average

7. How computer literate are your pupils?
   They vary from beginners to experienced

8. Do you and your pupils have access to computers at school?
   Yes

9. If so, are these computers connected to the internet?
   Yes

Date: *11/10/11* Time: *10:00* Location: *Library, St John's* Interviewee: *Mrs Munt*

Analysis of Interview 1

After conducting this interview with Mrs Munt, I now know that most the work she doses with her pupils, is during the class time, as she only sets *a bit* of homework, so the new system will be used mainly in the class room. Also with a large age range of pupils, as well as some of them being experienced computer users while others are beginners, the new system must be easy to use and suitable for all ages. Both Mrs Munt and her pupils have access to computers as school, and these computers are connected to the internet, so if the new system required internet access, that would be fine.

Results from interview 2 – The Current System

1. What is the current system you use to set class work and short tests?
   I set targets for the pupils to improve, and as to work towards

2. Is this system done completely manually?
   Yes

3. What problems are there with this system?
   It all needs to be recorded in books, which can be hard

4. What do your pupils think of your current system?
   They just accept that's how it is

Date: *12/10/11* Time: *11:30* Location: *Library, St John's* Interviewee: *Mrs Munt*

Analysis of Interview 2

My second interview, about the current system has shown me that it is done nearly completely manually, although she would like to be able to incorporate more computer orientated exercises into her lessons. She spoke about setting targets to allow and motivate the pupils to improve, so the new system a feature that will allow this. One of the problems she had with the current system is having to duplicate information manually.

Results from interview 3 – the new system

1. What roles must the new system be able to do?
   It needs to be easily accessible; if it is too complicated it may put people off. It needs to look lively and friendly and have a variety.

2. How would you prefer the new system to run?
   Well, it needs to be able to run on older computers, which have an older version of Windows running on them, and on Apple computers. It should be able to be used on desktop computers and laptops.

3. Any installation preferences?
   Downloading it from the internet, or installing it from a disk is okay.

4. What colour scheme would you like?
   Not yellow. I don't really mind, blue and purple is quite nice

5. Would a detailed built in help manual be of help?
   If it is complicated then yes, although keep instructions to a minimum.

6. Are there any other features you can think of that you will need to program to have?
   Print off, copy and paste results and quizzes, this will give it a wider scope of use.

Date: *18/10/11* Time: *10:00* Location: *Library, St John's* Interviewee: *Mrs Munt*

Analysis of interview 3

This was the third interview with Mrs Munt who will be the end user. It has been helpful in getting a lot of information for the requirements specification that I will need to create before I start designing the program. She has said mainly that the program should be easy to use, and be very accessible, so should be able to run on older computers and computers with different systems. There will be built in help manual in the program in case the user gets stuck, or is unsure of something.

Original Interviews

The Client

The new System

Questionnaires

I am going to create some questionnaires to collect additional information from some pupils, as they will be the primary users of the new system. This will allow me to collect some quantative data, that can be analysed and to derive some useful statistics. Most the questions in the questionnaire will be multiple choice, so it will be quick and easy for people to fill in. I will try not to use language to influence the expected answer.

These are the questions I am going to ask:

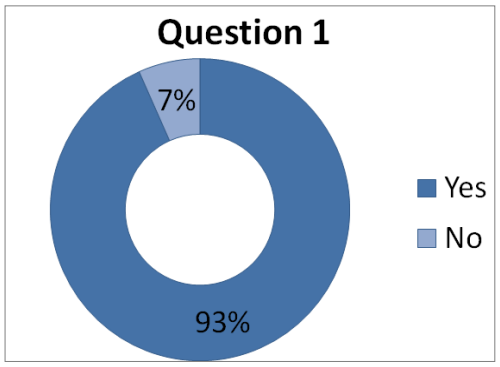
- Would you use a computer program based on testing your knowledge to aid your revision/ learning if it was available?
- Do you currently use a similar system, if so what?
- How many different subjects do you do, and to what level?
- How many hours of out of lesson revision, extended learning and testing your knowledge, do you do in your own time in a week?
- Do you find there is a lack of revision resources that will help you test your knowledge, and pin point weak points that may need more revision on?
- What additional features do you think a quiz program should include to make it more useful?

Most of the questions are multiple choice, that will allow people to fill them in quickly, and I have been careful not to ask leading questions to get the results I want. I am going to hand out 50 questionnaires, so that I will have a range of results.

Results from Questionnaires

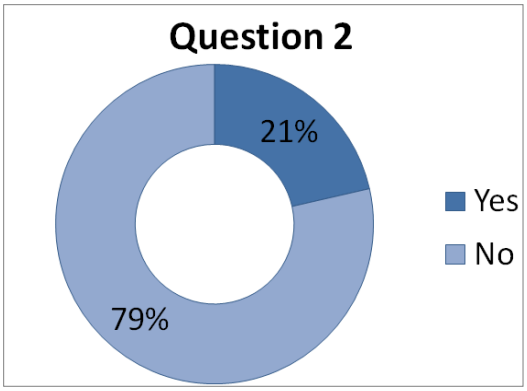
Q1) Would you use a computer program based on testing your knowledge to aid your revision/ learning if it was available?

For question1, there were 2 options, yes and no. 28 people said yes, and 2 said no. I was pleased with this response, as it means there is a high demand amongst pupils for a program like this. I expected the no count to be higher.



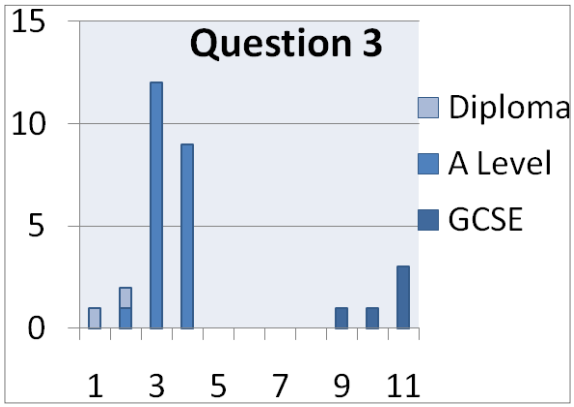
Q2) Do you currently use a similar system, if so what?

Again there were 2 options for question 2, yes and no also there was a line so people could specify if they currently used a similar system. 6 people said they did currently use a similar system, and 22 people said they did not. 4 of these people specified which piece of software they currently used, 2 of the used examstutor.com, one used getrevising.co.uk and the other used mymaths.com. I plan to research these applications as part of this investigation.



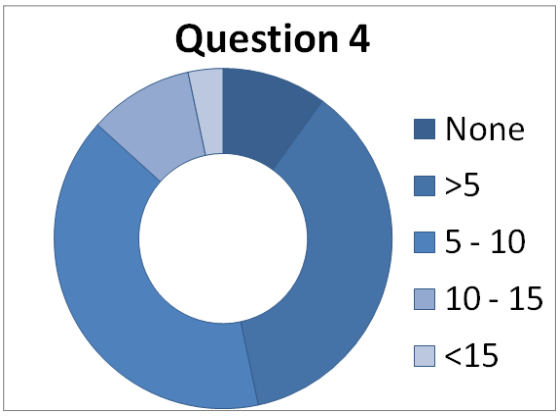
Q3) How many different subjects do you do, and to what level?

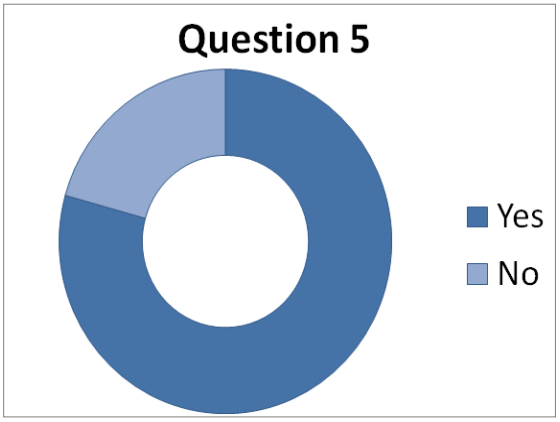
This question was written so I knew what level the people completing the questionnaire were, pupils in year 7 are likely to do less hours of work and use a smaller range of resources than pupils doing A levels. The results showed that most the pupils be questioned were doing AS or A level and were taking either 3 or 4 subjects, a few pupils were doing GCSE's and were taking between 9 and 11 subjects, and younger pupils in KS1, KS2 and KS3 did about 12 – 15 subjects.



Q4) How many hours of out of lesson revision, extended learning and testing your knowledge, do you do in your own time in a week?

This question is important, because if most people didn't do any of out of classroom work a week, they are unlikely to use other resources such as a quiz program to test their knowledge. The results showed that most people did either under 5 hours or between 5 and 10 hours, only 3 people said they didn't do any work outside lessons and 1 person did more than 15 hours.





Q5) Do you find there is a lack of revision resources that will help you test your knowledge, and pin point weak points that may need more revision on?

This question was designed to show whether there was demand for a quiz based revision program that will do the above. The results showed that there is as 23 people said yes and only 6 people said no.

Q6) What additional features do you think a quiz program should include to make it more useful?

Question 6 will be very helpful when creating the requirements specification, as lots of people have suggested things that they think will make the program more appealing for pupils. Not everyone filled in the last question, but below are some of the responses that were relevant.

- Practical leaning, with problem solving
- Previous results, calculate averages, specific weak areas
- View average scores for the quiz, not just your own
- Good appearance with colours
- Progress tracking, additional teacher notes.
- Record how long it took the student
- Do some questions in a random order, like a quick play option
- Interaction with other users
- Room for diagrams and pictures
- Games (x2)
- Additional links and resources that will increase understanding
- Scores for all modules and overall scores
- Random questions from a number of related quizzes
- Different types of questions
- High scores
- Animations
- Leader boards
- A summary at the end, showing which questions were wrong, would help show which areas need more revising.
- Compare your own results, and track your progress.

- To change the theme / colours if it's for students of different ages.

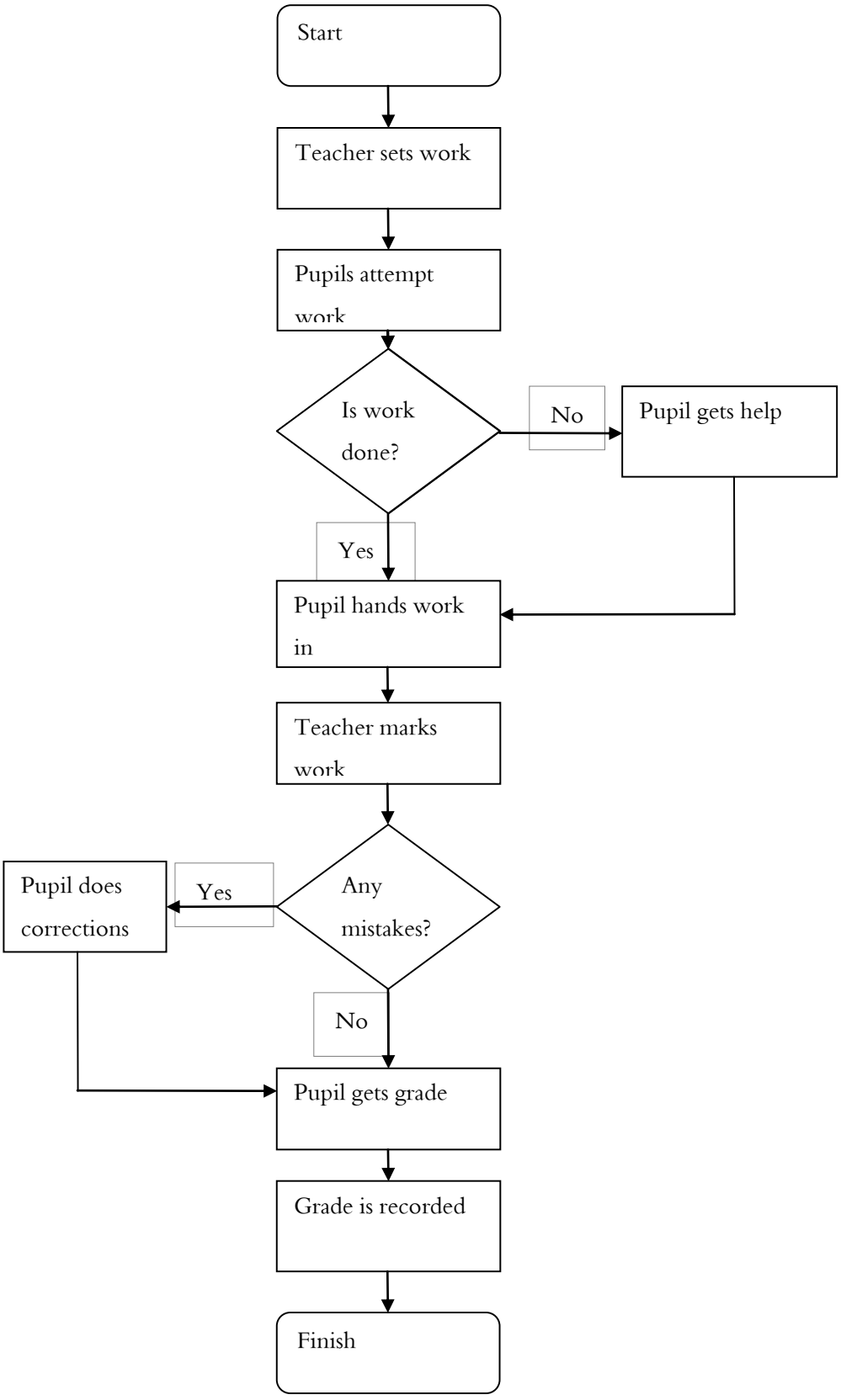
These points show that the main features pupils want is for the final system to be fun, good appearance, leader board containing high scores for them and other pupils and being able to track their progress. I will take all these points into account when designing the system.

Analysis from questionnaire results

The questionnaire has helped me come up with some ideas for the new system, from a pupil's point of view – as it is them who will be using it the most, I think this is important. I printed 30 questionnaires out, and got all 30 back, which was good, although some of the questions were left blank. The results have shown me that there is a demand for software like this, and both pupils and teachers would find it very useful for consolidating knowledge and testing. It also showed that the appearance would have to be suitable for younger and older pupils, because there was a range of ages wanting to use the system. I will also create a high scores list type thing, that will allow pupils to track their progress and improvement as well as comparing to averages, targets and previous scores. All of this information will help me create a detailed requirements specification.

The end user would like a program that will allow them, and others to create and set quizzes to pupils.

Flow Chart of Current System



Summary of user's current computer system

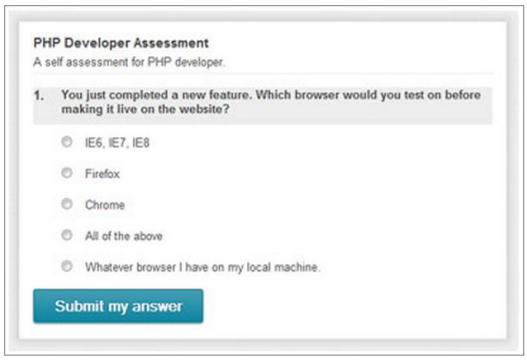
Mrs Munt will run the new system on a computer at St John's, below are the specs of the computer

- **RAM**: 2 GB
- **Processor**: AMD Athalon Dual Core Processor 2.1 GHz
- **HHD Space**: Access to a 500GB network drive
- **OS**: Windows 7 64bit
- **Range**: HP Compaq

The end users computer system is quite average although it will be adequate for the running of the program. Windows 7 is the OS, which is the most up-to-date version of Windows; it is 64 bit.

Available Software

1. Quiz school – created by proprofs.com

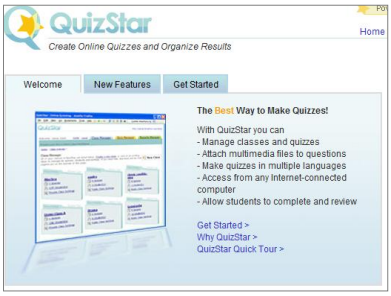


Quiz school is and online based quiz program that allows a user to write a quiz for others to do. Although some features of it seem good, it has many disadvantages, the biggest being the cost, the basic plan is $10 a month and the professional edition is $20 a month. Also the quizzes can only be done by other members, who also need to pay to use it. There are no ways to search for available quizzes which are a major drawback, in order to do a quiz you must have been sent a link.

2. Makeaquiz.com

This is a website that allows any user to make a quiz quite quickly. Its benefits are that it's free and easy, and adding information is reasonably quick, the finished quiz can be shared with anyone with internet access. However it is very limited in functionality, there is no way to record scores, and other than modification of the colour screen, there is no way of changing the layout or graphics that appear to the user during the running of the quiz. A lot of the layout is quite basic HTML style, which could be improved on.

3. Quizstar.4teachers.org



This is another internet based quiz program, although this one is aimed directly at teachers, so it tries to fulfil all there teaching needs. I think it fails at that. The website is hard to navigate around, and quite slow, as everything is on different pages, and there are many animated adverts all over it.

4. PowerQuizPoint – DigitalOfficePro.com

This one is slightly different, because it allows the quiz to be created offline, and quickly imported into Microsoft Power Point presentations, which can be useful for lessons. It also has some nice layout templates for teachers to use. However it does not allow scores to be kept and is limited in functions that would allow pupils to do the quiz in their own time. Also it costs $177, which is a bit much.

Requirement Speciation

Computer requirements:

- **RAM**: 512 MB
- **Processor**: 1 GHz or faster
- **HHD Space**: Very small amount required.
- **OS**: Any
- **Type of computer**: Any

The end user has already got a computer with good enough specs to run the final program, so nothing will need to be changed.

The final program must be able to:

- **Allow user to create quizzes**, It must allow users to create quizzes that can then be stored in a database and selected when a user wants to do it. There should be an option for quizzes to be public, or private.
- **Allow users to do quizzes**, It should be easy for users to do the quizzes, it should be efficiently written so it loads fast and keeps score all the way through.
- **Clear customizable interface**, the interface must be clear and easy to understand, it must be suitable for all ages, and people of all technical abilities.
- **Keep scores**, It must be able to record the users score, and compare it to their previous score so they can track their improvement, and see which areas they will need to improve on. It would also be good if it could allow users to compare scores to the overall averages and their previously set targets.
- **Use a low level of resources**, It must not use lots of commuter resources, or be inefficient. As slow running programs are really annoying.
- **Efficiently and securely store a large amount of quizzes**, It must store all the information both efficiently and securely. It must be quick and easy for the program to access it, and not allow other users who do not have permission to access it.
- **Easy to select a quiz**, It must be easy for the quizzes to be sorted by level and subject and be quick and easy to find by the user.
- **Time how long the quiz took**, It needs to be able to automatically time how long it took each pupil to complete the quiz; it should have time limits and record the time taken.
- **It should give a summary at the end of each quiz**, It should show the pupil which questions they got wrong, so they know which areas they need to improve on.
- **Compatibility**, Many pupils will have different software on their home computer than on the school computers, so it will have to be able to run on many different platforms.
- **Built in help manual**, This would be very useful to people who are quite new or less experienced to computers. It would also outline all the main features of the program.

Below the end user has confirmed all the requirements above

Confirmation of End user involvement

Signed End user [Mrs Munt] 

Signed Developer [Alicia Sykes] 

Section 2 – Design

Nature of the Solution

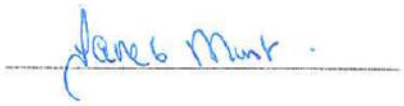
Next I will draw up the design objectives, which will highlight what the new system should aim to have. I will create these objectives, closely following to the specification I drew up from Mrs Munt and the pupil's responses collected from the interviews and questionnaire. I will meet with the end user and have a meeting to confirm and expand upon the following points

- To enter quiz information, the user should be able to input the following details about each quiz:
    - Quiz Name
    - Level (should be a drop down menu)
    - Subject (Dropdown menu)
    - Questions (as many as required up to 100)
    - Answers (As many or as few as required)
    - The correct answer for each question
    - A place to insert an image or diagram
- To user should be a able to complete a quiz which should:
    - Loop through each question in order
    - Show user if they got it wrong or right for each question
    - Question laid out on a clear screen
- Different modes, how the quiz will run
    - Questions in order, or out of order, and randomise questions
    - Show user the correct answer of question immediately
    - Show score and breakdown at the end
    - Change how many times quiz can be retaken
- Enable user to search for a quiz
    - Quick efficient way to search
    - Key word searching

    - Quizzes sorted by level and subject
    - Unique ID for each quiz, quick access
- Keep score of quizzes
    - Record users score while they do the quiz
    - Show total score at the end of the quiz
    - Show breakdown of score with all the questions that were correct or incorrect at the end
    - Show averages
    - Record score for the quizzes
- Store quizzes
    - A storage system to keep hold of all quizzes
    - Must be fast, efficient and effective
    - Must be able to store quizzes, questions, answers and necessary additional information
    - Must store the information securely, and be regularly backed up
- Admin control panel
    - Allow an admin to securely log in with password
    - Allow admin to delete quizzes, questions and answers
    - Allow admin to edit quizzes, questions and answers
- Good user interface
    - Easy to use
    - Clear
    - Welcoming
- Simple installation routine and compatibility
    - Quick
    - Will work on computers with restricted permissions (like school computers)
    - Will be compatible with older computers (e.g. Windows XP, Windows 2003…)
    - Compatible with computers with different a OS, like OS X or Linux

- Other features
    - Should allow the user to print off their scores at the end
    - Allow whole quizzes to be printed off to be done manually
    - Allow answer sheet to be printed off
    - Allow all of the above to be downloaded in PDF format
    - A built in easy to use help manual

Confirmation of End user involvement

Signed End user [Mrs Munt] 

Signed Developer [Alicia Sykes] 

Data Inputs and outputs

Requirement	Data Type	Size	Input	Output
Create quizzes	varchar	150 bytes	Title	Stored in quizzes database Question screen
Create quizzes	varchar	15 bytes	Level	Stored in quizzes database Question screen
Create quizzes	varchar	15 bytes	Subject	Stored in quizzes database Subject database amended Question screen
Add questions	varchar	250 bytes	Question	Stored in questions database Either 'create question' or finish screen is displayed
Add questions	varchar	150 bytes	Answer	Stored in questions database Either 'create question' or finish screen is displayed
Add questions	Char	1 byte	Correct answer	Stored in questions database Either 'create question' or finish screen is displayed
Add questions	Text	500 bytes	Answer exp	Stored in questions table
Finish adding quiz	Variable	-	More options	Added into quizzes database Start screen will then be shown
Search	varchar	-	Search box	Searched from databases, results are outputted.
Do quiz	-		Answer	Adds results to array, results are outputted on the next screen

Data Sources

The data to create a quiz will be inputted from the create a quiz page, the create a question page and can be edited from the finish page.

Data	Input Name	Form	Input Method	Data Size	Validation
Quiz Title	txtTitle	Create quiz	Text box	150 bytes	Presence, range, length, character
Quiz Level	txtLevel	Create quiz	Text box, predictive autofill	15 bytes	Preview
Quiz Subject	txtSubject	Create quiz	Text box, predictive autofill	15 bytes	Preview
Quiz options	-	Create quiz	Check boxes, radio buttons, drop downs, text boxes …	-	Preview
Question	txtQuestion	Create question	Text box	250 bytes	Presence, range, length, character, preview
Answers	txtAnswer[n]	Create question	Text box	150 bytes	Presence, range, length, character, preview
Correct answer	radCorAns	Create question	Dynamic radio button	1 byte	Presence, preview

The data inputted during the completion of a quiz will be inputted on the quiz page, by selecting a radio button. See below

Data input

Data will be inputted on a number of forms.

Search Quiz Form: The user will enter a search term, which will then be searched for in the database, and all quizzes whose title match some part of the search term will be returned and displayed to the user in a user friendly format. The quiz entries will then be able to be clicked to go to the quiz screen.

Create Quiz Form: This form will allow the user to enter a quiz title, level and subject. It will then save this information in the database, and direct the user to the 'create question's form.

'create question's Form: This is the form that the user will be directed to once they have created a quiz. The quiz ID will be sent to this form. It will detect whether it is the first number or not, depending on which form it was sent from. If it was sent from the create quiz form, then it is the first question, so question number will equal zero. If it has been sent from the 'create question' form, it will get the question number from the last form, and then increment it. This will be added to the question number field of the questions table. Then the question inputted on the last form by the user will be added into the question field of the questions table. It will count the number of answers imputed by the user, and add these to the answers field, with the question ID in the answers table. The question ID will be found using an SQL query. There is also a boolean field in the answers table which will either be true of false. This result will be gathered when the user selects a check box, which represents the correct answer. The answer ID will not be entered by the system, it will be automatically assigned by the database (the storage engine) when that field is created, and will auto-increment, so it will always be unique and can be used to represent that answer.

Data output

Data will be outputted in a number of locations.
The Start form: This is the first page, and will output the 12 top quizzes, ordered by number of times taken highest to lowest. The quiz titles, levels and subjects of these quizzes will be outputted to the user.

The Search form: This form will display the search results of the search term entered. It will display the quiz, subject and level of every quiz whose title matches some part of the search term.

The quiz form: The question, with a question ID equal to that of the current question will be outputted, and the number of answers with a question ID will also be outputted to the user.

The results form: the results from the quiz just done will be outputted to the user. There will be a drop down menu showing a breakdown of the score, showing the user which questions they got right, and which they didn't.

The 'create question' form: will output question number of the current question, as well as the title of the quiz.

The finish create form: This will output that the quiz and questions just added by the user have been successfully added to the database, unless they haven't then it will output an appropriate error message and code.

Data Structures

Data will be stored in a database, and will use MySQL queries to run it. There will be three main tables and then a few smaller tables.

Quiz table. This will store information about the quizzes in the system. It will include a unique ID for each quiz that will be automatically created for each quiz, and auto-incremented. It will be used to identify each quiz, it will be of integer type, and a maximum of 5 characters. The second field will be for the quiz title, this will be what will be outputted to the user when they search for a quiz, it has a maximum length of 100 characters, and will be VarChar data format. The third column will be for level, it will be enum, and will be a drop down menu containing all the possible levels. It will also include a field for subject ID, which will link it to the subjects table. It will be of integer data type and a maximum of 5 characters. It will be linked with the subjects table, the reason for linking it to another table is that users will be able to quickly add additional subjects that are needed. This will go through the admin approval process. The final field in the quiz table will be the number of takes. This will be an integer, with a maximum length of 5, and will be used as part of keeping the average score.

Field Name	ID	Name	Level	Subject_ID	Takes
Data Type	Integer	Varchar	Enum	Integer	Integer
Length	5	100	-	5	5
Details / Validation	Auto-increment Primary Field Unique Id	Not Null	Drop down menu	Linked to subjects table	Increments after every take
Description	The Unique ID used to identify each quiz	The title of the quiz to be displayed to the user, and used for searching.	The level that the quiz is for. (KS1, KS2, KS3, GCSE, A level)	A link to the subjects table, to say what subject the quiz is	The number of times the quiz has been taken, used for average score.

The questions table. The questions table will store the questions for the quizzes. The first field will be ID, which will be used to identify the question, it will be unique and a primary key. The data type will be integer, and it will be 5 characters long. It will be auto-incremented, and will be added it automatically when a new record is added. The second field will be the quiz ID, this is linked with the quiz table, and s the quiz knows what questions are in it. It will be an integer with a maximum length of 5. The third field will b question number, which will make it easier to loop through the questions in order in the program, it will start from one and go to how ever many questions there are in that particular quiz. The final field will be question, which will be outputted to the user, it will be Varchar with a maximum length of 250 characters.

Field Name	ID	Quiz_ID	QuestionNumber	Question
Data Type	Integer	Integer	Integer	Varchar
Length	5	5	2	250
Details/ Validation	Auto-increment Primary Field Unique Id	Foreign key linking to quiz ID	Not null	The question to be asked
Description	The Unique ID used to identify each question	Which quiz does the question relate to	Used for keeping track of questions during quiz	Will be outputted to the user during the quiz, and at the end.

The answers table will consist of ID, question_ID, answer, correct answer and numCorr. ID will be a unique field and will be a primary key, it will be used to identify the answers individually, it will be auto-incremented and added automatically. The question_ID will be a foreign key linking to the question_ID in the questions table. It will be used to identify which question the answer relates to. The answer field will contain the answer it will be a varchar format with a maximum length of 100 characters. There will be a a correct answer field, which will be a boolean indicating whether or not the answer is correct. The final field will be the number of correct answers, this will be used for calculating the average score.

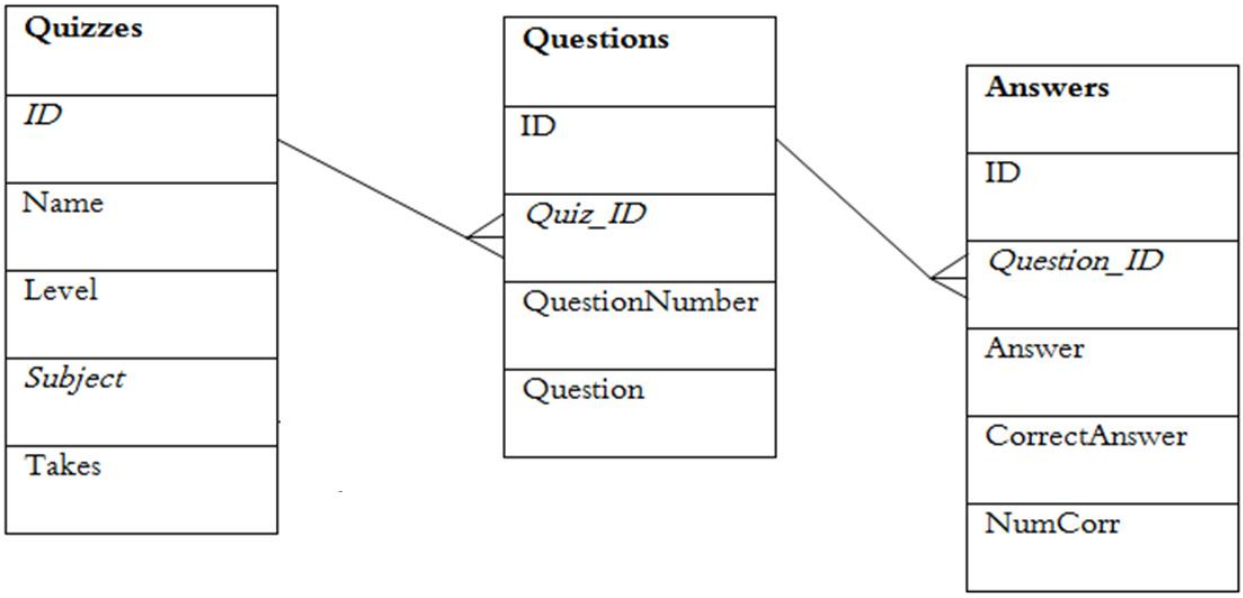
Field Name	ID	Question_ID	Answer	Correct	NumCorr
Data Type	Integer	Integer	Varchar	Boolean	Integer
Length	5	5	100	-	5
Details/ Validation	Auto-increment Primary Field Unique Id	Foreign key linked with the questionID	The answer	Will record whether or not answer is correct	Records how many people have got it right
Description	The Unique ID used to identify each answer	Shows what question the answers are with.		Will be true if it's correct or false if not.	Will be used to calculate average scores.

File structures

Entity relationships:

- ID from the quizzes table, will be linked by a 1 to many relationship with quiz_ID in the questions table
- ID from the questions table will be linked with question_ID in the answers table, also with a 1 to many relationship

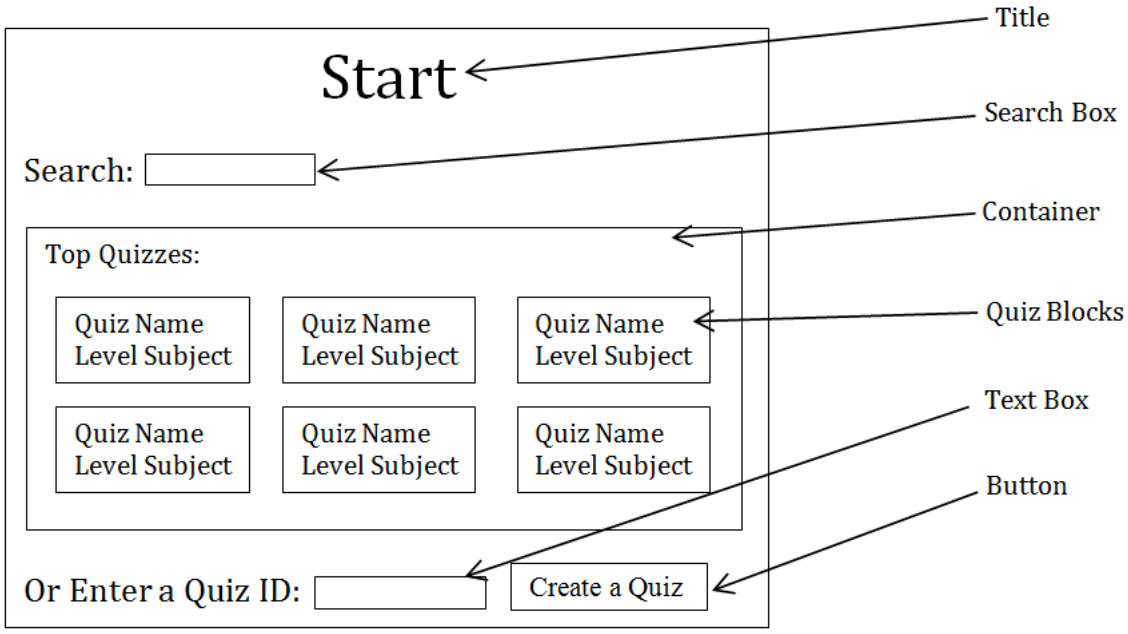
The diagram below illustrates the entity relationships



Screen Designs

I will include the basic designs of the screens for the main forms. The appearance may be slightly different after I have created it as a web page.

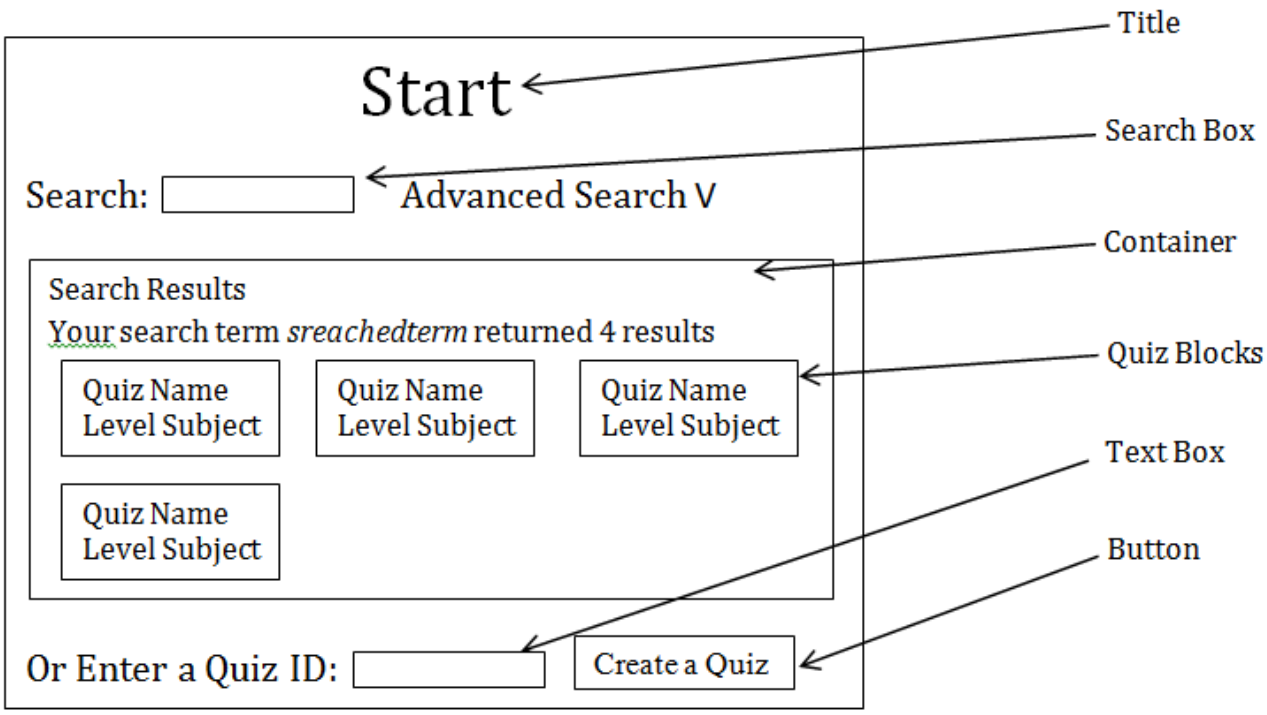
The start Screen:



- The title at the top of the screen will inform the user as to what screen they are on, on the start screen the title will be start.
- The search field will allow the user to enter a search term; there will be a label just before indicating to the user what they should enter in the search box. No search button will be necessary because it will instantly filter all results.
- All the quizzes will be displayed within a container, which will keep everything together, and will have a tab at the top of the page indicating what the page is showing.
- The quiz blocks will show all the quizzes in a summarised way, with their title, level and subject. Users will be able to filter search results so that the quiz blocks will show other information, such as average score or number of takes.
- There is a textbox lower down the page, labelled quiz ID. This will allow the user to just input the quiz ID of a specific quiz, if they know it. This will make is quicker and more user efficient to get to a specific quiz immediately.
- The final button is a link to the create a quiz page. It will show the text, create quiz. It is positioned near the bottom right hand corner of the screen, because that would be where the user is most likely to look for that button. Anyone can create a quiz.

The Search Screen:

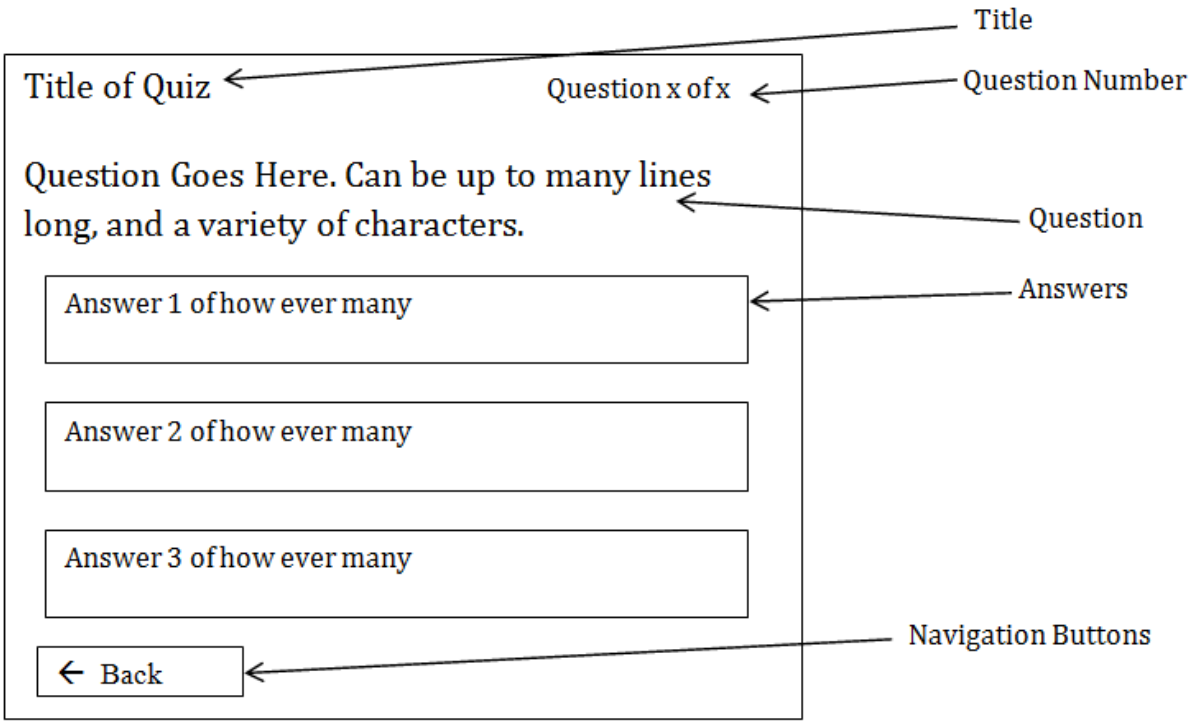
The search screen will be part of the start screen, because it will be dynamically updating what the user has searched for. It will look like as follows:



- The title will still be start, because it is still the start page
- The search box will still be visible, and will allow users to change their search terms, to filter the results.
- The container will also remain, although the tab will say search results
- There will also be an advance search option; this will be a slide down menu, containing more options about filtering, sorting and displaying the results. This menu will not become visible until the users curser is in the search box, or there is text in the search field.
- The search results will be displayed in quiz blocks, the same as before, although only the relevant quiz blocks will be visible.
- The quiz ID text box, and create a quiz button will be the same as before.

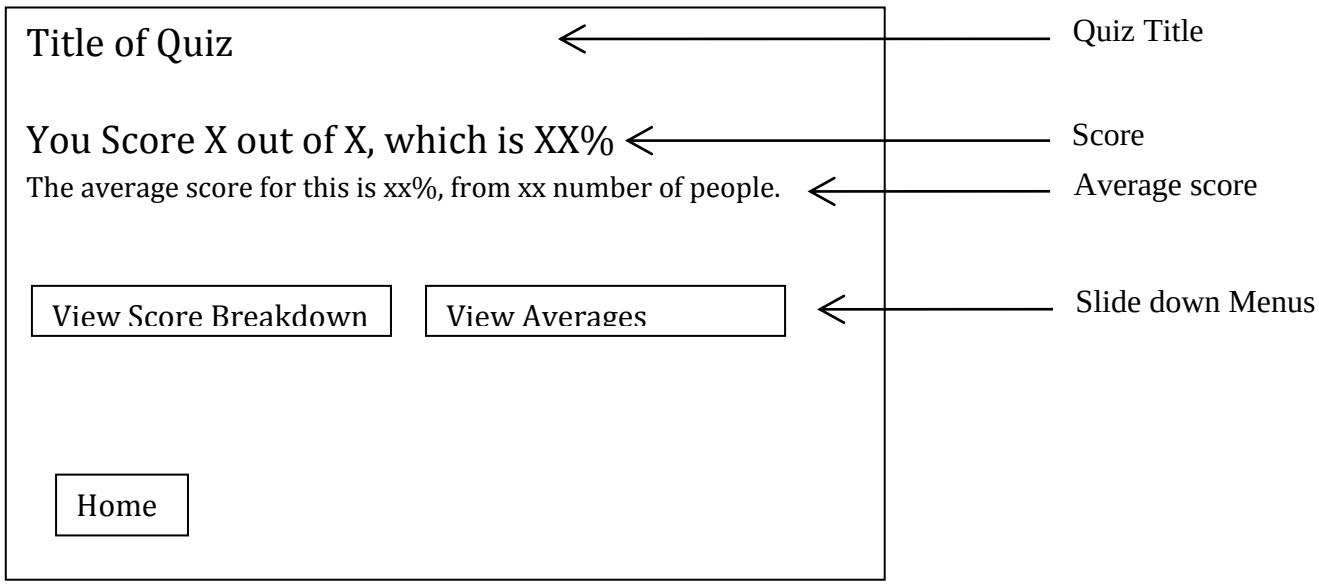
The Quiz form:

The quiz form will be where the user actually does the quiz. It will have to fetch the quiz title, questions and answers from the database. The only piece of data it will start off with will be the quiz ID.



- The quiz title, this will be just plain text, of font size 15px. It will get the information to display from the database, where it is equal to quiz ID.
- On the right hand side, it will display what question the user is on, and how many questions are left.
- Below the question will show, it can be many lines long and will be font size 15px.
- The questions will be displayed in boxes below. There will be one box for each question, between 2 and 6 questions. The user will not have to type anything, which will reduce mistakes, and make it quiz and easy to do.
- At the bottom there will be some navigation buttons, that will allow the user to navigate through questions and forms.

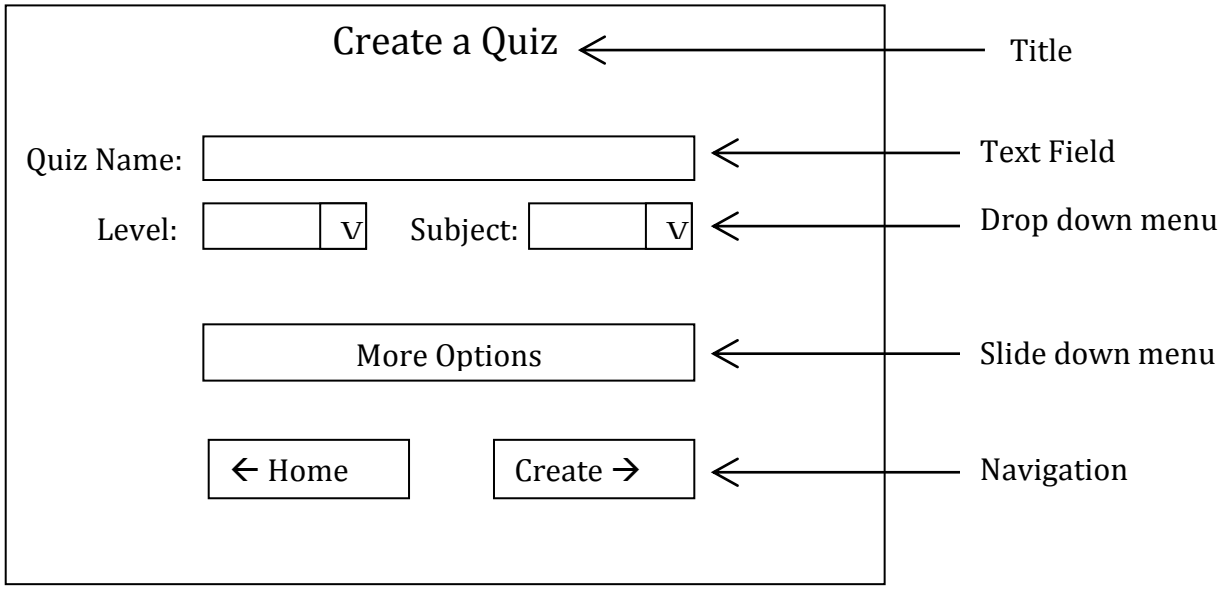
The results form:



- The title of the quiz will be the same as before, it, it will be displayed in a medium font in the top left hand corner of the screen
- The score will be outputted on the line below, in both marks and percentage.
- Below that there will be the average score, from other people.
- There will be a slide down menu, that has a breakdown of the scores, showing which questions the user got correct, and which they didn't.
- At the bottom of the page there will be navigation buttons, with links to the start page, or to retake the quiz.

The create a quiz form:

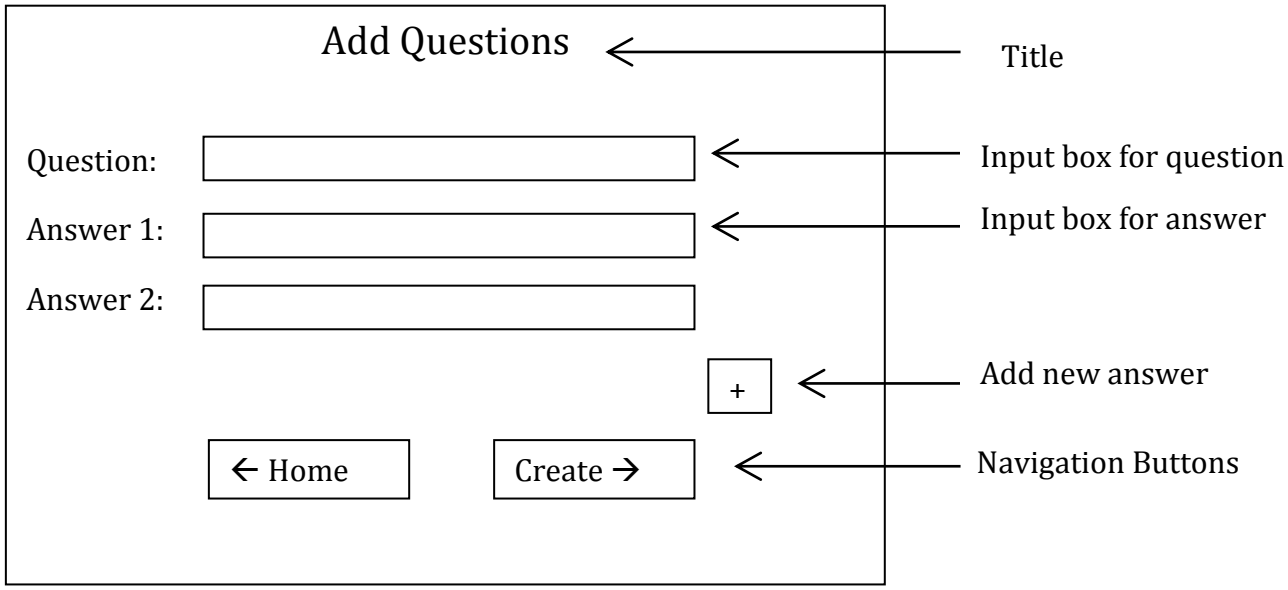
This form will allow the user to create a quiz. It will be quite plain, to keep it simple. It will be able to be accessed from the start page, and will lead onto the create a question page.



- The title will be Create a Quiz. It will be in a big font in the centre at the top, to keep it clear.
- There will be a text field where the user can enter the name of the quiz. It will be positioned near the top centre, because it is an important field.
- The level and subject will be a drop down menu, this will reduce mistakes, and make it faster to add information
- There will be a slide down menu for more options, which will allow the user to add more options. These will also be able to be amended at the end of the quiz.
- There will be navigation buttons at the bottom.

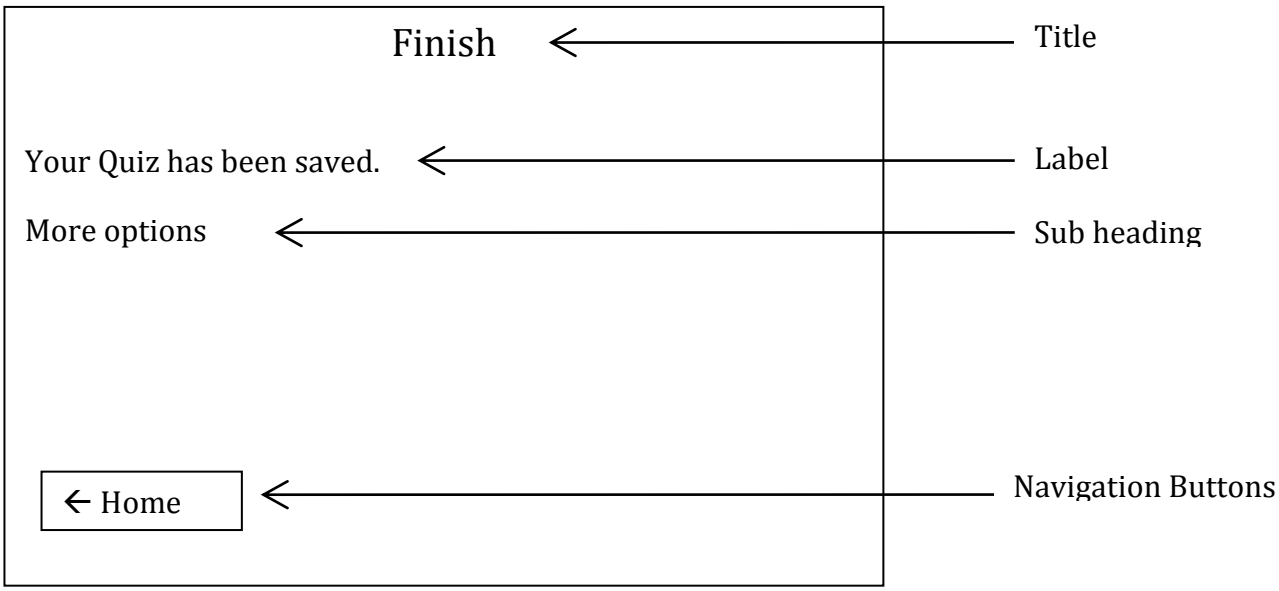
Anyone can create a quiz, not just an admin.

The create a question form:



- The title will be add questions in a medium large font, indicating to the user what form they are on.
- There will be an input field for the question, it will be big enough to fit the question in, and dynamically expanding using JavaScript, up to a certain length that will be the maximum question length.
- There will then be two answer fields, for the user to enter answers, once both have a value, using JavaScript another textbox will appear, and so on until there are a maximum of eight textboxes.
- There will be navigation buttons at the bottom.

The create finish form:



- There will be a title in a similar way to the other forms to keep it all uniform, it will be centred and a medium large font, and will say finish because it is the end of creating a quiz.
- Below this there will be some more options that will include setting a time limit for the quiz, order of questions and some more.
- There will be a home button at the bottom, redirecting the user to the start page.

Verification and Validation

Input	Verification or Validation check
Search Box in start form	Field length check – there will be no minimum length, and the maximum length will be 100 characters, because this is the longest a quiz title could possibly be. Format check – that there are no symbols, that would return no results.
Quiz ID Box in start form	Field presence check – Checks that there is a value entered in the quiz ID box before the user attempts to do that quiz, which will return an error. Field length check – To ensued that the value entered is of the right length, between 1 and 5 characters long. Range check – to ensure that the data inputted within the range. Between 0 and 99999. Format check – ensures that the data is made up of the right characters, so no letters of symbols, only numbers. Check digit – checks the number from the database and makes sure it is valid
Answers in quiz form	Presence check – checks a radio button or label has been selected before the user can proceed
Quiz name in create a quiz form	Field presence check – ensures that there is a value in the text box. Type check – checks that the data entered is of the right type, so contains words. Field length check – ensures that the length is no more than 100 character, because this will cause an error in the SQL. Also ensures that is it longer than 3 characters, because otherwise the title is likely to be too vague. Format check – to ensure that the data is of the correct format, and doesn't contain characters that could not be stored in varchar format in the database. Also will check that the data is sensible, so will reject it if all the letters are the same, of if there is too much punctuation. It will also search the database to ensure that there are no other quizzes with the same level and subject with that name.
Subject field in create a quiz form	Presence check – As it is a drop down menu, it must be selected.
Add subject field in create a quiz form	Field presence check – to ensure it has not been left blank if it is needed. Length check – checks that the data added is within the valid length between 3 and 50 characters. Format check – ensures that the data entered is spelt right, not yet in the database, of the tight format and correct.
Level field in create quiz	Presence check – checks that one of the values other than the default one is selected in the drop

form	down menu.
Question field in the 'create question's form	Field presence check – ensures that there is a value in the text box. Type check – checks that the data entered is of the right type, so contains words. Field length check – ensures that the length is no more than 150 characters, because this will cause an error in the SQL. Also ensures that is it not shorter than 3 characters, because otherwise the title is likely to be too vague. Format check – to ensure that the data is of the correct format, and doesn't contain characters that could not be stored in varchar format in the database. Also will check that the data is sensible, so will reject it if all the letters are the same, of if there is too much punctuation.
Answers field in the 'create question's form	Field presence check – ensures that there is a value in the text box. Type check – checks that the data entered is of the right type, so contains words. Field length check – ensures that the length is no more than 150 characters, because this will cause an error in the SQL. Also ensures that is it not shorter than 3 characters, because otherwise the title is likely to be too vague. Format check – to ensure that the data is of the correct format, and doesn't contain characters that could not be stored in varchar format in the database. Also will check that the data is sensible, so will reject it if all the letters are the same, of if there is too much punctuation.
More options radios in the finish create form	These will all be check boxes or optional menus, which can be left blank or selected from a list, so there are very few mistakes that could be made in this form.

Data Manipulation

As data is entered into the system, in all the forms it will be manipulated so it is in a suitable and efficient form and then added to an array with several dimensions. Once the array is complete and the user has reviewed what they have inputted, it will be inserted from the array to the correct place in the database. The array will then be destroyed.

The score array – will contain the users choices the be added into the database and marked at the end:

`QuestionID => userChoice -` Answer ID

Example data:

```
Array(Score)
  821 => int 1763
  822 => int 1767
  823 => int 1770
```

The array that will store the data in a session while the quiz is being created is a multi-dimensional array. Below are the array structures containing sample data to illustrate it.

The quiz info array – stores the information relating to the quiz, including title, level and subject.

```
Array(quizInfo)
  'name' => string 'Processor Components' (length=20)
  'level' => string 'A Level' (length=7)
  'sub' => string 'Computing' (length=1)
```

The questions array – stores all the questions in a one dimensional array with the question number as the identifier.

```
Array(questions)
  1 => string 'What part of the processor keeps track of the next instruction to
be executed?' (length=78)
  2 => string 'What is the current Instruction register?' (length=41)
  3 => string 'What is the function of the ALU?' (length=32)
```

The answers array is a multidimensional array, with the first dimension storing the question number, and the second dimension containing the answers and correct answers and then the values.

```
array
  1 =>
    array
      'answer' =>
        array
          'a' => string 'MAR' (length=3)
          'b' => string 'PC' (length=2)
          'c' => string 'CIR' (length=3)
          'd' => string 'ACC' (length=3)
      'correctaAns' =>
        array
          'a' => string 'f' (length=1)
          'b' => string 't' (length=1)
          'c' => string 'f' (length=1)
          'd' => string 'f' (length=1)
  2 =>
    array
      'answer' =>
        array
```

```
        'a' => string 'A register that contains the data r…' (length=67)
        'b' => string 'A register that contains the inform…..' (length=113)
    'correctaAns' =>
      array
        'a' => string 'f' (length=1)
        'b' => string 't' (length=1)
3 =>
  array
    'answer' =>
      array
        'a' => string 'Responsible for supervising the opera…' (length=72)
        'b' => string 'Provides the computer with logical a….' (length=66)
        'c' => string 'A storage location inside the process..' (length=40)
    'correctaAns' =>
      array
        'a' => string 'f' (length=1)
        'b' => string 't' (length=1)
        'c' => string 'f' (length=1)
```

System limitations

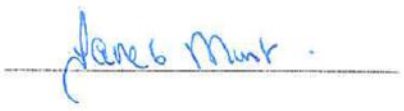
There will be some physical limitations to the new system, such as number limits, which I will address below.

- Maximum number of quizzes. Because the quiz ID field in the database can only hold up to 5 characters in each record, and gets auto incremented each time, the maximum number of quizzes stored in the database will be 99999 quizzes. Although it is unlikely that Mrs Munt will ever need to create more than 100, 000 quizzes.
- The questions cannot be more than 255 characters in length. This is another limitation caused by the database, because it would be too inefficient to reserve more than 255 characters per question, when the average question is 150 characters.
- The answers field will be limited to 150 characters.
- There is a maximum of 6 multiple choice answers, if there were any more than this, it would look messy while being displayed to the user.
- There will be a maximum of 100 questions per quiz.

SDLC

I plan on using the rapid application development (RAD) model of creating successive prototypes, which the end user will review, and give feedback and comments on, which I will then expand on and change. I have chosen this model, for a number of reasons. Firstly it is a linear model, which will be simple to implement and follow. Secondly documentation will be produced at every stage, which makes understanding the design very easy, and finally it will be tested after every stage, which will make the testing at the end much better. There will also be end user agreement after every stage, so I know that the final project will meet their exact needs.

Confirmation of End user involvement

Signed End user [Mrs Munt] 

Signed Developer [Alicia Sykes] 

Algorithm Development

Display search results

ALGORITHM DEVELOPMENT	
Algorithm Name:	Display Search Results
Algorithm Description:	Displays the quizzes matching the users search term

Parameters Passed IN	Parameters passed OUT	Variables
Users search term Data base connection	Search term Quiz ID Quiz Name Level Subject	Database_quizzes Quizzes Find Results Count I

Pseudo-code:

```
If there's a value in in the search box THEN {
  Database_quizzes = ("GET the quizzes table FROM quiz database")
  Quizzes = Put into an array from MySQL (Database-quizzes)
    Find = (GET value from search text box)
    Find = remove unnecessary punctuation and turn to all lower caps
  Search for Find in Quizzes array
    Results = put search results into array
  Count = Count number of results
  PRINT "Your search term "Find"returned "Count" result"
    If(count!= 1){PRINT "s"}
  For I = 0, I<count, I++{
    Quiz ID = GET ID where quiz name = results[count]
    PRINT Quiz Name where quiz ID = I
    PRINT Level where quiz ID = I
    PRINT Subject where subject ID. quizzes = subject ID. Subjects
    Value = Quiz ID}
```

Outputs	Testing Strategies Used	
Quiz Name Level Subject	I will dry run this and white box testing. The results are recorded below	
Design Modifications Problems that arose...	**Design Modifications Changes made...**	**Design Modifications Possible refinements...**
A common or short search term could return a lot of results, which can cause the page to not load correctly.	Search results limited to 12 per page, with next page button. Advanced search features to help users refine their search	Take a dimension out of the multi-dimensional array created from the SQL, so it only contains necessary information.

ALGORITHM DEVELOPMENT

Algorithm Name:	Did you mean function
Algorithm Description:	Suggests alternative searches if the users search term is misspelled or returns no results

Parameters Passed IN	Parameters passed OUT	Variables
Users search term Data base connection	Search term Quiz ID Quiz Name Level Subject	Database_quizzes Quizzes Find Results Count I

Pseudo-code:

```
If (numberOfResults == 0){

SuggestedWords =array();
for (i=0; i<NumberOfQuizzes; i++){
SuggestedWords[i]  = QuizInformation[$i]['Name'];}
shortest = -1;

foreach (words as word) {
    lev = find distance between characters(searchTerm, word);
    if (lev == 0) {
        closest = word;
        shortest = 0;
    break; }
    if (lev <= shortest || shortest < 0) {
        closest  = word;
        shortest = lev;}
}

 dbdidyoumean = mysql_query("SELECT * FROM quizzes
                             WHERE Name LIKE'%$closest%'");
        while (b = mysql_fetch_array(dbdidyoumean)){
            didyoumean[] = b; }
    countDidyoumean=count(didyoumean);
```

Outputs	Testing Strategies Used	
Did you mean Quiz Name Level Subject	I will dry run this and white box testing. The results are recorded below	
Design Modifications Problems that arose...	**Design Modifications Changes made...**	**Design Modifications Possible refinements...**
The users search term may be very wrongly spelled and the computer may not be able to guess it	Include spell check in search box	Take a dimension out of the multi-dimensional array created from the SQL, so it only contains necessary information.

ALGORITHM DEVELOPMENT

Algorithm Name:	Do a quiz
Algorithm Description:	The algorithm for when the user does a quiz

Parameters Passed IN	Parameters passed OUT	Variables
Quiz ID	Title Level Subject Questions Answers Score	

Pseudo-code:

```
//Get the quiz ID
IF there's a value in quiz_ID input box THEN{
     quizID = GET value of quiz ID input box }
ELSE IF the user has pressed a quiz box THEN{
     quizID = GET value of button pressed by user}
ELSE IF sent from quiz form THEN{
     quizID = GET from last form}

//Get quiz information from quiz ID
dbquizzes = (SELECT ID, Title, Subject, Level FROM Quizzes in quiz
WHERE ID = quizID)
  quizzes = put db into array(dbquizzes)
  title = quizzes[0]['title']


//Alocate the question number
IF first question THEN{
     questionNumber = 1 }
ELSE{questionNumber = GET questionNumber FROM last form + 1 }

//Get question information from quiz ID and question number
dbquestions = (SELECT ID, quiz ID, questionNum, question, answer,
COUNT FROM questions in quiz WHERE ID = quizID AND questionNum =
questionNumber)
  questions = put db into array (dbquestions)
  dispquestion = questions[0]['question']
  numquestions = questions[0]['count']
  questionID = questions[0]['ID']

//Get answer information
dbanswers = (SELECT ID, question ID, answer, correctAnswer, COUNT
FROM answers in quiz WHERE question ID = quiestionID)
  answers = put db into array(dbanswers)
  numAnswers = answers[0]['count']
FOR (i=0; i<numAnswers; i++){
  dispanswer = answers[i]['answer']
  correctAns = answers[i]['correctAnswer']}
```

```
//Add to the score
IF questionNumber>1 THEN{
  GET flatScroe Array
  scoreArray =unflatten flatScoreArray
  GET last correct answer
  GET user answer
  IF lastUserAnswer == lastCorrectAnswer THEN{
    scoreArray[questionID] = 1}
    ELSE scoreArray[questionID] = 0}
    flatScoreArray = flatten scoreArray
    SEND flatScoreArray
  }}

//Output the questions and answers to the user
PRINT title
PRINT "Question ".questionNumber. "out of ". numquestions
PRINT dispquestion
FOR (i=0; i<numAnswers; i++){
PRINT answers[i]['answer']
PRINT a radio button}
PRINT a submit button

//Send results to quiz form
IF questionNumber < numberOfQuestions THEN{
    Send results to quiz form}
ELSE IF questionNumber == numberOfQuestions THEN {
    Send results to results form}
SEND questionNumber
SEND selectedRadio
SEND quizID
SEND correctAnswer
```

Outputs	**Testing Strategies Used**	
Quiz Title Question Number Number of Questions Question Answers	White box testing, in the form of dry runs will be used to test the algorithm. The results are recorded below.	
Design Modifications Problems that arose...	**Design Modifications Changes made...**	**Design Modifications Possible refinements...**
Sending all the information along each form is inefficient	Information could be stored in a session, and only accessed when needed	The use of multi-dimensional arrays rather than multiple variables, would be more efficient and organised

ALGORITHM DEVELOPMENT	
Algorithm Name:	Results Page
Algorithm Description:	Get, record and display the results to the user

Parameters Passed IN	Parameters passed OUT	Global Variables	Local Variables
		quizID score numberOfQuestions questionNumber questionID question answer answerID	quizTitle flatScoreArray scoreArray total percentage takes dbquestions

Pseudo-code:

```
Connect to the database
GET quizID from last form
quizTitle = (SELECT title, ID FROM quizzes.quiz WHERE ID = quizID)
PRINT “Results”
PRINT quizTitle

//Mark last question
GET flatScroe Array
scoreArray =unflatten flatScoreArray
GET last correct answer
GET user answer
IF lastUserAnswer == lastCorrectAnswer THEN{
scoreArray[questionID] = 1}
ELSE scoreArray[questionID] = 0}

//Calculates how many questions are in the quiz
Dbquestions= (SELECT ID, quizID, question, COUNT WHERE quizID =
quizID)
NumberOfQuestions = dbquestions[0][‘count’]

//Prints total score for user
Total = add up sum of values in score array
Percentage = (total / numberOfQuestions * 100)
PRINT “Score = ” percentage “ %”

//Prints score breakdown for user
For(questionNumber = 1; questionNumber<numberOfQuestions;
questionNumber++){
GET questionID WHERE questionNumber == questionNumber
PRINT “Question Number: ”+questionNumber
PRINT question[questionNumber][‘questionNumber’]
PRINT “You got this question ”
IF (score[‘questionID]==1) THEN{
    PRINT “Correct” in green }
ELSE IF (score[“questionID”] ==0) THEN{
    PRINT “Incorrect” in red }
```

```
PRINT "The Answer was: " (GET answerID WHERE questionID = questionID
AND correctAnswer = true) PRINT answer(WHERE answerID = answerID)
}

//Add to average scores, and increment takes
For(questionNumber = 1; questionNumber<numberOfQuestions;
questionNumber++){
GET questionID WHERE questionNumber == questionNumber
answerID = GET selected answerID WHERE questionID = questionID
(UPDATE NumberofSelects in Answer.quiz to ++ WHERE answerID =
answerID)
(UPDATE takes in quizzes.quiz to takes ++ WHERE quizID = quizID)
```

Outputs	**Testing Strategies Used**	
	White box testing, in the form of dry runs will be used to test the algorithm. The results are recorded below.	
Design Modifications Problems that arose...	**Design Modifications Changes made...**	**Design Modifications Possible refinements...**

ALGORITHM DEVELOPMENT

Algorithm Name:	Create a quiz
Algorithm Description:	Create a quiz page, add information to database

Parameters Passed IN	Parameters passed OUT	Global Variables	Local Variables
Quiz title Subject ID Level	Array with quiz information	quizName Level SubjectID	

Pseudo-code:

```
Connect to the database
  QuizName = GET value of quiz name textbox
  Level = GET value of level drop down menu
  SubjectID = GET value of subject dropdown menu
    IF subjectID = AddNew THEN {
      newSubject = value of textbox
     (INSERT newSubject into subjects.quiz)
  SubjectID = (SELECT ID,subject FROM subjects.quiz WHERE subject=
newSubject) }
  Subject = (SELECT ID, subject FROM subjects.quiz WHERE
ID=subjectID)
(INSERT into quizzes.quiz name, level, subject ID: quizName, Level,
SubjectID)
```

Outputs	Testing Strategies Used	
Quiz created successfully, Data stored in db Add questions form shown	White box testing, in the form of dry runs will be used to test the algorithm. The results are recorded below.	
Design Modifications Problems that arose...	**Design Modifications Changes made...**	**Design Modifications Possible refinements...**
If quiz is created, then no questions are created there is a blank question	Could be stored in an array, and not added to the database until the quiz is finished	Subject and subject ID could be more efficient

ALGORITHM DEVELOPMENT

Algorithm Name:	Create a quiz a question
Algorithm Description:	Creates a question and adds it to the database

Parameters Passed IN	Parameters passed OUT	Variables
quizID questionNumber	Question Answers	quizID questionNumber question answer ansCount i correctAnswer

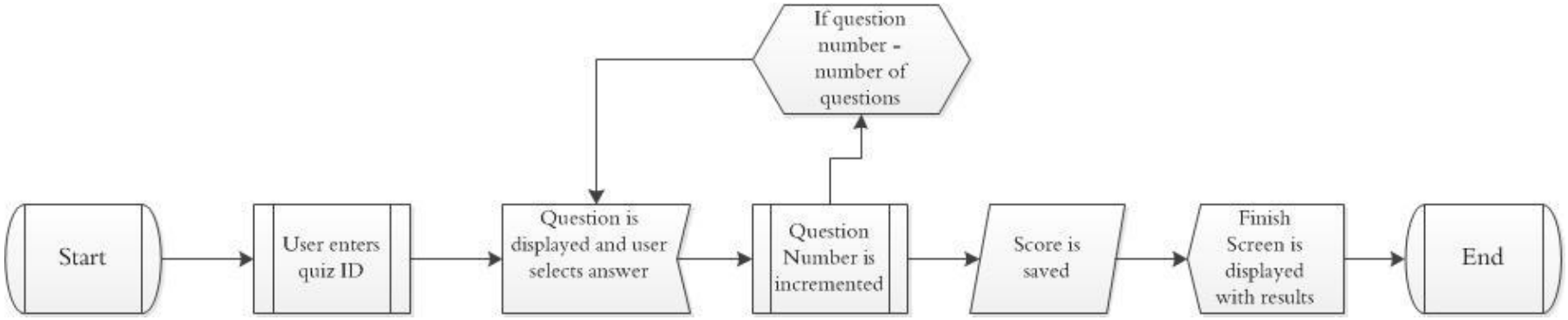
```
Connect to database
Find out if it is the first question
GET 'quizID'
(SELECT COUNT questions WHERE quizID = quizID)
questionNumber = count
question = GET value of question box
ansCount = GET number of answers
for (i=0; i<ansCount; i++)
answer[i]
correctAnswer = GET radio value correct answer
(INSERT into question.quiz quizID, questionNumber, question)
GET questionID WHERE question = question and quizID= quizID)
(INSERT into answers.quiz questionID answer[i] correctAnwer)
```

Outputs	Testing Strategies Used	
Data entered into database Message informing user that data was imputed correctly Or error message	Dry run	
Design Modifications Problems that arose...	**Design Modifications Changes made...**	**Design Modifications Possible refinements...**
If the user presses finish, then goes back to this screen to add another question, when they press next it will add a duplicate of this question into the database with the same question number, which will cause an error during the running of the quiz	The questions could be added to an array stored in a session, then not added into the database until the final screen	SQL queries could be reduced, they could all be moved to the finish slide, and not added into the database until the finish screen. The information could be stored in an array in a session until then.

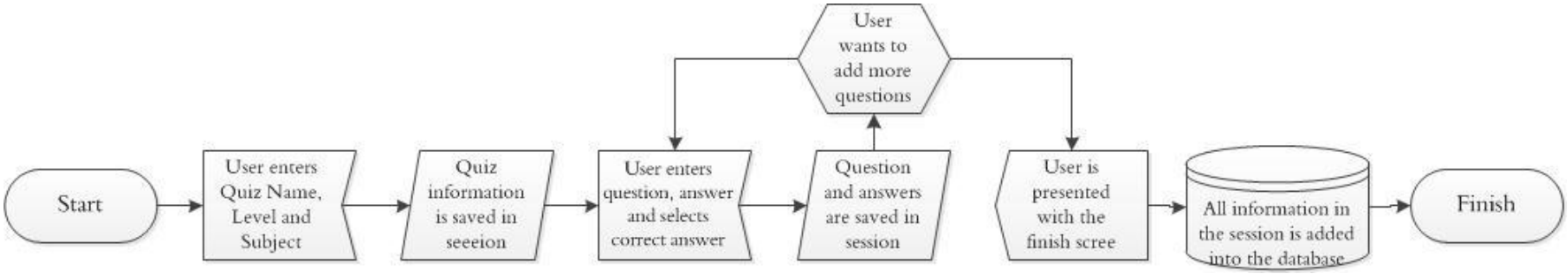
System Data Flow

I will create some data flow diagrams to demonstrate the flow of data through the program.

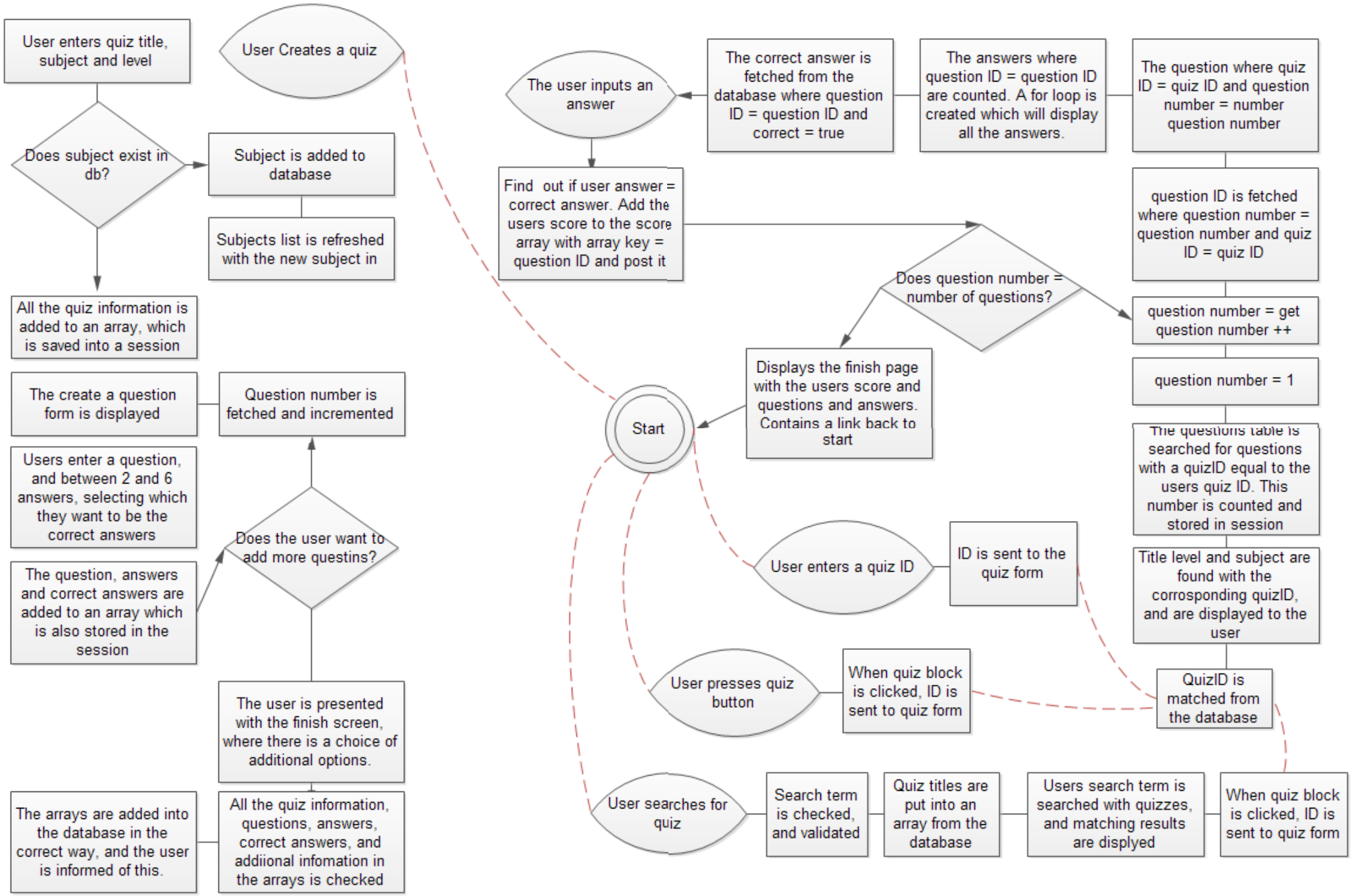
Do a quiz:



Create a quiz:



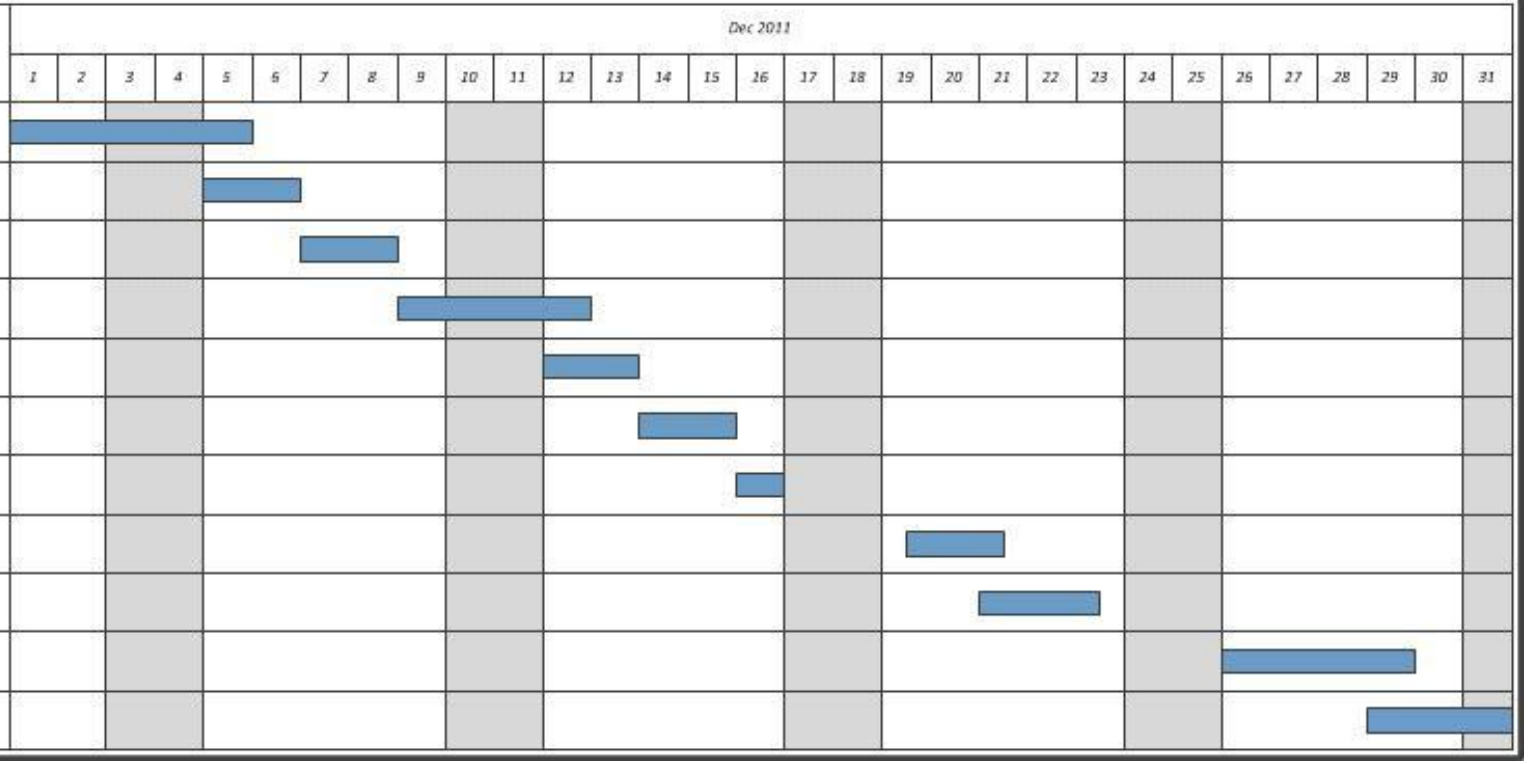
The data flow diagrams show how the data flows through the program, from when it is inputted by the user, how it is processed, what is saved and what is outputted back to the user. This fits in with the pseudo code created above.


User Creates a quiz
User enters quiz title, subject and level
Does subject exist in db?
Subject is added to database
Subjects list is refreshed with the new subject in
All the quiz information is added to an array, which is saved into a session
The create a question form is displayed
Question number is fetched and incremented
Users enter a question, and between 2 and 6 answers, selecting which they want to be the correct answers
Does the user want to add more questins?
The question, answers and correct answers are added to an array which is also stored in the session
The user is presented with the finish screen, where there is a choice of additional options.
All the quiz information, questions, answers, correct answers, and addiional infomation in the arrays is checked
The arrays are added into the database in the correct way, and the user is informed of this.
Start
The user inputs an answer
The correct answer is fetched from the database where question ID = question ID and correct = true
The answers where question ID = question ID are counted. A for loop is created which will display all the answers.
The question where quiz ID = quiz ID and question number = number question number
Find out if user answer = correct answer. Add the users score to the score array with array key = question ID and post it
question ID is fetched where question number = question number and quiz ID = quiz ID
Does question number = number of questions?
question number = get question number ++
question number = 1
Displays the finish page with the users score and questions and answers. Contains a link back to start
The questions table is searched for questions with a quizID equal to the users quiz ID. This number is counted and stored in session
User enters a quiz ID
ID is sent to the quiz form
Title level and subject are found with the corrosponding quizID, and are displayed to the user
User presses quiz button
When quiz block is clicked, ID is sent to quiz form
QuizID is matched from the database
User searches for quiz
Search term is checked, and validated
Quiz titles are put into an array from the database
Users search term is searched with quizzes, and matching results are displyed
When quiz block is clicked, ID is sent to quiz form

Time plan

I have created a GANT chart to plan the timings needed for designing effectively. I will start the design at the end of November, and hope to have finished it be then end of December, when I will test the final version, and then install and implement the solution for the end user. Because I am using the waterfall model, I will create successive prototypes and meet with the end user regularly to get their feedback and modify it accordingly.

ID	Task Name	Start	Finish	Duration
1	V0.1	01/12/2011	05/12/2011	3d
2	V0.2	05/12/2011	06/12/2011	2d
3	V0.3	07/12/2011	08/12/2011	2d
4	V0.4	09/12/2011	12/12/2011	2d
5	V0.5	12/12/2011	13/12/2011	2d
6	V0.6	14/12/2011	15/12/2011	2d
7	V0.7	16/12/2011	16/12/2011	1d
8	V0.8	19/12/2011	21/12/2011	2d
9	V1.0	21/12/2011	23/12/2011	2d 4h
10	Testing	26/12/2011	29/12/2011	4d
11	Instaling	29/12/2011	04/01/2012	5d



Test Plan

Each part of the program will be tested with a variety of data.
Normal data – The system will be tested with data that it was expecting, and what it was designed to use. Its path will be traced through the program.
Extreme or boarder line data – The program will be tested with data that it is not expecting, like bi numbers, long words and unexpected characters.
Erroneous data – invalid data, to check that the system refuses the data. If it is not refused then it may cause an error later on in the program, when it is unable to process the data. There may be some unexpected results.

Test Strategy

Before I can pass the solution to the end user, I must test it toughly to make sure that it is fully stable, no bugs and works how the user would expect it to. I will test every part of the solution fully.

White box testing – I will do dry runs through the pseudo code to test the algorithms paths.

Black box testing- I will test the functionally of my program, without looking at the code at all. I will use it as I was the end user, and I would test for bugs, or not fully working parts. I will test my pseudo code through using dry runs, and trace tables to check if I get the expected results.

Alpha testing – During the development of my program, at every stage I will carry out a series of tests. I will test the inputs and outputs of all the new or modified features of that version. I will record the results in a table and include them.

Beta testing – When the program is nearly fully developed I will allow potential users to use it as it will be used, and ask them to send feedback and report bugs.

Dry Runs

I am going to test all the algorithms, and record the results below, white box testing, I will just trace the values through the pseudo code, and record the inputs and outputs in the tables below

The search form:

Test Name	**Test Description**	**Type of data**	**Input**	**Expected output**	**Actual Output**	**Notes**
Normal term with a result	A normal and expected term will be searched for	Normal	Atom	Your search result *Atom* returned 1 result	Your search result *Atom* returned 1 result	Passed
Search term with no results	A search term that has no results	Erroneous	Aotm	Your search term *Aotm* returned 0 results. Did you mean *Atom*?	Your search term *Aotm* returned 0 results. Did you mean *Atom*?	Passed, the similar string function should work to return suggestions similar to words in a list, in this case the list of quiz titles
Search term with many results	A short search term that will have many quiz results.	Extreme	A	Your search term of *A* returned 7 results.	Your search term of *A* returned 7 results.	Passed. It correctly returned all 7 quizzes with A in their title
Blank search	Blank search should return all quizzes in database, on several pages	Erroneous	[blank]	Showing all quizzes	Showing all quizzes	To start with this did not pass, because I had forgotten to add the else show all quizzes at the end. But now it works
Punctuation	A search result containing punctuation	Erroneous	A;t!o?m	Showing 1 search result for *Atom* corrected from *A;t!o?m*	Showing 1 search result for *Atom*	This correctly removed illegal punctuation from ;!?, but didn't include corrected from part

The do a quiz form:

Test Name	**Test Description**	**Type of data**	**Input**	**Expected output**	**Actual Output**	**Notes**
Expected answer	The user clicks a radio button within the expected range	Normal	B radio	Input will be recorded, and the next page will be shown	Input was recorded, and the next page was shown	As expected
No answer selected	To test if the form will allow the user will proceed without selecting a radio button	Erroneous	[blank]	Please select a radio	Please select a radio	The output box tells the user to select a radio button, as expected

The create a quiz form:

Test Name	Test Description	Type of data	Input	Expected output	Actual Output	Notes
Quiz name as expected	If quiz name is added as expected	Normal	Quiz Name	Quiz saved successfully	Quiz saved successfully	Works as long as subject and level were also correct
Quiz name with illegal characters	Quiz name has punctuation	Erroneous	Quiz Na;m!!e	Characters are not allowed	Characters are not allowed	Works as long as subject and level were also correct
Level not selected	To check that the user has entered a level and avoid SQL error	Normal	[blank]	Please select a level	Please select a level	As expected
Level selected as expected	To check that when the level is selected it will work correctly	Normal	GCSE	Next page shown, level stored in array in session	Next page shown, level stored in array in session	As expected, as long at title and subject are also selected
Subject selected as expected	To check that when the subject is selected it will work correctly	Normal	Maths	Next page shown, subject stored in array in session	Next page shown, subject stored in array in session	As expected, as long at title and level are also selected
Subject not selected	To check that the user has entered a subject in the drop down menu	Normal	[blank]	Please select a subject	Please select a subject	As expected, as long as the level and name are alright
If the whole form is incomplete	To check that the user hasn't just left the whole form blank	Erroneous	[blank]	Please enter a quiz name, subject and level	Please enter a quiz name, subject and level	As expected

Create a question form:

Test Name	Test Description	Type of data	Input	Expected output	Actual Output	Notes
Expected question	The user enters a question and answers as expected	Normal	What is the square root of 25?	The add new question form should be shown or finish screen	The add new question form was shown	As expected, and question was added to the array
Question is too short	The user enters a question into the question box, that is under 10 characters, or leaves it blank	Erroneous	Wha	A dialog box should appear, telling the user that the question is too short or has been left blank	A dialog box appeared, telling the user that the question is too short	As expected
Question is too long	The user enters a question that is more than 250 characters long	Erroneous	Some very long question	A dialog box should appear, telling the user that the question is too long	A dialog box appeared, telling the user that the question is too long	As expected
Less than two answers are entered	The user enters a question, and only one or zero answers	Erroneous	Under two answers	A dialog box appeared, telling the user that there is a minimum of two answers	A dialog box appeared, saying minimum of two answers to be entered	As expected
No correct answer	The user forgets to select which answer is correct	Erroneous	No correct answer	A dialog box appeared, telling the user that they must select	A dialog box appeared, telling to select a correct	As expected

				a correct answer	answer	

Section C – Software Development

I am going to develop the solution using the waterfall model, this will include creating successive prototypes and then building on them after each stage closely following the system specifications and the users input. I will meet with the end user after I have developed each stage in order to get their input on things which may need to be changed.

Stages of Development:

- Form Development
- Code Development
- White box testing
- Black box testing
- Discuss with end user

Stages of the program

- V0.1
- V0.2
- V0.3
- V0.4
- V0.5
- V0.6
- V0.7

For each version I will give details on:

- Objectives for that version
- Annotated screen designs
- Verification and Validation added (if any)
- Data Structures modified
- The new functions added into this version, all annotated code
- The user specifications that this meets
- Testing details and results
- Details on the issues found during testing and how they were corrected
- End user involvement
- Review of that version

Version 0.1

Objectives – V0.1

The main objectives of version 0.1 is to create a system that will loop through questions in a quiz displaying the answers to the user in the form of 4 radio buttons, and allow the user to select one and proceed to the next question. When all questions are finished the results screen should be displayed and should show the user the sum of their score. There must also be a start screen that will let the user enter the quiz ID of a quiz in a database from which they would like to do.

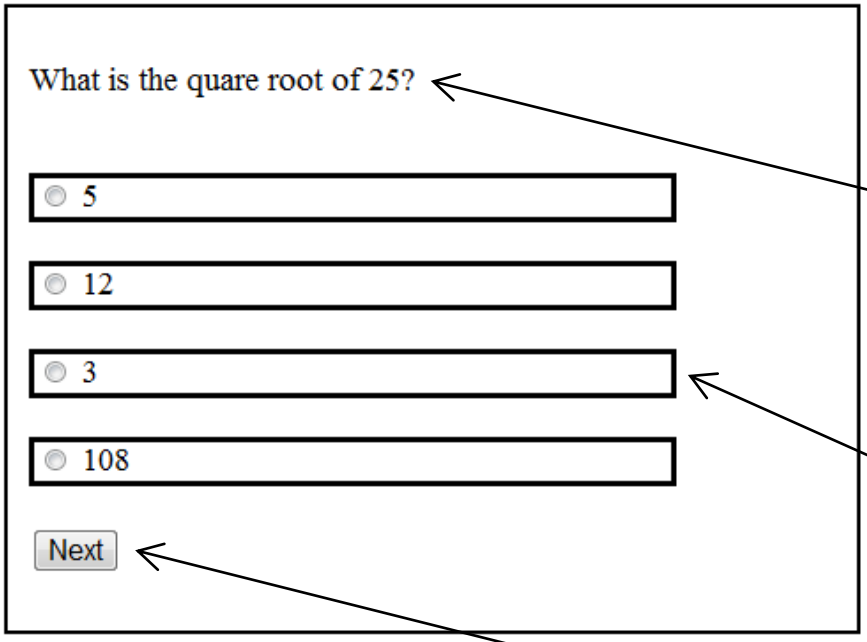
Screen Designs – V0.1

The Start form:

Label, indicating to the user what they should enter in the text box

Text box, for the user to enter quiz ID. Called quizID

Button for the user to submit the form. Located near to the textbox to make it clear to the user where it is. Alternatively the user could press the enter key.

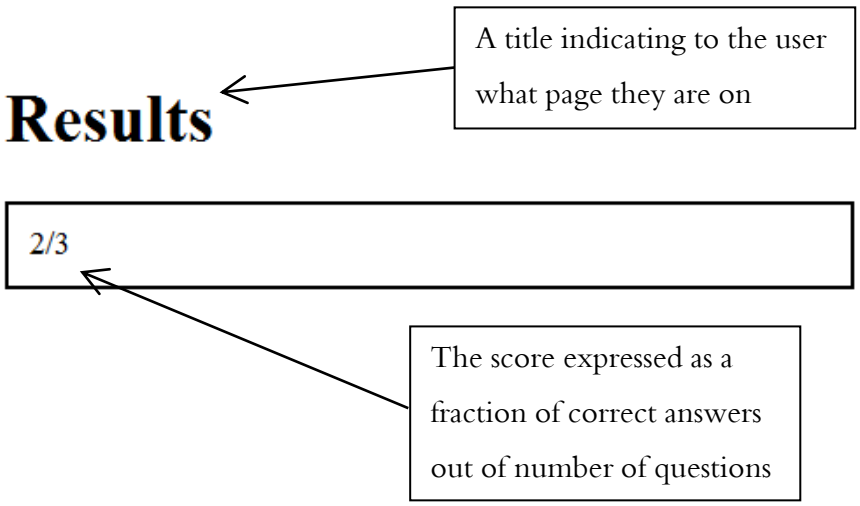


The question printed in a clear font at the top of the screen

The answers printed in radio buttons inside boxes to keep them separate and improve clarity

A summary of the quiz title, and question number at the bottom of the screen

A next button at the bottom of the screen, for the user to press once they have selected an answer


A title indicating to the user what page they are on
Results
2/3
The score expressed as a fraction of correct answers out of number of questions

Code – V0.1:

- Start form – There is no PHP for the start screen, it is just a HTML form with and input box and label.
- The questions form has some PHP and SQL on it that fetches the questions from the database and displays them accordingly.

Get the quiz ID, in this bit of code, it checks that if the use answer is set, because it is a radio button it will send the value. If it is not set then it must be the first question, the quiz ID should be fetched from the value that the use inputted on the start screen. If it is set the quiz ID is returned from the hidden input on the previous screen.

```
<?php // Get quiz ID
   if (isset($_POST['userAnswer'])) {
$quizID = ($_POST['quizID']);}
    else{ $quizID = ($_GET['quizID']);} ?>
```

To calculate the number of questions in each quiz I have used an SQL query that counts every question ID where the quiz ID field is equal to the quiz ID fetched from above. This result will be stored in a variable called $noq, number of questions, and will be used on numerous occasions throughout the form.

```
<?php //Calculate the number of questions in selected quiz
$dbnoq = mysql_query("SELECT ID, COUNT(quiz_ID), quiz_ID FROM Questions WHERE
quiz_ID =  $quizID ");
$d = mysql_fetch_array($dbnoq);
$noq = $d[1];
?>
```

This function finds out which questions and answers to output to the user. It starts of by finding out if it is the first question in the quiz, by whether or not there was a question before it, using the isset function. If it is not the first question, it must then check whether or not it's the last question. It does this by finding the last question number from the radio buttons, then adding one to get the current question number, if this is equal to the number of the questions in the quiz. If it is it calls the function lastQ() which just changes what form the question is submitted to. If it is not the last question, then it calls the function notLastQ(), which sends the form back to the quiz form when it is submitted.

```
<?php //Which question and answers to output
if (isset($_POST['userAnswer'])) {
        $selected_radio = $_POST['userAnswer'];
        $lqn =  $_POST['qn'];
    if ($lqn + 1 < $noq) {
            $qn = $lqn + 1;}
    else{ "----END OF QUIZ----";
            }
}
else{
        $qn = 0;}
        $lqn = $qn -1;

    if ($lqn == $noq -2){
        lastQ(); }
    else{ notLastQ(); }
?>
```

The next function fetches the quiz information, question and answers from the database or the current question. It uses SQL queries and then the MySQL_fetch_array() function to store the results in an array.

```
<?php //Fetches the right questions and answers from db
$dbquizzes = mysql_query("SELECT * FROM `quizzes` WHERE 1")or die(mysql_error());
$quiz = array();
while ($i = mysql_fetch_array($dbquizzes)){
    $quiz = $i;}

$dbquestions = mysql_query("SELECT * FROM questions WHERE quiz_ID = $quizID AND
QuestionNumber = $qn + 1") or die(mysql_error());
$question = array();
while ($i = mysql_fetch_array($dbquestions)){
    $question= $i;}
$questionID =  $question['ID']; //Get the unique ID of current question

$dbanswers = mysql_query("SELECT * FROM answers WHERE question_ID =
$questionID")or die(mysql_error());
$answers = array();
while ($i = mysql_fetch_array($dbanswers)){
    $answers[] = $i;
}
?>
```

The information from above is then put into variables from the arrays, so that less code is needed for printing them to the user later on.

```
<?php //Put db values of questions and answers into variables
$title = ($quiz[$quizID-1]['Name']); // The title of the quiz
$ask = $question[0]['Question']; //The current question
$optionA = $answers[0]['Answer1']; // option a of current question
$optionB = $answers [0]['Answer2']; // option b of current question
$optionC = $answers [0]['Answer3']; // option c of current question
$optionD = $answers [0]['Answer4']; // option d of current question
$correctAnswer = ($answers[0]['CorrectAnswer']); //The correct answer of current
question
?>
```

The score function keeps track of the score throughout the running of the program. Firstly it declares the score array. Then it checks whether or not it's the first question, using an if statement and the isset function checking the radio button on the previous form. If it's not the first question, it fetches the serialised score from the hidden input on the previous form. It then unserializes it. The user answer and last correct answer are fetched. An if statement is used to compare the two, if they are equal then the question ID is used as an array key and 1 is the value, else the value is 0. The array is then reseralised and resent to the next form.

```
<?php //Calculates and keeps track of the score
$score = array();
if (isset($_POST['userAnswer'])) {
    $seriScore = $_POST['seriScore'];
    $score = unserialize($seriScore);
    $userAnswer = $_POST['userAnswer'];
    $lastCorrectAnswer = $_POST['correctAnswer'];
    if($userAnswer == $lastCorrectAnswer){
        $score[$questionID] = 1; }
    else{$score[$questionID] = 0; }
        $newSeriScore = serialize($score);}
else{$newSeriScore = serialize($score);}
?>
```

There will also be a HTML form at the bottom of the screen containing the label's, text, radio buttons, div's and submit button.

- The results from just fetches the array that the score is stored in and finds the sum of it.
  The score function will calculate and output the final score out of the number of questions to the user at the end of the quiz. It fetches the score from the previous form, unserializes it, finds the sum of the array and then outputs it. It also recalculates the number of questions in the same was as above, and outputs that too.

```php
<?php // Calculate and echo the final score
$seriScore = ($_POST['seriScore']);
$score = unserialize($seriScore);
$scoreValue = array_sum($score);
echo $scoreValue. "/".$noq;
?>
```

User specifications met – V0.1:

- Loop through each question in order
- Record users score while they do the quiz
- Show total score at the end of the quiz
- A storage system to keep hold of all quizzes

Data Structures – V0.1

Below is the structure for the database in V0.1.

Name	Records	Storage	Type	Size
answers	10	MyISAM	latin1_swedish_ci	2.6 KiB
questions	10	MyISAM	latin1_swedish_ci	2.9 KiB
quizzes	3	MyISAM	latin1_swedish_ci	2.1 KiB
3 tables	23	InnoDB	latin1_swedish_ci	7.6 KiB

This is the quizzes table. There is space for the quiz ID, Name, Level and Subject.

Field	Type	Collation	Attributes	Null	Default	Extra
ID	int(5)			No	None	AUTO_INCREMENT
Name	varchar(100)	latin1_swedish_ci		No	None	
Level	enum('KS1','KS2','KS3','GCSE','A Level')	latin1_swedish_ci		No	None	
Subject	varchar(100)	latin1_swedish_ci		No	None	

This is some sample data entered into the quizzes table:

ID Unique ID of Quiz	Name Descriptive Text	Level	Subject
1	test	KS1	Maths
2	Atoms	GCSE	Physics
3	World War II	KS3	History

This is the questions table, it will be where the questions are stored.

Field	Type	Collation	Attributes	Null	Default	Extra
ID	int(5)			No	None	AUTO_INCREMENT
quiz_ID	int(5)			No	None	
QuestionNumber	int(3)			No	None	
Question	varchar(250)	latin1_swedish_ci		No	None	

Below is some sample data in the questions table:

ID	quiz_ID	QuestionNumber	Question
1	1	1	What is 5 + 5 ?
2	1	2	What is 12 - 4 ?
3	1	3	What is the quare root of 25?
4	2	1	What is the centre of an atom called?
5	2	2	What is the nucleus made up of?
6	3	1	At the start of the European theatre in 1939, ther...
7	3	2	The offensives in Poland and France taught the Eng...
8	3	3	Which country denied much needed border access to ...
9	3	4	Germany turned its aggression on its old foe Franc...
10	3	5	Italy decided to join Hitler's march of conquest i...

This is the answers table, it will hold information on the answers.

Field	Type	Collation	Attributes	Null	Default	Extra
ID	int(5)			No	None	AUTO_INCREMENT
question_ID	int(5)			No	None	
Answer1	varchar(100)	latin1_swedish_ci		No	None	
Answer2	varchar(100)	latin1_swedish_ci		No	None	
Answer3	varchar(100)	latin1_swedish_ci		No	None	
Answer4	varchar(100)	latin1_swedish_ci		No	None	
CorrectAnswer	char(1)	latin1_swedish_ci		No	None	

This is some sample data entered into the answers database.

ID	question_ID	Answer1	Answer2	Answer3	Answer4	CorrectAnswer
1	1	8	9	10	11	c
2	2	7	6	9	8	d
3	3	5	12	3	108	a
4	4	Proton	Neutron	Nucleus	Electron	c
5	5	Protons and Electrons	Protons and Neutrons	Electrons and Neutrons	Protons, Electrons and Neutrons	b
13	9	Battle of Gibraltar	Battle for the English Channel	Battle of Britain	Battle of Dover	c
12	8	Finland	Norway	Denmark	Sweden	a
11	7	Lightning War	Panzer Assault	Mechanized War	Combined Forces	a
10	6	Poland and the Soviet Union	Japan and China	Japan and Malaya	North Korea and South Korea	b
14	10	Ruini	Umberto II	Mussolini	De Gasperi	c

Testing – V0.1:

Test Name	Test Description	Input	Expected output	Actual Output	Notes
Enter quiz ID correctly	To check that the user is redirected to the quiz after they have correctly entered the quiz ID	76	AS Economics quiz	AS Economics quiz	The enter quiz ID box works correctly, and the quiz form gets the information from the start form correctly.
Loop through each question in order	To check that the quiz program will start at question 1 of the selected quiz and display each question until the last question, when it should display the results form	Quiz ID, then the answer of question 1, presses next, the answer of question 2, press next…	The quiz title, then question 1 with all the possible answers, then question 2… until the last question then the finish screen	The quiz title, then question 1 with all the possible answers, then question 2… until the last question then the finish screen	Works correctly, although had to be modified, because at first instead of finishing it just went on to the next quiz.
Show total score at end of quiz	To see if the score function works. Some questions will be entered correctly, some incorrectly	Quiz ID, correct answer, incorrect answer, incorrect answer	Results. 1/3	Results. 1/3	Everything worked as expected
Show total score at end of quiz	To see if the score function works. All questions will be entered correctly.	Quiz ID, correct answer, correct answer, correct answer	Results. 3/3	Results. 2/3	The score was one lower than it should have been by one mark. The last question was not scored

Test the storage system	Run an SQL query. I will do this for the quizzes, questions and answers table	SELECT * FROM `quizzes` LIMIT * ORDER `ID`	Return all the quizzes	Returned all quizzes	Worked

Issues found and addressed – V0.1:

There was an issue with the last question not being scored, this was because the score was incremented when the result was resent to the quiz form, but on the last question it was sent to the results form, where a score is not added to. I have corrected this now.

Also when the user enters the quiz ID incorrectly, by either entering a non-existent ID, illegal characters or leaving it bank, there are no validation checks in place to stop that returning a syntax error, from trying to access a non-existent SQL record.

Also if the user forgets to enter an answer there is nothing in place to stop them proceeding, and there will be an undefined index syntax error in the next screen, and the score will be one lower.

Another issue is that the code is not as efficient as it could be, and some of it relies on information that could be missed out or incorrect. This makes the program quite unstable and volatile. I will correct this in V0.2

Finally there are many functions missing, so it would be of no use to the end user. The only way of writing a quiz is to do it directly into the database, and there is no way to search for quizzes, or to view a breakdown of the scores.

End user involvement – V0.1:

Version 0.1	
Pros: • Clear • Works	Cons: • The user can not create a quiz. • The layout is quite Plain. • There seems to be a problem with the score (last question isn't scored)
Notes:	

Confirmation of End user involvement

Signed End user [Mrs Munt] Janet Munt

Signed Developer [Alicia Sykes] Alicia Sykes

Above is the response for V0.1 from my end user.

Review of V0.1

In version 0.1 there is limited functionality, and the design is minimal, although it does work and meets the objectives of that version and was developed on time. The testing has shown that there was a bug where the last question wasn't scored, I will aim to correct this in the next version. My end user involvement showed that the next most important feature would be the ability to create quizzes. I will develop this for the next version.

Version 0.2

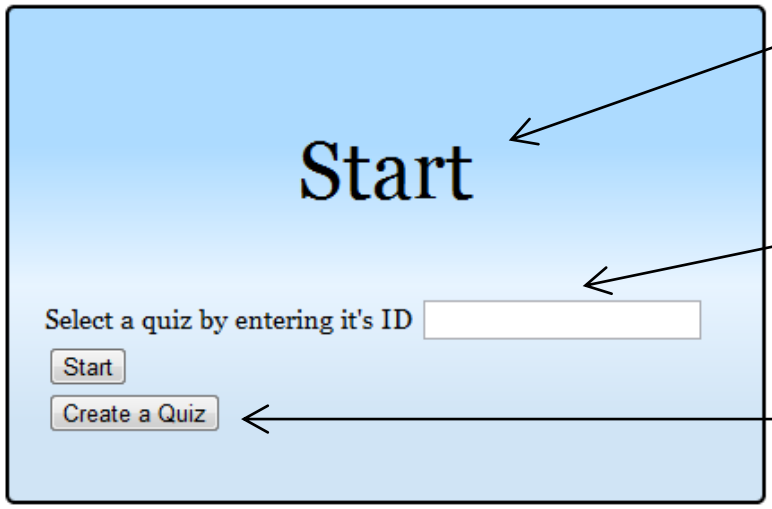
Objectives – V0.2

The main objectives of version 0.2 are to create the form that will allow the user to create quizzes and save them to an SQL database. I will also correct the issues found during the testing stage of V0.1 such as the last question not being scored. I plan to improve the design a little bit too.

Screen Designs – V0.2

The Start form:

The start screen is very similar to before, the main difference being the link to 'The Create a Quiz Page'. Also there is a gradient fill that makes it clearer to the user, and the corners of the block have a border radius of 4px.



Title at the top, so the user knows what screen they are on. Large font.

The select quiz ID label and textbox and he start button have stayed the same.

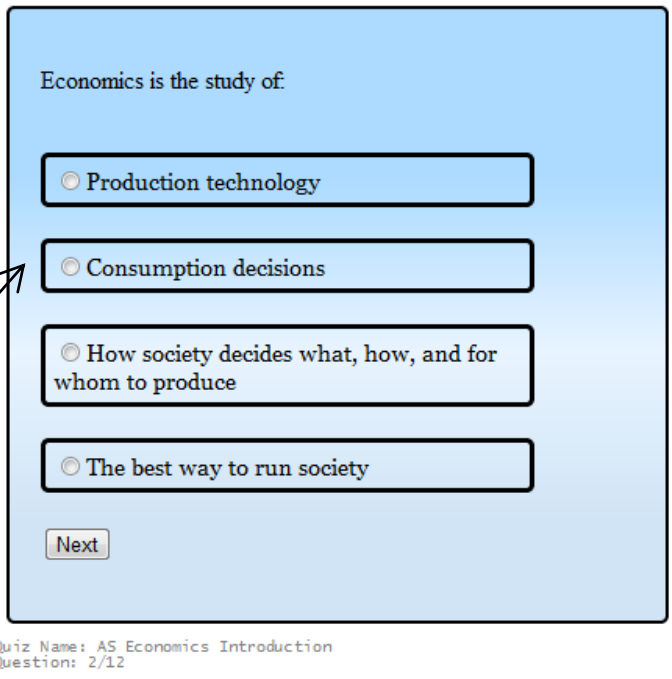
Button linking toThe Create a Quiz Page.

The quiz form:

The quiz form is also very similar from last time, with a background fill gradient and curved corners.

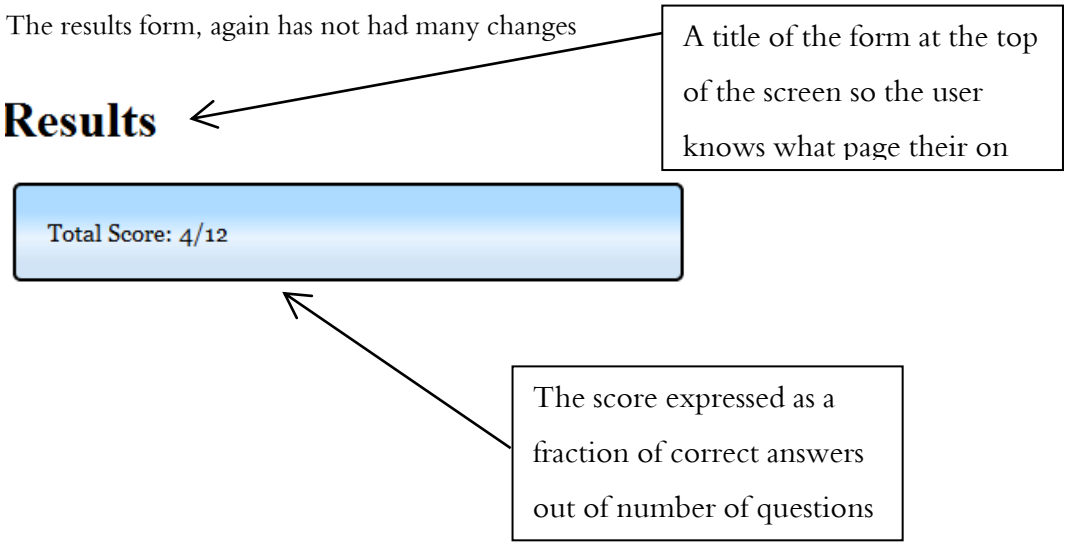
The questions and answers are displayed in a similar way as to how they were before.

There is still a next button at the bottom and a summary below that.



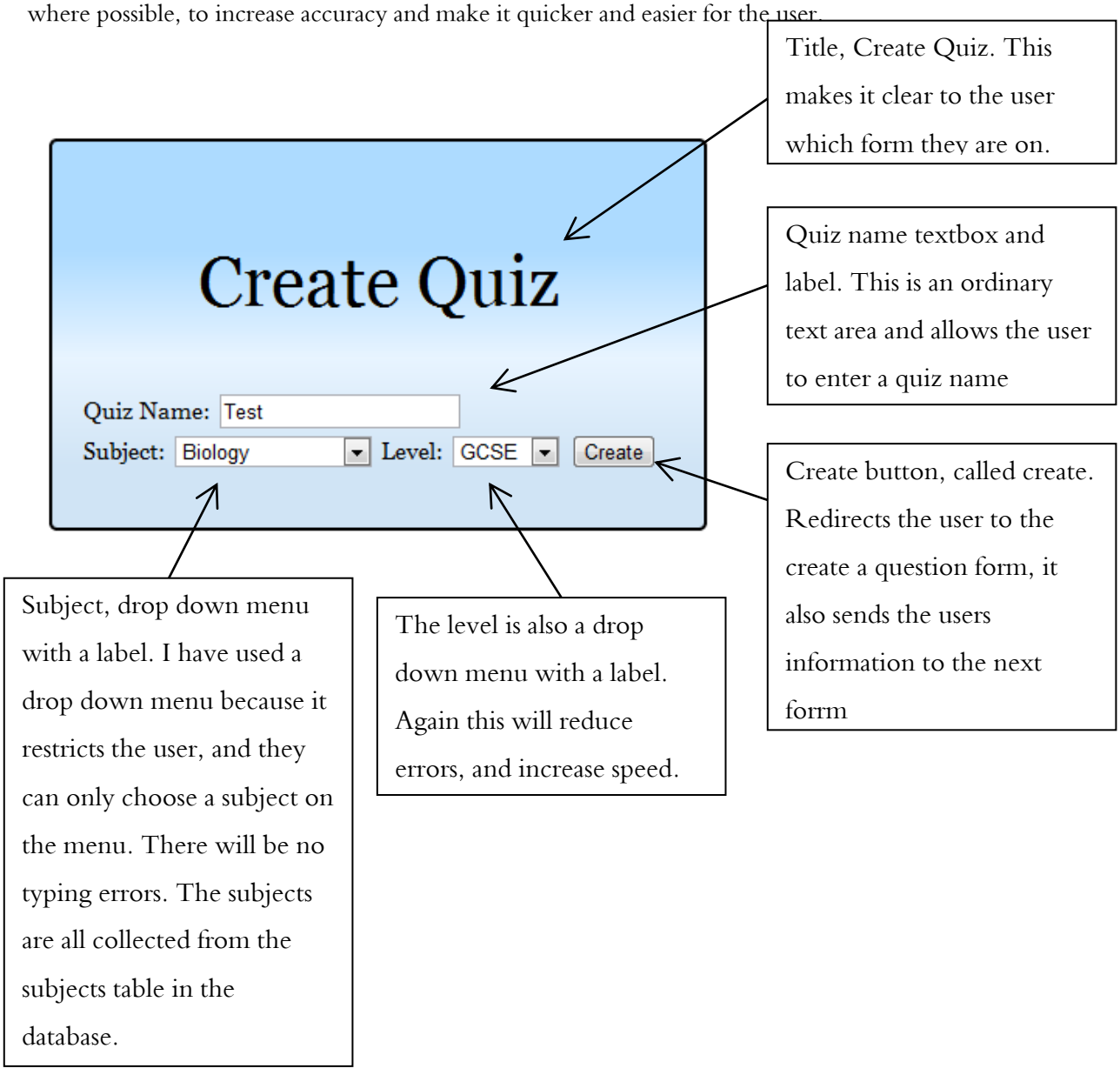
The Results Form:

The results form, again has not had many changes



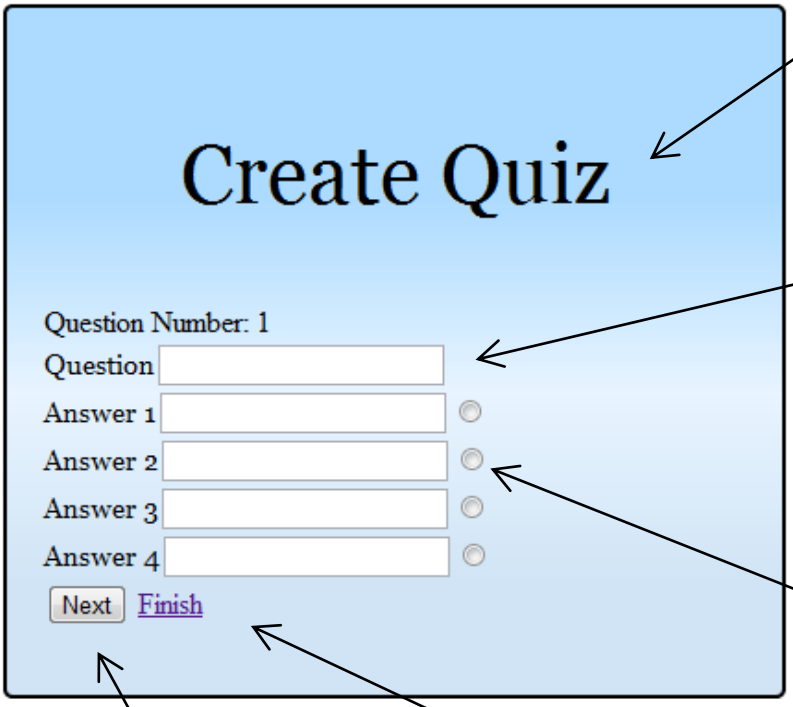
The Create a Quiz form:

This is a new form added, and will allow the user to create a quiz that will then be saved to the MySQL database in the quizzes table. It will allow the user to specify a quiz title, subject and level. I have used drop down menus where possible, to increase accuracy and make it quicker and easier for the user.



The Add questions to quiz form:

This from will allow users to add questions to the quiz just created.



The title will be in large text and centred, clearly indicting to the user what form they're in.

The question field will allow for 250 characters to be entered. There is a label indicating to the user what should be entered.

Correct answer radio buttons. The user will select a radio button indicating which of the answers thy entered is correct. Only one correct answer can be entered.

The next button will take the user to the next form, and save the current questions and answers entered.

The finish link will take the user back to the start screen.

Code – V0.2:

- The quiz form:

There is a basic validation check that checks the use has checked/ selected a radio button before they proceed to the next form. It uses JavaScript, and it's not very good, but it works for the four answers.

```
function validateForm(){
valid = true;
 if (  ( document.questions.userAnswer[0].checked == false )
    && ( document.questions.userAnswer[1].checked == false )
    && ( document.questions.userAnswer[2].checked == false )
    && ( document.questions.userAnswer[3].checked == false ) )
    { alert ( "Please select an answer" );
        valid = false;
    }
    return valid;
    }
```

- The create a quiz form:

First it declares and initialises the main variables that will be used, to avoid errors.

```
<?php //Declare Variables
$level = null;
      $subject = null;
      $selectedSub = null;
      $qn = 0;
 ?>
```

There is also a MySQL fetch query that gets all the subjects from the subjects table, and puts them into an array, which is then looped through, to display all the subjects in a drop down menu.

- The add questions form:

Firstly the form fetches the question number from the previous form, using the POST method.

```
<?php //Get question number
$qn = ($_POST['qn']);
?>
```

Next it decides what form to display, if the question number is equal to -1 then it was sent from the create quiz page, so an if statement tells it to do all the code that will save the quiz information.

```
<?php // what screen to display
if ($qn == -1)  { // is this the first question?
?>
```

If it's sent from the create quiz form, one of the first things it will do is initialise question number to zero, so that when it's sent to the next form and incremented it will equal zero.

```
<?php //Question number = 1
$qn = 0;
?>
```

Then it will fetch the users inputted data from the Create a Quiz Screen and store it in variables to make it easier to use throughout the process.

```
<?php //Fetch the users inputed data
$quizName=($_POST['quizName']);
$subject=($_POST['subject']);
$level=($_POST['level']);
```

```
?>
```

Next it will find the subject ID from the subjects table, and match their ID's. Because in the quiz table only the subject ID is stored and not the actual subject.

```
<?php //Finds subject from table and matches ID
$dbsubjects = mysql_query("SELECT ID, Subject FROM Subjects WHERE Subject =
'$subject' ") or die(mysql_error());
$subject = array();
while ($a = mysql_fetch_array($dbsubjects)){
$subject[] = $a;}
$subjectID = $subject[0]['ID'];
?>
```

Next it adds the quiz information into the database, using an SQL insert into statement, adding in the quiz name, subject ID and level. The quiz ID is automatically incremented from the last record.

```
<?php //Add Quiz information into db
mysql_query("INSERT INTO quizzes (Name, Subject_ID, Level)
VALUES ('$quizName', '$subjectID' ,'$level')");
?>
```

To return the quiz ID of the new quiz I will use the mysql_insert_id function which returns the last ID of something just added into a database.

```
$quizID =  mysql_insert_id(); // return quiz ID
```

If it is the first question, then the first thing to happen is to get the quiz ID from the previous form. This is done using the POST method.

```
<?php // Get quizID
$quizID = ($_POST['quizID']);
?>
```

Then the question number from the previous question is fetched, also using the POST method. It is incremented to get the current question number.

```
<?php //Get question number, and increment
$qn = ($_POST['qn'])+1;
?>
```

The user's questions and answers are fetched from the previous screen, using the POST method and are stored in variables, ready to put into the SQL database.

```
<?php // Get users questions and answers
$question = ($_POST['question']);
$optionA = ($_POST['optionA']);
$optionB = ($_POST['optionB']);
$optionC = ($_POST['optionC']);
$optionD = ($_POST['optionD']);
$correctAns = ($_POST['correctAns']);
?>
```

Next the questions are entered into the database using an SQL query. The quiz ID, question number and question are added into the questions table.

```
<?php // Add question to db
mysql_query("INSERT INTO questions (Quiz_ID, QuestionNumber, Question)
VALUES ('$quizID', '$qn' ,'$question')"); //sort out question number in a min
?>
```

The question ID from the questions just inserted into the SQL database is fetched using the same function as before again.

```
$questionID =  mysql_insert_id(); // return question ID
```

Next the answers are added into the database, in a similar was as the question using an SQL insert statement and inserting question ID, all the answers and the correct answer.

```
<?php // Add answers into db
mysql_query("INSERT INTO answers (Question_ID, Answer1, Answer2, Answer3, Answer4,
CorrectAnswer)
VALUES ('$questionID', '$optionA'
,'$optionB','$optionC','$optionD','$correctAns')");
?>
```

Finally the necessary information is posted to the next form, such as quiz ID and question number.

User specifications met – V0.2:

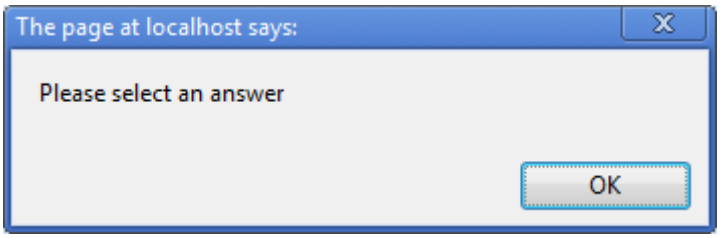
From the user specification list, below are the points met that weren't met in the last version

- Quiz Name
- Level (should be a drop down menu)
- Subject (Dropdown menu)
- Questions (as much or little as needed)
- The correct answer for each question (radio button)

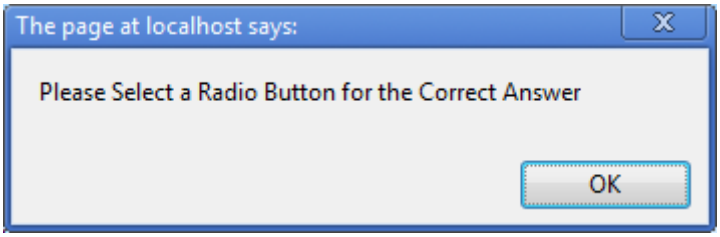
Validation checks – V0.2:

- Check an answer was selected:

If the user doesn't select a radio button during the quiz process, a dialog box will pop-up informing the user that they haven't selected an answer.



If they forget to select a correct answer for the question while they are creating it there will be a similar dialog informing them that there needs to be a correct answer.



Data Structures – V0.2

The data structures have remained the same since the last version, although now it is possible to add a quiz from the program. This adds information into the quizzes, questions and answers table. This information cannot yet be edited from a user interface.

Quizzes table before data was added:

ID Unique ID of Quiz	Name Descriptive Text	Level	Subject_ID

Quizzes table after data was added:

ID Unique ID of Quiz	Name Descriptive Text	Level	Subject_ID
76	AS Economics Introduction	A Level	5

Questions table before data was added:

ID	quiz_ID	QuestionNumber	Question

Questions table after data was added:

ID	quiz_ID	QuestionNumber	Question
68	76	2	Economics is the study of:
69	76	3	A resource is scarce if supply exceeds demand at z...
70	76	4	The opportunity cost of a good is
71	76	5	A market can accurately be described as
72	76	6	A command economy decides resource allocation by g...
73	76	7	In a free market _____ _____
74	76	8	In the mixed economy
75	76	9	Positive economics studies objective explanations ...
76	76	10	Normative economics forms _____ based on ___...
77	76	11	Microeconomics is concerned with
78	76	12	Macroeconomics is the study of _____
79	76	1	An economic model is a physical representation of ...

Answers table before data was added:

ID	question_ID	Answer1	Answer2	Answer3	Answer4	CorrectAnswer

Answers table after data was added:

ID	question_ID	Answer1	Answer2	Answer3	Answer4	CorrectAnswer
75	68	Production technology	Consumption decisions	How society decides what, how, and for whom to pro...	The best way to run society	c
76	69	True	False			b
77	70	The time lost in finding it	The quantity of other goods sacrificed to get anot...	The expenditure on the good	The loss of interest in using savings	b
78	71	A place to buy things	A place to sell things	The process by which prices adjust to reconcile th...	A place where buyers and sellers meet	c
79	72	True	False			a
80	73	Governments intervene	Governments plan production	Governments interfere	Prices adjust to reconcile scarcity and desires	d
81	74	Economic problems are solved by the government and...	Economic decisions are made by the private sector ...	Economic allocation is achieved by the invisible h...	Economic questions are solved by government depart...	a
82	75	True	False			a
83	76	Positive statements, facts	Opinions, personal values	Positive statements, values	Opinions, facts	b

Testing – V0.2:

Test Name	Test Description	Input	Expected output	Actual Output	Notes
Entering a quiz Name	To test if the quiz name is correctly stored and can be accessed from the database	Test quiz name	Your quiz has been stored. Then show the 'create question's form	Your quiz has been stored. Then show the 'create question's form	Works correctly
Entering the level	To select a level from the drop down list and see if it saves correctly	A Level	Your quiz has been stored. Then show the 'create question's form	Your quiz has been stored. Then show the 'create question's form	Works correctly
Entering the subject	To select a subject from the drop down list, fetched from the subjects table in the database	Geography	Your quiz has been stored. Then show the 'create question's form	There was a syntax error caused by an undefined offset.	Didn't work because geography was the last subject in the list and caused an undefined offset when the ID from subjects was linked with the subject ID in the quiz table
Adding questions	To test that the form will be able to store the questions into the database	question	Your question and answers have been stored. And shows the next screen	Your question and answers have been stored. And showed the next screen	Worked correctly, although on the last screen the user had to go onto the next question before finishing the process in order to save the last question
Adding answers	To test that the form will be able to store the answers into the database	question	Your question and answers have been scored	Your question and answers have been stored. And showed the next screen	Worked correctly, although on the last screen the user had to go onto the next question before finishing the process in order to save the last q and answers.

Issues found and addressed – V0.2:

On the creating a quiz process if the subjects selected was the last subject in the list a syntax error was returned informing of an undefined offset. This was because the subjects are fetched from a database, and they are all identified by unique ID's that are in the subject's field in the quiz database. This is done to take up less memory and improve flexibility and efficiency. However because the subject ID starts at one and the array values start at zero, they have to be incremented by one to be the same, so it finds the subject, then goes one down, but if it is the last subjects it tries to find a non-existent subject, returning an error as it has no ID to decrement.

Another issue is that the user can only enter 4 answers, no more and no less, although this isn't an error it's more of a design fault that I hope to improve on in a later version.

Also in order to finish the quiz the user has to go forward to the next question and then finish it. This is because in order to save a question they must go to the 'add new questions' form for it to be saved and added to the database. Although this is inconvenient for the user as they should just be able to click finish and it save the quiz. I will work on this in the next version.

End user involvement – V0.2:

Version 0.2	
Pros: • The creating a quiz process works fully • The last question is now scored.	Cons: • There is no way to browse quizes, the only way to start is to enter it's ID. Also you can't search for a quiz. • Navigation during the quiz • you can't see where you scored your points in which q. • Appears to be no way to exit from creating a quiz.
Notes:	

Confirmation of End user involvement

Signed End user [Mrs Munt] Janet Munt

Signed Developer [Alicia Sykes] Alicia Sykes

Review of V0.2

In version 0.2 the user is able to create a quiz, this was the biggest feature lacking in version 0.1 as was showed by the end user involvement. I have corrected the bug where the last question was not scored. My testing showed that there were a few syntax errors, where a for loop wasn't ended in the right place, and a semicolon was missed out, I corrected those. My end user involvement showed that there were quite a lot of additional features still lacking. A major one being the lack of a start screen which would give the user the user the ability to search and browse quizzes. There was also very poor navigation, there are no home button links, nor next or back buttons. I hope to improve on all of this in the next version.

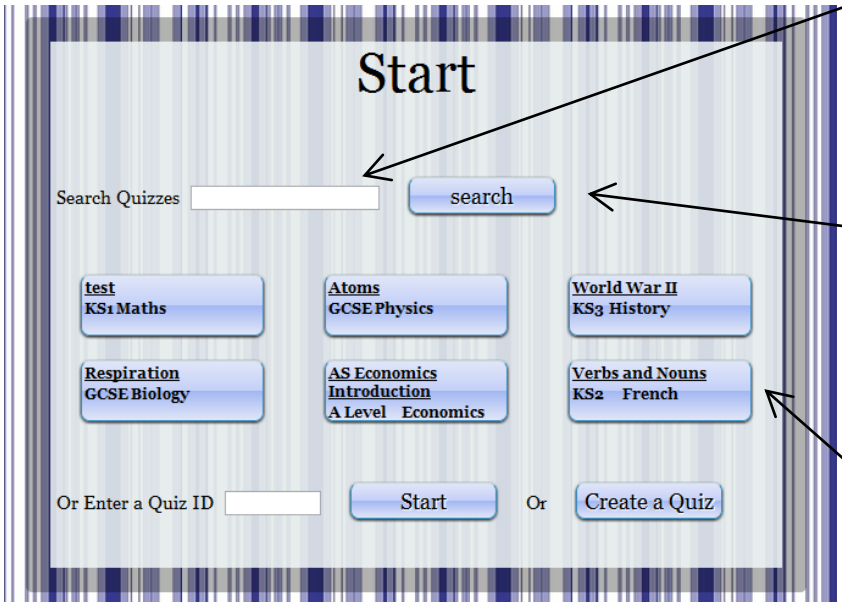
Version 0.3

Objectives – V0.3

The main objectives of version 0.3 are to create a start screen that displays the quizzes, in a way which the user can see the title, subject and level for each quiz. Also I will start to develop a search system, that will return the users search results, and make finding a quiz easier. Further to this I will try and make the quiz process more user friendly by adding navigation buttons and easier labels. I will try and improve the scoring system to give a breakdown of scores. Finally I will add a finish screen for when creating a quiz, which will have a number of further options on it.

Screen Designs – V0.3

The start form:



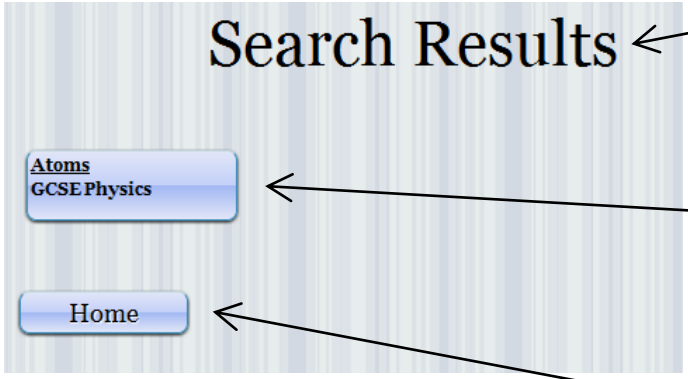
Search textbox, with a label indicating to the user what they should enter into the search box.

Search button, placed next to the search box.

Quiz blocks, showing the quizzes with their title, level and subject. The value is equal to the quiz ID, and so they can be clicked to do the quiz.

The title, enter quiz ID textbox, and create button have remained similar to in the last start screen.

The search results form

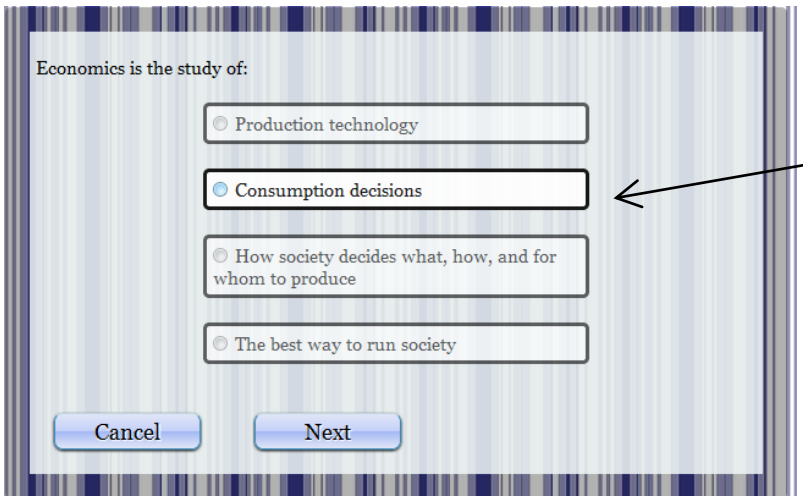


There is a title at the top of the screen, informing the user of what screen they are on. Search Results.

The results will be displayed here with their title, level and subject.

Navigation buttons to allow the user to return home after searching.

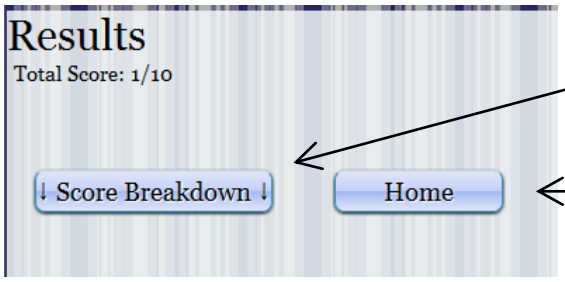
The quiz screen:



The questions are now inside labels, which allows the user to click anywhere on the text to select a question. This will improve the speed and accuracy, and make it more user-friendly. When the user hovers over it or selects it, the opacity will increase

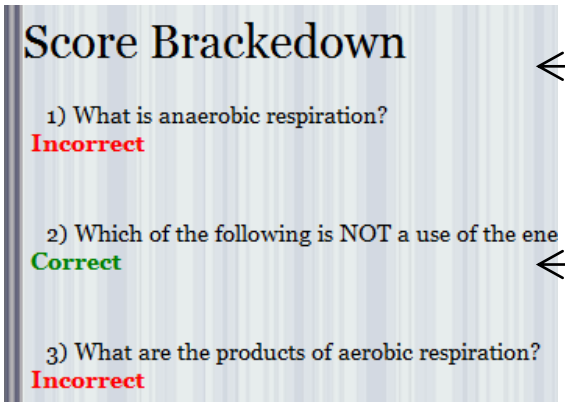
The rest of the form has remained similar to the last version

The results form:



There is a JavaScript slide down menu showing a breakdown of the scores.

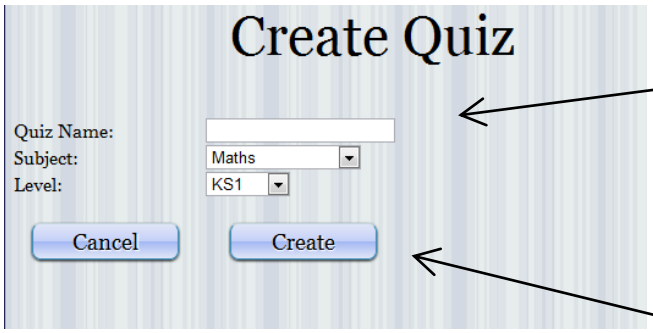
There is a navigation button redirecting the user to the start page.



This is what is shown, when the user clicks the score breakdown button.

It shows each question, and whether the user got it wrong or right.

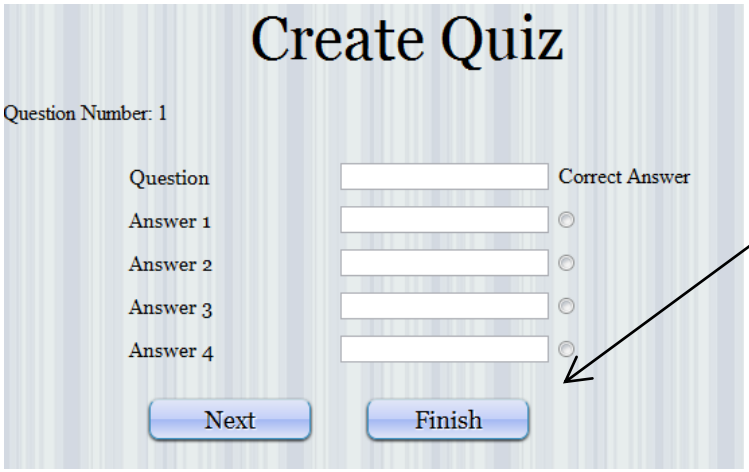
The Create a quiz screen:



The Create a Quiz form has not really changed much from the last version, other than the new navigation buttons and add new subject.

These new navigation buttons add usability to the form, allowing the user to return home at any time.

The Create a Question screen:



The add a question page also hasn't changed that much since the last version, except for the navigation buttons

Code – V0.3:

The start form:

First it counts the number of quizzes in the database and creates a for loop to display each quiz as gone along, through using the quiz ID.

```
<?php // Get quiz titles and levels
    $num = (mysql_num_rows($dbquizzes));
    for($q=0; $q<$num; $q++)
    $subjectID = $quiz[$q]['Subject_ID'];
    ?>
```

Then it finds the subject in the database using an SQL query still inside the for loop so it will happen each time and will be displayed for each block.

```
<?php //Finds subject from the ID
    $dbsubjects = mysql_query("SELECT ID, Subject FROM Subjects") or
die(mysql_error());
    $sub = array(); while ($a = mysql_fetch_array($dbsubjects)){ $sub[] = $a;}
    $subject = $sub[0]['Subject'];
    ?>
```

Then there is another for loop and inside that a button containing the quiz title, subject and level. Also the value of the button is the quiz ID, so the user can click it.

The search results form:

First it will connect to the database, and fetch the quiz information. It will put this into an array.

```
    <?php  //Conect to quizzes database
    $dbquizzes = mysql_query("SELECT * FROM `quizzes`")or die(mysql_error());
    $quiz = array();
    while ($a = mysql_fetch_array($dbquizzes)){
    $quiz[] = $a;}
    ?>
```

Next I get the users search term and remove the unnecessary parts, like punctuation, start and end spaces and make it all into capital letters, then put it in the variable.

```
<?php // Get users search term
        $find = ($_GET["search"]);
        $find = strtoupper($find);
        $find = strip_tags($find);
        $find = trim ($find);
        ?>
```

Next I search for the users search term in the database using the like function in an SQL query.

```
<?php //Search in db
        $dbsearch = mysql_query("SELECT * COUNT(*) FROM quizzes WHERE Name
LIKE'%$find%'");
        while ($b = mysql_fetch_array($dbsearch)){
    $results[] = $b;
$count = $results[0]['COUNT'];
        }?>
```

Then it gets the subject information from the database where quiz ID is equal to the quiz ID called.

```
<?php //Finds subject from the ID
    $dbsubjects = mysql_query("SELECT ID, Subject FROM Subjects") or
die(mysql_error());
    $sub = array(); while ($a = mysql_fetch_array($dbsubjects)){ $sub[] = $a;}
    $subject = $sub[0]['Subject'];
```

```
        ?>
```

Finally, it will output the search results, which will be structured with each quiz in each block and the level and subject under the title. The quiz ID will be the value of the button, and it will have the action quiz.php and method will be get.

The results form:

The breakdown of scores uses the code below. Firstly it gets the array keys from the score array, because the array keys used is the question ID. It then declares and initialises the user answer $userA. A for loop going up to the number of questions, previously defined. Because the question ID is the array key in the score array, the score keys array to inside the for loop. From the question ID, an SQL query is used to fetch the question where the ID is equal to the question ID. This question is then outputted. Then either correct or incorrect is outputted depending on the score value.

```
<?php  //Display Results
    $scoreKeys = (array_keys($score)); //Get the array keys of score array
    $userA = 0; //Declair and initialise user answers
    for($i=0; $i<$noq; $i++){    //for loop for each question in the array
        $questionID=$scoreKeys[$i]; //Gets the question ID from the array key
        $dbquestions = mysql_query("SELECT ID, QuestionNumber, Question
                                    FROM Questions
                                    WHERE quiz_ID =  $quizID
                                    AND ID = ($questionID-1)"); //get question from
ID
            $e = mysql_fetch_array($dbquestions); //put SQL results into an array
            $question = $e['Question']; //get the actual qurstion from array
        if($score[$scoreKeys[$i]]==1) //if the score for that question is 1 then..
            {$userA='<p class=correct>Correct</p>';}// output correct
        else if($score[$scoreKeys[$i]]==0) //if the score is zero
            {$userA='<p class=incorrect>Incorrect</p>';} //output incorrect
        else {$userA = 'Result Unknown';} // if the score isnt there

        echo ($i+1).") ".$question.$userA; // print the question
        }
?>
```

User specifications met – V0.3:

Below is the user specification met in version 0.3 from the list at the beginning of section B.

- Show score and breakdown at the end
- Key word searching
- Record users score while they do the quiz
- Show total score at the end of the quiz
- Show breakdown of score with all the questions that were correct or incorrect at the end

Testing – V0.3:

est Name	est Description	put	xpected utput	ctual utput	otes
ow score at e end	o test if the score at the d is correct	nswers to uestions 7 ght, 3 wrong	otal Score: /10	otal Score: /10	orked
ow score at e end	o test if the score at the d is correct	nswers to uestions 1 ght, 9 wrong	otal Score: /10	otal Score: /10	orked
ow score eakdown	o test if at the end the ore breakdown is correct d marks which questions ere done correctly	nswers to uestions 7 ght, 3 wrong	e question llowed by hether it was rrect or not r each	e question llowed by hether it was rrect or not r each	orked
eyword arching	o test to see if the search nction works to a basic vel	rt of the name a quiz	e full name that quiz d any other atches	e full name that quiz d any other atches	orked

Issues found and addressed – V0.3:

There were no major issues found during testing, although the efficiency and layout of the program could be improved greatly.

End user involvement – V0.3:

Version 0.3	
Pros: • It is easy to browse through quizes. • There is a basic Search Function • The navigation during the quiz is improved. • There is a Score brakedown • IES easier to exit from create a quiz	Cons: • only being able to have 4 answers Per question is very limiting. • answer text bokes are taking up a lot of Space. • The Screen Showing results is slow • Background.
Notes:	

Confirmation of End user involvement

Signed End user [Mrs Munt] 

Signed Developer [Alicia Sykes] 

Review of V0.3

From the end user involvement I will need to improve the efficiency of the program, they said that it is inflexible only being able to have 4 answers.

Version 0.4

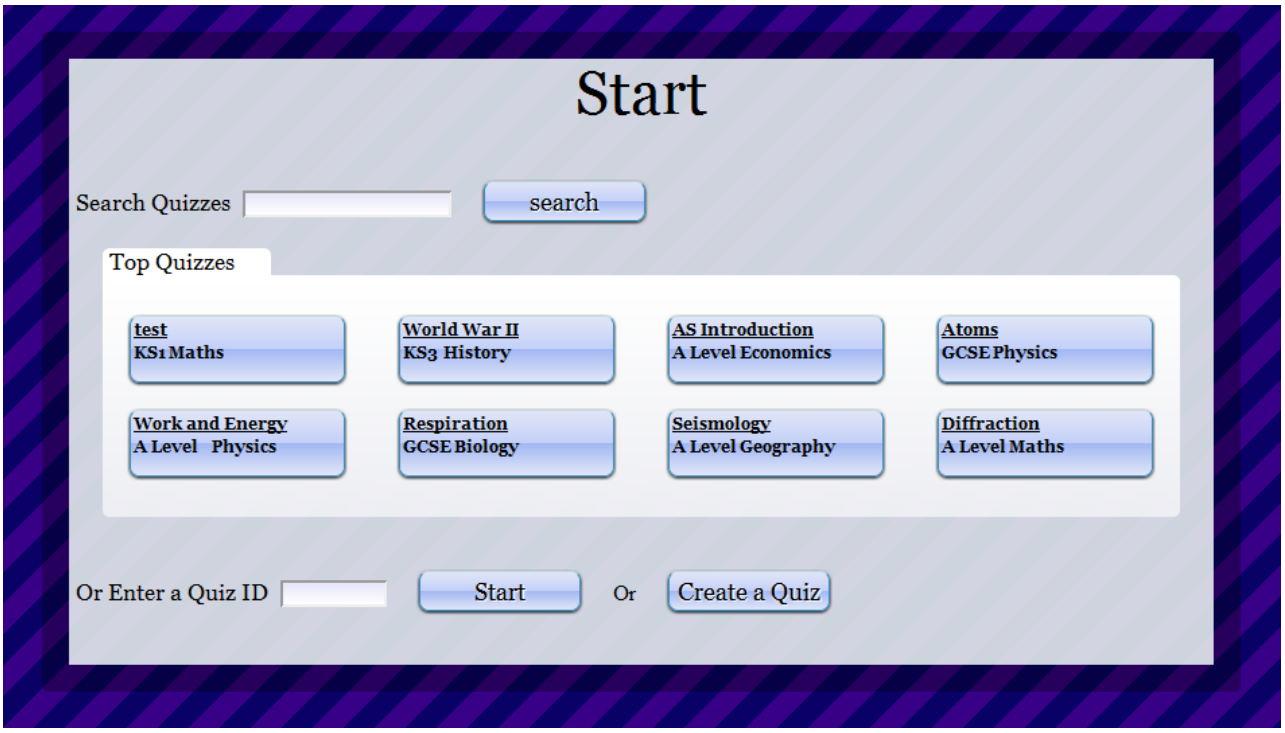
Objectives – V0.4

In version 0.4 I am going to try and improve the usability of it. I will also change the database structure to allow there to be between 2 and 6 answers to each question, previously there were only 4 answers, and this was inflexible. On The Create a Question form I will allow the user to add new input boxes with a button. I will also start making the saved scoring system, which will save the scores of users in the database, and record which answers people select. There will be a new search screen, that will be part of the quiz screen, this will make it quicker, easier and clearer to search.

Screen Designs – V0.4

The start screen: -

Screen shot showing quizzes ordered by number of takes:



Tab box keeping quizzes ordered by top quizzes, with a label at the top, so the user knows what they are viewing.

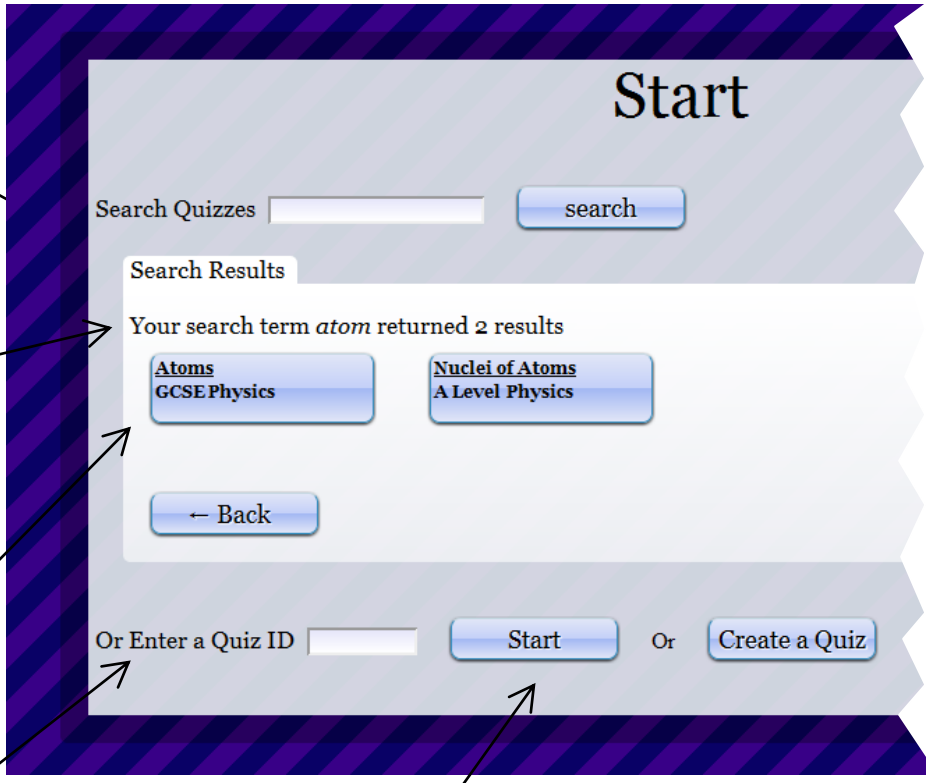
The search results (start screen):

The name of the tab will change when there is a value in the search box

The top sentence will repeat what the user searched for and say how many results there were

The quiz blocks will be displayed in a similar was as to how they were on the start screen.

The lower part of the form will remain unchanged.



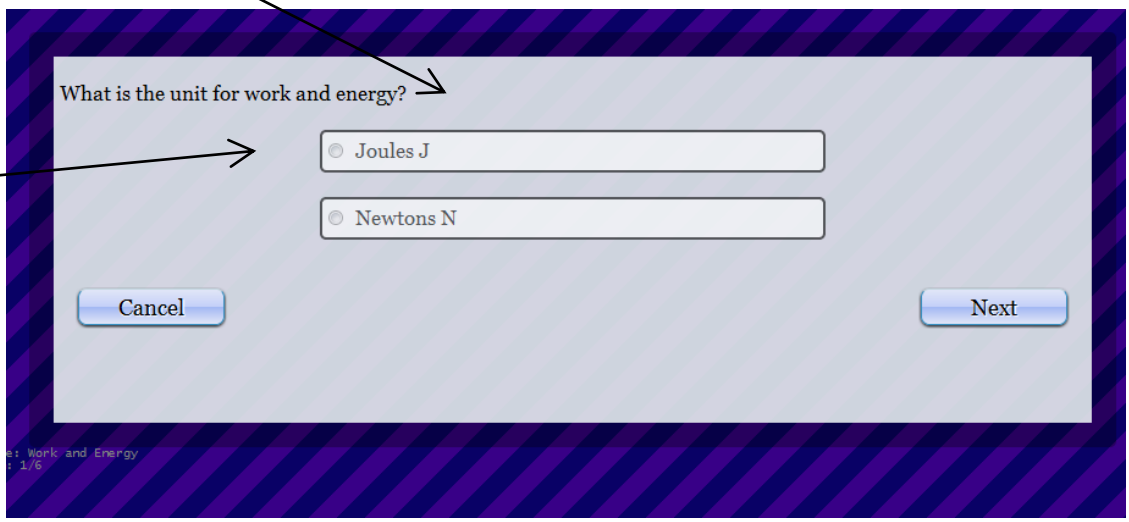
The buttons and quiz blocks will get darker on hover.

(The squiggly line was where the form was too big to fit on the page)

The quiz form:

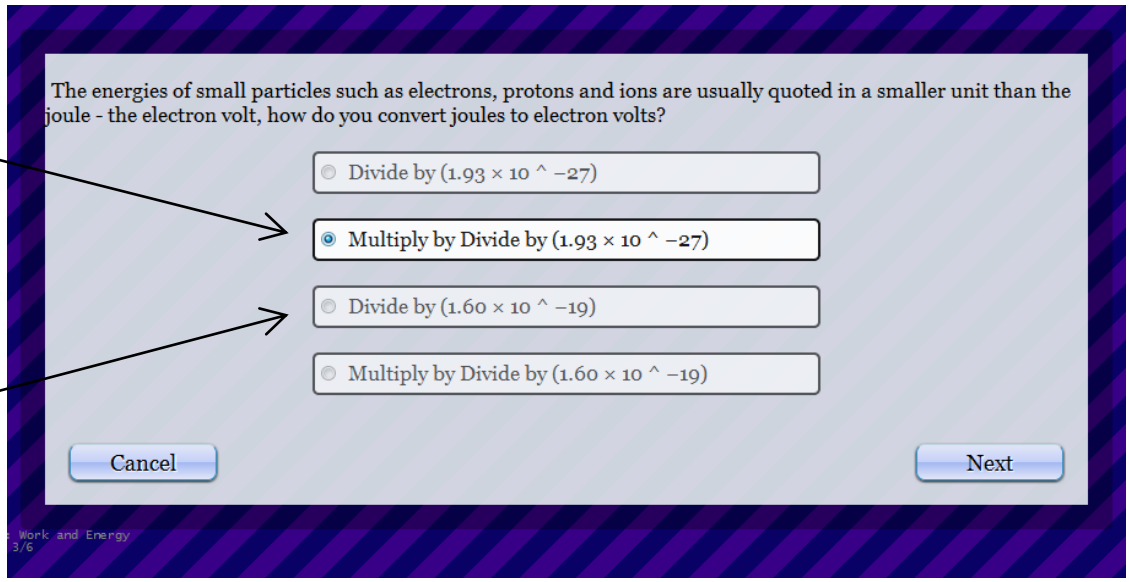
The questions and answers are laid out in a similar way to

If there are only two answers, there will only be two answer blocks. There can be up to 6 answers per question.

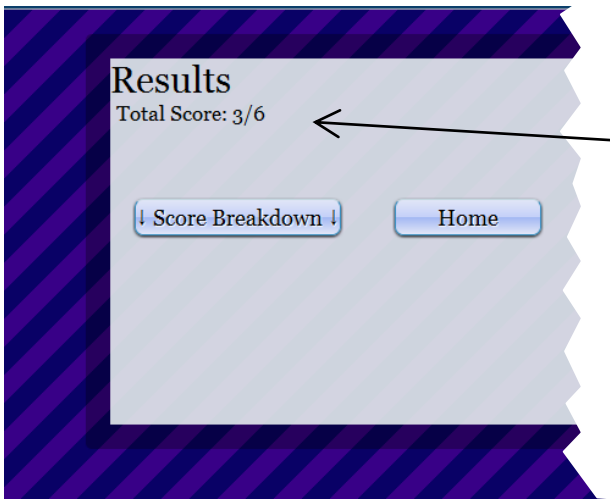


When an answer is hovered or selected the transparency will be decreased to 0%

If there are more than 2 answers they will just be displayed like this.

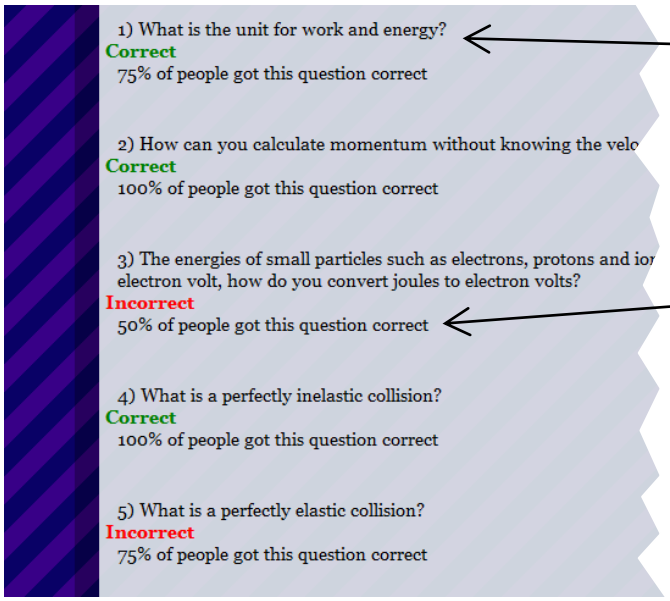


The results screen:



There is still a title, score and navigation buttons. This for hasn't changed.

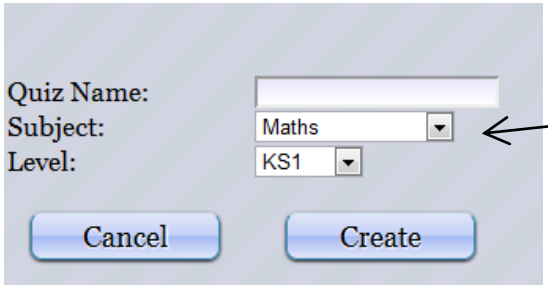
Score breakdown – results screen.



There is still the question and if they got it correct or not.

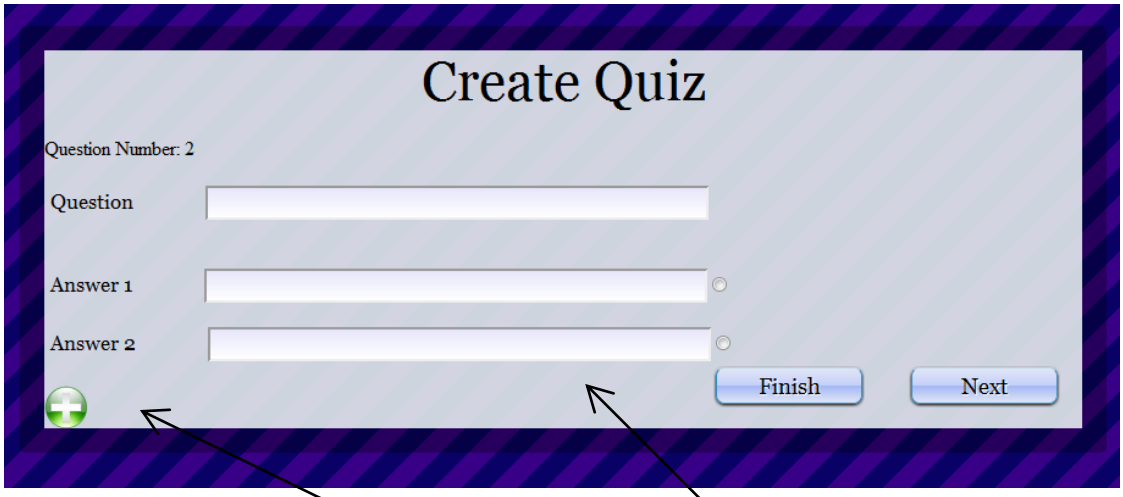
Percentage of people that have got that question correct.

The create a quiz form



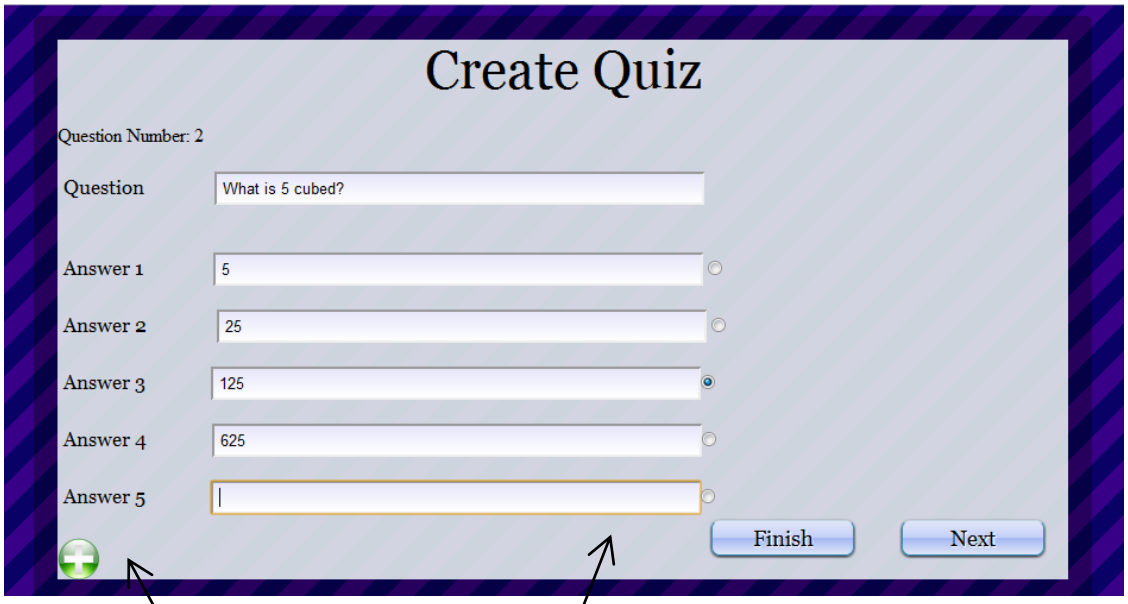
The Create a Quiz form has not really changed. There is still an input box for the quiz name, two drop down menus for subject and level. As well as labels and navigation buttons.

The create a question form:



There is a button to add more text boxes.

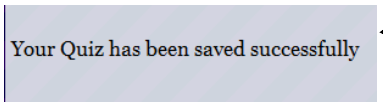
By default there are only two text boxes.



There is a button to add more text boxes. It loses functionally and disables when there are 6 boxes visible

The form will automatically resize when there are more text fields present.

The finish creating screen:



The finish screen just informs the user that there quiz was saved successfully and adds the

Code – V0.4:

There is not much new PHP in the start form, just a small amount of refinement of the current code, and some changes in the CSS properties.

In the quiz form, the only main difference is that the users score is recorded in the database, so it can be used for the averages. This is done using this code. This checks that the user answer is set, before trying to process non-existent information. It then fetches the value from the number of answers field in the answers database. If the user selected that answer, then the value in that field is incremented.

```
<?php // Adds to the average score
if (isset($_POST['userAnswer'])) {
    $dbavscore= mysql_query("SELECT `ID`,`NumCorr`
                             FROM `answers`
                             WHERE `ID`=$userAnswer");
    $avscore = array();
while ($m = mysql_fetch_array($dbavscore)){
    $avscore[] = $m;}
        $avscore=$avscore[0]['NumCorr'];
        $avscore = $avscore + 1;
mysql_query("UPDATE  `quiz`.`answers`
            SET  `NumCorr` =  '$avscore'
            WHERE  `answers`.`ID` = $userAnswer;");
}
?>
```

A similar function is used in the results form to increment the number of times the quiz has been taken. The only difference is that there is no condition that has to be met.

There are not that many changes in the create quiz form either.

In The Create a Question form, users can now add between 2 and 6 answers for each question. There is a JavaScript function that creates a new label, new textbox and new radio button each time the plus button is pressed. Firstly it has a for loop, to make sure that no more than 6 boxes are created, it starts of at 2, because there are already 2 textboxes before the function is called. Count is incremented, and only if it is not bigger than 6 will the rest of the function be executed. The new element is created and a switch case statement gets the ID from the number of count. Another switch case statement is used to create the name for the new element. Then the parameters are added to the element and it is appended to the document. The number is incremented each time, and cannot be more than 6. Before the function there is a variable declared and initialised to 2. When the function is executed count is incremented, and the function can't run if count is not smaller than 7, because that would create 7 text boxes. The count will form part of the ID, and will be converted into a letter, like a = 1, b = 2…. In the HTML form, there are hidden fields which are deleted when a new textbox is created. The point of this is that when the form is submitted, it will never be trying to access a non-existent element.

```
var count = 2;
function addElement(parameters) {
        count++;
    if (count < 7){
```

```
        //Create element
        var newLabel = document.createElement('p');
        var newBox = document.createElement('INPUT');
        var newRad = document.createElement('INPUT');
        var line = document.createElement('P');
        var tar = 'ans';

        //create id
        var id
    switch (count) {
            case 3:
                id = "c";
        break;
            case 4:
            id = "d";
        break;
            case 5:
                id = "e";
        break;
            case 6:
                id = "f";
        break; }

        var name
        switch (id) {
            case "c":
                name = "optionC"
                break;
            case "d":
                name = "optionD"
                break;
            case "e":
                name = "optionE"
                break;
            case "f":
                name = "optionF"
                break; }

        //Add parameters
       for (parameter_name in parameters) {
            newLabel.innerHTML = "Answer "+count;
            newLabel.className = "subFont";
            newBox.setAttribute("name",name );
            newBox.className = "textboxQuestions";
            newRad.setAttribute("type","radio");
            newRad.setAttribute("value","correctAns")
            newRad.setAttribute("id",id)
            line.innerHTML = "<br>";
          }

        //Append element to target
        document.getElementById(tar).appendChild(newLabel);
        document.getElementById(tar).appendChild(newBox);
        document.getElementById(tar).appendChild(newRad);
        document.getElementById(tar).appendChild(line);

        //Remove old element
        var child = document.getElementById(id);
        var parent = document.getElementById('ans');
        parent.removeChild(child);
}
}
```

User specifications met – V0.4:

- Answers (As many or as few)
- Show averages
- Record score for the quizzes

Data Structures – V0.4

In V0.4 the users can enter between 2 and 6 answers per question, rather than just 4 as in previous versions. As a result there needs to be a change of structure of the answers table. I was initially going to have 6 columns for answers and just use the ones needed. Although this wouldn't make good use of space, and would be inefficient and messy. Instead I have just got the question ID, answer and whether it is correct or not.

Below is the new structure of the database:

Field	Type	Collation	Attributes	Null	Default	Extra
ID	int(5)			No	*None*	AUTO_INCREMENT
question_ID	int(5)			No	*None*	
Answer	varchar(100)	latin1_swedish_ci		No	*None*	
CorrectAnswer	char(1)	latin1_swedish_ci		No	f	
NumCorr	int(5)			No	0	

ID	question_ID	Answer	CorrectAnswer	NumCorr
13	4	Proton	f	0
14	4	Neutron	f	1
15	4	Nucleus	t	2
16	4	Electron	f	0
17	5	Protons and Electrons	f	0
18	5	Protons and Neutrons	t	1
19	5	Electrons and Neutrons	f	2
20	5	Protons, Electrons and Neutrons	f	0
21	9	Battle of Gibraltar	f	1
22	9	Battle for the English Channel	f	2
23	9	Battle of Britain	t	4
24	9	Battle of Dover	f	1
25	8	Finland	t	4
26	8	Norway	f	1
27	8	Denmark	f	1
28	8	Sweden	f	2
29	7	Lightning War	t	4
30	7	Panzer Assault	f	3
31	7	Mechanized War	f	1
32	7	Combined Forces	f	0

Below is some example data added into the database through the quiz program, it shows how it is stored

Testing – V0.4:

est Name	**est Description**	**put**	**xpected utput**	**ctual utput**	**otes**
o Answers	o check that if the user ters at least 2 answers	None]	ease enter at ast two swers	ease enter at ast two swers	ssed
Answer, but textboxes sible	o check that if a textbox visible, but hasn't had a lue added into it, it on't count	answer	ease enter at ast two swers	ease enter at ast two swers	ssed
aximum of 6 xtboxes	o check that when there e 6 textboxes visible the er can't add more	answers and ess	ney can't add ore	ney can't add ore	ssed
ow averages	o check that in the score eakdown the averages e displayed correctly	uizzes done ultiple times, ould be corded in the tabase	% of people ose this swer	% of people ose this swer	Vorked
kes should be cremented	hen a quiz is completed e value of takes in the tabase is incremented	quiz is taken 7 nes	ne value of kes is creased by 7	ne value of kes is creased by 7	Vorked

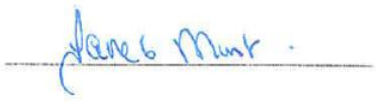
Issues found and addressed – V0.4:

- The validation checks to ensure that a radio button is selected did not work with the JavaScript produced radio buttons. Initially this was because it wasn't sure how many radio buttons to check for a value in, but I changed it so that it just checks for one value in that group.
- If a user starts writing a quiz, then stops half way and closes the window, there will be a half created quiz, because it saved at each stage into the database, I will try and correct this in the next version.

End user involvement – V0.4:

Version 0.4	
Pros: • Being able to enter between 2 and 6 answers is a positive. • Search results screen is better.	Cons: • Quizzes arnt timed, timing feature would be very useful • When quiz starts it goes strait into it, screen before could be useful. • The layout is clear, but it doesn't make best use of space.
Notes:	

Confirmation of End user involvement

Signed End user [Mrs Munt] 

Signed Developer [Alicia Sykes] 

Review of V0.4

In version 0.4 I improved the layout from the previous version and changed the design of the answers table to allow the user a have a variable number of answers per question. During the testing I found and fixed a problem with the validation, where the user could forget to select a radio button. Also half made quizzes could be stored if the user exited The Create a Quiz process half way, and duplicate data could be added if the user used the back and forward navigation buttons in their browser. My end user involvement showed that a timing feature would be very useful, and modifying the design would improve the usability. I will change the way quizzes are entered into the database in the next version.

Version 0.5

Objectives – V0.5

In V0.5 I would like to create a timing function that will keep track of the user's time, and display it at the end. Also I will add a new screen that will be displayed after the user clicks a quiz, but before the first question comes up. Also I will change the layout to increase functionality and make better use space. There will be a side pane with the search bar on the start page, and the question list on the quiz page, it will allow faster navigation. There will also be a top pane with more navigation buttons like a home button.

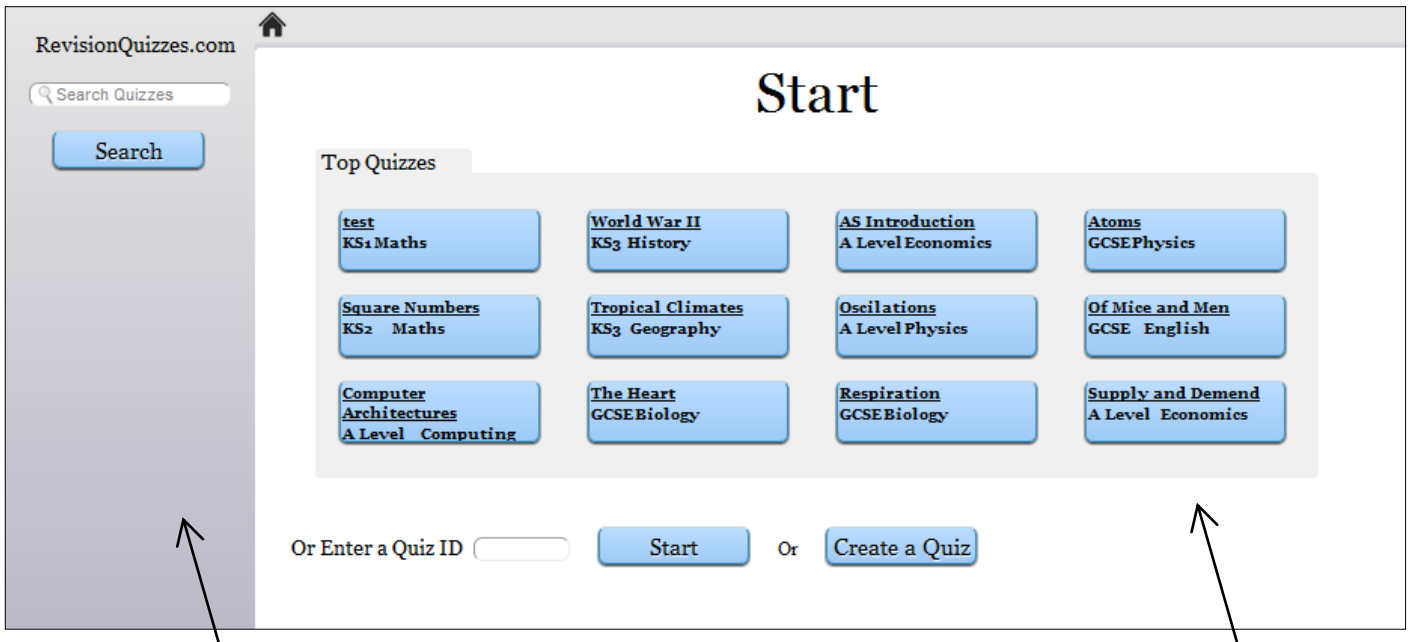
Further to this, I would like to make the quizzes quicker for the user to do, one way in which I will do this is to remove the submit button after each question. Instead the user which will just press the answer they want, and it will automatically proceed to the next question, using JavaScript. I also am going to implement a more dynamic way to enter additional answers to each question, because pressing a button to make the new textbox will slow the process down.

I will create a better way for the user to see the breakdown of their scores at the end, using the side menu for navigation, and showing the question, answer they selected and correct answer.

I will change the creating a quiz process so that the information is all stored in an array in a session, then added into the database right at the end. This will avoid having half created quizzes stored in the database.

Screen Designs – V0.5

The start form:

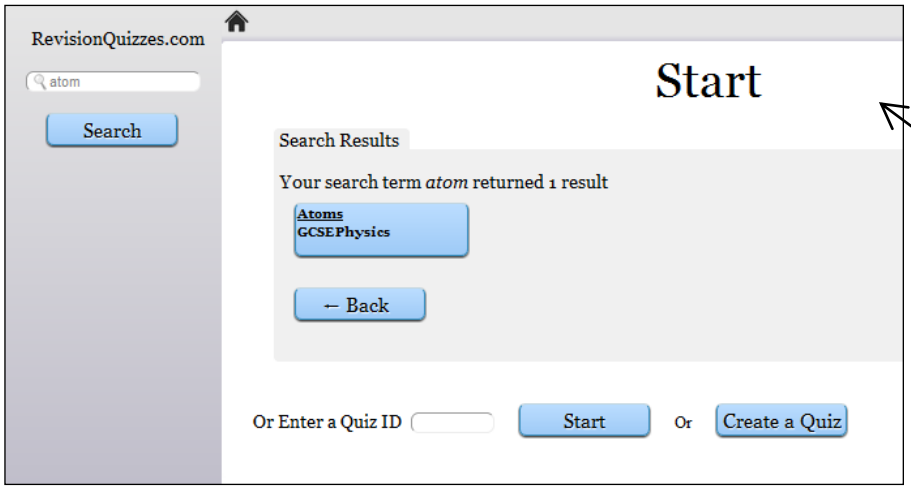


The side pane has the search box in, and in later versions will include advanced search filters

There is also a bar along the top, which will in future versions have some navigation buttons, further to the home button.

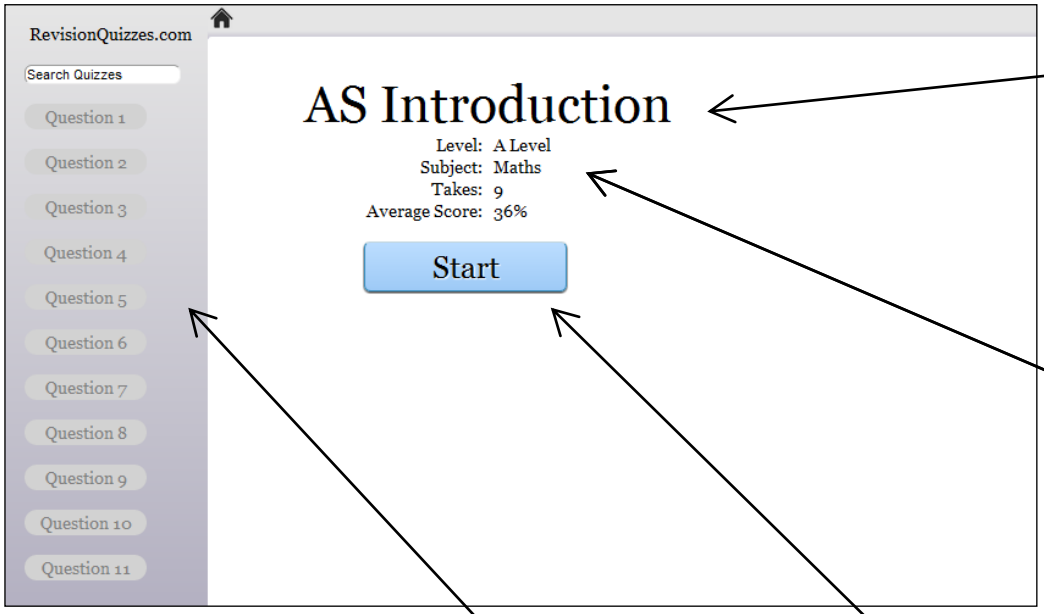
The main tab browser has remained the same.

The search form:



The search results form has remained similar to in the last version. In future versions I will add more search filters to the side bar.

The begin form:



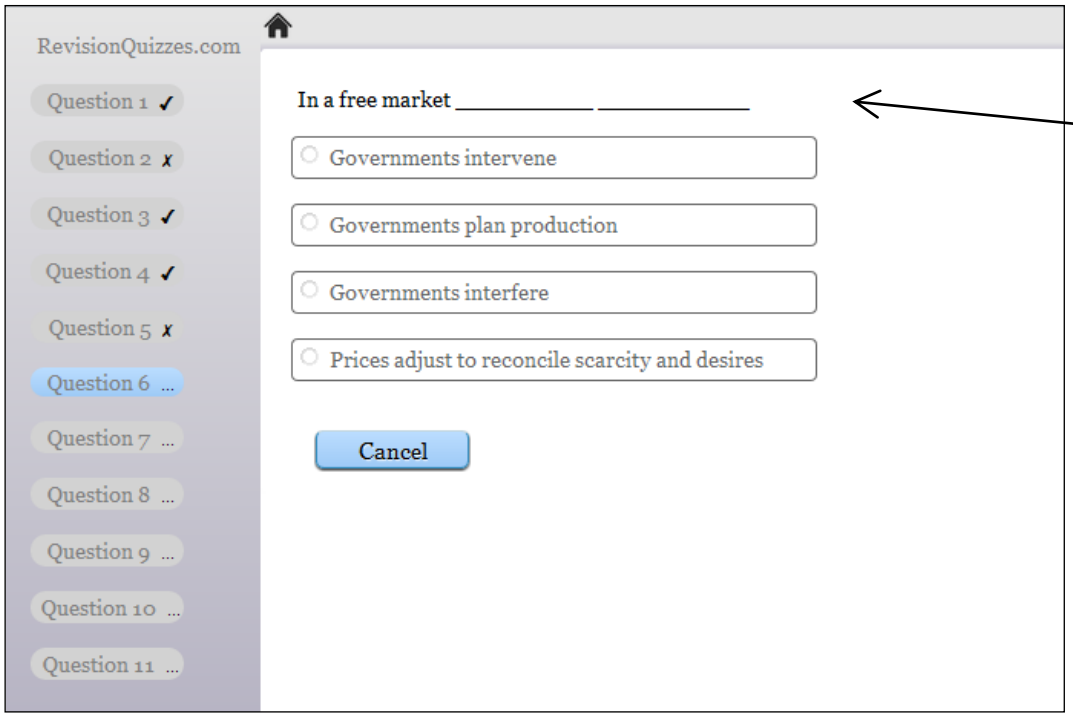
Quiz title.
This will make it clear to the user what quiz they are doing, and will be the title for that page.

Quiz stats.
These will only show if they are available, so if there is no subject for that quiz, it won't show up at all.

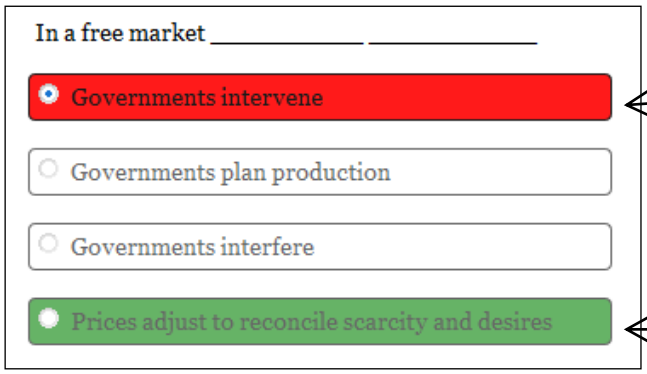
Side bar.
Will not be made active on this screen, but will be visible.

Start button.
Dominant button on the form, when it is pressed the timing will start.

The quiz screen:



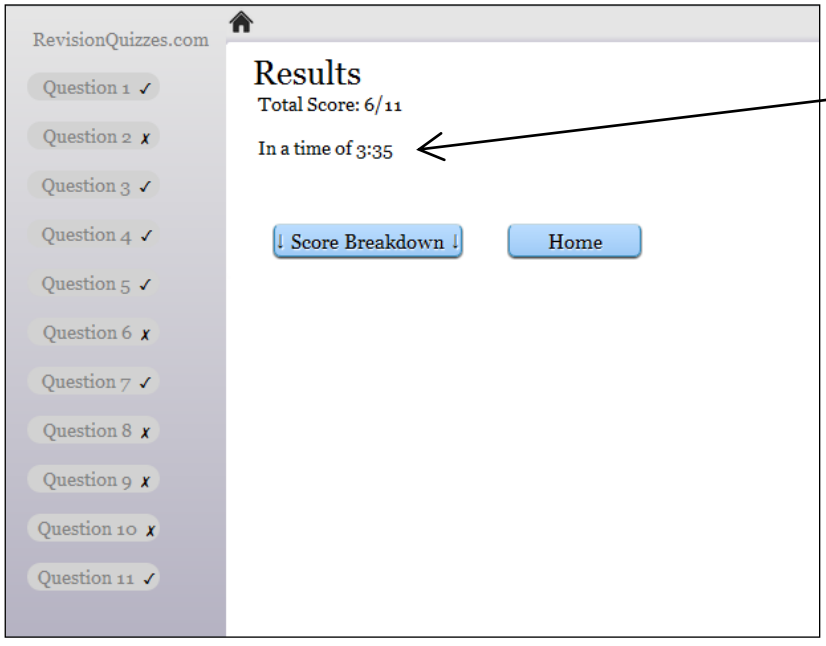
Question, and answers displayed in a similar way as before.



When an answer is selected, the correct answer is shown up in green, and if the user got it wrong, than the incorrect answer is shown in red

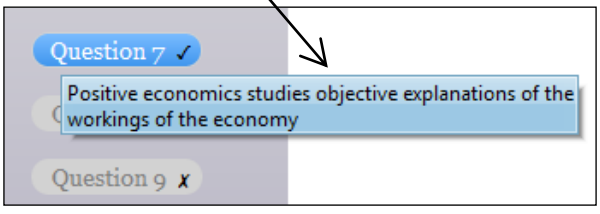
The box is only a solid colour on mouse over, like the top one, usually it is slightly lighter like the bottom one

The Results screen:

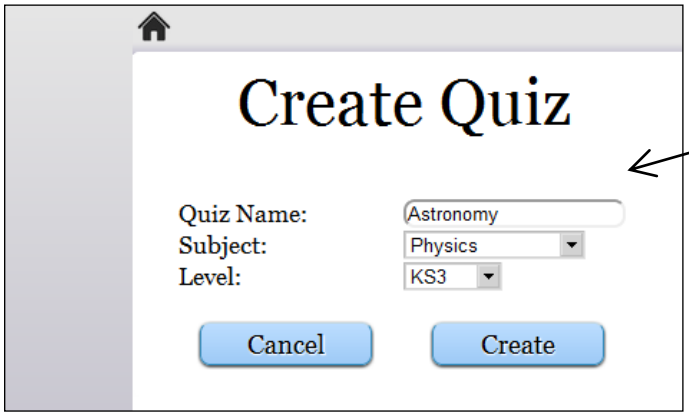


The time is displayed in minutes and seconds

The questions are summarised on the side. The question is displayed on mouse over



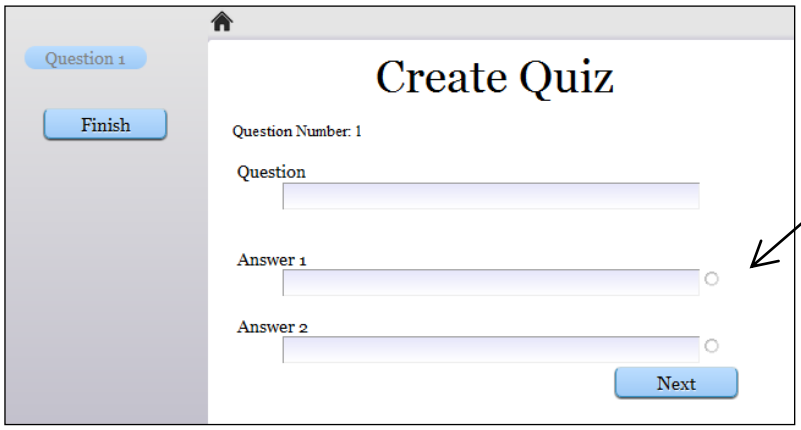
The create quiz form:



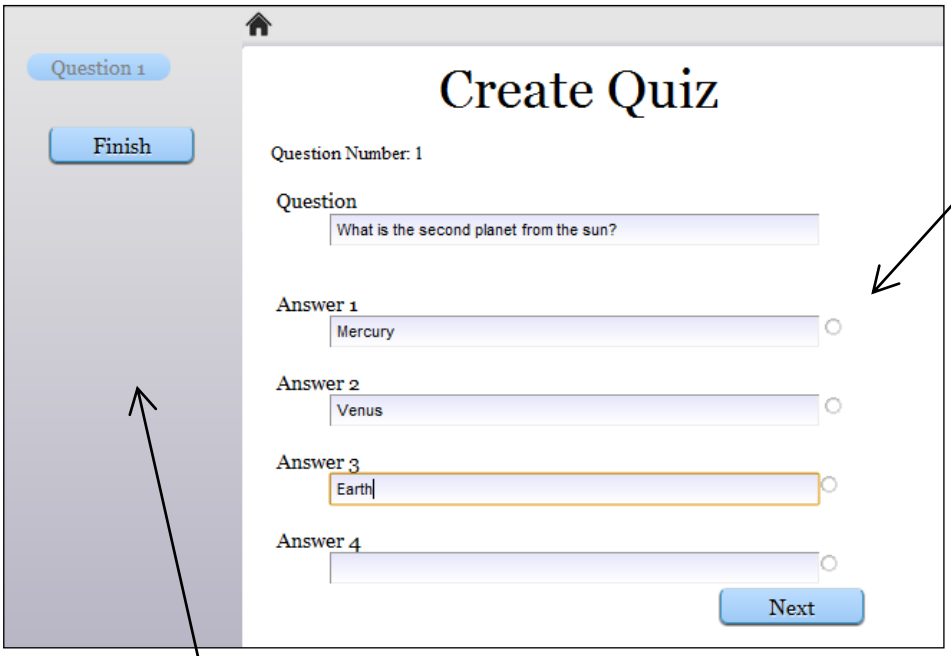
The quiz form has not changed very much. The new side and top bar have been implemented

The 'create question' form:

Initially appears as:



To start with only two textboxes will be visible. This should mean the user is not greeted with a cluttered screen with 6 big textboxes



As the user puts there answer in one textbox, the next one will be visible, and when they put a value in that one the next one will become visible until all 6 boxes are showing. The idea of this is to speed up the time it takes to write a quiz, and it means the user doesn't have to keep switching between mouse and keyboard, which can be quite annoying and slow down the process. If the unused textboxes are deleted on submit.

When a question is created by the user it will be added to the side pane.

Data Structures – V0.5

Now all the information about the quiz is stored in a session while the quiz is being created, below is the structure of the session array, with some example data in it.

The quiz info array – stores the information relating to the quiz, including title, level and subject.

```
array
  'name' => string 'Processor Components' (length=20)
  'level' => string 'A Level' (length=7)
  'subID' => string '9' (length=1)
```

The questions array – stores all the questions in a one dimensional array with the question number as the identifier.

```
array
  1 => string 'What part of the processor keeps track of the next instruction to be
executed?' (length=78)
  2 => string 'What is the current Instruction register?' (length=41)
  3 => string 'What is the function of the ALU?' (length=32)
```

The answers array is a multidimensional array, with the first dimension storing the question number, and the second dimension containing the answers and correct answers and then the values.

```
array
  1 =>
    array
      'answer' =>
        array
          'a' => string 'MAR' (length=3)
          'b' => string 'PC' (length=2)
          'c' => string 'CIR' (length=3)
          'd' => string 'ACC' (length=3)
      'correctaAns' =>
        array
          'a' => string 'f' (length=1)
          'b' => string 't' (length=1)
          'c' => string 'f' (length=1)
          'd' => string 'f' (length=1)
  2 =>
    array
      'answer' =>
        array
          'a' => string 'A register that contains the data rela…' (length=67)
          'b' => string 'A register that contains the informati…' (length=113)
      'correctaAns' =>
        array
          'a' => string 'f' (length=1)
          'b' => string 't' (length=1)
  3 =>
    array
      'answer' =>
        array
          'a' => string 'Responsible for supervising the oper…' (length=72)
          'b' => string 'Provides the computer with logical …' (length=66)
          'c' => string 'A storage location inside the processor.' (length=40)
      'correctaAns' =>
        array
          'a' => string 'f' (length=1)
          'b' => string 't' (length=1)
          'c' => string 'f' (length=1)
```

Code – V0.5:

The begin form:

One of the features of the begin form, is that it displays all the information about the quiz to the user before they start.

Below is the subroutine that declares the variables needed.

```
<?php //Get basic quiz information
    $dbquizzes = mysql_query("SELECT *
                        FROM Quizzes
                        WHERE ID =  $quizID ");
    $quiz = array(); while ($b = mysql_fetch_array($dbquizzes)){
            $quiz[] = $b;}
$title = $quiz[0]['Name'];
$level = $quiz[0]['Level'];
    $dbsubjects = mysql_query("SELECT ID, Subject
                                FROM Subjects");
    $sub = array(); while ($c = mysql_fetch_array($dbsubjects)){
            $sub[] = $c;}
$subject = $sub[0]['Subject'];
$takes = $quiz[0]['takes'];
    $dbquestions = mysql_query("SELECT ID FROM Questions
                                WHERE quiz_ID =  $quizID ");
    $question = array(); while ($d = mysql_fetch_array($dbquestions)){
            $question[] = $d;}
    $firstQID = ($question[0][0]);
    $lastQID =  ($question[$noq-1][0]);
    $dbanswers = mysql_query("SELECT  `ID` ,
                                sum(`NumCorr`)AS total ,  `question_ID`
                                FROM answers
                                WHERE CorrectAnswer =  't'
                                AND  `question_ID`
                                BETWEEN $firstQID
                                AND $lastQID");
    $answer = array(); while ($f = mysql_fetch_array($dbanswers)){
            $answer[] = $f;}
    $total = $answer[0]['total'];
$avScore = round($total/$noq/$takes*100);
$tagScore= null;
$avTime= null;
$tagTime= null;
$creator= null;
$tags= null;
?>
```

Next it will declare and initialise the variables and arrays needed for the running of the quiz.

```
<?php // Declair the necissary varibles, arrays and sessions
$_SESSION['score'] = null;
for ($j=0; $j<($noq); $j++){
    $dbquestions = mysql_query("SELECT *
                                FROM questions
                                WHERE quiz_ID = $quizID
                                AND  QuestionNumber = $j+1");
    $question = array();
    while ($g = mysql_fetch_array($dbquestions)){
        $question = $g;
    }
    $questionID =  $question['ID'];
    $_SESSION['score'][$questionID] = "null";
    $_SESSION['userChoices'][$questionID] = "null";
    $_SESSION['startTime']=time();
}
?>
```

The quiz form:

The JavaScript function that shows the correct answer and the answer the user selected after they have done the question. The user will be able to navigate through the questions, and see what they put, and what was correct when they finished the quiz. The JavaScript function will only be called if the user has finished the quiz.

```
function mark(userAnswer, correctAnswer){
    if (userAnswer == correctAnswer){
        document.getElementById(userAnswer).setAttribute
                        ("style", "background-color:green") }
    else if (userAnswer != correctAnswer){
        document.getElementById(userAnswer).setAttribute
                        ("style","background-color:red")
        document.getElementById(correctAnswer).setAttribute
                        ("style", "background-color:green") }
    document.forms['questions'].submit();
    var userAnswer = document.getElementByName('userAnswer');
    userAnswer.setAtribute('disabled','disabled');
        }
```

The score sub-routine has been modified so that it adds the score into an array stored in a session, rather than an array being serialised and posting to the next screen. This will be more efficient and easier to keep track of.

```
<?php //Calculates and keeps track of the score
 $score = $_SESSION['score'];
if (isset($_POST['userAnswer'])) {
$lqid = ($_POST['questionID']);
$userAnswer = $_POST['userAnswer'];
$lastCorrectAnswer = $_POST['correctAnswer'];
if($userAnswer == $lastCorrectAnswer){
$score[$lqid] = 1; }
else{$score[$lqid] = 0; }
$_SESSION['score']=$score;
$_SESSION['userChoices'][$lqid]=$userAnswer;
}
?>
```

The create quiz form:

There has also been some modification of the create quiz form. Now all the variables are declared on this form, and put into a session. This will make the 'create question' more efficient.

```
        <?php //Declaire Varibles
        $level = null;
        $subject = null;
        $selectedSub = null;
        $qn = 0;
        $questions = array();
        $answers = array();
        $quizInfo = array('name' => "", 'level' => "", 'subID' => "");
        ?>

        <?php //Put varibles in session
        $_SESSION['quizInfo']=$quizInfo;
        $_SESSION['questions']=$questions;
        $_SESSION['answers']=$answers;
        ?>
```

The 'create question' form

Another change in the 'create question' form, is that all the information is added to an array in a session where previously it was added straight to the database.

```
<?php //Get quiz information, questions and answers
```

```php
$quizInfo = ($_SESSION["quizInfo"]);
$quizName=$quizInfo['name'];
$quizLevel=$quizInfo['level'];
$quizSubID=$quizInfo['subID'];
$questions = ($_SESSION["questions"]);
$answers = ($_SESSION["answers"]);
?>

<?php //Add quiz information into database
mysql_query("INSERT INTO quizzes (Name, Subject_ID, Level)
VALUES ('$quizName', '$quizSubID' ,'$quizLevel')");
$quizID =  mysql_insert_id(); // return quiz ID
?>

<?php // Add questions into database
$noq = $qn + 1; // Find number of questions
for($j=1; $j<$noq; $j++){
    mysql_query("INSERT INTO questions (Quiz_ID, QuestionNumber, Question)
    VALUES ('$quizID', '$j' ,'$questions[$j]')");
$questionID =  mysql_insert_id(); // return question ID

?>

<?php // Add answers into database
$enoa = count($answers[$j]['answer']);//get number of answers for each question
$lastLetter = array_pop(array_keys($answers[$j]['answer']));
foreach(range('a',$lastLetter) as $k){
    $dbans = $answers[$j]['answer'][$k];
    $dbcorans = $answers[$j]['correctaAns'][$k];
 mysql_query("INSERT INTO answers (Question_ID, Answer, CorrectAnswer)
    VALUES ('$questionID', '$dbans', '$dbcorans')");
}}
?>
```

User specifications met – V0.5:

- Show user the correct answer of question immediately

Testing – V0.5:

est Name	**est Description**	**put**	**xpected utput**	**ctual Output**	**otes**
me feature	o test that the timing orks, and is accurate	ser does the quiz 2 minutes 30 conds	me: 2:30	me: 2:30	orked
egin screen	o test that the new reen works, and curately displays all the formation.	uiz ID	l the formation out the quiz splayed in the ble	l the formation about e quiz displayed the table	expected
he auto bmit feature	o test that while the er is doing the quiz, the rm will automatically bmit when they select answer	n answer	he next screen owed, and er answer corded	he next screen as showed, and e answer was corded.	expected
ppending swer boxes	o test that while the er is creating a uestion, the answer oxes will automatically eate	question and veral answers	hen one xtbox has a lue in it, the xt one tomatically ows	hen the text ox has a value I an new textbox as added	orked

Issues found and fixed from testing – V0.5

There are still some problems with the subject while creating a quiz, it seems to always be offset by a place, and it is inefficient for the user to have to select their subject from a drop down menu, it is also quite inflexible, so in the next version I will consider getting rid of the subjects table all together.

Other than that, all the tests recorded above returned zero errors, although while writing the code there were some minor errors caused by missing semicolons and brackets, though they were easy to find.

End user involvement – V0.5:

Version 0.5	
Pros: • New layout makes better use of space • timing feature	Cons: • The questions are of limited use is you can't add additional details, or maybe an image. • Don't like the color scheme • can't edit quiz after
Notes:	

Confirmation of End user involvement

Signed End user [Mrs Munt] 

Signed Developer [Alicia Sykes] 

Review of V0.5

The end user involvement showed that being able to add additional information to quizzes and questions would be beneficial, and also maybe to be able to edit a quiz immediately after it has been created.

Version 0.6

Objectives – V0.6

In version 0.6, I would like to improve the functionality, I will do this by allowing the user to add additional information to The Create a Quiz form, and The Create a Question form. This will include extra information like images and text to be displayed either during or after the quiz is complete. Also the timing information, creator and other information about the quiz will be able to be stored in the database.

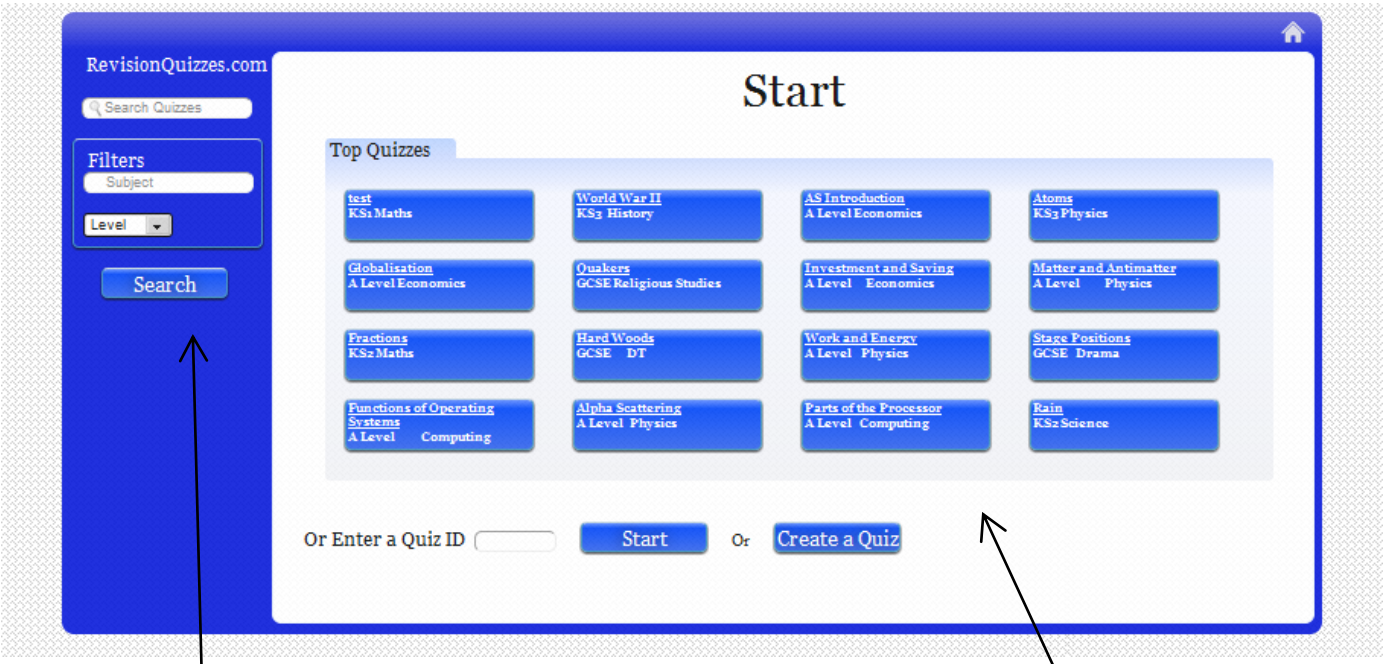
I will improve the search by adding some basic filters including subject and level, which can either be left blank or have items selected.

I will allow the user to edit basic quiz information when the quiz is complete.

I will also adjust the colour scheme from blue and grey to blue and white.

Screen Designs – V0.6

The start screen:



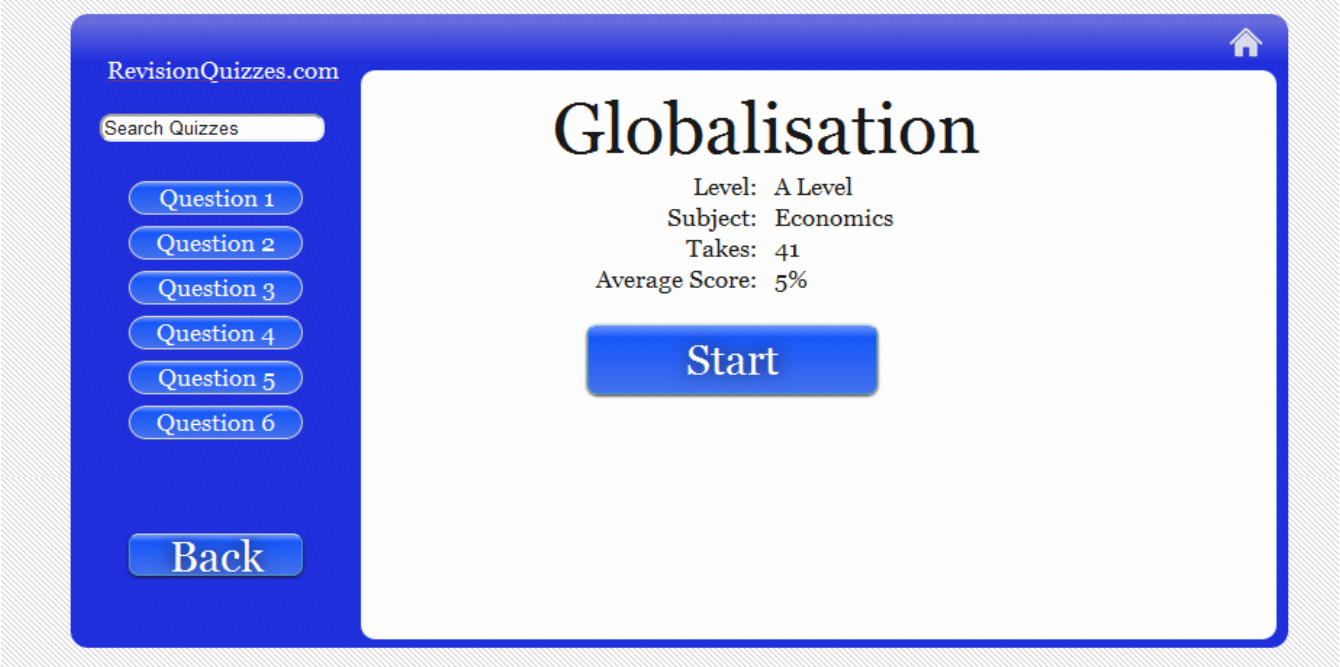
The search filters box contains a drop down menu for level, and a text box for subject

The general layout has stayed the same as in the last version, although the colour scheme has changed, and the background is stripy. The quizzes are still in a main block, and The Create a Quiz button is at the bottom of the screen.

The Search form:

The search form has remained the same as before, other than the change in colour, which is the same as above.

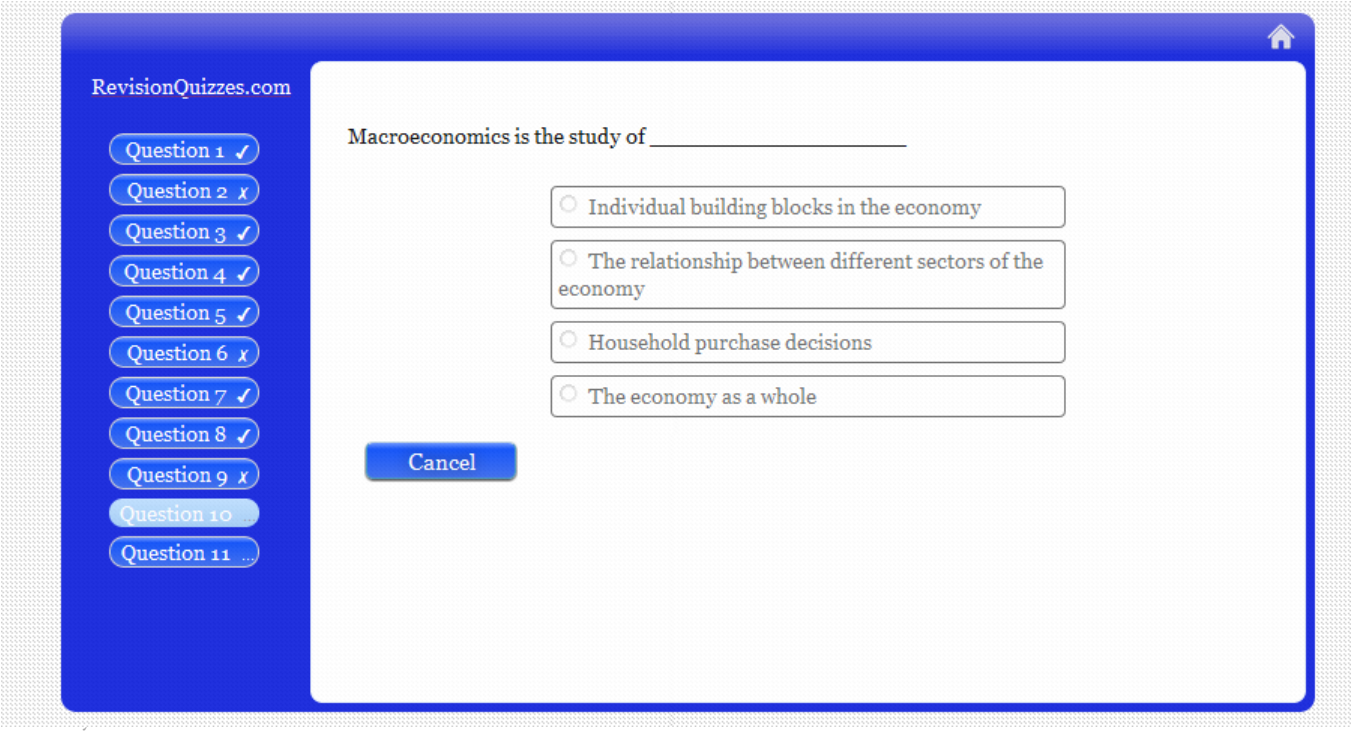
The Begin Screen:



The begin form again has remained similar as before, although the functionality has been improved. If there are zero takes, it will not show at all, same as with the average score. If there is a creator name linked with the quiz it will show, same is with the average and target time and all other details.

The Quiz Screen:

If there are no additional details added then the questions will be displayed in the centre, if there are any images or notes attached to the question, then the answers will be on the left to make best use of space.



If there are additional notes or an image uploaded to the question the answers will display on the left, with the notes or image on the right.

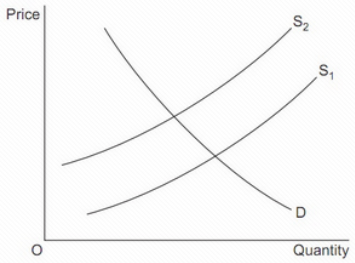
RevisionQuizzes.com

Question 1 ...

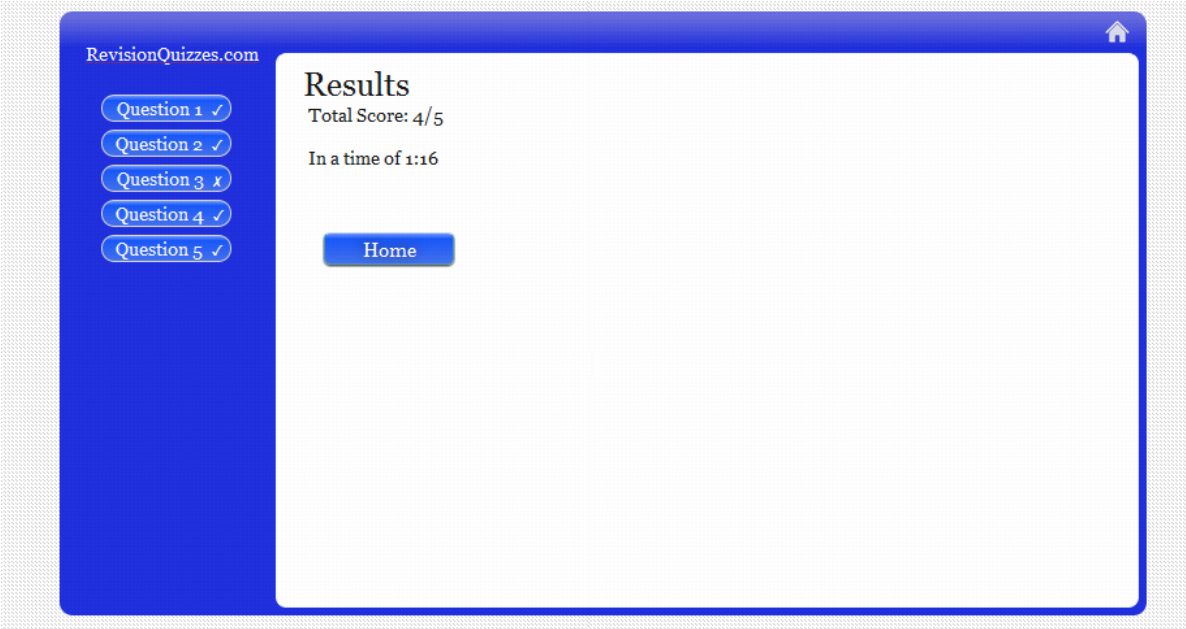
The diagram below shows the demand and supply curves for sports equipment. Which one of the following could explain the shift of the supply curve from S1 to S2 ?

- An improvement in the technology for making sports equipment
- The granting of a subsidy to sports equipment producers
- An increase in the price of a complementary good
- An increase in wages in the industry

Cancel



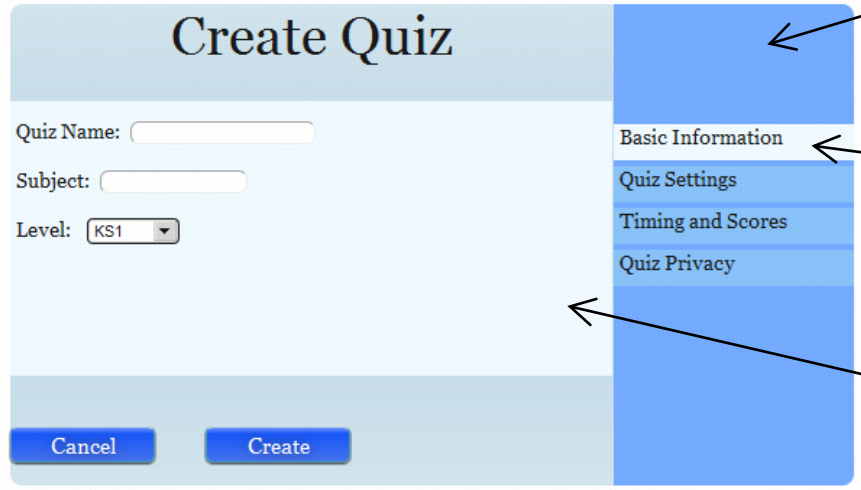
The Results form:



The results screen has changed from the last version, because for the user to see which questions they got right or wrong they use the navigation buttons along the side. This allows for more information about the question and answers to be displayed, as space was limited before.
The time and side pane and home button are still displayed as before.

When the user clicks a question, it will show the question with the correct answer highlighted in green, and if their answer was incorrect it will be highlighted in red. If an answer explanation was added by the user during the process of creating a quiz it will be displayed.

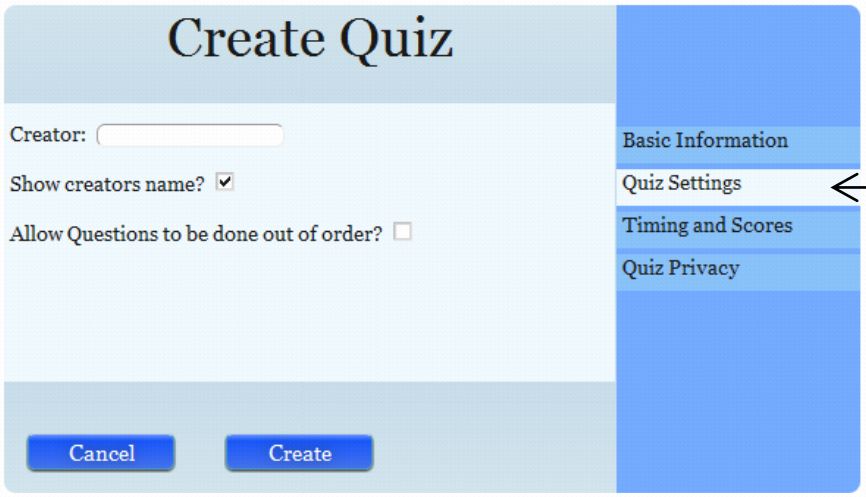
The Create a quiz form:



The side pane will be where all the tabs for additional options will be.

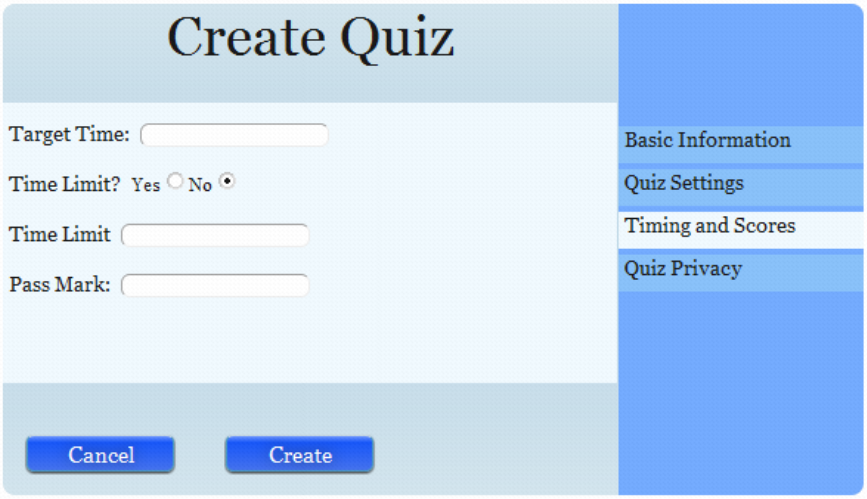
The selected tab will be highlighted in a different colour to improve clarity.

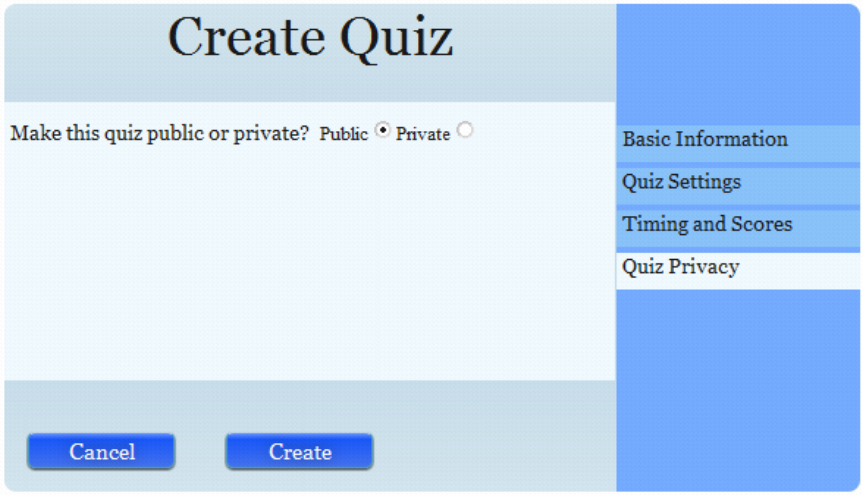
The main part has stayed the same since the last version. This is the only part that has to be filled in, the rest is all optional.



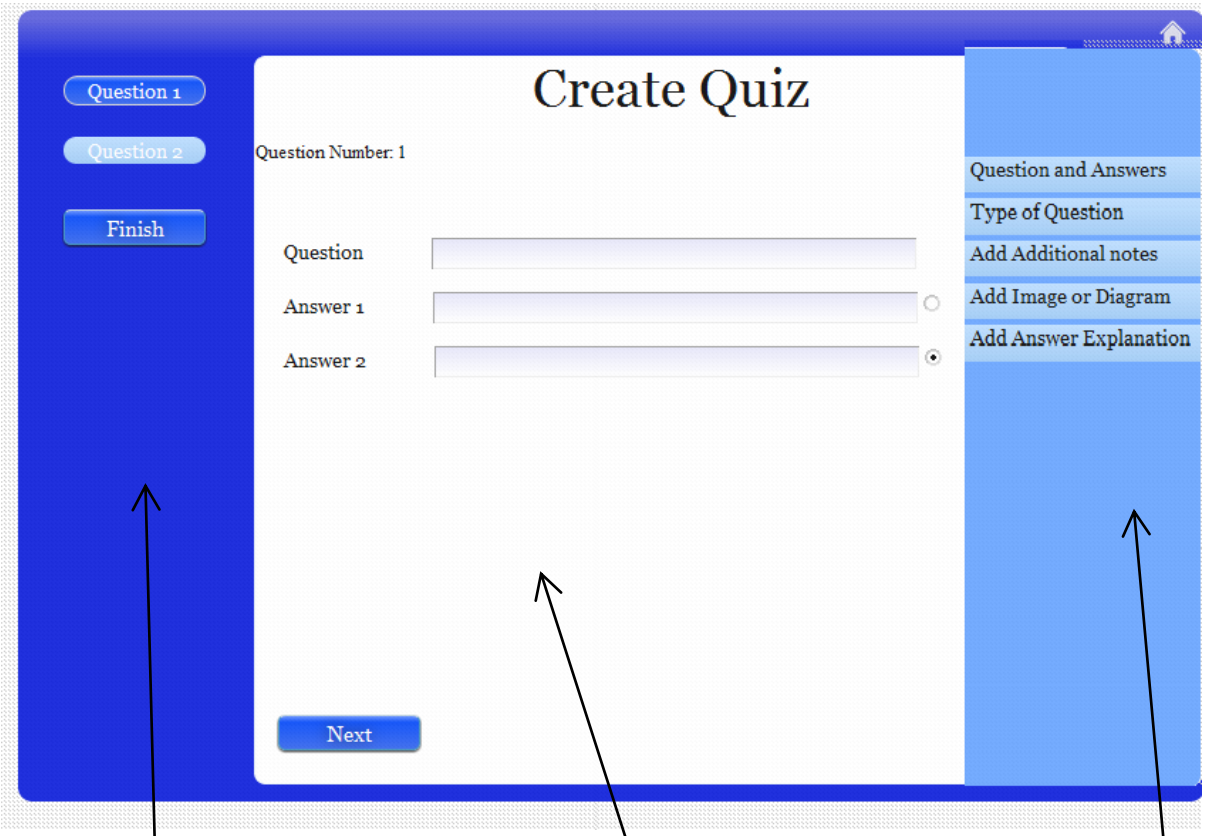
This is part of the same screen and uses JavaScript to hide and display different tabs, so the page doesn't have to be reloaded.

Where possible I have used checkboxes, radio buttons and drop down menus to improve the accuracy of the user's inputs.





The Create a question form:

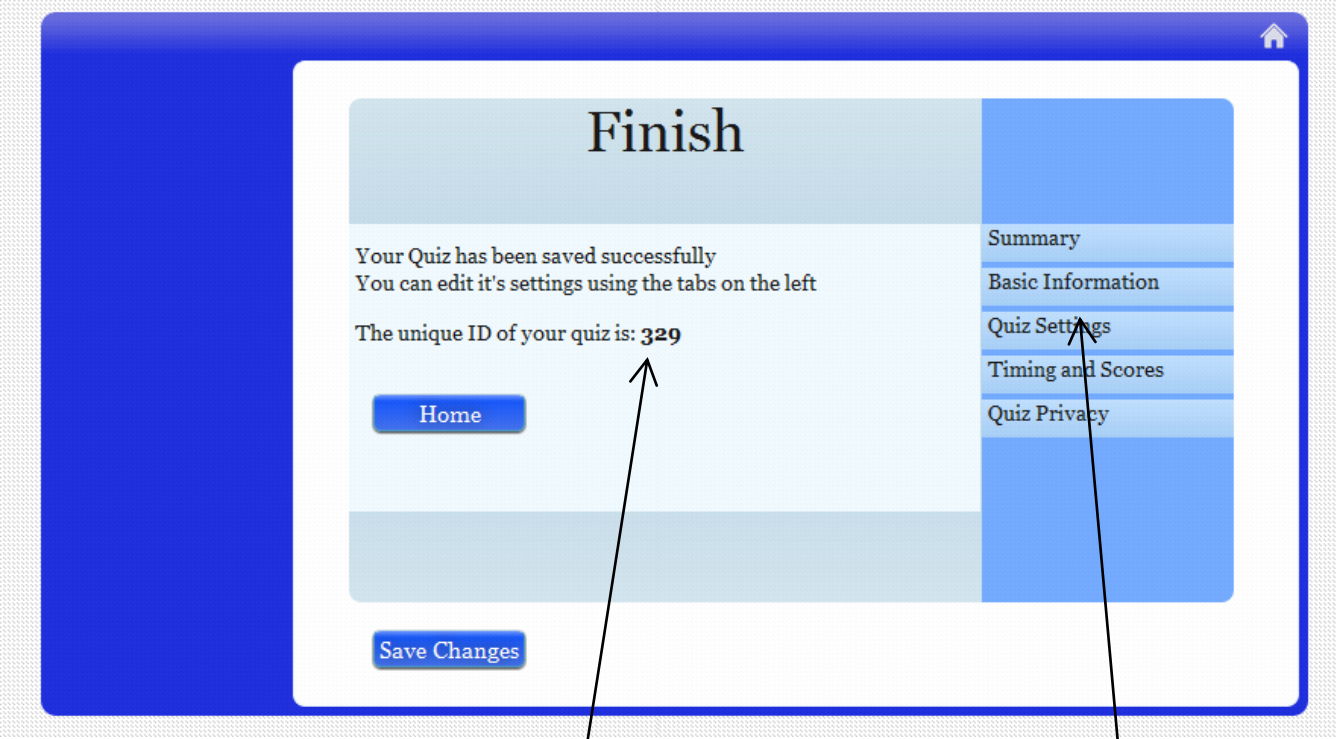


The questions will be listed on the left, so the user can click it to return and edit a question.

More answer text boxes will appear when the one above has a value in.

The tabs on the right can be used to enter additional details, answer explanation or an image to the question.

The create finish screen:



The information about the quiz will be displayed here.

The tabs on the right can be used to edit the additional details relating to the quiz.

There was a small error with left and right

Data Structures – V0.6

I have deleted the subjects table, and now instead of a subject ID being saved in the database, there is the whole subject again.

Code – V0.6:

The search function has been changed, the filters have been added in:

```
<?php if (isset($_GET['search'])) {

    // Get users search term
        $find = ($_GET["search"]);
        if ($find == "Search Quizzes"){
            $find = null;}
        $find = strtoupper($find);
        $find = strip_tags($find);
        $find = trim ($find);

//Check filters
        $subject = ($_GET['subject']);
        if($subject == "Subject"){
            $subject = null; }

         $level = ($_GET['level']);
        if($level == "level"){
            $level = null; }
//Search in db
        $dbsearch = mysql_query("SELECT * FROM quizzes
                WHERE Name LIKE'%$find%'
                AND Subject LIKE'%$subject%'
                AND Level LIKE '%$level%'");
        while ($b = mysql_fetch_array($dbsearch)){ $results[] = $b; }
```

To add additional details to questions:

```
<?php // Add the image or diagram to file
if($_FILES['imgupload']['name'] != "") {
define("LOCATION", "C:/Users/Lissy/wamp/www/quiz/data/images/");
//check image meets the criteria
if (($_FILES["imgupload"]["type"] == "image/gif")
  || ($_FILES["imgupload"]["type"] == "image/jpeg")
  || ($_FILES["imgupload"]["type"] == "image/png" )
  && ($_FILES["imgupload"]["size"] < 10000))
  {
    move_uploaded_file($_FILES["imgupload"][timestart()],
    LOCATION . $_FILES["imgupload"]["name"]);
  }
// Rename and save file name
  $fileName= ($_FILES["imgupload"]["name"]);
  $fileame=preg_replace("/'/","", $fileName);
  $ext = preg_split('/(?<![\\\])\./', $fileName);
  $n = count($ext)-1;
  $ext = $ext[$n];
  $newName = time().".".$ext;
  rename(LOCATION.$fileName, LOCATION.$newName);

$_SESSION['options'][$qn]['img']=$newName;
}
else {  $_SESSION['options'][$qn]['img']=null; }
?>

<?php // Add additional notes to question
if(isset($_POST['addNotes'])){
    $addNotes = ($_POST['addNotes']);
    $_SESSION['options'][$qn]['addNotes']=$addNotes;
}
else {$_SESSION['options'][$qn]['addNotes']=null;}
?>

<?php // Add add explanation of answer to  question
if(isset($_POST['ansExp'])){
    $ansExp = ($_POST['ansExp']);
    $_SESSION['options'][$qn]['ansExp']=$ansExp;
}
else {$_SESSION['options'][$qn]['ansExp']=null;}
?>
```

Also I have majorly refined the way new answer blocks are displayed. Now they are all created before, and just not visible until the JS function is called, which changes the visibility. This will make it easier when editing questions to display the previous value.

```
function addElement(ID) {
       document.getElementById('ansBlock'+ID).style.visibility = 'visible';  }
```

User specifications met – V0.6:

- A place to insert an image or diagram
- Question laid out on a clear screen
- Questions in order, or out of order
- Quizzes sorted by level and subject
- Show averages

Testing – V0.6:

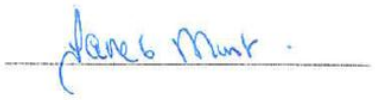
Test Name	Test Description	Input	Expected Outcome	Actual Output	Notes
Additional information	To test that additional information entered by the user is stored, put in database and shown during the quiz process.	[Additional Information]	For the information to be stored in session, then put in database then outputted to the user	Dumping the value of the session showed that it was stored, the database shows that it was submitted, and it was shown during quiz process.	Worked
Add Image or Diagram	To check that the user can add an image or diagram to their question, which the address of which will be stored in database, and the file itself will be stored in a noted directory	[File Uploaded] (JPEG file under 2MB)	The file to be uploaded to directory and address stored in the database, and the image shown during the quiz.	The file was uploaded, and address saved and the image was shown during the quiz process. A message telling the user that file upload was successful was outputted.	Worked
Add an unauthorised or unsupported file to the file upload option	To check that the system rejects any files which cannot be displayed during the quiz, or are harmful, or will take up too much space.	[File Uploaded] (over 2MB, or not JPG, PNG, GIF or BMP type)	The system should reject the file, and not save it to the directory. The user should be informed there file was rejected	The file was rejected, nothing was entered into the database and the file was not uploaded. A message was outputted to the user to inform them.	Worked
Answer Explanation	To check that the answer explanation is saved and outputted at the correct time (only once the user has completed that	[Answer Explanation]	The answer explanation should be added to the database.	The answer explanation was stored in the database, and only displayed to the user once the question	Worked

	quiz)			was done.	
Ignore blank fields	To check that if no additional details, answer explanation or image is selected, nothing should be added to the database	[Blank]	The index in the session array should be defined, but given the value of null. Nothing should be displayed to the user	The index was defined and given the value of null. During the quiz process, nothing was displayed to the user, and when all are left blank the questions will be in the middle.	Worked.
Edit quiz information	To edit the quiz information immediately after the quiz has been submit.		That the original quiz information is displayed to the user after completion and the changes they make to this will be saved in the database	The original quiz information was outputted in the textboxes, the changes the user makes to this data was updated in the quiz table of the database.	Worked

End user involvement – V0.6:

Version 0.6	
Pros: • blue • very usefull to be able to add all the additional information to quizzes and questions. • good to be able to edit quiz after completion	Cons: • can't edit a quiz from main menu • A basic user system would be good. • A better way to view score averages would be good too. • The layout doesn't fit on screen.
Notes:	

Confirmation of End user involvement

Signed End user [Mrs Munt] 

Signed Developer [Alicia Sykes] 

Review of V0.6

In version 0.6 I have fixed the errors caused by subjects being offset by one, and there is no longer a subjects table. My end user involvement showed that being able to edit a quiz after completion was useful, although this could be improved further to allow the user to edit the quiz and all the questions any time after completion. Also there were some problems with the layout, and not fitting on screens of some sizes, with parts being cut off, I will address this in the next version too. Finally my end user wanted a better way to be able to view scores and the score breakdown after the quiz was completed. I think that it could benefit from better and more complete validation checks.

Version 0.7

Objectives – V0.7

In version 0.7 I would like to create a user system, that will allow users to log on, to create a quiz, this will mean that they can come back to edit or delete the quiz. It will be optional for users to sign up.

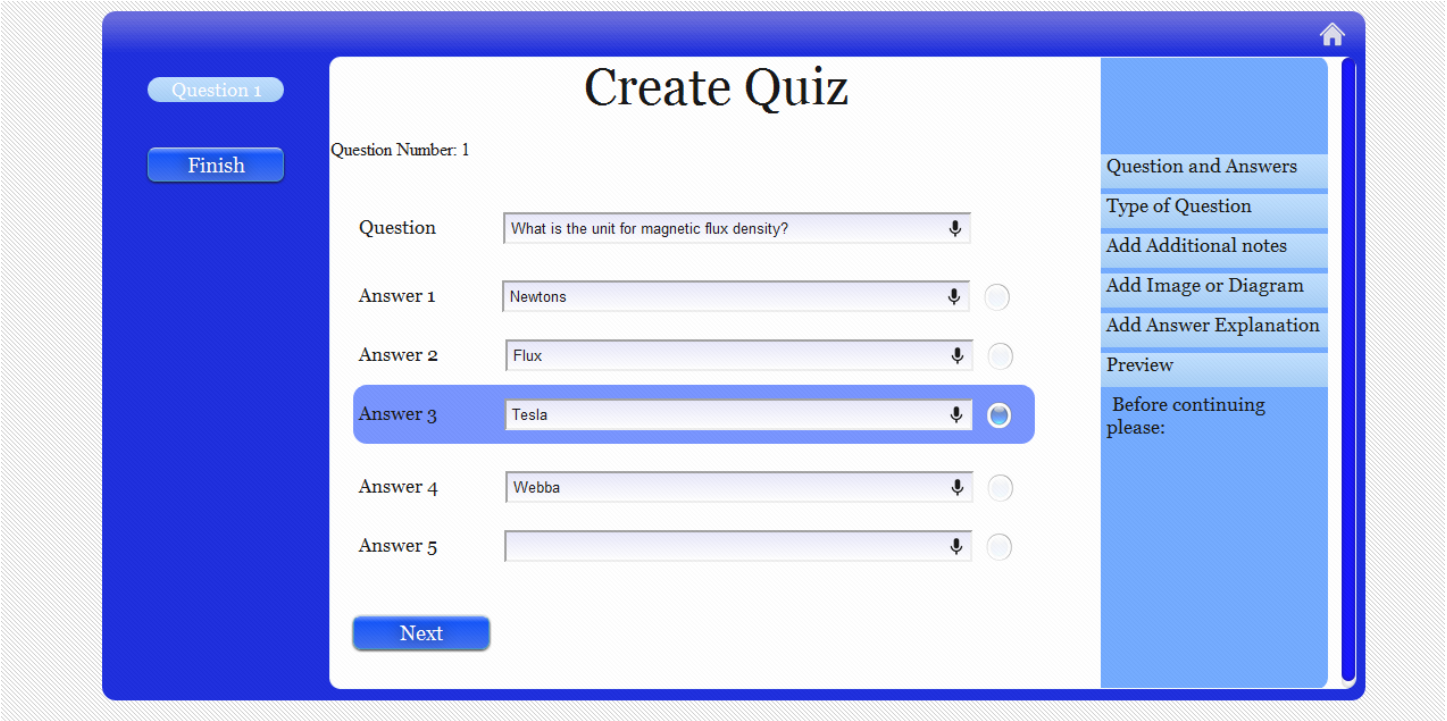
I am also going to further develop the search function, to include an instant search the results before the user presses enter, and a 'did you mean' suggestion if the users search term matches no results, searching from existing quiz titles.

Also I am going to improve the design, because previously some objects did not fit on smaller or zoomed in screens, and there were no scrollers, and also the CSS gradients were slow to load.

I will also improve the way averages are shown, using lengths of a div. I will add more validation too.

Screen Designs – V0.7

The Create a Question Form:



The preview question tab will show what the user has typed as they type it, below is a cropped screenshot of the contents of the preview tab:

Question Number: 1

Question) What is the unit for magnetic flux density?

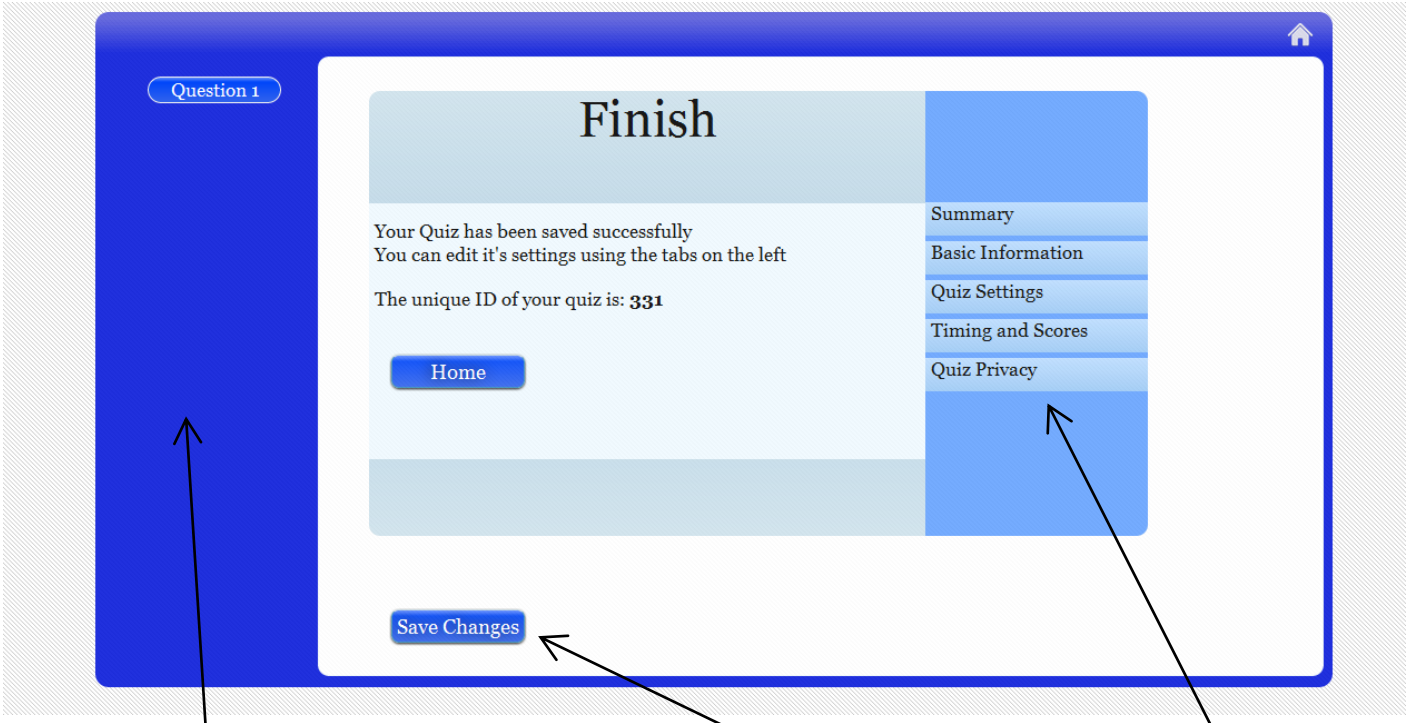
a) Newtons

b) Flux

c) Tesla [CORRECT]

d) Webba

The create finish/ create quiz form:

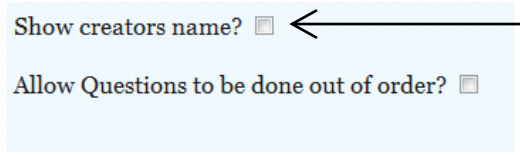


The questions will be listed on the left [in this example there is only 1 question] .

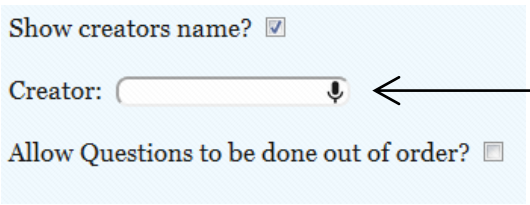
The create quiz and create finish form are virtually the same. From this form the user can make changes to all the quiz information and press save changes to update the database

The user will use these tabs to add additional information to their

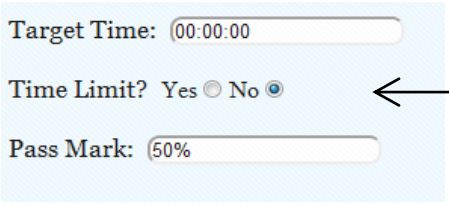
Under the quiz settings tab there are the following fields:



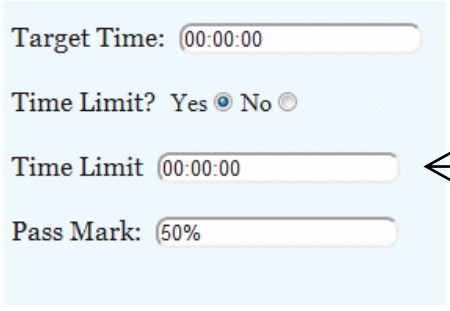
When the user checks this box the creators name field will appear with a blinds animation and the rest of the check boxes below will slide down to make more room.



If the user is logged in, there user name will be in the textbox by default. If not they will need to type the name they would like to appear.

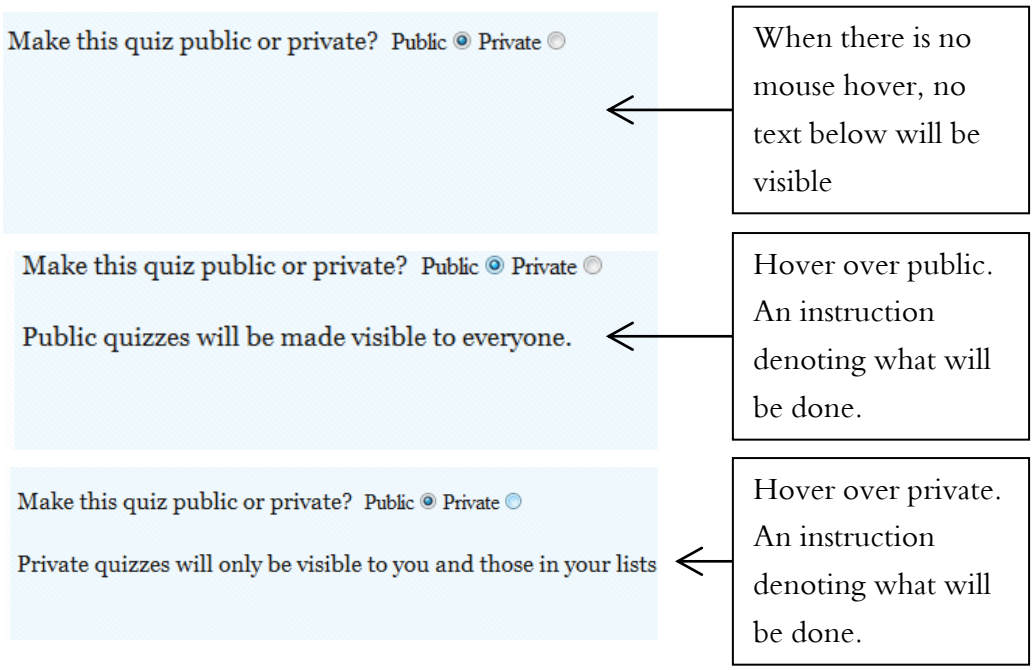


A similar thing happens to the time limit radio button, when yes is selected the text box becomes visible, and when no is selected it is hidden.

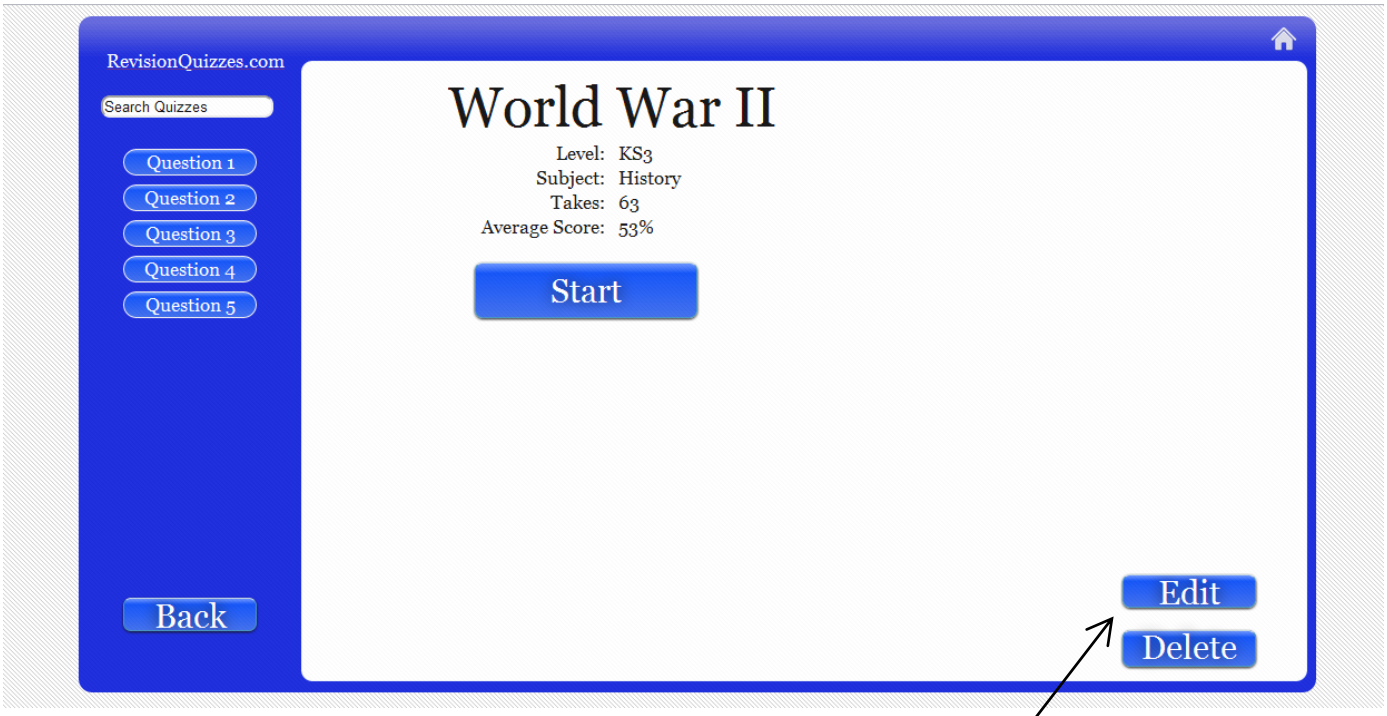


I will add a time picker or something to make selecting the time easier in the next version.

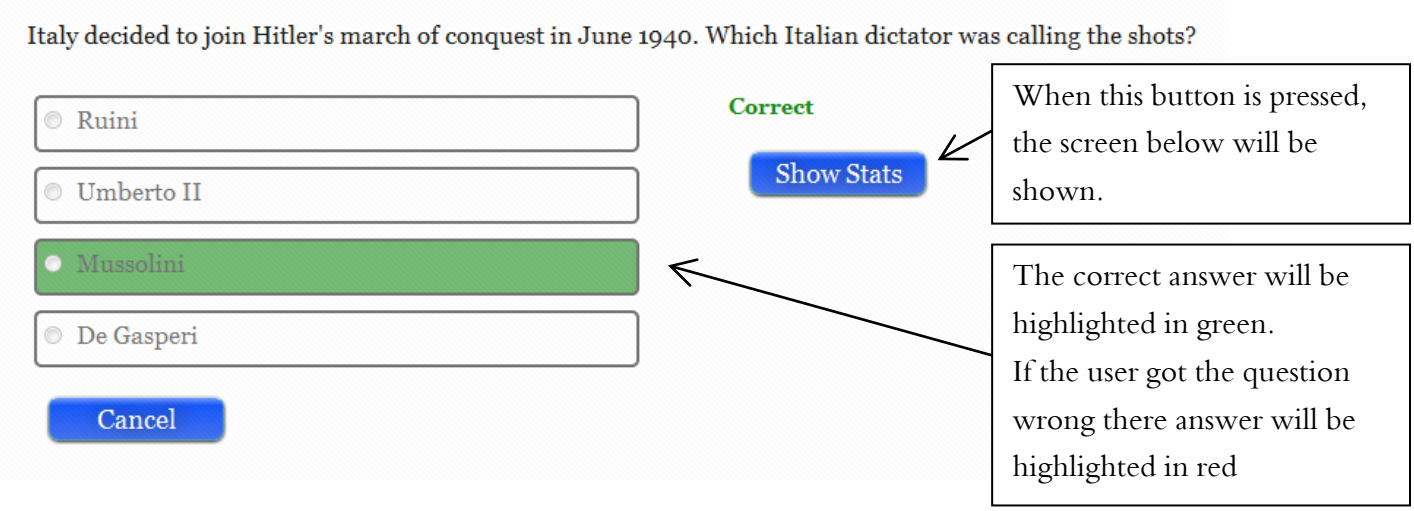
As the end user was not sure what all the options did I also added text that will be shown when the mouse is hovered over an option. For example:

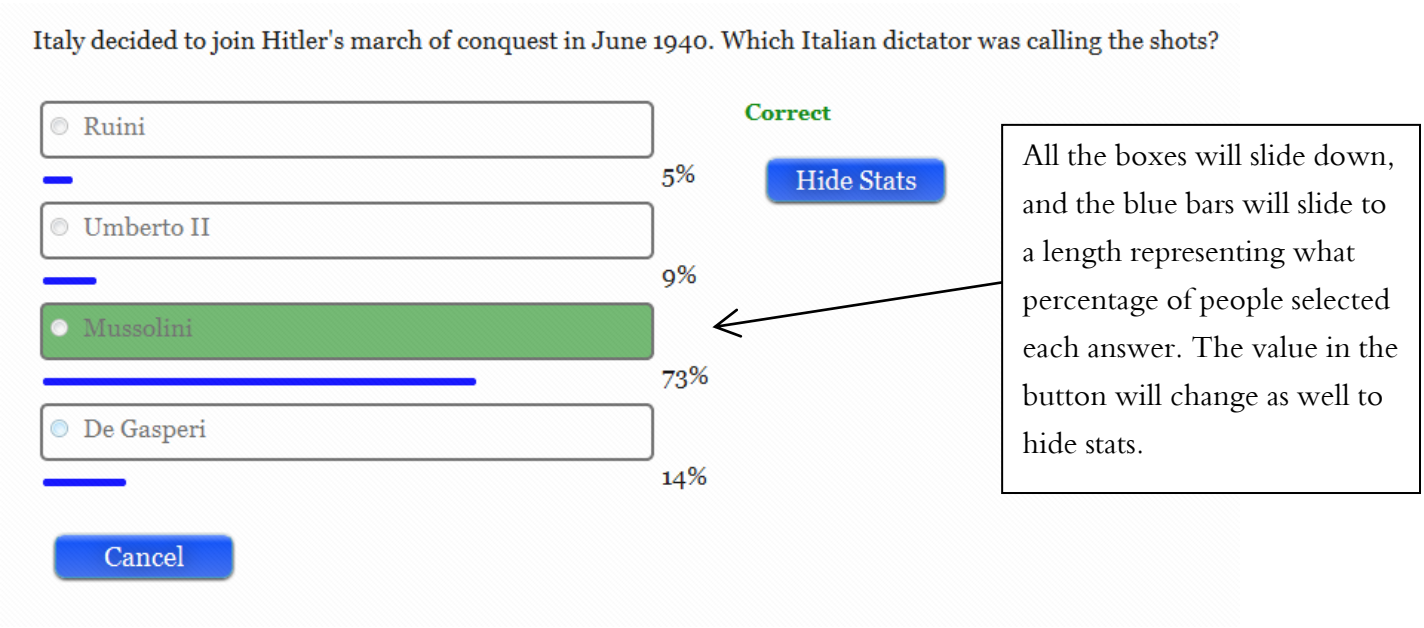


The begin screen:



The results (Showing on the quiz screen)

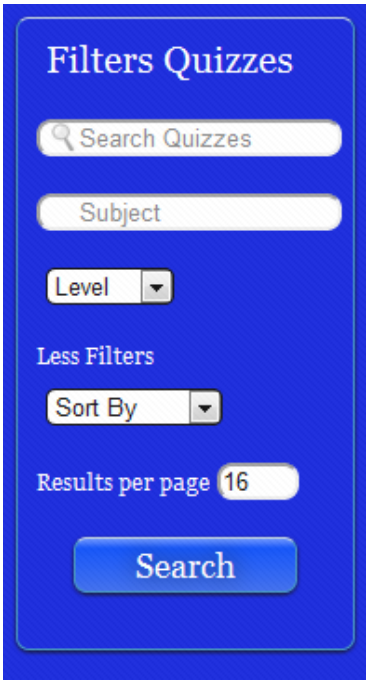


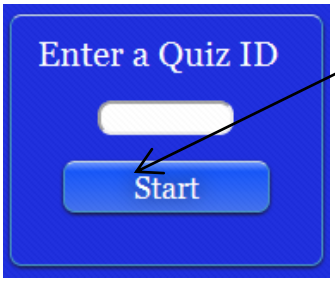


The search screen:

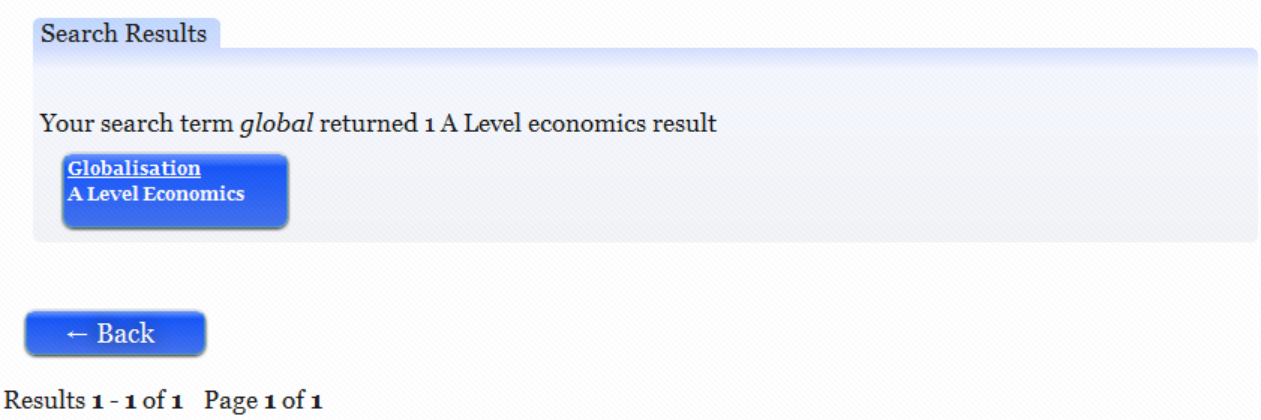
On the side of the screen on the main page the search filters box will appear:

When the user clicks more filters, the filters box will increase in size to show the additional options.



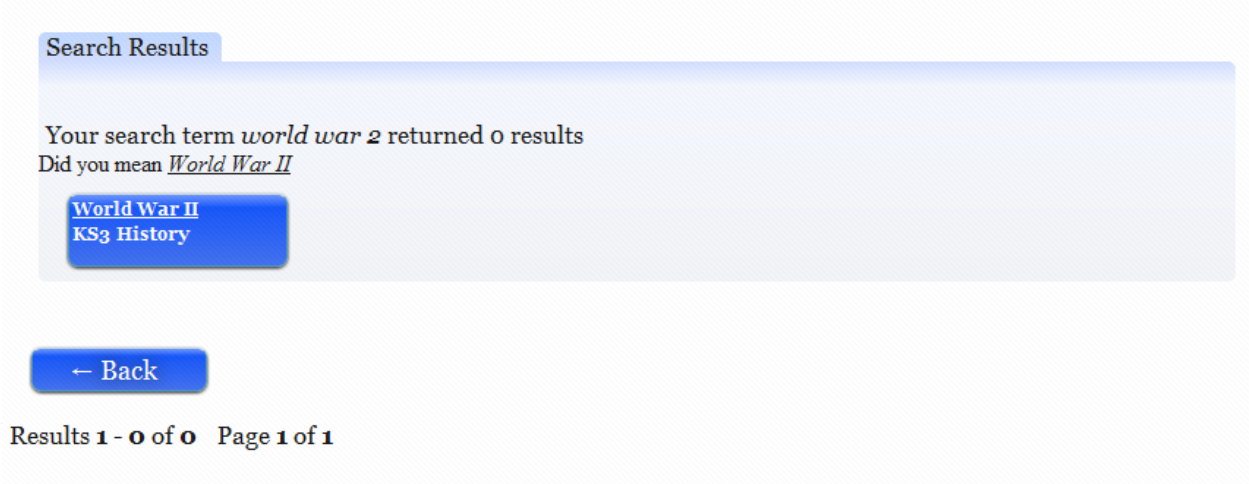


The user can still go state to a quiz if they know it's ID.



The instant search:

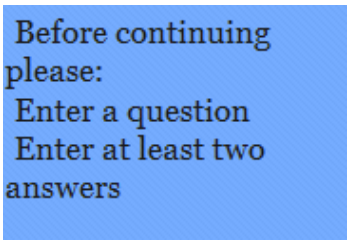
The 'did you mean' function:



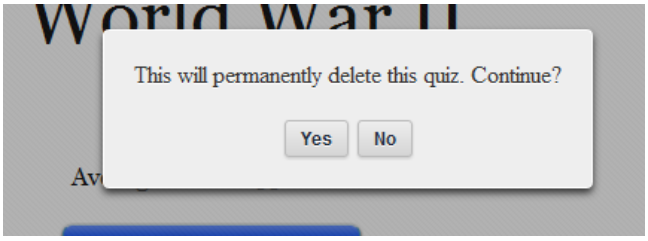
Validation – V0.7

While the user is creating a quiz, they cannot forget to miss out any important data like question, answers or correct answer, they will not be able to continue.

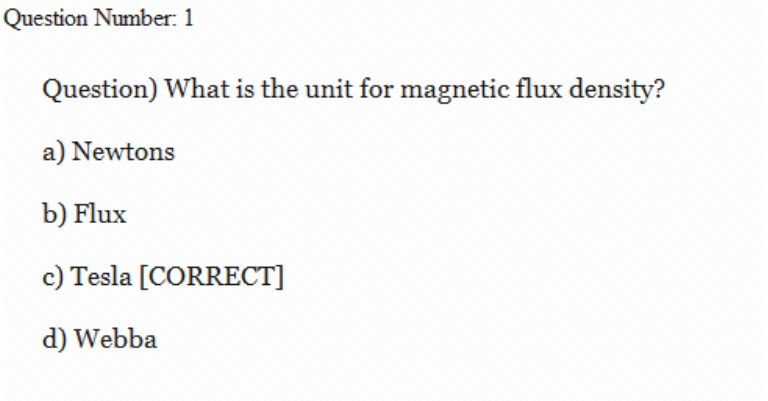
There is also some live validation on the right hand side of the screen telling the user what they must do before they can continue. It automatically updates itself on any key press.



There is also a warning before the user can delete a quiz



The question preview will improve data verification



Code – V0.7:

The start form:

The declaration and initialisation of the variables used by the jQuery:

```
    var visFilters = false;
    var visDidyoumean = false;
    var visLogin = false;
    var visIDblock = false;
```

To show and hide some of the elements:

Enter quiz ID block:

```
$("div#IDblock").click(function() {
  if (visIDblock == false){
    $("div#showIDblock").show('slow');
    visIDblock = true;}
 else if (visIDblock == true){
    $("div#showIDblock").hide('slow');
    visIDblock = false;}
 });
```

Login bar:

```
$("div#logButton").click(function() {
  if (visLogin == false){
    var mainHeight = $("div#main").css("height");
    mainHeight = (mainHeight - 350);
    $("div#loginBlock").show('fast');
    $("div#main").css("height", "85%");
    visLogin = true;}

  else if (visLogin == true){
    var mainHeight = $("div#main").css("height");
    mainHeight = (mainHeight + 350);
    $("div#loginBlock").hide('fast');
    $("div#main").css("height", "93%");
    visLogin = false;}
});
```

Show more filters

```
$("p#more").click(function() {
  if(visFilters == false){
    $("div#moreFilters").show('slow');
    $(this).text('Less Filters');
    visFilters = true
  }
  else if (visFilters == true){
    $("div#moreFilters").hide('slow');
    $(this).text('More Filters');
    visFilters = false;
  }
});
```

'did you mean' options:

```
$("p#txtDidYouMean").click(function() {
  if(visDidyoumean == false){
    $("div#didYouMean").show('slow');
      visDidyoumean = true
  }
```

```
  else if (visDidyoumean == true){
    $("div#didYouMean").hide('slow');
    visDidyoumean = false;
  }
});
```

The sign up box

```
$("#signUp").click(function() {
  $("div#dim").fadeIn('slow');
  $("div#createAccount").show('slow');
  $("div#createAccount").draggable();
});

$("#dim").click(function() {
 $("div#dim").hide();
 $("div#createAccount").hide('slow');
});
```

The instant search function:

```
$(document).ready(function(){
    $("#search").keyup(function(){
        var filter = $(this).val(), count = 0;

        $(".browse button").each(function(){
            if ($(this).text().search(new RegExp(filter, "i")) < 0) {
                $(this).fadeOut(); }
            else {
                $(this).show();
                count++; }
        });
        if (count > 1||count==0){ s='s'}
        else {s=''}

        if (filter != ''){
        $("#filter-count").text("Your search term    "+
            filter+
            " returned "+
            count+
            " result"+
            s);}
        else  $("#filter-count").text('')
    });
});
```

Prepare the results onto pages

```
$pages = intval($numRes/$rpp); // Number of results pages
        if ($numRes % $rpp) {
            $pages++;}
        $current = ($page/$rpp) + 1;
        if (($pages < 1) || ($pages == 0)) {
            $total = 1;}
       else {
            $total = $pages;}
        $first = $page + 1;
        if (!((($page + $rpp) / $rpp) >= $pages) && $pages != 1) {
            $last = $page + $rpp;}
        else{
            $last = $numRes;}
```

The ‘did you mean’ function

```
<?php //did you meen
```

```
if ($numRes == 0){

$words=array();
for ($i=0; $i<$numNames; $i++){
$words[$i]  = $names[$i]['Name'];}

$shortest = -1;

foreach ($words as $word) {

    $lev = levenshtein($find, $word);
    if ($lev == 0) {
        $closest = $word;
        $shortest = 0;
    break; }

        if ($lev <= $shortest || $shortest < 0) {
            $closest  = $word;
            $shortest = $lev;}
}


 $dbdidyoumean = mysql_query("SELECT * FROM quizzes
                WHERE Name LIKE'%$closest%'");
        while ($b = mysql_fetch_array($dbdidyoumean)){ $didyoumean[] = $b; }
    $countDidyoumean=count($didyoumean);
        }
?>
```

The 'create a quiz form':

To show the creators name when radio is selected:

```
$('input#showCre').click(function() {
  if($(this).is(':checked')){
    $('#showCreator').show('slow');}
  else { $('#showCreator').hide('slow'); }
});
```

To show the target time box when radio is selected:

```
$("input[name='time']").click(function() {
  if($('#limitYes').is(':checked')){
    $('#timeLimit').show('slow'); }
  else if ($('#limitNo').is(':checked')){
    $('#timeLimit').hide('slow');}
});
```

To give explanation of public and private quizzes, and other features on hover:

```
$("input#public").hover(function() {
  $("p#pub").show('fast'); },
function() {
  $("p#pub").hide('fast'); });

$("input#private").hover(function() {
  $("p#priv").show('fast'); },
function() {
  $("p#priv").hide('fast'); });
```

The ‘create question form’:

To add change the radio buttons:

```
$("[name='rad']").click(function () {
    $("[name='rad']").attr("src", 'img/radOff.png');
    var presetValue = $(this).attr('id');
    $("div.selectedAns").removeClass('selectedAns');
    $("#ansBlock"+presetValue.toUpperCase()).toggleClass('selectedAns');
    $('input[name="correctAns"]').removeAttr('checked');

$("[name='correctAns']").filter("[value="+presetValue+"]").attr("checked","checked
");
    var src = $(this).attr("src").replace("Off", "On");
    $(this).attr("src", src);
 });
```

To show the list of question types:

```
$("#typeSide").click(function () {
  if (visible == false){
   $(this).fadeTo(250, 0.95, function () {
   $("#typeList").show("slow");
   $("#typeList").toggleClass('typeQuestion');
   $(this).animate({height: 150},500);
    });
    visible = true;
  }
  else if (visible == true){
   $(this).fadeTo(250, 1, function () {
   $("#typeList").hide("slow");
   $("#typeList").removeClass('typeQuestion');
   $(this).animate({height: 30},500);
    });
    visible = false;
  }
});
```

For the instant validation on the side:

```
$("input#question").keyup(function () {
  if ($(this).val() === ''){
    $("p#queV").text('Enter a question');}
  else {$("p#queV").text('');}
}).keyup();

$("input[name='optionB']").keyup(function () {
  if ($(this).val() === ''&& $("input#optionA").val()!=''){
    $("p#ansV").text('Enter at least two answers');}
  else {$("p#ansV").text('');}
}).keyup();

$("input[name='optionA']").keyup(function () {
  if ($(this).val() === ''){
    $("p#ansV").text('Enter some answers');}
  else if(($(this).val() != '') && ($("input[name='optionB']").val() != '')) {
    $("p#ansV").text('');}
  else if(($(this).val() != '') && ($("input[name='optionB']").val() === '')) {
    $("p#ansV").text('Enter at least two answers');}
}).keyup();

$("input[name='correctAns']").click(function(){
  if (!$("input[name='correctAns']:checked").val()) {
    $("p#canV").text('Select a correct Answer'); }
  else if ($("input[name='correctAns']").is(':checked')){
    $("p#canV").text('blank');}
});
```

```
$(document).ready(function(){
  if (($("p#queV").val() != '')
  && ($("p#ansV").val() != '')
  && ($("p#canV").val() != '')){
    $("p#val").text('You can add additional details using the tabs on the
right');}
else {$("p#val").text('Before continuing please:');}
});
```

For the live preview tab:

```
$("input#question").keyup(function () {
  var val = $(this).val();
      $("p#prevQuestion").text(val);
    }).keyup();

$("input#[name='optionA']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsA").text(val);
}).keyup();

$("input#[name='optionB']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsB").text(val);
}).keyup();

$("input#[name='optionC']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsC").text(val);
}).keyup();

$("input#[name='optionD']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsD").text(val);
}).keyup();

$("input#[name='optionE']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsE").text(val);
}).keyup();

$("input#[name='optionF']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsF").text(val);
}).keyup();
```

For the login page:

```
<?php

$username=$_POST['username'];
$password=$_POST['password'];


$username = stripslashes($username);
$password = stripslashes($password);
$username = mysql_real_escape_string($username);
$password = mysql_real_escape_string($password);
$password = md5($password);

$dbusers=("SELECT * FROM users WHERE `username`='$username' AND
`password`='$password'");
  $result=mysql_query($dbusers);
  $count=mysql_num_rows($result);
  $userData = mysql_fetch_array($result);
```

```
if($count==1){
    echo "Login Succesful";
$_SESSION['logIn']['username'] = $username;
$_SESSION['logIn']['userID'] = $userData['ID'];

}
else {
  echo "Wrong Username or Password";
}

?>
```

For the new global page, that will be linked with all the pages:

```
<?php
//Database Connection
$servername = 'localhost';
$username = 'root';
$password = null;
$database = 'quiz';

$con = mysql_pconnect($servername,$username,$password);
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db($database);


//Constants and varibles
define("LOCATION", "C:/Users/Lissy/wamp/www/quiz/data/images/"); //location for
images

?>
```

Code Listing

File Structure

I have organised my code into several separate files. There are several advantages of this:

- Each script is short enough to be understandable
- Many scripts are standard procedures, I will use them several times in different places
- Maintenance becomes easier
- It is more organised, to find a file as they are sorted into relevant folders
- Testing becomes easier as each script can be tested independently with test data

I have separated the different languages where possible to increase clarity, so PHP, JavaScript and HTML are all in different files.

- **Arr** (JSON arrays to be passed to JavaScript)
  - **Subjects.json** (the list of subjects from database)
  - **Usernames.json** (the list of usernames)
  - **quizTitles.json** (list of quiz titles)
- **Actions**(small standard procedures)
  - **addScore.php** (adds high scores to database)
  - **contact.php** (sends messages to admin)
  - **createAccount.php** (adds user details to database)
  - **deleteQuiz.php** (deletes a quiz and associated data)
  - **leaveFeedback.php** (sends data from leave feedback form)
  - **loading.html** (defult loading screen reused)
  - **logIn.php** (logs the user in)
- **Create** (contains files relating to creating a quiz)
  - **addQuestion.php** (adds questions and answers to db)
  - **addQuiz.php** (adds or updates quiz and associative data)
  - **addToDatabase.php** (adds session arrays to database)
  - **answerBox.php** (this is a dynamic DOM object)
  - **createFinish.php** (this is the finish screen)
  - **createQuestion.php** (this is the add question form)
  - **createQuiz.php** (this is the start form for creating)
  - **questionValidate.js** (this is the validating scripts)
- **Data** (contains all user uploaded data, eg images…)
- **Forms** (contains reused forms)
  - **top.html** (this is the top bar and html objects)
  - **side.html** (this is the side bar and html objects)
- **img** (contains all graphics used throughout the program)
  - **bg.gif** (the background image)
  - **close.png** (the close button)
  - **home.png** (the home button)
  - **mag.png** (the search icon)
  - **radOff.png** (the radio button off)
  - **radon.png** (the radio button on)
  - **reviLogo.png** (the main logo)
- **lib** (contains all imported libraries, not my work)
- **quiz** (contains all scripts relating to the running of quiz)
  - **begin.php** (the first screen of a quiz)
  - **calculateBegin.php** (the calculations for begin screen)
  - **calculateQuiz.php** (the calculations for the quiz screen)
  - **calculateResults.php** (the calculations for the results)
  - **highScores.php** (calculating, adding, retrieving scores)
  - **quiz.php** (displays questions and answers)
  - **results.php** (displays results)
- **start** (contains scripts relating to searching + starting quiz)
  - **getQuiz.php** (contains queries to fetch quizzes)
  - **homeScript.js** (controls all the jquery for home screen)
  - **instantSearch.js** (the JavaScript live search script)
  - **redirectMessages.php** (messages displayed to the user)
  - **search.php** (sub routine for all search features)
  - **start.php** (the home form)
- **styles** (contains all the cascading style sheets)
  - **general.css** (styles that apply to everything)
  - **fonts.css** (all font faces, styles and effects)
  - **bigObjects.css** (big objects like wrappers)
  - **smallObjects.css** (small objects like text boxes)
  - **gradiesnts.css** (all fill gradients)
- **global** (files which will be accessed through out)
  - **dbConection.php** (the database conection login)

addScore.php

```php
<?php
/*
 * This script adds the users name to the database if they are not logged in
 */
header("Refresh: 1; url=../start/start.php");
require("../global.php");
$output = 'Adding Score...';
include 'loading.php';

$username = ($_POST['scoreName']);
$scoreID = ($_POST['scoreID']);

$username = ucwords(strtolower($username)).'(Guest)';

mysql_query("UPDATE `scores` SET `user` = '$username'
WHERE `ID` = '$scoreID'");

?>
```

contact.php

```php
<?php
/*
 * This script sends the users message via email
 */

// fetch users data
$usersName = ($_POST['contactName']);
$usersMail = ($_POST['contactEmail']);
$usersSubj = ($_POST['contactSubject']);
$usersMesg = ($_POST['contactMessage']);


// Prepare data for sending
$usersMesg = wordwrap($usersMesg);


// Create headers
$headers = 'From:'.$usersName."\r\n" .
           'Reply-To:'.$usersMail. "\r\n" .
           'Sent on: '.date('d-m-y h:i:s a', time()).
           'X-Mailer: PHP/' . phpversion();


//send mail
$sendTo = 'contact@revisionquizzes.com';
mail($sendTo, $usersSubj, $usersMesg, $headers);


// inform user and redirect
$output = 'Sending Message...';
 header("Refresh: 1; url=../start/start.php");
```

```
require('loading.php');
?>
```

createAccount.php

```
<?php
/*
 * The create account page
 */

require("../global.php");

$output = 'Creating Account...';

// fetch users inputed data, and prepaire it for db
$newUsername = ucwords(strtolower(mysql_escape_string(($_POST['SUserName']))));
$newPassword = md5(mysql_escape_string(($_POST['SPassword'])));
$newUserType = mysql_escape_string(($_POST['SstudentTeacher']));


//check that the username isnt taken
$dbusername = mysql_query("SELECT COUNT(`username`)
                    AS num FROM `users`
                    WHERE `username` = '$newUsername'");
$numConflictingUsernames = mysql_fetch_assoc($dbusername);
$numConflictingUsers = $numConflictingUsernames['num'];
if ($numConflictingUsers > 0){
    echo 'Username is already taken';
    $output ='Username Taken';
    header("Refresh: 1; url=../start/start.php");
    }


//check that the passwors match
if (($_POST['SPassword']) != ($_POST['SConPassword'])){
    echo 'Passwords don\'t match';
    $output ='Passwords don\'t match';
    header("Refresh: 1; url=../start/start.php");
}


// insert into database
mysql_query("INSERT INTO users (username, password, userType)
VALUES ('$newUsername', '$newPassword', '$newUserType')");


// Log in
$_SESSION['logIn']['username'] = $username;
$_SESSION['logIn']['userID'] = $userData['ID'];


//redirect back to start page
header("Refresh: 1; url=../start/start.php");
include 'loading.php';
?>
```

deleteQuiz.php

```
<?php
/*
 * This script deletes a quiz
 */

require("../global.php");
header("Refresh: 1; url=../start/start.php");
$output = 'Deleting quiz...';
require('loading.php');
$quizID = ($_POST['qtd']) ;
mysql_query("DELETE FROM `quizzes` WHERE `ID`= '$quizID'");
mysql_query("DELETE FROM `questions` WHERE `quiz_ID` = '$quizID'");
?>
```

logIn.php

```
<?php
/*
 * This script logs the user in
 */
session_start();

require("../global.php");

//Get username and password and put into correct format
$username=$_POST['username'];
$password=$_POST['password'];
$username = stripslashes($username);
$password = stripslashes($password);
$username = mysql_real_escape_string($username);
$password = mysql_real_escape_string($password);
$username = ucwords(strtolower($username));
$password = md5($password);


//Search in database
$dbusers=("SELECT *
           FROM users
           WHERE `username`='$username'
           AND `password`='$password'");
  $result=mysql_query($dbusers);
  $count=mysql_num_rows($result);
  $userData = mysql_fetch_array($result);

if($count==1){
    $output =  "Loging In...";
$_SESSION['logIn']['username'] = $username;
$_SESSION['logIn']['userID'] = $userData['ID'];
}
else {
  $output =  "Wrong Username or Password"; }

 // inform user and redirect
header("Refresh: 1; url=../start/start.php");
require('loading.php');
```

```
?>
```

loading.php

```
<!DOCTYPE html>
<html>
    <head>
        <title>Loading...</title>
        <style type="text/css">
          body{background-image:url('../img/bg.gif'); height:100%; width: 100%;
              }
          div{
             margin:0 auto; margin-top: 100px; border-radius: 12px;
             background-color: blue; height: 100px; width: 250px;
             border-right: 2px solid #2b2b2b; border-bottom: 2px solid #2b2b2b;
             padding: 15px;
            }
          p{
            font-family:  Georgia, Verdana, Tahoma; font-size:40px;
            text-align:center; display: inline; padding: 5px;
            color: #d4d6ff; text-shadow:0px 2px 3px #000552; font-weight: bold;
            }
        </style>
    </head>

    <body>
        <div>
            <p> <?php echo $output ?> </p>
        </div>
    </body>

</html>
```

addQuestion.php

```php
<?php
/*
 * This script adds the question and all assosiated data into array
 */


//Get question number, and increment
$qn = ($_POST['qn'])+1;


// Get users questions and answers
$question = ($_POST['question']);
$optionA = ($_POST['optionA']);
$optionB = ($_POST['optionB']);
$optionC = ($_POST['optionC']);
$optionD = ($_POST['optionD']);
$optionE = ($_POST['optionE']);
$optionF = ($_POST['optionF']);


//Find number of answers
$noa = 0;
if ($optionC==""){
    $noa = 2; }
else if($optionD==""){
    $noa = 3; }
else if($optionE==""){
    $noa = 4; }
else if($optionF==""){
    $noa = 5; }
else{
    $noa= 6; }


//Find correct answer
$ansA=$ansB=$ansC=$ansD=$ansE=$ansF='f';
$correctAns = ($_POST['correctAns']);
switch ($correctAns) {
    case 'a':
        $ansA='t';
        break;
    case 'b':
        $ansB='t';
        break;
    case 'c':
        $ansC='t';
        break;
    case 'd':
        $ansD='t';
        break;
    case 'e':
        $ansE='t';
        break;
```

```
    case 'f':
        $ansF='t';
        break;
}



// Add question to array
$questions = ($_SESSION["questions"]);
$questions[$qn]=$question;
$_SESSION['questions']=$questions;


// Add answers into array
$answers = ($_SESSION["answers"]);


switch ($noa){
    case 2:
        $answers[$qn]['answer']['a'] = $optionA;
        $answers[$qn]['correctAns']['a'] = $ansA;

        $answers[$qn]['answer']['b'] = $optionB;
        $answers[$qn]['correctAns']['b'] = $ansB;
    break;

    case 3:
        $answers[$qn]['answer']['a'] = $optionA;
        $answers[$qn]['correctAns']['a'] = $ansA;

        $answers[$qn]['answer']['b'] = $optionB;
        $answers[$qn]['correctAns']['b'] = $ansB;

        $answers[$qn]['answer']['c'] = $optionC;
        $answers[$qn]['correctAns']['c'] = $ansC;
    break;

    case 4:
        $answers[$qn]['answer']['a'] = $optionA;
        $answers[$qn]['correctAns']['a'] = $ansA;

        $answers[$qn]['answer']['b'] = $optionB;
        $answers[$qn]['correctAns']['b'] = $ansB;

        $answers[$qn]['answer']['c'] = $optionC;
        $answers[$qn]['correctAns']['c'] = $ansC;

        $answers[$qn]['answer']['d'] = $optionD;
        $answers[$qn]['correctAns']['d'] = $ansD;
    break;

    case 5:
        $answers[$qn]['answer']['a'] = $optionA;
        $answers[$qn]['correctAns']['a'] = $ansA;

        $answers[$qn]['answer']['b'] = $optionB;
        $answers[$qn]['correctAns']['b'] = $ansB;
```

```
        $answers[$qn]['answer']['c'] = $optionC;
        $answers[$qn]['correctAns']['c'] = $ansC;

        $answers[$qn]['answer']['d'] = $optionD;
        $answers[$qn]['correctAns']['d'] = $ansD;

        $answers[$qn]['answer']['e'] = $optionE;
        $answers[$qn]['correctAns']['e'] = $ansE;
    break;

    case 6:
        $answers[$qn]['answer']['a'] = $optionA;
        $answers[$qn]['correctAns']['a'] = $ansA;

        $answers[$qn]['answer']['b'] = $optionB;
        $answers[$qn]['correctAns']['b'] = $ansB;

        $answers[$qn]['answer']['c'] = $optionC;
        $answers[$qn]['correctAns']['c'] = $ansC;

        $answers[$qn]['answer']['d'] = $optionD;
        $answers[$qn]['correctAns']['d'] = $ansD;

        $answers[$qn]['answer']['e'] = $optionE;
        $answers[$qn]['correctAns']['e'] = $ansE;

        $answers[$qn]['answer']['f'] = $optionF;
        $answers[$qn]['correctAns']['f'] = $ansF;
    break;
}
$_SESSION['answers']=$answers;


// Add the image or diagram to file
if($_FILES['imgupload']['name'] != "") {
//check image meets the criteria
if ((($_FILES["imgupload"]["type"] == "image/gif")
  || ($_FILES["imgupload"]["type"] == "image/jpeg")
  || ($_FILES["imgupload"]["type"] == "image/png" )
  && ($_FILES["imgupload"]["size"] < 10000))
  {
        move_uploaded_file($_FILES["imgupload"]["tmp_name"],
        LOCATION . $_FILES["imgupload"]["name"]);
      }
      $fileName= ($_FILES["imgupload"]["name"]);
      $ext = preg_split("[/\\.]", $fileName);
      $n = count($ext)-1;
      $ext = $ext[$n];
      $newName = time().$ext;
      rename(LOCATION.$fileName, LOCATION.$newName);

    $_SESSION['options'][$qn]['img']=$newName;
    }
else {  $_SESSION['options'][$qn]['img']=null; }


// Add additional notes to question
if(isset($_POST['addNotes'])){
```

```
        $addNotes = ($_POST['addNotes']);
        $_SESSION['options'][$qn]['addNotes']=$addNotes;
}
else {  $_SESSION['options'][$qn]['addNotes']=null;}


// Add add explanation of answer to  question
if(isset($_POST['ansExp'])){
        $ansExp = ($_POST['ansExp']);
        $_SESSION['options'][$qn]['ansExp']=$ansExp;
}
else {  $_SESSION['options'][$qn]['ansExp']=null;}

?>
```

addQuiz.php

```php
<?php
/*
 * This script adds and updates the quiz to the session array
 */

//Fetch the users inputed data
$quizName=($_POST['quizName']);
if (isset ($_POST['jff'])){
    $level = null;
    $subject = 'Just for fun';}
else {
    $level=($_POST['level']);
    if ($level =='other'){
      $level = ($_POST['levelOther']);
      if ($level == 'level'){
        $level = null;} }
    $subject=($_POST['subject']);}
if(isset($_POST['showCre'])){
   $creator = ($_POST['creator']);}
else {$creator = null;}
if(isset($_POST['tarTime'])){
   $tarTime = ($_POST['tarTime']);}
else $tarTime = null;
if (isset($_POST['timeLimit'])){
   $timeLimit = ($_POST['timeLimit']);}
else $timeLimit = null;
if(isset($_POST['passMark'])){
   $passMark = ($_POST['passMark']);}
else $passMark = null;
$privacy = ($_POST['priv']);
if (isset ($_POST['ooo'])){
    $jumpOrder = 'true';}
else{$jumpOrder = 'false';}
?>

<?php // Remove illegal characters
$quizName = mysql_escape_string($quizName);
$creator  = mysql_escape_string($creator);
$subject  = mysql_escape_string($subject);
?>

<?php //Add Quiz information into an array
$quizInfo=array('name' => "$quizName",
    'level' => "$level",
    'subject' => "$subject",
    'creator' => "$creator",
     'tarTime' => "$tarTime",
    'timeLimit' => "$timeLimit",
    'passMark' => "$passMark",
    'jumpOrder' => "$jumpOrder",
    'privacy' => "$privacy");
$_SESSION['quizInfo']=$quizInfo;
?>
```

```php
<?php //Declaire varibles and arrays
$questions = ($_SESSION['questions']);
$answers = ($_SESSION['answers']);
?>
```

answerBox.php

```
<?php
/*
 * This is the input box where the user enters the answers
 * It is a lot of code, only because it may already contain values
 * It also needs to diplay correctly depending on what is in it
 * The purpose of it is to make less code,
 * and to make it easier to adjust, e.g max number of answers...
 */
?>
<?php
$num = 6; // Max number of answers per question (max = 12)
$vis = 2; //The number of answer blocks visible by defult
$let = array(1=> 'a', 2=> 'b', 3=>'c', 4=>'d', 5=>'e', 6=>'f',
             7=> 'g', 8=> 'h', 9=>'i', 10=>'j', 11>'k', 12=>'l'); //convert
 for ($i=1; $i<=$num; $i++) { //start loop?>
<div id="<?php echo 'ansBlock'.strtoupper($let[$i]);?>"
     class="ansBlock"
     name="<?php echo $let[$i]?>"
            style= "<?php if ($done==true) {
            if (array_key_exists($let[$i], $answers[$cqn]['answer'])
                && $answers[$cqn]['answer'][$let[$i]] != null){?>
                    visibility:visible; <?php }}
            else if ($i<=$vis) {?>
                visibility:visible; <?php }
            else { ?> visibility:hidden <?php } ?>" />
          <p class="subFont"><?php echo 'Answer '.$i;?></p>
          <input type="text"
                 name="<?php echo 'option'.strtoupper($let[$i])?>"
                 class="textboxQuestions"
                 <?php if ($i != $num){ ?>
                 onFocus="addElement('<?php echo strtoupper($let[$i+1]) ?>');"
                     <?php } ?>
                 <?php if ($done==true) {
                if (array_key_exists($let[$i], $answers[$cqn]['answer'])) {?>
                 value="<?php echo $answers[$cqn]['answer'][$let[$i]]?>"
                 <?php }}?>/>
          <input type="radio" name="correctAns" value="<?php echo $let[$i]?>"
                 style="visibility: hidden"
                 <?php if ($done==true) {
                     if (array_key_exists($let[$i], $answers[$cqn]['correctAns']))
{
                     if ($answers[$cqn]['correctAns'][$let[$i]]=='t'){?>
                         checked ="true"
             <?php }}} ?>/>
             <img
                 src="<?php if ($done==true) {
                     if (array_key_exists($let[$i], $answers[$cqn]['correctAns']))
{
                     if ($answers[$cqn]['correctAns'][$let[$i]]=='t'){?>
                         ../img/radOn.png <?php }}}
                 else { ?>
                         ../img/radOff.png <?php } ?> "
                 width="25" height="25"
```

```
                style="margin-left: -10px; margin-bottom: -8px;"
                id="<?php echo $let[$i]; ?>"
                name="rad"
                />
                </div>
<?php } // end loop ?>
```

addToDatabase.php

```
<?php

/*
 * This script adds the sessoin arrays into the database
 */

define("LOCATION", "C:/Users/Lissy/wamp/www/quiz/data/images/"); //location for
images

//finished adding in the questions
if (($_POST['qn'])=='null'){ //The quiz information has been updated

require 'addQuiz.php';
$quizID = ($_SESSION['quizInfo']['quizID']);
mysql_query( "UPDATE quizzes SET
                `Name` = '$quizName',
                `Subject` = '$subject',
                `Level` = '$level',
                `creator` = '$creator',
                `tarTime` = '$tarTime',
                `timeLimit`= '$timeLimit',
                `passMark` = '$passMark',
                `jumpOrder` = '$jumpOrder',
                `privacy` = '$privacy'
            WHERE `ID` = '$quizID' ")or die(mysql_error());



} // finish updating quiz information

else if (($_POST['qn'])=='edit'){ //the quiz is being edited


// add the quiz information into the array
$quizID = ($_POST['edit']);
$dbquizzes = mysql_query("SELECT * FROM Quizzes WHERE ID =  $quizID ");
    $quiz = array(); while ($b = mysql_fetch_array($dbquizzes)){ $quiz[] = $b;}

$Qname = $quiz[0]['Name'];
$Qsubject = $quiz[0]['Subject'];
$Qlevel = $quiz[0]['Level'];
$Qcreator = $quiz[0]['creator'];
$QtarTime = $quiz[0]['tarTime'];
$QtimeLimit = $quiz[0]['timeLimit'];
$QpassMark = $quiz[0]['passMark'];
$QjumpOrder = $quiz[0]['jumpOrder'];
$Qprivacy = $quiz[0]['privacy'];
$QquizID = $quiz[0]['ID'];

  $quizInfo=array('name' => "$Qname",
    'level' => "$Qlevel",
    'subject' => "$Qsubject",
    'creator' => "$Qcreator",
    'tarTime' => "$QtarTime",
```

```
    'timeLimit' => "$QtimeLimit",
    'passMark' => "$QpassMark",
    'jumpOrder' => "$QjumpOrder",
    'privacy' => "$Qprivacy",
    'quizID' => "$QquizID");
$_SESSION['quizInfo']=$quizInfo;

$creator = $quiz[0]['creator'];
$jumpOrder= $quiz[0]['jumpOrder'];
$qn=null;



} else if(($_POST['qn'])!='null'){ //If form has been sent from questions
  require 'addQuestion.php';


//Get quiz information
$quizInfo = ($_SESSION["quizInfo"]);
$quizName=$quizInfo['name'];
$quizLevel=$quizInfo['level'];
$quizSubject=$quizInfo['subject'];
$creator= $quizInfo['creator'];
$tarTime=$quizInfo['tarTime'];
$timeLimit=$quizInfo['timeLimit'];
$passMark=$quizInfo['passMark'];
$jumpOrder=$quizInfo['jumpOrder'];
$privacy=$quizInfo['privacy'];


// Get questions, answers and question options
$questions = ($_SESSION["questions"]);
$answers = ($_SESSION["answers"]);
$options = ($_SESSION["options"]);


//Add quiz information into database
mysql_query( "INSERT INTO quizzes (
                Name,
                Subject,
                Level,
                creator,
                tarTime,
                timeLimit,
                passMark,
                jumpOrder,
                privacy)
            VALUES (
                '$quizName',
                '$quizSubject' ,
                '$quizLevel',
                '$creator',
                '$tarTime',
                '$timeLimit',
                '$passMark',
                '$jumpOrder',
                '$privacy')") or die(mysql_error());
```

```
$quizID =  mysql_insert_id(); // return quiz ID
$_SESSION['quizInfo']['quizID'] = $quizID; //updates quizID


// Add questions into database
$noq = $qn + 1; // Find number of questions
for($j=1; $j<$noq; $j++){
   $addNotes = $options[$j]['addNotes'];
   $img = $options[$j]['img'];
   $ansExp = $options[$j]['ansExp'];

 mysql_query("INSERT INTO questions
         (Quiz_ID,
         QuestionNumber,
         Question,
         Notes,
         Image,
         Explanation)
VALUES ('$quizID',
         '$j' ,
         '$questions[$j]',
         '$addNotes',
         '$img',
         '$ansExp')")or die(mysql_error());
$questionID =  mysql_insert_id(); // return question ID


// Add answers into database
$enoa = count($answers[$j]['answer']);//get number of answers for each question
$lastLetter = array_pop(array_keys($answers[$j]['answer']));
foreach(range('a',$lastLetter) as $k){
    $dbans = $answers[$j]['answer'][$k];
    $dbcorans = $answers[$j]['correctAns'][$k];
     mysql_query("INSERT INTO answers (Question_ID, Answer, CorrectAnswer)
    VALUES ('$questionID', '$dbans', '$dbcorans')") or die(mysql_error());
}}
}


// get questions for side panel
$dbnoq = mysql_query("SELECT ID, COUNT(quiz_ID) FROM questions WHERE quiz_ID =
$quizID ");
$d = mysql_fetch_array($dbnoq);
$noq = $d[1];

$dbquestions = mysql_query("SELECT * FROM questions WHERE quiz_ID =  $quizID ");
$question = array(); while ($b = mysql_fetch_array($dbquestions)){ $question[] =
$b;}
for ($i=0; $i<$noq; $i++){
$questions = array();
$questions[$i]=$question[$i]['Question'];
}


$lvl = $quizInfo['level'];

?>
```

createFinish.php

```
    <?php session_start(); ?>
    <!DOCTYPE html >
    <html>
        <head>
            <script type="text/javascript" src="../lib/jquery.js"></script>
            <link rel="stylesheet" type="text/css" href="../styles/stylesheet.css" />
            <script type="text/javascript">

    function showTab(tab){

    document.getElementById('summary').style.visibility = 'hidden';
    document.getElementById('basicInfo').style.visibility = 'hidden';
    document.getElementById('furtherInfo').style.visibility = 'hidden';
    document.getElementById('timingScores').style.visibility = 'hidden';
    document.getElementById('quizPrivacy').style.visibility = 'hidden';

    document.getElementById(tab).style.visibility = 'visible';
    document.getElementById(tab).setAttribute("style","background-color: #F0FAFF;")
    }
    function timeLimit() {
       if (document.getElementByName('time')=='yes'){
          document.getElementById('timeLimit').style.visibility = 'visible';
       }}
    </script>

    </head>

    <body>

     <?php
     require("../global.php");
     require 'addToDatabase.php';
     ?>

<div class="back backFill">
<?php include '../forms/top.php'; ?>

     <div class="main">
        <form method="post" action="createFinish.php" name="cf">

             <div class="form">

                 <!-- TABS -->
                 <div class="sideForm sideFormFill">
<div class="formTabs formTabsFill"                    onClick="showTab('summary')">
                        <p class="subFont">Summary</p>
                    </div>

                    <div class="formTabs formTabsFill" onClick="showTab('basicInfo')">
                        <p class="subFont">Basic Information</p>
                    </div>

                    <div class="formTabs formTabsFill"
onClick="showTab('furtherInfo')">
```

```
                        <p class="subFont">Quiz Settings</p>
                    </div>


                    <div class="formTabs formTabsFill"
onClick="showTab('timingScores')">
                        <p class="subFont">Timing and Scores</p>
                    </div>

                    <div class="formTabs formTabsFill"
onClick="showTab('quizPrivacy')">
                        <p class="subFont">Quiz Privacy</p>
                    </div>
                </div>

          <?php if (($_POST['qn'])=='edit'){ ?>
             <p class="titleFont">Edit Quiz</p>
          <?php } else { ?>
             <p class="titleFont">Finish</p>
          <?php } ?>
             <br></br>

         <!-- SUMARY TAB -->
          <div id="summary"  class="innerForm">
                 <?php if(($_POST['qn'])=='null'){ ?>
                 <p class="subFont">Your changes have been saved successfully</p>
                 <?php } else if(($_POST['qn'])=='edit'){ ?>
                 <p class="subFont">You are editing this quiz</p>
                 <?php } else { ?>
                 <p class="subFont">Your Quiz has been saved successfully</p>
                 <?php } ?><br/>
                 <p class="subFont">
                     You can edit it's settings using the tabs on the left
                 </p><br></br>
                 <p class="subFont">
                     The unique ID of your quiz is:  <b><?php echo $quizID ?></b>
                 </p><br></br>
                 <input type="button"
                        value="Home"
                        class="button"
                        onClick="window.location.href='../start/start.php'"/>
                 <br></br>
         </div>

         <!-- BASIC INFO TAB -->
         <div id="basicInfo" style="visibility:hidden" class="innerForm innerFormFill">
             <p class="subFont">Quiz Name: </p>
             <input name="quizName"
                    class="textbox"
                    value="<?php echo $quizInfo['name'];?>"/>
             <br></br>


             <p class="subFont">Subject: </p>
            <input type="text" name="subject" id="subject" style="width:120px; "
class="textbox" value="<?php echo $quizInfo['subject'];?>"/><br></br>
```

```
            <p class="subFont">Level: </p>
                <select name="level" class="dropDown" id="searchLevel">
                    <option value="level" title="Select a Level">Level</option>
                    <option value="KS1" title="Ages 4-6">KS1</option>
                    <option value="KS2" title="Ages 7-10">KS2</option>
                    <option value="KS3" title="Ages 11-13">KS3</option>
                    <option value="GCSE" title="Ages 14-15">GCSE</option>
                    <option value="A Level" title="Ages 16-17">A Level</option>
                    <option value="higher">Higher</option>
                    <option value="other"
                            title="An input box will appear for you
                            to type a non-specified level">Other...
                    </option>
                </select>
        </div>


        <!-- FURTHER INFO TAB -->
        <div id="furtherInfo" style="visibility:hidden" class="innerForm">
             <p class="subFont" style="display:inline;">Show creators name?  </p>
            <input type="checkbox"
                   name="showCre"
                   id="showCre"
                   <?php if ($creator != null){?>checked="checked"<?php }?>/>
            <br></br>

            <div id="showCreator" style="display:none;">
                <p class="subFont">Creator: </p>
                <input name="creator"
                       class="textbox"
                       value="<?php echo $quizInfo['creator'];?>" />
                <br></br>
            </div>

           <p class="subFont" style="display:inline;">Allow Questions to be done out
of order?  </p>
            <input type="checkbox" name="ooo"
                   <?php if ($jumpOrder == 'true'){?>checked="checked"<?php }?>/>
            <br></br>

        </div>

        <!-- TIMING AND SCORES -->
        <div id="timingScores" style="visibility:hidden;" class="innerForm
innerFormFill">
            <p class="subFont">Target Time: </p>
            <input id="tarTime"
                   name="tarTime"
                   class="textbox"
                   value="<?php echo $quizInfo['tarTime'];?>"/>
            <br></br>


            <p class="subFont">Time Limit? </p>
            Yes <input type="radio"  name="time" value="yes" id="limitYes"/>
            No <input  type="radio"  name="time" value="no" id="limitNo"
checked="checked"/><br></br>
```

```
            <div id="timeLimit" style="display:none;">
                <p class="subFont">Time Limit </p>
                <input name="timeLimit"
                       id="timeLimit"
                       class="textbox"
                       value="<?php echo $quizInfo['timeLimit'];?>"/>
                <br></br>
            </div>


            <p class="subFont">Pass Mark: </p>
            <input name="passMark"
                   class="textbox"
                   value="<?php echo $quizInfo['passMark'];?>"/>
            <br></br>
        </div>


        <!-- QUIZ PRIVACY TAB -->
        <div id="quizPrivacy" style="visibility:hidden" class="innerForm
innerFormFill">
            <p class="subFont">Make this quiz public or private? </p>
            Public <input  type="radio" name="priv" value="public" checked="checked"
id="public"/>
            Private <input  type="radio" name="priv" value="private"
id="private"/><br></br>
            <p class="subFont" style="display:none;" id="pub">
                Public quizzes will be made visible to everyone.
            </p>
            <p class="subFont" style="display:none;" id="priv">
                Private quizzes will only be visible to you and those in your lists.
            </p>
        </div>   <input type="hidden" name="qn" value="null"/>
                 <input type="submit" value="Save Changes" id="saveChanges"
class="button" style="position: absolute; bottom:20px;"/>

                 </div>
                 </form>
      </div>
  </div>



   <div class="side" style="width: 200px; position:fixed; top: 20px; left: 12%;">
<br></br>

            <?php
               for ($i=0; $i<$noq-1; $i++){ ?>

            <button
                name="jump"
                id="jump"
                method="post"
                value="<?php echo $i;?>"
                class="questionList"
                title="<?php echo $question[$i]['Question']; ?>">
            <p class="questionListP">
```

```
                <?php echo "Question ".($i+1); ?>
            </p>
        </button>
        <br></br>
        <input type="hidden" name="qn" value="<?php echo $i ?>">
           <?php } ?>

        <button type="submit[]"
                name="nothing"
                id="jump"
                action="createQuestion.php"
                method="post"
                value="jump"
                class="questionList">
            <p class="questionListP"
                title="<?php echo $questions[$i]; ?>">
                <?php echo "Question ".($noq); ?>
            </p>
        </button>  <br></br>

    </div>



</div>

    <script>
    $('input#showCre').click(function() {
      if($(this).is(':checked')){
        $('#showCreator').show('slow');}
      else { $('#showCreator').hide('slow'); }
    });

    $("input[name='time']").click(function() {
      if($('#limitYes').is(':checked')){
        $('#timeLimit').show('slow'); }
      else if ($('#limitNo').is(':checked')){
        $('#timeLimit').hide('slow');}
    });

    $("input#public").hover(function() {
      $("p#pub").show('fast'); },
    function() {
      $("p#pub").hide('fast'); });

    $("input#private").hover(function() {
      $("p#priv").show('fast'); },
    function() {
      $("p#priv").hide('fast'); });

    $("#searchLevel").change(function() {
     if ($(this).val() == 'other'){
       $("#levelOther").show('slow'); }
     else{ $("#levelOther").hide('slow'); }
    });
    </script>
```

```
</body>
```

createQuestion.php

```
<!DOCTYPE html >
<?php session_start(); ?>

<html>
    <head>
        <script type="text/javascript" src="../lib/jQuery.js"></script>
         <script type="text/javascript" src="../lib/apprise.js"></script>
        <link rel="stylesheet" type="text/css" href="../styles/stylesheet.css" />

         <script type="text/javascript">
function validateForm(){

validRad = true;
 if (  ( document.questions.correctAns[0].checked == false )
    && ( document.questions.correctAns[1].checked == false )
    && ( document.questions.correctAns[2].checked == false )
    && ( document.questions.correctAns[3].checked == false )
    && ( document.questions.correctAns[4].checked == false )
    && ( document.questions.correctAns[5].checked == false ) )
    { apprise ( "Please Select a Radio Button for the Correct Answer" );
        validRad = false;
    }
    return validRad;
    }

function addElement(ID) {
    document.getElementById('ansBlock'+ID).style.visibility = 'visible';
    }

function showTab(tab){

document.getElementById('ans').style.display = 'none';
document.getElementById('questionType').style.display = 'none';
document.getElementById('addNotes').style.display = 'none';
document.getElementById('addImg').style.display = 'none';
document.getElementById('ansExplain').style.display = 'none';
document.getElementById('prev').style.display = 'none';

document.getElementById(tab).style.display = 'block';
}

var visSideTab = false;
function slideTab(list) {
    if (visSideTab == false){
        document.getElementById(list).style.visibility = 'visible';
        document.getElementById(list).className = 'typeQuestion';
        document.getElementById('typeSide').style.height = '150px';

        visSideTab = true;
        }
```

```
    else if (visSideTab == true){
        hideTab(list);
        visSideTab = false;
    }
}

var visible = false;
var valid = false;

</script>

</head>

<?php define("LOCATION", "C:/Users/Lissy/wamp/www/quiz/data/images/"); //location
for images ?>

<?php //Get question number
$qn = ($_POST['qn']);
?>



<?php // what screen to display
if ($qn == -1)  { // is this the first question?
    require 'addQuiz.php';
    $qn = 0; // initialise question number
    }
else{ // if this is not the first question
    require 'addQuestion.php';
}?>


<?php //has this question already been filled out?
if (isset($_POST['next'])){
    $qn = ($_POST['qn'])+1;
    }
else if (isset($_POST['jump'])){
   $qn = ($_POST['jump'])-1; }

   $cqn = $qn + 1;

if (array_key_exists($cqn, $questions)) {
        $done = true; }
else {
        $done = false; }
        ?>

<body>

     <form method="post"
           action="createQuestion.php"
           name="questions"
           onsubmit="return validateForm()"
           enctype="multipart/form-data">


    <div class="back backFill">
```

```
    <div class="main">

        <div class="formQuestion">

            <div class="sideFormQuestion sideFormFill">

                    <div class="formTabs formTabsFill" onClick="showTab('ans')">
                        <p class="subFont">Question and Answers</p>
                    </div>

                     <div class="formTabs formTabsFill" id="typeSide">
                        <p class="subFont">Type of Question</p>
                        <div id="typeList" style="visibility: hidden;"> Here will be a
list of question types... </div>
                    </div>

                    <div class="formTabs formTabsFill" onClick="showTab('addNotes')">
                        <p class="subFont">Additional notes</p>
                    </div>


                    <div class="formTabs formTabsFill" onClick="showTab('addImg')">
                        <p class="subFont">Image or Diagram</p>
                    </div>

                    <div class="formTabs formTabsFill"
onClick="showTab('ansExplain')">
                        <p class="subFont">Answer Explanation</p>
                    </div>

                    <div class="formTabs formTabsFill" onClick="showTab('prev')">
                        <p class="subFont">Preview</p>
                    </div>
                    <div style="display:block;">
                        <p class="subFontTiny"
                           id="showVal"
                           title="Show instant validation"
                           style="display:none">
                            Show Instant Validation
                        </p>
                    </div>
                    <div id="validate" class="validationCheck">
                        <p class="subFontV" id="val">Before continuing please:</p>
                        <li id="queVB"><p id="queV" class="subFontV"></p></li>
                        <li id="ansVB"><p id="ansV" class="subFontV"></p></li>
                        <li id="canVB"><p id="canV" class="subFontV"></p></li>
                        <br></br>
                        <p class="subFontTiny" id="showSuggestions" title="Show more
sugestions on improving your question" style="display:none">Show More
Suggestions</p>
                        <div id="moreSuggestions" style="display:none;">
                            <p id="CV" class="subFontV" style="display:none;">
                                    You could consider improving your question by:
                            </p>
                            <li id="addeVB" style="display:none">
                                <p class="subFontV">
                                    Adding additional details
                                </p>
```

```
                        </li>
                        <li id="iodVB" style="display:none">
                            <p  class="subFontV">
                                Uploading an image or diagram
                            </p>
                        </li>
                        <li id="anseVB" style="display:none">
                            <p class="subFontV">
                                Adding an answer explanation
                            </p>
                        </li>
                    </div>
                    <p class="subFontTiny" id="hideVal" title="Hide instant
validation">
                        Hide
                    </p>
                </div>

              <input type="submit" name="next" value="Next &rarr;" class="button"
id="proceed"/>
            </div>


        <p class="titleFont">Create Quiz</p><br>
        <?php echo "Question Number: ".($qn+1) ?><br>

        <div id="ans" class="innerFormQu">   </br>
                <p class="subFont">Question</p> <input type="text"
                   name="question" class="textboxQuestions" id="question"
                       <?php if ($done==true) {?>
                   value="<?php echo $questions[$cqn]?>"
                       <?php }?>/><br></br>

              <?php include 'answerBox.php'; // display answer text boxes ?>


            </div>


            <div class="innerFormQu " id="questionType"
style="visibility:hidden;">
                 <p class="subFont"> Feature Coming Soon </p>
            </div>

            <div class="innerFormQu " id="addNotes" style="display: none;">
               <p class="subFont">
                Text written here will be made visible during the quiz at this
question.
                </p><br></br>
                 <textarea cols=65 rows=9 name="addNotes" id="additNotes"
                           value="<?php if ($done==true) {?>
                       <?php echo $_SESSION['options'][$cqn]['addNotes']?>
                       <?php } else { echo '';}?>"></textarea>
            </div>

            <div class="innerFormQu " id="addImg" style="display:none;">
               <p class="subFont"> Choose an image or diagram: </p>
               <input name="imgupload" type="file" id="imgordi"/>
```

```
            </div>

            <div class="innerFormQu " id="ansExplain" style="display:none;">
               <p class="subFont">
                You can add an explanation of the answer in the space below.
               </p><br></br>
               <p class="subFont">
               It will be made visible to the user once they have done the
question.
               </p><br></br>
                <textarea cols=65 rows=9 name="ansExp" id="ansExplanation"
                        value="<?php if ($done==true) {?>
                        <?php echo $_SESSION['options'][$cqn]['ansExp']?>
                        <?php } else { echo '';}?>"></textarea>
            </div>

            <div id="prev" class="innerFormQu" style="display:none;">
                <p class="subFont" id="prevQuestion"></p><br></br>
                <p class="subFont" id="prevAnsA"></p><br></br>
                <p class="subFont" id="prevAnsB"></p><br></br>
                <p class="subFont" id="prevAnsC"></p><br></br>
                <p class="subFont" id="prevAnsD"></p><br></br>
                <p class="subFont" id="prevAnsE"></p><br></br>
                <p class="subFont" id="prevAnsF"></p><br></br>

            </div>

            <input type="hidden" name="qn" value="<?php echo $qn ?>" >

        </div>
    </div>

    <div class="side" style="width: 200px; position:fixed; top:20px">
        <br></br>
        <?php
        $noq = count($questions);
        for ($i=1; $i<$noq+1; $i++){ ?>

        <button type="submit[]"
            name="jump"
            id="jump"
            action="createQuestion.php"
            method="post"
            value="<?php echo $i;?>"
            class="<?php if ($i-1 == $qn){?> questionListCurrent <?php }
                                    else{ ?> questionList <?php } ?>">
        <p class="questionListP"
            title="<?php echo $questions[$i]; ?>">
            <?php echo "Question ".($i); ?>
        </p>
    </button>

        <br></br>
       <?php } ?>

    <button type="submit[]"
            name="nothing"
            id="jump"
```

```
            action="createQuestion.php"
            method="post"
            value="jump"
            class="questionListCurrent">
        <p class="questionListP"
            title="<?php echo 'Question'.$noq+1; ?>">
            <?php echo "Question ".($noq+1); ?>
        </p>
    </button>  <br></br>
            <input type="submit"
                    value="Finish"
                    name="button"
                    id="finish"
                    class="button"
                    onClick="document.questions.action='createFinish.php'"/>

    </div>

</div>
      </form>
<script>
$("[name='rad']").click(function () {
    $("[name='rad']").attr("src", '../img/radOff.png');
    var presetValue = $(this).attr('id');
    $("div.selectedAns").removeClass('selectedAns');
    $("#ansBlock"+presetValue.toUpperCase()).toggleClass('selectedAns');
    $('input[name="correctAns"]').removeAttr('checked');

$("[name='correctAns']").filter("[value="+presetValue+"]").attr("checked","checked
");
    var src = $(this).attr("src").replace("Off", "On");
    $(this).attr("src", src);
 });

$("#typeSide").click(function () {
  if (visible == false){
   $(this).fadeTo(250, 0.95, function () {
   $("#typeList").show("slow");
   $("#typeList").toggleClass('typeQuestion');
   $(this).animate({height: 150},500);
    });
    visible = true;
  }
  else if (visible == true){
   $(this).fadeTo(250, 1, function () {
   $("#typeList").hide("slow");
   $("#typeList").removeClass('typeQuestion');
   $(this).animate({height: 30},500);
    });
    visible = false;  }});

</script>

<script type="text/javascript" src="questionValidation.js"></script>

    </body>
</html>
<script>
```

```
$('input#showCre').click(function() {
  if($(this).is(':checked')){
    $('#showCreator').show('slow');}
  else { $('#showCreator').hide('slow'); }
});

$("input[name='time']").click(function() {
  if($('#limitYes').is(':checked')){
    $('#timeLimit').show('slow'); }
  else if ($('#limitNo').is(':checked')){
    $('#timeLimit').hide('slow');}
});

$("input#public").hover(function() {
  $("p#pub").show('fast'); },
function() {
  $("p#pub").hide('fast'); });

$("input#private").hover(function() {
  $("p#priv").show('fast'); },
function() {
  $("p#priv").hide('fast'); });

$("#searchLevel").change(function() {
 if ($(this).val() == 'other'){
   $("#levelOther").show('slow'); }
 else{ $("#levelOther").hide('slow'); }
});

$("#jff").change(function(){
    $(this).is(':checked') ?
        $("#subject").prop('disabled', true)&&
        $("#searchLevel").prop('disabled', true)&&
        $("#txtSubject").css('color','grey')&&
        $("#txtLevel").css('color','grey'):
        $("#subject").prop('disabled', false)&&
        $("#searchLevel").prop('disabled', false)&&
        $("#txtSubject").css('color','black')&&
        $("#txtLevel").css('color','black');
})

$("#txtJFF").hover(function() {
  $("p#expJFF").show('fast'); },
function() {
  $("p#expJFF").hide('fast'); });

$("#jff").hover(function() {
  $("p#expJFF").show('fast'); },
function() {
  $("p#expJFF").hide('fast'); });


$(function() {
        $('#tarTime').timepicker({});
    });
var loggedIn = ("<?php echo $loggedIn; ?>");
if (loggedIn == ''){
    document.getElementById('private').disabled = true;
```

```
    document.getElementById('private').disabled = true;}
else if (loggedIn =='1') {
document.getElementById('private').disabled = false;
document.getElementById('private').disabled = false;}
</script>
```

questionValidate.js

```
/*
 * This file contains the validation script used to stop the user entering
 * incalid data into the database, and to display to the user what is still
 * needed or if there ar any errors in what they have inputed
 */


var answerValid = false;
var questionValid = false;
var corAnsValid = false;


$("input#question").keyup(function () {
  if ($(this).val() === ''){
    $("p#queV").text('Enter a question');
    questionValid = false;}
  else if ($(this).val().length < 5){
    $("p#queV").text('Question too short');
    questionValid = false;}
  else if ($(this).val().length > 250){
    $("p#queV").text('Question too long');
    questionValid = false;}
  else {$("p#queV").text('');
     questionValid = true;}
 if (questionValid == true){
     $("#queVB").hide('slow'); }
 else { $("#queVB").show('slow');}
}).keyup();


$("input[name='optionA']").keyup(function () {
  if ($(this).val() === ''){
    $("p#ansV").text('Enter some answers');
    answerValid = false;}
  else {
    $("input[name='optionB']").keyup(function () {
    if ($(this).val() === ''){
      $("p#ansV").text('Enter at least two answers');
      answerValid = false;}
    else if ($(this).val() != '') {$("p#ansV").text('')
      $("#ansVB").hide('slow');;
     answerValid = true;}
    }).keyup(); }
if (answerValid == false) { $("#ansVB").show('slow');}
}).keyup();


$(document).ready( function() {
  if( $('input[name=correctAns]:radio:checked').length > 0 ) {
    $("p#canV").text('');
    corAnsValid = true; }
```

```
  else {
    $("p#canV").text('Select a correct answer');
     corAnsValid = false;}
$('.ansBlock').click( function() {
  if( $('input[name=correctAns]:radio:checked').length > 0 ) {
    $("p#canV").text('');
    $("#canVB").hide('slow');
    corAnsValid = true;}
  else { $("p#canV").text('Select a correct answer');
    corAnsValid = false;}
    });
});


$('form').submit(function () {
  if( $("input#question").val().length > 4 &&
    $("input[name='optionB']").val() != '' &&
    $('input[name=correctAns]:radio:checked').length > 0){
    return true;
       }
  else {
  $("div#validate").css("background-color", "#ffd4d4");
  $("div#validate").css("border-color", "#ff0000");
return false;}
});


function isValiadate(){
if (questionValid == true && answerValid == true && corAnsValid ==true){
  $("div#validate").css("background-color", "#bcfca6");
  $("div#validate").css("border-color", "#6afa3a");
  $("p#val").text('Ready to continue!');
  $("#showSuggestions").show('fast');
  if(  $('#additNotes').val() =='' ||
       $('#ansExplanation').val() =='' ||
       $('#imgordi').val()=='') {
     $("#CV").show('fast');
       if ($('#additNotes').val() ==''){
         $('#addeVB').show('fast');}
       else {$('#addeVB').hide('fast');}
       if ($('#imgordi').val() ==''){
         $('#iodVB').show('fast');}
       else {$('#iodVB').hide('fast');}
       if ($('#ansExplanation').val() ==''){
         $('#anseVB').show('fast');}
       else {$('#anseVB').hide('fast');} }
   else { $("#CV").hide('fast');}

} };


function notValid(){
if (questionValid == false || answerValid == false || corAnsValid == false){
  $("div#validate").css("background-color", "#ffffa1");
  $("div#validate").css("border-color", "#fad73a");
  $("p#val").text('Before continuing:');
  $("#showSuggestions").hide('fast');}
};
```

```
$("div").on("click", isValiadate);
$("input").on("keydown", isValiadate);
$("textarea").on("keydown", isValiadate);
$("div").on("click", notValid);
$("input").on("keydown", notValid);


$("#showSuggestions").click(function(){
  $("div#moreSuggestions").toggle(400);
  $("#showSuggestions").text('Show / Hide Suggestions');
})


$("#hideVal").click(function(){
  $("div#validate").hide('slow');
  $("#showVal").show('slow');
})


$("#showVal").click(function(){
  $("div#validate").show('slow');
  $(this).hide('fast');
})


$("input#question").keyup(function () {
  var val = $(this).val();
      $("p#prevQuestion").text(val);
    }).keyup();


$("input#[name='optionA']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsA").text(val);
}).keyup();


$("input#[name='optionB']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsB").text(val);
}).keyup();


$("input#[name='optionC']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsC").text(val);
}).keyup();


$("input#[name='optionD']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsD").text(val);
}).keyup();


$("input#[name='optionE']").keyup(function () {
```

```
  var val = $(this).val();
  $("p#prevAnsE").text(val);
}).keyup();


$("input#[name='optionF']").keyup(function () {
  var val = $(this).val();
  $("p#prevAnsF").text(val);
}).keyup();
```

begin.php

```
<?php session_start(); ?>
<!DOCTYPE html>
<html>
    <head>
        <title>Begin</title>
        <script type="text/javascript" src="../lib/jquery.js"></script>
        <link rel="stylesheet" type="text/css" href="../styles/stylesheet.css" />
        <script type="text/javascript" src="../lib/apprise.js"></script>

        <?php require("../global.php"); ?>

    <script type="text/javascript" >
        function confirmDelete(){

        }
    </script>

    </head>


    <body>

<?php require 'calculateBegin.php'; ?>

    <div class="back backFill">

        <div style="float:right; margin-top: -15px; margin-right: 15px;">
            <a href="../start/start.php">
                <img src="../img/home.png"
                     title="Home"
                     style="margin:2px;
                     border: none;"
                     width="25"
                     height="25"/>
            </a>
        </div>


<div class="main">
<table  style="table-layout: fixed">  <col width=250/> <col width=250/>
<tr>
    <td align="center" colspan="2">
        <p class="subFont" style="font-size: 50px;">
            <?php echo $title ?> </p>
    </td>
</tr>
<?php if($level != null) {?>
    <tr>
        <td align="right">
            <p class="subFont">
                Level:
            </p>
        </td>
```

```
        <td>
            <p class="subFont">
                <?php echo $level ?>
            </p> <?php } ?>
        </td>
    </tr>
<?php if($subject != null) {?>
    <tr>
        <td align="right">
            <p class="subFont"> Subject: </p>
        </td>
        <td>
            <p class="subFont">
                <?php echo $subject ?>
            </p>
<?php } ?>
        </td>
    </tr>
<?php if($takes != null) {?>
    <tr>
        <td align="right">
            <p class="subFont">
                Takes:
            </p>
        </td>
        <td>
            <p class="subFont">
                <?php echo $takes ?>
            </p>
<?php } ?>
        </td>
    </tr>
<?php if($avScore != 0) {?>
    <tr>
        <td align="right">
            <p class="subFont">
                Average Score:
            </p>
        </td>
        <td>
            <p class="subFont">
                <?php echo $avScore."%" ?>
            </p>
<?php } ?>
        </td>
    </tr>
<?php if($passMark != 0) {?>
    <tr>
        <td align="right">
            <p class="subFont">
                Target Score:
            </p>
        </td>
        <td>
            <p class="subFont">
                <?php echo $passMark ?>
            </p>
<?php } ?>
```

```
            </td>
        </tr>
    <?php if($avTime != null) {?>
        <tr>
            <td align="right">
                <p class="subFont">
                    Average Time:
                </p>
            </td>
            <td>
                <p class="subFont">
                    <?php echo $avTime ?>
                </p>
    <?php } ?>
            </td>
        </tr>
    <?php if($tagTime != '00:00:00') {?>
        <tr>
            <td align="right">
                <p class="subFont">
                    Target Time:
                </p>
            </td>
            <td>
                <p class="subFont">
                    <?php echo $tagTime ?>
                </p>
     <?php } ?>
            </td>
        </tr>
    <?php if($creator != null) {?>
        <tr>
            <td align="right">
                <p class="subFont">
                    Created by:
                </p>
            </td>
            <td>
                <p class="subFont">
                    <?php echo $creator ?>
                </p>
    <?php } ?>
            </td>
        </tr>
    <?php if($tags != null) {?>
        <tr>
            <td align="right">
                <p class="subFont">
                    Tags:
                </p>
            </td>
            <td>
                <p class="subFont">
                    <?php echo $tags ?>
                </p>
    <?php } ?>
            </td>
        </tr>
```

```
</table>
<form action="quiz.php" method="get">
    <button  type="submit"
             class="button"
             name="start"
             id="start"
             value="<?php echo $quizID?>"
             style="width: 200px;
                    height: 50px;
                    margin-left: 130px;
                    cursor: pointer;">
        <p class="subFontLa"> Start</p>
    </button>
</form>

    <div class="quizOptions" style="display:none">

        <form action="../actions/printQuiz.php" method="post" id="printQuiz">
            <input type="hidden" name="qtd" value="<?php echo $quizID; ?>"/>
            <button  class="button" name="print"  id="delete" type="submit">
                <p class="subFontLa"> Print</p>
            </button>
        </form>

        <form action="downloadQuiz.php" method="post" id="downloadQuiz">
            <input type="hidden" name="qtd" value="<?php echo $quizID; ?>"/>
            <button  class="button" name="download"  id="download">
                <p class="subFontLa"> Download</p>
            </button>
        </form>

        <form action="createFinish.php" method="post">
            <input type="hidden" name="qn" value="edit"/>
            <button  type="submit"
                     class="button"
                     name="edit"
                     id="edit"
                     value="<?php echo $quizID?>">
                <p class="subFontLa"> Edit</p>
            </button>
        </form>

        <form action="actions/deleteQuiz.php" method="post" id="deleteQuiz">
            <input type="hidden" name="qtd" value="<?php echo $quizID; ?>"/>
            <button  class="divButton button" name="delete"  id="delete">
                <p class="subFontLa"> Delete</p>
            </button>
        </form>
    </div>
</div>




<div style="width: 200px; position:fixed; top:20px;">
        <br>
        <a href="start.php">
```

```
            <img src="../img/reviLogo.png"
                 style="margin-left:-18px;
                        margin-top: -27px;
                        margin-bottom: -10px;
                        border: none;"
                 width="195"
                 height="68"/>
        </a>
        <br></br>
        <input type="text"
               name="search"
               id="search"
               class="textbox"
               value="Search Quizzes"
               onfocus="disapearText()"/>
        <br></br>
            <?php //Show all questions
             for ($loop=0; $loop<$noq; $loop++){
                $dbquestionslist = mysql_query("SELECT ID, Question
                                                FROM questions
                                                WHERE quiz_ID = $quizID");
                $questionList = array();
                while ($p = mysql_fetch_array($dbquestionslist)){
                    $questionList[] = $p;} $questionList; ?>

        <button class="questionList" >
            <p class="questionListP"><?php echo "Question ".($loop+1); ?></p>
        </button>

            <?php } ?>

</div>
        <button  class="button"
                 style="width: 100px;
                        height: 30px;
                        position: relative;
                        bottom:100px;
                        float:left;"
                 onclick="window.location.href='../start/start.php'">
            <p class="subFontLa"> Back</p>
        </button>

        </div>
<script>
$("div#delete").click(function(){
  apprise('This will permanently delete this quiz. Continue?',
    {'verify':true}, function(r){
    if(r){
      document.forms["deleteQuiz"].submit();}
    else{ }
    });
  });

</script>


    </body>
</html>
```

calculateBegin.php

```
<?php

/*
 * This script fetches the necisary information about the quiz from the database
 *  it also makes calculations for scores and averages
 *   and it declares and initialises some session varibles and their indexes
 */


// Get quiz ID
if(isset($_GET['textbox'])){
    $quizID = ($_GET['quizID']);}
else{
    $quizID = ($_GET['button']);}


//Calculate the number of questions in selected quiz
$dbnoq = mysql_query("SELECT `ID`, COUNT(quiz_ID)
                      FROM questions
                      WHERE quiz_ID =  $quizID ");
$a = mysql_fetch_array($dbnoq);  $noq = $a[1];


//Get basic quiz information
    $dbquizzes = mysql_query("SELECT * FROM quizzes WHERE ID =  $quizID ");
    $quiz = array(); while ($b = mysql_fetch_array($dbquizzes)){ $quiz[] = $b;}
$title = $quiz[0]['Name'];
$level = $quiz[0]['Level'];
$subject = $quiz[0]['Subject'];
$takes = $quiz[0]['takes'];
    $dbquestions = mysql_query("SELECT ID FROM questions WHERE quiz_ID =  $quizID
");
    $question = array();
    while ($d = mysql_fetch_array($dbquestions)){
        $question[] = $d;}
    $firstQID = ($question[0][0]);
    $lastQID =  ($question[$noq-1][0]);
    $dbanswers = mysql_query("SELECT  `ID` ,  sum(`NumCorr`)AS total ,
`question_ID`
    FROM answers
    WHERE CorrectAnswer =  't'
    AND  `question_ID`
    BETWEEN '$firstQID'
    AND '$lastQID'")or die(mysql_error());
    $answer = array();
    while ($f = mysql_fetch_array($dbanswers)){
        $answer[] = $f;}
    $total = $answer[0]['total'];
    if ($takes == 0) {$avScore = 0;}
    else if ($total/$noq/$takes > 1) {$avScore = 100;} else {
$avScore = round($total/$noq/$takes*100); }
$passMark= $quiz[0]['passMark'];
$avTime= null;
```

```php
$tagTime= $quiz[0]['tarTime'];
$creator= $quiz[0]['creator'];
$tags= null;


// Declair the necissary varibles, arrays and sessions
$_SESSION['score'] = null;
for ($j=0; $j<($noq); $j++){
    $dbquestions = mysql_query("SELECT *
                                FROM questions
                                WHERE quiz_ID = $quizID
                                AND  QuestionNumber = $j+1");
    $question = array();
    while ($g = mysql_fetch_array($dbquestions)){
        $question = $g;
    }
    $questionID =  $question['ID'];
    $_SESSION['score'][$questionID] = "null";
    $_SESSION['userChoices'][$questionID] = "null";
    $_SESSION['startTime']=time();
    $_SESSION['finished'] = false;
}
?>
```

calculateQuiz.php

```
<?php

/*
 * This script contains the process gone through for the quiz screen
 */

//Get quiz ID
if (isset($_POST['quizID'])) {
$quizID = ($_POST['quizID']); }
else{
$quizID = ($_GET['start']);}


//Calculate the number of questions in selected quiz
$dbnoq = mysql_query("SELECT ID, COUNT(quiz_ID)
                      FROM questions
                      WHERE quiz_ID =  $quizID ");
$d = mysql_fetch_array($dbnoq);
$noq = $d[1];


$answers = 0; $questionID = 0;  //declairing varibles for the function


//Which question and answers to output
if (isset($_POST['userAnswer'])) {
    $selected_radio = $_POST['userAnswer'];
    $lqn =  $_POST['qn'];
    if ($lqn + 1 < $noq) {
        $qn = $lqn + 1;}
    else{ echo "----END OF QUIZ----";}}
else{ $qn = 0;
$lqn = $qn -1;}
if (isset($_POST['quizID'])) {
    if (($_POST['sentFrom'])=='jump'){
        $qn = ($_POST['nqn']);}}


//Gets the question ID
if (isset($_POST['quizID']) && ($_POST['sentFrom'])=='jump'){
$questionID = ($_POST['jump']);
$dbquestions = mysql_query("SELECT *
                            FROM questions
                            WHERE quiz_ID = $quizID
                            AND  QuestionNumber = $qn + 1");
$question = array();
while ($g = mysql_fetch_array($dbquestions)){
    $question[] = $g;}
}
else{
$dbquestions = mysql_query("SELECT *
                            FROM questions
                            WHERE quiz_ID = $quizID
```

```
                            AND  QuestionNumber = $qn + 1");
$question = array();
while ($g = mysql_fetch_array($dbquestions)){
    $question[] = $g;}
$questionID =  $question[0]['ID'];}


//Calculates number of answers for the question
$dbnoa = mysql_query("SELECT ID question_ID, COUNT(question_ID), question_ID
                      FROM answers
                      WHERE question_ID =  $questionID ");
    $noa = array();
    while ($h = mysql_fetch_array($dbnoa)){ $noa[] = $h;}
    $noa = ($noa[0][1]);


//Gets answers
$dbanswers = mysql_query("SELECT *
                          FROM answers
                          WHERE question_ID = $questionID");
$answers = array();
while ($r = mysql_fetch_array($dbanswers)){
    $answers[] = $r; }
$dbquizzes = mysql_query("SELECT *
                          FROM `quizzes`
                          WHERE ID = '$quizID'");
$quiz = array();
while ($a = mysql_fetch_array($dbquizzes)){
    $quiz[] = $a;}


//Calculates and keeps track of the score
$score = $_SESSION['score'];
if (isset($_POST['userAnswer'])) {
  $lqid = ($_POST['questionID']);
  $userAnswer = $_POST['userAnswer'];
  $lastCorrectAnswer = $_POST['correctAnswer'];
  if($userAnswer == $lastCorrectAnswer){
    $score[$lqid] = 1; }
  else{$score[$lqid] = 0; }
  $_SESSION['score']=$score;
  $_SESSION['userChoices'][$lqid]=$userAnswer;
}


// Adds to the average score
if (isset($_POST['userAnswer'])) {
$dbavscore= mysql_query("SELECT `ID`,`NumCorr`
                         FROM `answers`
                         WHERE `ID`=$userAnswer");
$avscore = array();
while ($m = mysql_fetch_array($dbavscore)){
    $avscore[] = $m;}
        $avscore=$avscore[0]['NumCorr'];
        $avscore = $avscore + 1;
mysql_query("UPDATE  `answers`
             SET  `NumCorr` =  '$avscore'
             WHERE  `answers`.`ID` = $userAnswer;");
```

```
}

//Put db values of questions and answers into variables
$title = $quiz[0]['Name']; // The title of the quiz
$ask = $question[0]['Question']; //The current question
$correctAnswer = ($answers[0]['CorrectAnswer']); //The correct answer of current
question


// calculate percentages for answers
    $dbansav = mysql_query("SELECT ID
                            FROM answers
                            WHERE question_ID =  $questionID ");
    $ansAv = array();
    while ($d = mysql_fetch_array($dbansav)){
        $ansAv[] = $d;}
    $firstAID = ($ansAv[0][0]);
    $lastAID =  ($ansAv[$noa-1][0]);
    $dbcount = mysql_query("SELECT  `ID` ,  sum(`NumCorr`)AS total, `question_ID`
    FROM answers
    WHERE `ID`
    BETWEEN $firstAID
    AND $lastAID")or die(mysql_error());
    $countArray = array();
    while ($f = mysql_fetch_array($dbcount)){
        $countArray[] = $f;}
    $total = $countArray[0]['total'];



//iz question done?
if (($_SESSION['score'][$questionID]) === 0 ||
    ($_SESSION['score'][$questionID]) === 1){
        $izdone = true; }
else {  $izdone = false;}


//Finds correct answers
$dbcorrectAns = mysql_query("SELECT *
                            FROM `answers`
                            WHERE CorrectAnswer = 't'
                            AND question_ID = '$questionID'");
$correctAnswer = array();
while ($l = mysql_fetch_array($dbcorrectAns)){
    $correctAnswer[] = $l;}
        $correctAnswer = $correctAnswer[0]['ID'];
        ?>
```

quiz.php

```
<?php session_start(); ?>
    <!DOCTYPE HTML >

<html>
```

```
<head>
<title> Quiz </title>
<script type="text/javascript" src="../lib/jQuery.js"></script>
<script type="text/javascript" src="../lib/apprise.js"></script>
<?php require("../global.php"); ?>
<link rel="stylesheet" type="text/css" href="../styles/stylesheet.css" />

<script type="text/javascript">

function mark(userAnswer, correctAnswer){
    if (userAnswer == correctAnswer){
        document.getElementById(userAnswer).setAttribute("style", "background-
color:green;") }
    else if (userAnswer != correctAnswer){
        document.getElementById(userAnswer).setAttribute("style","background-
color:red")
        document.getElementById(correctAnswer).setAttribute("style", "background-
color:green") }
    document.forms['questions'].submit();
            }

     function showCorrect(correctAns){
        document.getElementById(correctAns).setAttribute("style", " margin-left:
10px; background-color:green;");
         }

var statsVisible = false;


</script>

</head>

<body>

<?php require 'calculateQuiz.php'; ?>

    <div class="back backFill">

        <div style="float:right; margin-top: -15px; margin-right: 15px;">
            <a href="../start/start.php">
                <img src="../img/home.png"
                     title="Home"
                     style="margin:2px;
                     border: none;"
                     width="25"
                     height="25"/>
            </a>
        </div>



    <?php // Outputs the questions and options to the user ?>
        <div class="main">

        <form action="
                 <?php if ($lqn == $noq -2){ ?>
                    results.php
```

```
                    <?php } else { ?>
                        quiz.php <?php } ?>"
                    method="post"
                    onsubmit="return validateForm()"
                    name="questions">  <br></br>

            <p class="subFont"><?php echo $ask; ?></p> <br><br>

            <?php // loops through the answers and displays them
            for ($j=0; $j<$noa; $j++){
                $quid = $answers[$j]['ID'];
                $done = ($_SESSION['score'][$questionID])?>


<?php if ($question[0]['Notes']!= null ||
                $question[0]['Image'] != null ||
                $done === 1 ||
                $done === 0) {
    $shiftLeft =true; }
    else{ $shiftLeft = false; }?>

             <label>
                <div class="<?php if ($shiftLeft == true) {?>
                     questionBoxLM <?php } else {?>
                     questionBox <?php } ?>"

                            id="<?php echo $answers[$j]['ID']?>" >
                                    <input  type="radio"
                            name="userAnswer"
                            value="<?php echo $quid?>"
                            onClick="mark('<?php echo $quid?>', '<?php echo
$correctAnswer; ?>')"/>
                    <p class="subFont">
                            <?php echo ($answers[$j]['Answer']); ?>
                    </p>

                </div>
            </label>
            <?php if ($izdone==true) {
            $numSelected = $answers[$j]['NumCorr'];
            $averageSelect = round($numSelected /$total * 100);
            $length = $averageSelect * 4; ?>
            <div id="statsPercentage" class="statsPercentage">
            <p class="subFont"><?php echo $averageSelect.'%'; ?></p>
            </div>
            <div class="stats"
                 id="stats"
                 style="width:<?php echo $length.'px' ?>">
            </div>
                <?php } ?>


<?php } ?>
            <input type="hidden" name="sentFrom" value="strait">
        <input type="hidden" name="correctAnswer" value="<?php echo $correctAnswer;
?>">
        <input type="hidden" name="qn" value="<?php echo $qn ?>" >
        <input type="hidden" name="quizID" value="<?php echo $quizID ?>">
```

```
        <input type="hidden" name="questionID" value="<?php echo $questionID ?>" >
        <input type="button" value="Cancel" class="button"
onClick="window.location.href='../start/start.php'">

</form>


<?php // Calculate average quiz information
$dbtakes = mysql_query("SELECT ID, Takes
                        FROM quizzes
                        WHERE ID = $quizID");
$takes = array();
while ($j = mysql_fetch_array($dbtakes)){
    $takes[] = $j;}
$takes = $takes[0]['Takes'];

$scoreKeys = (array_keys($score));
$userA = 0;
$questionID=$scoreKeys[$qn];

 //Calculate average correct answers for each question
$dbavq= mysql_query("SELECT *
                     FROM `answers`
                     WHERE `Question_ID` = ($questionID)
                     AND `CorrectAnswer` = 't'");
$avqb = array();
while ($v = mysql_fetch_array($dbavq)){
    $avqb[] = $v;}
            $avq = $avqb[0]['NumCorr'];
        // Find out if user answer is correct
        if($score[$scoreKeys[$qn]]==1)
            {$userA='<p class=correct>Correct</p>';}
        else if($score[$scoreKeys[$qn]]==0)
            {$userA='<p class=incorrect>Incorrect</p>';}
        else {$userA = 'Result Unknown';}
        ?>


   <?php if (($_SESSION['score'][$questionID]) === 1 ||
             ($_SESSION['score'][$questionID]) === 0 ) {    ?>
        <div class="ext">
            <p class="subFontP"><?php echo $userA;?></p>
                <?php $comp = round(($avq / $takes) * 100); ?>

            <button class="button" id="showStats">Show Stats</button>
            <p class="subFontP">
                <?php if($question[0]['Explanation']!= null)  {
                 printf("Explanation of Answer: "."
\n".$question[0]['Explanation']);
                 } ?>
            </p>
            <br></br>
        </div>
     <?php } ?>


       <?php if ($question[0]['Notes']!= null ||
                 $question[0]['Image']!= null ) { ?>
```

```
        <div class="ext"
                <?php if (($_SESSION['score'][$questionID]) === 0 ||
                          ($_SESSION['score'][$questionID]) === 1){ ?>
                style="top: 330px" <?php } ?> >
           <p class="subFontP">
              <?php if($question[0]['Notes']!= null)  {
              printf("Notes: "." \n".$question[0]['Notes']); } ?>
          </p>
            <?php if($question[0]['Image']!= null)  { ?>
             <img
                src="../data/images/<?php echo $question[0]['Image']?>"
                style="margin:2px; border: none;"
                width="300" />

            <?php } ?>

            <br></br>
        </div>
     <?php } ?>
        </form>
</div>
<?php // If the question has been done, show correct answer
if (($_SESSION['score'][$questionID])===1 || ($_SESSION['score'][$questionID])
===0){
    ?> <script type="text/javascript">showCorrect(<?php echo $correctAnswer
?>)</script>
    <?php } ?>



<div class="side" style="width: 200px; position:fixed; top:20px;">
        <br></br>
        <a href="../start/start.php">
            <img src="../img/reviLogo.png"
                style="margin-left:-18px;
                       margin-top: -27px;
                       margin-bottom: -10px;
                       border: none;"
                width="195"
                height="68"/>
        </a>
        <br></br>
             <?php
             for ($loop=0; $loop<$noq; $loop++){
                $dbquestionslist = mysql_query("SELECT ID, Question
                                                FROM questions
                                                WHERE quiz_ID = $quizID");
                $questionList = array();
                while ($p = mysql_fetch_array($dbquestionslist)){
                    $questionList[] = $p;} $questionList;
                $score = ($_SESSION['score']);
                ?>

    <form name="jump" method="post">
        <button
                name="jump"
                id="jump"
                method="post"
```

```
                action="quiz.php"
                <?php if (($_SESSION['finished'])==false){ ?>
                disabled="disabled" <?php } else {?>
                title="<?php echo $questionList[$loop]['Question']; }?>"
                value="<?php echo $questionList[$loop]['ID'];?>"
                class="<?php if($questionID == $questionList[$loop]['ID']){ ?>
                questionListCurrent <?php } else { ?>
                questionList <?php } ?> ">
            <p class="questionListP">
                <?php echo "Question ".($loop+1); ?>
            </p>

                 <?php //makr question display in side
                 $score = ($_SESSION['score']);

                 if ($score[$questionList[$loop]['ID']]==='null'){ ?>
                     <p style="display:inline;"> ...  </p> <?php }
                 else if ($score[$questionList[$loop]['ID']]===1){ ?>
                     <p style="display:inline;"> &#10004;  </p> <?php }
                else if ($score[$questionList[$loop]['ID']]===0){ ?>
                     <p style="display:inline;"> &#10007;  </p> <?php }  ?>

        </button>

            <input type="hidden" name="sentFrom" value="jump"/>
            <input type="hidden" name="quizID"  value="<?php echo $quizID; ?>" />
            <input type="hidden" name="nqn"  value="<?php echo ($loop); ?>" />
    </form>
    <?php  } ?>
</div>

</div>

</body>
<script>
  $("button#showStats").click(function () {
    if (statsVisible == false){
      $("div#stats").slideToggle("slow");
      $("div#stats").addClass('stats')
      $("div#statsPercentage").show("slow");
      $(this).html('Hide Stats');
      statsVisible = true; }
    else if (statsVisible == true){
      $("div#stats").hide("slow");
      $("div#stats").removeClass('stats');
      $("div#statsPercentage").hide("slow");
      $(this).html('Show Stats');
      statsVisible = false; }
})
</script>


</html>
```

calculateResults.php

```php
<?php

/*
 * This script calculates and displays the users score
 * it also adds to the average scores in the database
 */


//Get the quiz ID and qn
$quizID = ($_POST['quizID']);
$qn = ($_POST['qn'] + 1);
$userAnswer = $_POST['userAnswer'];


//Increments the number of times the quiz has been taken
mysql_query("UPDATE `quizzes` SET `Takes` = (Takes+1)
WHERE `ID` = '$quizID'");


// Adds to the average score
$dbavscore= mysql_query("SELECT `ID`,`NumCorr`
                        FROM `answers`
                        WHERE `ID`=$userAnswer");
$avscore = array();
while ($m = mysql_fetch_array($dbavscore)){
    $avscore[] = $m;}
        $avscore=$avscore[0]['NumCorr'];
        $avscore = $avscore + 1;
mysql_query("UPDATE `answers`
             SET  `NumCorr` =  '$avscore'
             WHERE  `answers`.`ID` = $userAnswer;");


// marks the quiz as finished
$_SESSION['finished']=true;


//Fetches the right questions and answers from db
$dbquestions = mysql_query("SELECT *
                            FROM questions
                            WHERE quiz_ID = $quizID
                            AND  QuestionNumber = $qn");
$question = array();
while ($g = mysql_fetch_array($dbquestions)){
    $question[] = $g;}
$qID =  $question[0]['ID'];
$questionID = $qID + 1; //Get the unique ID of current question


//Calculates and keeps track of the score
$score = $_SESSION['score'];
$lqid = ($_POST['questionID']);
$userAnswer = $_POST['userAnswer'];
$lastCorrectAnswer = $_POST['correctAnswer'];
if($userAnswer == $lastCorrectAnswer){
```

```
$score[$lqid] = 1;}
else{ $score[$lqid] = 0;}
$_SESSION['score']=$score;
$_SESSION['userChoices'][$lqid]=$userAnswer;


//Calculate the number of questions in selected quiz
$dbnoq = mysql_query("SELECT ID, COUNT(quiz_ID), quiz_ID
                      FROM questions
                      WHERE quiz_ID =  $quizID ");
$d = mysql_fetch_array($dbnoq);
$noq = $d[1];


// Calculates the time
$startTime=$_SESSION['startTime'];
$endTime=time();
$timeDiff = $endTime - $startTime;
$totalTime = scoreTime($timeDiff);


// Set cookie
setcookie("takes[$quizID]", "$timeDiff");


// is user logged in
if (isset($_SESSION['logIn'])){
    $loggedIn = true;
    $username = $_SESSION['logIn']['username'];
    }
else { $loggedIn = false;}


// Has user already done quiz
$alreadyDone = false;

if ($loggedIn == true){
$dbscore = mysql_query("SELECT COUNT(`user`)
                    AS num FROM `scores`
                    WHERE `user` = '$username'");
$numConflictingScores = mysql_fetch_assoc($dbscore);
$numConflictingScores = $numConflictingScores['num'];
if ($numConflictingScores > 0){
    $alreadyDone = true;
    $lastTime = 0;
    }}

if((isset($_COOKIE['takes'][$quizID]))){
    $alreadyDone = true;
    $lastTime = ($_COOKIE['takes'][$quizID]);
}


// Create time format
function scoreTime($timeDiff){
    $time = floor($timeDiff / 60) . ":" . $timeDiff % 60;
    return $time;
}
```

```
// Create score %
function scorePer($score, $noq){
   $percentage = round(($score/$noq*100))."%";
   return $percentage;
}
?>
```

highScores.php

```
<?php
/*
 * This script calculates the users score, fetches the high scores,
 *  compares the users score and if applicable inserts the users score
 */


//Calculate Points
function calculateScore($time, $score){
$points = round((((1/$time)*1000)+($score*2))*10);
return $points;
}
$points = calculateScore($timeDiff, $scoreValue); // the users points

// Get username
if ($loggedIn == true){
    $username = $_SESSION['logIn']['username']; }
else {$username = 'Guest'; }


// Insert into database
if ($alreadyDone == false){
mysql_query("INSERT INTO `scores` (`quiz_ID`, `user`, `time`, `score`, `points`)
VALUES ('$quizID', '$username', '$timeDiff', '$scoreValue', '$points')");
}
// Get score ID if user wants to view/ edit/ delete score
$scoreID = mysql_insert_id();


// Fetch High scores from database
$scores = array();

$dbscores = mysql_query("SELECT *
FROM  `scores`
WHERE `quiz_ID` = '$quizID'
ORDER BY  `scores`.`points` DESC
LIMIT 0 , 6");

while($s = mysql_fetch_array($dbscores)){
$scores[] = $s;}

$numScores = count($scores); // Number of scores if < 6


//Check to see if user has new high score
if ($points>$scores[$numScores-1]['points']){
    $newScore = true; }
else {$newScore = false; }
```

```
// If user is logged in and has high score add score to database
if ($newScore == true && $loggedIn == true){
    }

?>
```

results.php

```
<?php session_start(); ?>
<!DOCTYPE html >
<html>

<head>
<script type="text/javascript" src="../lib/jQuery.js"></script>
<link rel="stylesheet" type="text/css" href="../styles/stylesheet.css" />
<?php
require("../global.php");
require 'calculateResults.php';
?>

</head>

<body>

<div class="back backFill">

    <div style="float:right; margin-top: -15px; margin-right: 15px;">
        <a href="../start/start.php">
            <img src="../img/home.png"
                 title="Home"
                 style="margin:2px;
                 border: none;"
                 width="25"
                 height="25"/>
        </a>
    </div>

    <div class="main">
        <p class="subFontLa">Results</p><br>

<?php // Calculates and outputs final score
$scoreValue = array_sum($score);
?><p class="subFont" id="score1">
    <?php echo "Total Score: ".$scoreValue. "/".$noq;?>
</p><br></br>
  <p class="subFont" id="score2" style="display:none;">
    <?php echo "Percentage : ".($scoreValue/$noq*100)."%"."<br></br>";?>
</p>
<p class="subFont">
     <?php echo "In a time of ".$totalTime;
     if ($alreadyDone == true){
```

```
            if ($timeDiff < $lastTime){
                $speed = 'quicker';
            }
            else {
                $speed = 'slower';
            }
            echo ', that was '.
                   scoreTime($timeDiff - $lastTime).
                   ' '.$speed.' than last time';
        }
        ?>
</p>  <br></br>
 <p>
 You can view how you did in each question using the buttons on the side
 </p>
 <br></br>


<div id="scoreWrapper">

<?php require('highScores.php'); ?>
    <p class="subFontScores">High Scores</p>
    <div <?php if ($loggedIn == true || $alreadyDone == true){ ?>
                   style="display:none;"
            <?php } ?> class="addScore">
            <form action="../actions/addScore.php" method="post">
                <p class="subFont" style="color:white">Enter your Name</p>
                <input type="text" name ="scoreName" class="scoreName"/>
                <input type="hidden" name="scoreID" value="<?php echo $scoreID; ?>"/>
                <button type="submit">Submit Score</button>
            </form>
    </div>
<table border="1" class="scoreTable">
  <tr>
    <th>Place</th>
    <th>User</th>
    <th>Score</th>
    <th>Time</th>
  </tr>
  <?php for ($i=0; $i<$numScores; $i++){ ?>
  <tr
      <?php if($scores[$i]['ID'] == $scoreID){?>
      style="background-color:#acabff;"
          <?php }?>>
    <td><?php echo $i+1 ?></td>
    <td><?php echo $scores[$i]['user']?></td>
    <td><?php echo scorePer($scores[$i]['score'],$noq);?></td>
    <td><?php echo scoreTime($scores[$i]['time']);?></td>
  </tr>
  <?php } ?>
</table>

</div>


    <input type="button"
           value="Home"
           class="button"
```

```
            onClick="window.location.href='../start/start.php'">
    <br clear="all">
</div>


<div class="side" style="width: 200px; position:fixed; top:20px;">
        <br>
        <a href="../start/start.php">
            <img src="../img/reviLogo.png"
                 style="margin-left:-18px;
                        margin-top: -27px;
                        margin-bottom: -10px;
                        border: none;"
                 width="195"
                 height="68"/>
        </a>
        <br><br>
             <?php
             for ($loop=0; $loop<$noq; $loop++){
                $dbquestionslist = mysql_query("SELECT ID, Question
                                                FROM questions
                                                WHERE quiz_ID = $quizID");
                $questionList = array();
                while ($p = mysql_fetch_array($dbquestionslist)){
                    $questionList[] = $p;} $questionList;
                $score = ($_SESSION['score']);
                ?>

    <form name="jump" method="post" action="quiz.php">
        <button type="submit[]"
                name="jump"
                id="jump"
                action="quiz.php"
                method="post"
                value="<?php echo $questionList[$loop]['ID'];?>"
                class="questionList"
                style="cursor: pointer;">
            <p class="questionListP"
                title="<?php echo $questionList[$loop]['Question']; ?>">
                <?php echo "Question ".($loop+1); ?>
            </p>
         <?php //makr question display in side
         $score = ($_SESSION['score']);

         if ($score[$questionList[$loop]['ID']]==='null'){ ?>
             <p style="display:inline;"> ...  </p> <?php }
         else if ($score[$questionList[$loop]['ID']]===1){ ?>
             <p style="display:inline;"> &#10003  </p> <?php }
        else if ($score[$questionList[$loop]['ID']]===0){ ?>
             <p style="display:inline;"> &#10007  </p> <?php }  ?>

        </button>

            <input type="hidden" name="sentFrom" value="jump"/>
            <input type="hidden" name="quizID"  value="<?php echo $quizID; ?>" />
            <input type="hidden" name="nqn"  value="<?php echo ($loop); ?>" />
    </form>
    <?php  } ?>
```

```
</div>

    </div>
<script>
$("p#score1").click(function() {
  $("p#score2").show('fast');
  $("p#score1").click(function() {
    $("p#score2").hide('fast');
});});
</script>

</body>

</html>
```

getQuiz.php

```
<?php

/* This file fetches all the necissary information about the quizzes
 * from the database and populates an array with the results
 * also fetches information on subjects
 */

//Get information on the quizzes, put in array
    $dbquizzes = mysql_query("SELECT * FROM `quizzes` ORDER BY Takes DESC");
    $quiz = array();
    while ($a = mysql_fetch_array($dbquizzes)){
    $quiz[] = $a;}
    $numQuizzes=count($quiz);

// Get quiz titles and levels count
    $num = (mysql_num_rows($dbquizzes));
    for($q=0; $q<$num; $q++)
    $subject = $quiz[$q]['Subject'];

// Subjects
    $dbsubjects = mysql_query("SELECT DISTINCT Subject
                        FROM quizzes
                        ORDER BY name ASC ");
    while ($S = mysql_fetch_array($dbsubjects, MYSQL_ASSOC)){ $subs[] = $S; }
    $numSubs=count($subs);
?>
```

search.php

```php
<?php
require("../global.php");


// Get all quiz title for Did You Mean part
    $dbnames = mysql_query("SELECT DISTINCT Name FROM quizzes");
        while ($n = mysql_fetch_array($dbnames, MYSQL_ASSOC)){ $names[] = $n; }
        $numNames = count($names);

// Get users search term
        $find = ($_GET["search"]);
        if ($find == "Search Quizzes"){
            $find = null;}
        $find = strtoupper($find);
        $find = strip_tags($find);
        $find = trim ($find);

//Check filters
        $subject = ($_GET['subject']);
        if($subject == "Subject"){
            $subject = null; }

        $level = ($_GET['level']);
        if($level == "level"){
            $level = null; }
        if ($level =='other'){
            $level = ($_GET['levelOther']);
            if ($level == 'EnterLevel'){
                $level = null;} }

        $sortBy = ($_GET['sortBy']);
        if($sortBy == "null"){
            $sortBy = 'takes'; }

        if ($sortBy == 'takes'){
            $order = 'DESC';}
        else {
        $order = 'ASC';}

        if ($_GET['rpp']!=''){
            $rpp = ($_GET['rpp']);
        }

 // Get and count results
        $dbcount = mysql_query("SELECT *, Count(*) FROM quizzes
                WHERE Name LIKE'%$find%'
                AND Subject LIKE'%$subject%'
                AND Level LIKE '%$level%'");
        while ($c = mysql_fetch_array($dbcount)){
        $count[] = $c; $numRes = $count[0]['Count(*)'];}

//Prepaire to display
        $pages = intval($numRes/$rpp); // Number of results pages
        if ($numRes % $rpp) {
            $pages++;}
```

```
        $current = ($page/$rpp) + 1;
        if (($pages < 1) || ($pages == 0)) {
            $total = 1;}
       else {
            $total = $pages;}
        $first = $page + 1;
        if (!(($page + $rpp) / $rpp) >= $pages) && $pages != 1) {
            $last = $page + $rpp;}
        else{
            $last = $numRes;}

//Search in db
        $dbsearch = mysql_query("SELECT * FROM quizzes
                WHERE Name LIKE'%$find%'
                AND Subject LIKE'%$subject%'
                AND Level LIKE '%$level%'
                ORDER BY $sortBy $order LIMIT $page, $rpp");
        while ($b = mysql_fetch_array($dbsearch)){ $results[] = $b; }

//did you meen
if ($numRes == 0){

$words=array();
for ($i=0; $i<$numNames; $i++){
$words[$i]  = $names[$i]['Name'];}
$shortest = -1;

foreach ($words as $word) {
    $lev = levenshtein($find, $word);
    if ($lev == 0) {
        $closest = $word;
        $shortest = 0;
    break; }
    if ($lev <= $shortest || $shortest < 0) {
        $closest  = $word;
        $shortest = $lev;}
}

 $dbdidyoumean = mysql_query("SELECT * FROM quizzes
                            WHERE Name LIKE'%$closest%'");
        while ($b = mysql_fetch_array($dbdidyoumean)){
            $didyoumean[] = $b; }
    $countDidyoumean=count($didyoumean);

}

?>
```

redirectMessages.php

```php
<?php
/* This file will display messages to the user after they are redirected
 * from another location where an action has been taken eg created account
 * It should make it clearer to the user what the program is doing
 */

$message = null; //The message to be displayed
$cameFrom = getenv("HTTP_REFERER"); // what page the user was redirected from


// Create the message
if (strpos($cameFrom, 'actions/logIn.php') !== false) {
    $message = 'You have been succesfully logged in'; }
elseif (strpos($cameFrom, 'actions/contact.php') !== false) {
    $message = 'Your message was sent succesfully'; }
elseif (strpos($cameFrom, 'actions/leaveFeedback.php') !== false) {
    $message = 'Thank you for your feedback'; }
elseif (strpos($cameFrom, 'actions/deleteQuiz.php') !== false) {
    $message = 'Your quiz has been deleted'; }
elseif (strpos($cameFrom, 'actions/createAccount.php') !== false) {
    $message = 'Your account has been succesfully created'; }
elseif (strpos($cameFrom, 'actions/addScore.php') !== false) {
    $message = 'Your score has been added'; }
elseif (strpos($cameFrom, 'start.php') !== false) {
    if (isset($_GET['logOut'])==true){
        unset($_SESSION['logIn']);
            $message = 'You have been logged out';}
}

// Display the message
if ($message != null){
    echo "<script language=javascript>
            apprise('$message');
          </script>";
}
?>
```

instantSearch.js

```
$(document).ready(function(){
    $("#search").keyup(function(){
        var filter = $(this).val(), count = 0;

        $(".browse button").each(function(){
            if ($(this).text().search(new RegExp(filter, "i")) < 0) {
                $(this).fadeOut(); }
            else {
                $(this).show();
                count++; }
        });

        if (count > 1||count==0){ s='s'}
        else {s=''}

        if (filter != ''){
        $("#filter-count").text("Your search term    "+
            filter+
            " returned "+
            count+
            " result"+
            s);
        $("#subTitle").text("Quick Results");
        }

        else  $("#filter-count").text('')
        $("#subTitle").text("Top Quizzes");
    });
});


$.ajax({
  url: '../arr/subjects.php',
  async: false,
  dataType: 'json',
  success: function (response) {
     var subjects = (JSON.stringify(response));
  }
});


$( "#subject" ).autocomplete({
    source: response
    });
```

homeScript.js

```
// This script contains all the jquery used on the home screen

   var visFilters = false; //if search filters are visible
    var visDidyoumean = false;
    var visLogin = false;
    var visIDblock = false;

$("p#enterqid").click(function() {
  if (visIDblock == false){
    $("div#showIDblock").show('slow');
    visIDblock = true;}
 else if (visIDblock == true){
    $("div#showIDblock").hide('slow');
    visIDblock = false;}
 });

$("div#logButton").click(function() {
  if (visLogin == false){
    $("div#loginBlock").show('fast');
    $("div#main").css("height", "85%");
    visLogin = true;}

  else if (visLogin == true){
    $("div#loginBlock").hide('fast');
    $("div#main").css("height", "93%");
    visLogin = false;}
});

$("p#more").click(function() {
  if(visFilters == false){
    $("div#moreFilters").show('slow');
    $(this).text('Less Filters');
    visFilters = true
  }
  else if (visFilters == true){
    $("div#moreFilters").hide('slow');
    $(this).text('More Filters');
    visFilters = false;
  }
});

$("p#txtDidYouMean").click(function() {
  if(visDidyoumean == false){
    $("div#didYouMean").show('slow');
      visDidyoumean = true
  }
  else if (visDidyoumean == true){
    $("div#didYouMean").hide('slow');
    visDidyoumean = false;
  }
});

$("#signUp").click(function() {
```

```
  $("div#dim").fadeIn('slow');
  $("div#createAccount").show('slow');
});

$("#enterQuizIdLink").click(function() {
  $("div#dim").fadeIn('slow');
  $("div#enterQuizID").show('slow');
});

$("#contactText").click(function() {
  $("div#dim").fadeIn('slow');
  $("div#contact").show('slow');
});

$("#feedbackText").click(function() {
  $("div#dim").fadeIn('slow');
  $("div#leaveFeedback").show('slow');
});

function hideAll(object){
 $("div#dim").hide();
 $(object).hide('slow');
}

$("#searchLevel").change(function() {
 if ($(this).val() == 'other'){
   $("#levelOther").show('slow'); }
 else{ $("#levelOther").hide('slow'); }
});
```

start.php

```
<?php session_start(); // start session ?>
<!DOCTYPE html PUBLIC >
<html>
    <head>
        <title>Revision Quizzes</title>
        <script type="text/javascript" src="../lib/jQuery.js"></script>
        <script type="text/javascript" src="../lib/apprise.js"></script>
        <link rel="stylesheet" type="text/css" href="../styles/stylesheet.css" />
    <link
href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/themes/base/jquery-ui.css"
rel="stylesheet" type="text/css"/>
        <script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-
ui.min.js"></script>
    <script type="text/javascript" >
    function disapearText(box){
          document.getElementById(box).value=""; }
    </script>

    </head>

    <body>
```

```php
<?php

// Include global file
require("../global.php");

//define varibles
    $rpp = 16; // The results per page
    $page = 0; // Defult page value

// clear session
$_SESSION = array_intersect_key($_SESSION, array_flip(array('logIn')));

// include redirect messages file
require('redirectMessages.php');

// includes the file containing the SQL quesries
require('getQuiz.php');
?>


<!-- DIM THE BACKGROUND-->
<div class="dim" id="dim" style="display:none;"></div>

<!-- THE CREATE AN ACCOUNT FORM -->
<div class="createAccountWrapper" style="display:none" id="createAccount">
        <img src="../img/close.png"
         onClick="hideAll('#createAccount')"
         style="height: 40px;
                position: absolute;
                top:-12px;
                right:-10px;
                cursor: pointer;"/>
    <div class="title">
        <center><b><p class="subFontCA">Create Account</p></b></center>
    </div>
    <div style="padding:50px;">
        <form name="signUp" method="post" action="../actions/createAccount.php">
            <p class="subFontSU">Username:</p>
            <input type="text"
                   class="textbox"
                   name="SUserName"
                   style="position:absolute;
                          left: 250px;"/>
            <br></br>
            <p class="subFontSU">Choose a Password:</p>
            <input type="password"
                   class="textbox"
                   name="SPassword"
                   style="position:absolute;
                          left: 250px;"/>
            <br></br>
            <p class="subFontSU">Confirm Password:</p>
            <input type="password"
                   class="textbox"
                   name="SConPassword"
                   style="position:absolute;
                          left: 250px;"/>
            <br></br>
```

```
            <p class="subFontSU">Student or Teacher:</p>
            <select name="SstudentTeacher"
                    class="dropDown"
                    style="position:absolute;
                           left: 250px;">
                <option value="student">Student</option>
                <option value="teacher">Teacher</option>
                <option value="other" >Other</option>
            </select>
            <br></br>
            <button type="submit" class="button" style="position: relative; left:
220px;">Sign Up</button>
        </form>
    </div>
</div>

<!-- LEAVE FEEDBACK FORM -->
<div class="createAccountWrapper" style="display:none" id="leaveFeedback">
        <img src="../img/close.png"
         onClick="hideAll('#leaveFeedback')"
         style="height: 40px;
                position: absolute;
                top:-12px;
                right:-10px;
                cursor: pointer;"/>
    <div class="title">
        <center><b><p class="subFontCA">Leave Feedback</p></b></center>
    </div>
    <div style="padding:50px;">
        <form method="post" action="../actions/leaveFeedback.php">
            <p class="subFontSU">Ease of use</p>
            <p class="subFontSU">Speed of site</p>
            <p class="subFontSU">Layout</p>
        </form>
    </div>
</div>

<!-- CONTACT FORM -->
<div class="createAccountWrapper" style="display:none" id="contact">
    <img src="../img/close.png"
         onClick="hideAll('#contact')"
         style="height: 40px;
                position: absolute;
                top:-12px;
                right:-10px;
                cursor: pointer;"/>
    <div class="title">
        <center><b><p class="subFontCA">Contact</p></b></center>
    </div>
    <div style="padding-top:25px; padding-left: 60px; padding-bottom: 25px;">
        <form method="post" action="../actions/contact.php">
            <p class="subFontSU">Name:</p>
            <input type="text"
                   class="textbox"
                   name="contactName"
                   style="position:absolute;
                          left: 150px;"/>
             <br></br>
```

```
            <p class="subFontSU">Email:</p>
            <input type="text"
                   class="textbox"
                   name="contactEmail"
                   style="position:absolute;
                          left: 150px;"/>
            <br></br>
            <p class="subFontSU">Subject:</p>
            <input type="text"
                   class="textbox"
                   name="contactSubject"
                   style="position:absolute;
                          left: 150px;"/>
            <br></br>
            <p class="subFontSU">Message:</p>
            <textarea  name="contactMessage"
                       rows="5" cols="20"
                   style="position:absolute;
                          left: 150px;"></textarea>
                          <br></br>
          <button type="submit"
                   class="button"
                   style="position: relative;
                          top:30px;
                          left: 210px;">
              Send
          </button>
          <br></br>
        </form>
    </div>
</div>


<!-- ENTER QUIZ ID FORM -->
<div class="createAccountWrapper" style="display:none" id="enterQuizID">
    <img src="../img/close.png"
         onClick="hideAll('#enterQuizID')"
         style="height: 40px;
                position: absolute;
                top:-12px;
                right:-10px;
                cursor: pointer;"/>
    <div class="title">
        <center><b><p class="subFontCA">Enter a quiz ID</p></b></center>
    </div>
    <div style="padding:50px;">
        <form action="../quiz/begin.php" method="get">
            <p class="subFont">If you know the quizzes ID, you can enter it below
to go strait to it</p>
            <br></br>
            <input class="textbox"
                name="quizID"
                id="quizID"
                style="width: 80px;
                       height: 30px;
                       font-size: 22px;
                       font-weight: bold;
                       margin-top: 15px;
```

```
                        margin-bottom: -5px;
                        margin-left: 40px;
                        padding-left:3px;"/>
            <input type="submit"
                value="Start"
                name="textbox"
                class="button"
                action="../quiz/begin.php"/>
      </form>
    </div>
</div>

<div class="back backFill">

<!-- SIDE BAR WITH SEARCH FILTERS -->
<div class="side" id="side">
    <form method="get" action="start.php">
        <br/>
        <img src="../img/reviLogo.png"
             style="margin-left:-18px;
                    margin-top: -27px;
                    margin-bottom: -10px;"
             width="195"
             height="68"/>
        <br/><br/>
        <div class="filtersExpand" id="filters">
            <p class="subFontMW">Filter Quizzes</p>
            <br></br>
             <input type="text"
                    name="search"
                    id="search"
                    class="textbox search"
                    value="Search Quizzes"
                    onfocus="disapearText('search')"/>
             <div>
            <input type="text"
                   name="subject"
                   id="subject"
                   class="textbox level"
                   value="Subject"
                   onFocus="disapearText('subject')"/>
             </div>
            <select name="level" class="dropDown" id="searchLevel">
                <option value="level" title="Select a Level">Level</option>
                <option value="KS1" title="Ages 4-6">KS1</option>
                <option value="KS2" title="Ages 7-10">KS2</option>
                <option value="KS3" title="Ages 11-13">KS3</option>
                <option value="GCSE" title="Ages 14-15">GCSE</option>
                <option value="A Level" title="Ages 16-17">A Level</option>
                <option value="higher">Higher</option>
                <option value="other"
                        title="An input box will appear for you
                        to type a non-specified level">Other...
                </option>
            </select>
            <input type="text"
               name="levelOther"
               id="levelOther"
```

```
                class="textbox level"
                value="EnterLevel"
                onFocus="disapearText('EnterLevel')"
                style="display:none; margin-top: 8px;"/>
                <br></br>
               <p class="subFontWh" id="more" style="cursor:pointer">
                   More Filters
               </p>
               <div id="moreFilters" style="display:none; padding-top: 10px;">
                 <select name="sortBy" class="dropDown">
                     <option value="null">Sort By</option>
                     <option value="takes">Popularity</option>
                     <option value="name">Name A-Z</option>
                     <option value="ID">Date Added</option>
                 </select>
                 <br></br>
                   <p class="subFontWh">Results per page</p>
                     <input type="text"
                            name="rpp"
                            value="16"
                            class="textbox"
                            style="width:40px;">
                     </input>
                </div>
                <input type="submit" value="Search" class="button"/>
                <p class="subFontWh" id="enterQuizIdLink" style="cursor:pointer">
               Enter a quiz ID
               </p>
          </div>

        <br></br>

        <input type="button"
               value="Create a Quiz"
               onclick="window.location.href='../create/createQuiz.php'"
               class="button"/>
    </form>
</div>

<!-- TOP BAR -->

<?php include '../forms/top.php'; ?>

<!-- LOG IN -->
<div class="loginBlock" id="loginBlock">
    <form name="login" method="post" action="../actions/logIn.php">
        <p class="subFont">User Name</p>
        <input type="text" class="textbox logInTB" id="username" name="username"/>
        <p class="subFont" style="margin-left: 10px;">Password</p>
        <input type="password" class="textbox logInTB" id="password"
name="password"/>
        <button type="submit" class="buttonLogin">Log In</button>
        <p class="subFont" style="margin-left: 2px;">Or</p>
        <button type="button" class="buttonLogin" id="signUp">Sign Up</button>
    </form>
</div>
```

```
<!-- MAIN SCREEN -->
<div class="main" id="main">
    <p class="titleFont">Start</p><br>


<?php //if the user is searching
if (isset($_GET['search'])) {
require("search.php");
?>


    <div class="tab tabFill">
        <p class="subFont" style="color: black;">Search Results</p>
    </div>

   <div class="browse"><br><?php


//Echo what the user searched for
    ?><p class ="subFont"><?php
    echo "Your search term ".
         "<i>".strtolower($find)."</i>".
         " returned ".$numRes.
            " ".$level.
              " ".$subject.
            " result";
    if($numRes!=1)
        {echo "s";}?> </p> <br>
            <p class="subFont">
     <?php if ($numRes == 0){
         echo"<p id='txtDidYouMean' style='cursor:pointer'>".
                "Did you mean ".
                "<U><i>".$closest."</i></U>".
                "</p>";
     } ?></p>
<form method="get" action="../quiz/begin.php">
    <div id="didYouMean" style="display:none;">
       <?php for($i=0; $i<$countDidyoumean; $i++){ ?>
         <button type="submit"
            class="quizBlock"
            name="button"
            id="quizzes"
            action="../quiz/begin.php"
            method="get"
            value="<?php echo $didyoumean[$i]['ID']?>"
            style="font-weight:bold;
                   text-align:left;
                   padding-left:4px;">

            <p class="subFontMe" id="title">
                <?php echo $didyoumean[$i]['Name']?>
            </p>
        <br>
            <p class="subFontWh" id="levDisp">
                <?php echo $didyoumean[$i]['Level']?>
            </p>
             <p class="subFontWh" id="creDisp" style="display:none">
```

```
                <?php echo $didyoumean[$i]['creator']?>
            </p>
            <p class="subFontWh" id="subDisp">
                <?php echo $didyoumean[$i]['Subject']?>
            </p>
            <p class="subFontWh" id="timeDisp" style="display:none;">
                <?php echo $didyoumean[$i]['tarTime']?>
            </p>
        </br>
    </button>
        <?php } ?>
    </div>



        <?php
   for($i=0; $i<$last; $i++){ ?>

    <button type="submit[]"
            class="quizBlock"
            name="button"
            id="quizzes"
            action="../quiz/begin.php"
            method="get"
            value="<?php echo $results[$i]['ID']?>"
            style="font-weight:bold;
                   text-align:left;
                   padding-left:4px;">

            <p class="subFontMe" id="title">
                <?php echo $results[$i]['Name']?>
            </p>
        <br>
            <p class="subFontWh" id="levDisp">
                <?php echo $results[$i]['Level']?>
            </p>
             <p class="subFontWh" id="creDisp" style="display:none">
                <?php echo $results[$i]['creator']?>
            </p>
            <p class="subFontWh" id="subDisp">
                <?php echo $results[$i]['Subject']?>
            </p>
            <p class="subFontWh" id="timeDisp" style="display:none;">
                <?php echo $results[$i]['tarTime']?>
            </p>
        </br>
    </button> <?php } ?> <br>

</form>


    </div>
 <input type="button"
        value="&larr; Back  "
        class="button"
        onClick="window.location.href='start.php'"/>
 <div>
    <p class="subFont" align="centre">
```

```
        Results <b><?php echo $first?></b> -
        <b><?php echo $last?></b> of <b><?php echo $numRes?></b>
    </p>
    <p class="subFont" align="centre">
        Page <b><?php echo $current?></b> of <b><?php echo $total?></b>
    </p>
</div>



<?php } else {  ?>


<form method="get" action="../quiz/begin.php" >
    <div class="tab tabFill">
        <p class="subFont" style="color:black;" id="subTitle">Top Quizzes</p>
    </div>
       <div class="browse">
           <p id="filter-count"></p>

           <?php //Display matching quizzes
                 for($i=0; $i<$rpp; $i++){ ?>
            <button type="submit[]"
                    class="quizBlock"
                    name="button"
                    id="quizzes"
                    action="../quiz/begin.php"
                    method="get"
                    value="<?php echo $quiz[$i]['ID']?>"
                    style="font-weight:bold;
                           text-align:left;
                           padding-left:4px;">

                    <p class="subFontMe" id="title">
                        <?php echo $quiz[$i]['Name']?>
                    </p>
                <br>
                    <p class="subFontWh" id="levDisp">
                        <?php echo $quiz[$i]['Level']?>
                    </p>
                    <p class="subFontWh" id="subDisp">
                        <?php echo $quiz[$i]['Subject']?>
                    </p>
                     <p class="subFontWh" id="creDisp" style="display:none">
                        <?php echo $quiz[$i]['creator']?>
                    </p>
                    <p class="subFontWh" id="timeDisp" style="display:none;">
                        <?php echo $quiz[$i]['tarTime']?>
                    </p>
                </br>
            </button><?php }?>

<?php //Display matching quizzes
                 for($i=$rpp; $i<$numQuizzes; $i++){ ?>
            <button type="submit[]"
                    class="quizBlock"
                    name="button"
                    id="quizzes"
```

```
                    action="../quiz/begin.php"
                    method="get"
                    style="display:none;
                           font-weight:bold;
                           text-align:left;
                           padding-left:4px;"
                    value="<?php echo $quiz[$i]['ID']?>">


                    <p class="subFontMe" id="title">
                        <?php echo $quiz[$i]['Name']?>
                    </p>
                <br>

                    <p class="subFontWh"><?php echo $quiz[$i]['Level']?></p>

                    <p class="subFontWh"><?php echo $quiz[$i]['Subject']?></p>

            <br>
            </button><?php }?>
            <br clear="all"><br>

 <?php } ?>
                    </div>
            </form>
                        <br>
                    </div>
        </div>

<!-- THE WEB SITE INFORMATION FOOTER-->
<div style="position: absolute; bottom:2px; left: 35%;" >
<p class="smallGreyText" id="contactText">Contact</p>
<p class="smallGreyText" id="feedbackText">Feedback</p>
<a href="http://documentation.revisionquizzes.com"><p
class="smallGreyText">Documentation</p></a>
<p class="smallGreyText">&#169; Alicia Sykes 2012 - All rights reserved</p>
</div>

<!-- EXTERNAL JQUERY-->
<script type="text/javascript" src="homeScript.js"></script>
<script type="text/javascript" src="instantSearch.js"></script>

    </body>

</html>
```

databaseConection.php

```php
<?php
//Database Connection
$servername = 'localhost';
$username = 'admin';
$password = 'root';
$database = 'quiz';

$con = mysql_connect($servername,$username,$password);
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db($database);

?>
```

Testing

As I have gone through the program and after each stage I have used white box tested the new features added. I have also white box tested my algorithms in the design section. I am going to beta test my program where it will be released in order for others to use and find any faults or bugs. The acceptance testing will show to the end user that the program meets the requirements.

Acceptance Testing

Design Specification	**Completed**	**Reasons Why**
To enter quiz information		
Quiz Name	✔	The user agrees that they can add a quiz name. Quiz names can be used more than once, although must have a different subject or level.
Level	✔	The user can select a level from a dropdown menu. They can also choose to type it if it is not in the database.
Subject	✔	The user can type a subject, a distinct SQL query adds all the subjects from the database into a live drop down suggestions.
Questions (as much or little as needed)	✔	The user can enter between 1 and 100 questions per quiz.
Answers (As many or as few)	✔	The user can enter between 2 and 6 answers per question.
The correct answer for each question (radio button)	✔	The user agrees that they can select a correct answer for each question, and it is showed clearly by highlighting the whole field.
A place to insert an image or diagram	✔	The user agrees that they can upload an image or diagram onto the server, which will be displayed during the running of the quiz.
An optional place to insert additional details or an answer explanation	✔	The user agrees that they can add both additional details or an answer explanation relating to the question, if they want.

To be able to do quizzes		
Loop through each question in order	✔	The end user agrees that the quiz program can loop through each question in order. There is also an option while making the quiz to do the questions out of order.
Show user if they got it wrong or right for each question	✔	The answer will be highlighted either red or green, and there answer explanation, if there is one, will be made visible.
Questions laid out clearly	✔	The questions are laid out on a clear screen with 1question on the screen at a time, and nothing unnecessary. The user agrees that it is clear.
Searching for a quiz		
Quick efficient way to search	✔	The end user agrees that the live search is very quick and efficient at searching. Also the 'did you mean' feature is very useful.
Filter be subject and level	✔	The user does not have to enter a search term; they can browse by just subject and level if they want.
More filters	✔	The user can also choose how to sort the results, and how many results per page to display.
Unique ID for each quiz	✔	If the user knows the unique ID of the quiz, they can enter it to go straight there. The end user agrees that this is quick.
Keep score		
Record the users score while they do the quiz.	✔	The users score will be kept as they go through the quiz. It will be stored in an array in a session, and displayed at the end to the user.
Show total score at the end of the quiz	✔	The user agrees that the score outputted at the end is both accurate and in a clear form.
Show a breakdown of the score at the end	✔	The user can go through the questions at the end and see which they got right, and which they got wrong.
Record scores for each quiz in the	✔	The scores for each quiz are recorded and updated

database		in the database.
Show averages for question	✔	The user can view averages in a bar chart when the question is complete. It has a slide animation too.
Store quizzes		
A storage system to store all quizzes	✔	The storage system is a database and it can hold a large number of quizzes, quite organisdley and efficiently.
Must be fast, efficient and effective	✔	The user agrees that the database is effective at it's use
Must be able to store quizzes, questions, answers and necessary additional information	✔	The end user agrees that all the necessary information is stored in the database.
Must store the information securely, and be regularly backed up	✔	The end user can back up the database very quickly from the admin control panel. Also all user data is encrypted with MD5.

Contents

Choosing a compatible web browser

RevisionQuizzes.com is an online program, and requires a web browser to run. Some browsers may not be compatible, which could result in the program running slower, not displaying properly, or some features not working as expected. To avoid this, a compatible browser in the list below should be used.

Google Chrome 4 and later

Firefox 3.0 and later

Internet Explorer 8 and later

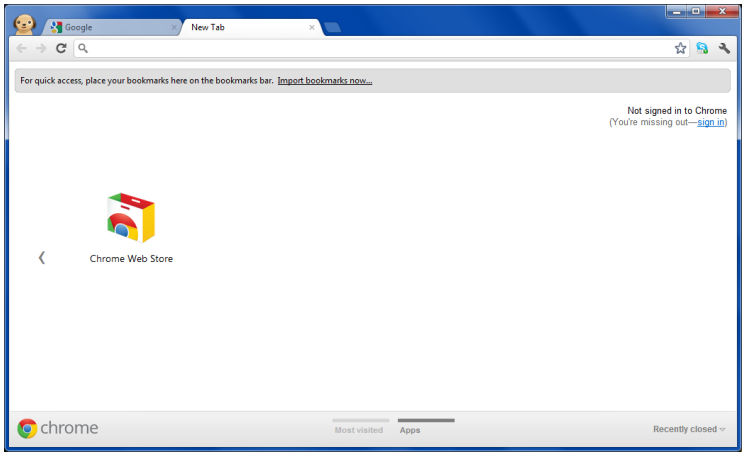
Opera 10.10 and later

Safari 4 and later

All the above web browsers are readily available online. See appendix 3 for the download links, and installation instructions.

Downloading and installing the Chrome App

There is a Chrome App for RevisionQuizzes.com. It is a free web application, that will allow all your setting, it will also be automatically updated, and be available 24/7.



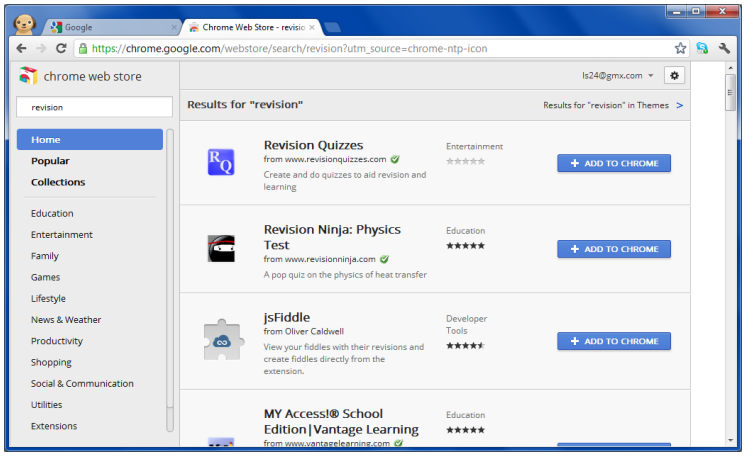
It can be downloaded and installed following the instructions below:

Open up Google Chrome

Open a new tab

Press right, to go to web apps

Click the Chrome Web Store



Once on the Web Store:

Search for 'Revision'

Revision Quizzes should be near the top

Click 'Add to Chrome'

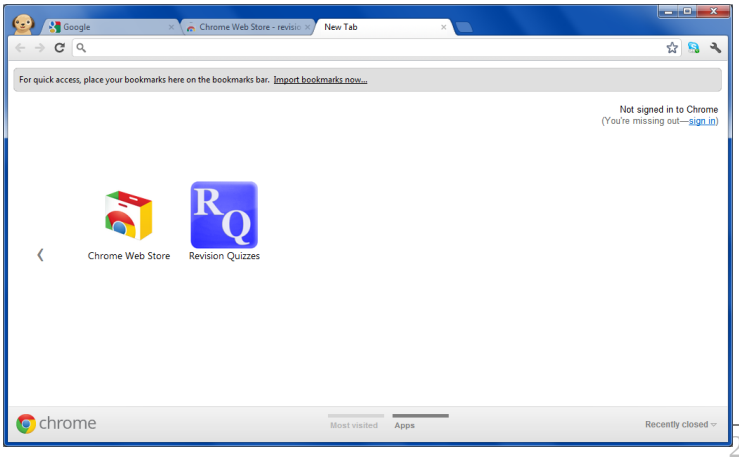
It will download and install automatically, and open a new tab. To use:

Click new tab, as before

There will now be a new icon present

Click Revision Quizzes

You will be sent to RevisionQuizzes.com



Opening your web browser

For Windows XP users:

Go to start, bottom left corner

All programs

Scroll down until you find your web browser

For Windows Vista and Windows 7 users:

Go to start, bottom left corner

If your web browser appears at the top of the menu, select it

if not, go to all programs and scroll down until you find it

For OS X users:

Select spotlight in the top right corner

Search for your default web browser

For Linux GNOME 3 users:

Go to activities in the top left hand corner

If your web browser appears on the dock click it

If not, go to applications at the top of the launcher, then internet on the right, then select your web browser.

For Linux GNOME 2 users:

Go to applications at the top left

Go to internet

Select your web browser

For Linux KDE users:

Click KDE in the bottom left corner

Select internet on the right

Then click the browser that appears

Or just search the name of your default web browser in the KDE bar

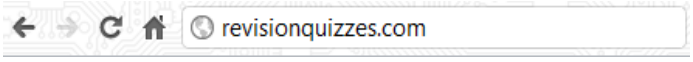
For Linux Ubuntu Unity users:

Click the dash home in the top left corner

Click quick internet, which will go straight to your default web browser

Or to choose another, under the internet menu

Navigating to the URL



Once in your web browser, you must go to the URL which Revision Quizzes is run under.

For nearly all web browsers this can be done by; selecting the address bar, at the top of the screen, deleting what is currently in it, typing one of the following: http://www.revisionquizzes.com/start.php, or just revisionquizzes.com or the short URL goo.gl/xHyeY

Searching for a quiz

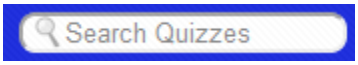
Hover your pointer over the input box in the upper left hand corner, which by default will have the text 'Search for a Quiz' in it

The cursor should appear automatically

Start typing a keyword relating to the quiz you want to find

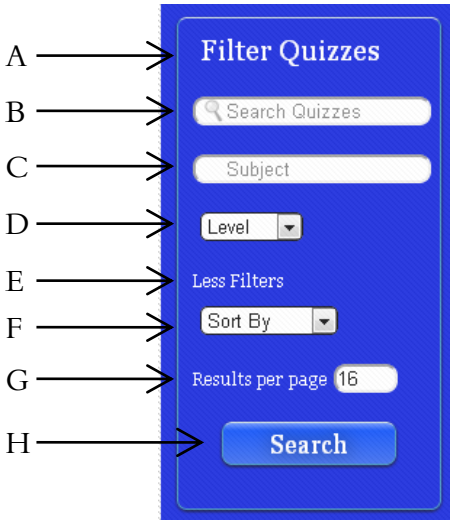
Your results should appear automatically in the browse pain where the top quizzes were previously displayed

To get more specific search results consider using the advanced search



Using advanced search

There are several advanced search features built into the filter quizzes pain [A]. The filters can be left blank, or have values in them, you can leave the fields you do not wish to search blank.



To filter by subject, hover over the field which says subject [C]

Once the cursor is in the input box type the subject you would like to filter by. Auto fill is enabled by default for the subjects textbox. It uses results from the database of subjects that already exist, and sorts them in order of relevance depending on what's in the quiz title text box.

The results will automatically appear on the right filtered to your subject, and any other values in any other fields

You can filter by level in a similar way, just select the level from the drop down menu [D], and the results will only show if the level matches.

If you select *Other* for level, an input box will appear, and you will need to type the level in.

If you do not have JavaScript enabled, or your connection is slow, you will need to press the search button which is below the level drop down menu

You can also choose how you would like to sort results

Click the *more filters* [E] text below the level box, and above the search button

The additional filtering options should slide down

Where the drop down says *Sort by* [F]select what you would like to sort your results by

Your results will refresh, now sorted by the filter you selected.

What each sorted by option does

Popularity – sorts results by the number of times users have taken it

Name – Sorts results by the name of quiz, alphabetically

Date added – sorts quizzes by the date they were added, most recent first.

You can also select how many results per page you would like to be displayed

Under the *more filters* option [G]

Hover over the text box to select it

Change the number in it to reflect how many results per page you would like to be displayed

By default the value for a standard sized screen is 16. This value may vary, as it is calculated from screen size to best optimize the start page.

Using the best guess function

If no results for your search term are found, the best guess function will suggest quizzes similar to your term.

Search for a term, which has no matching quizzes, in the category or level.

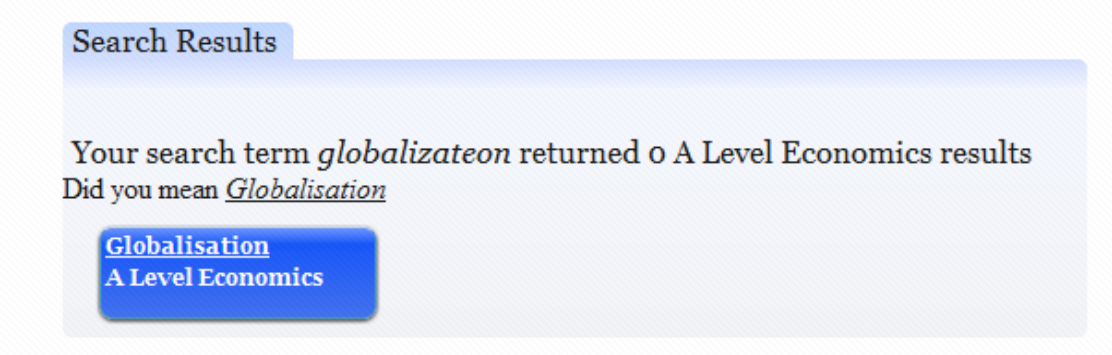
The screen below will be shown

Look to see if any of the underlined terms matches a quiz that you're looking for

If it does, click it, and the results will slide down

The results will still be filtered to your subject and level

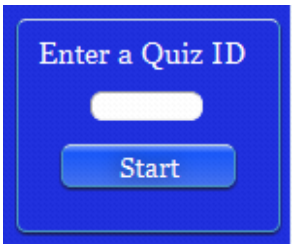
Unless there were no quizzes matching your search filters



Entering a quiz ID

If you know the unique ID of a quiz, you can jump strait to it by entering it's ID.

Click the *enter a quiz ID* button that is on the left hand side below the search filters.



A pop out block should appear

Enter the ID in the text box

Press start

You will be directed to the begin screen of the quiz

To find out a quiz ID of any quiz from the search results:

Click the quiz

You will be directed to the begin screen

You will see that where the information is displayed, one of the rows says quiz ID

This will be in the format of a number under 5 characters long

To find the quiz ID of a quiz you have just created:

On the finish screen, which you will be directed to after you finish creating a quiz

Click the summary tab on the right

There will be a sentence saying what the unique ID of your quiz is

Starting a quiz

After selecting a quiz from the start page, search results or entering the quiz ID you can start the quiz.

The begin screen will be displayed first

It will display relevant information relating to the quiz:

Title – this is the title of the quiz [A]

Level – this is the level that the quiz was created for [B]

Subject – this is the category the quiz falls under [C]

Takes – this is the number of people who have already taken the quiz [D]

Average score – this is the average score that people have achieved on that quiz

Target score – this is the score set as a target by the creator of the quiz [E]

Average time – this is the average time people spend on that quiz

Target time – This is the target time, set by the creator to do the quiz in under that time

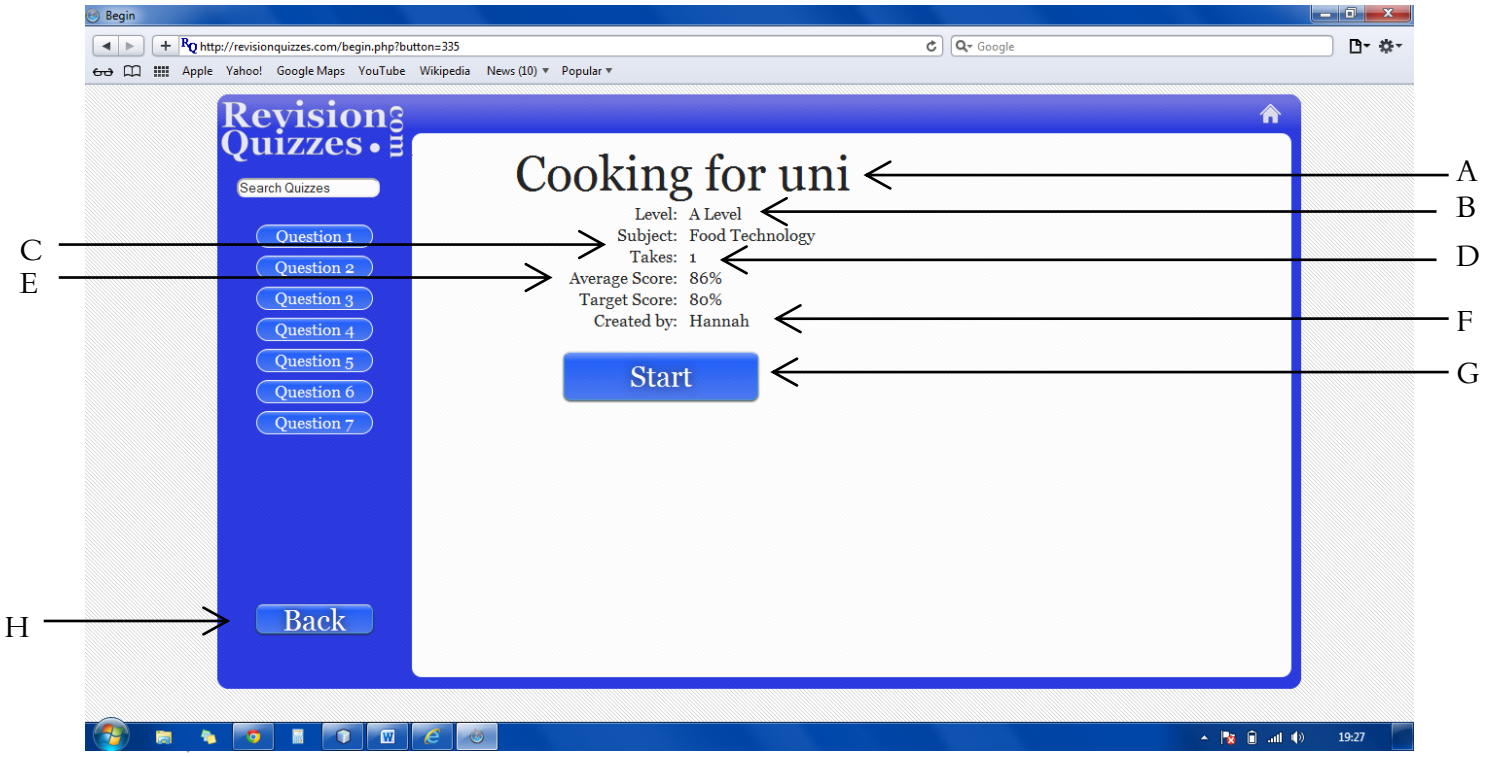
Creator – this is the user name of the person that created the quiz [F]

Tags – These are key words relating to the quiz, which make searching easier

Not all this information will be displayed. If for example the creator of the quiz wished to remain anonymous, there name will not appear. Or if there was no target time entered, this field would not show at all.

To start the quiz you must press *Begin* your time will start as soon as this button is pressed.

The begin button is the dominant button on the lower half of the screen. [G]



Doing a quiz

Once the user has begun the quiz following the instructions above, they can do the quiz.

The blue bar on the side lists all the questions, and if the creator of the quiz enabled questions to be done out of order, then these question links can be clicked to go strait to the question.

The Question will be printed at the top of the screen, on the white background.

The answers (between 2 – 6) will be displayed in blocks below the question

There is a cancel button on the lower left hand side of the screen

If JavaScript is not enabled, there will be a radio button inside each answer box

Also if JavaScript is not enabled there will be a next button in the lower right hand side

The display of additional details like images or text

If there are additional details, or an image uploaded by the creator of the quiz, the question and answers will be displayed on the left rather than in the centre, this will make room for the image or text, to be displayed on the right.

How to do the quiz

First the user must read the question and the answers, and if available, any additional details

To select answer, they must click the one they think is correct

There score will automatically be recorded, and the next question will be displayed.

If JavaScript is not enabled, there will be a radio button and next button visible, which the user will have to click to proceed.

Changing an answer

If the creator of the quiz has allowed the user to do the questions out of order, then the user can modify their answer.

They can do this by clicking the question they wish to jump to from the menu on the side.

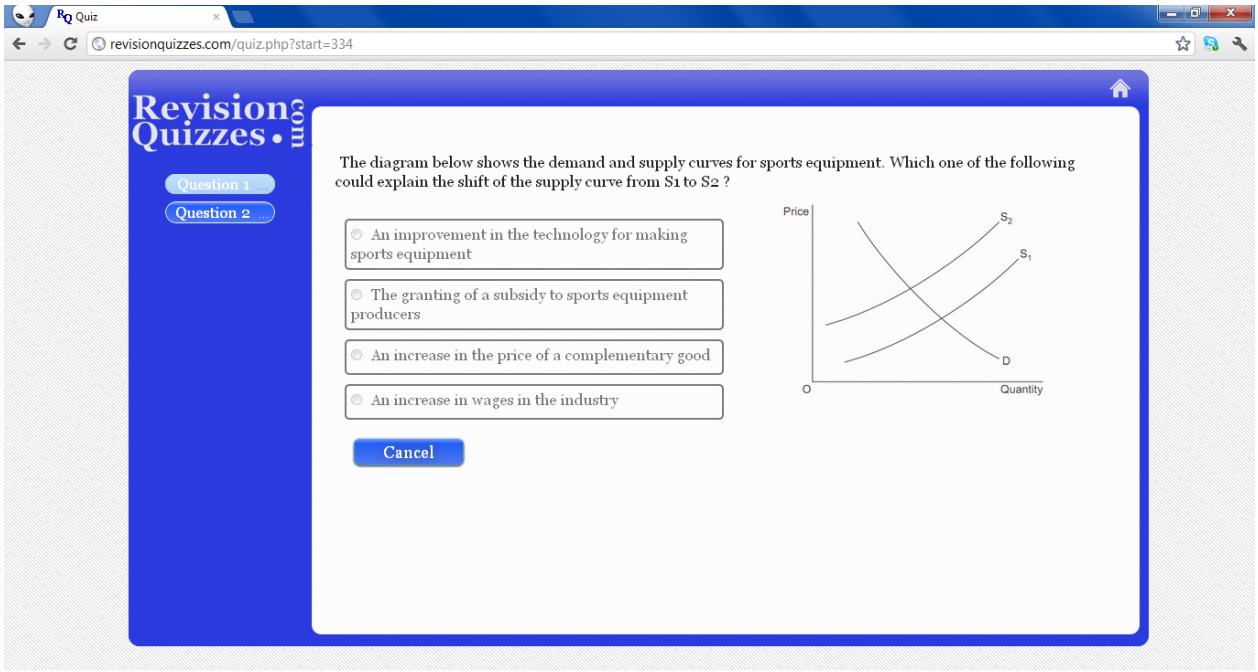
If they have already answered that question, there answer will be shown, but may be modified.

Exiting from the quiz half way through

If the user no longer wants to continue with the quiz process, then they can cancel.

This can be done by clicking the button saying cancel in the lower left of the screen

Also the user can click the home button in the top right corner, a warning message will appear checking that they are happy to leave that page.

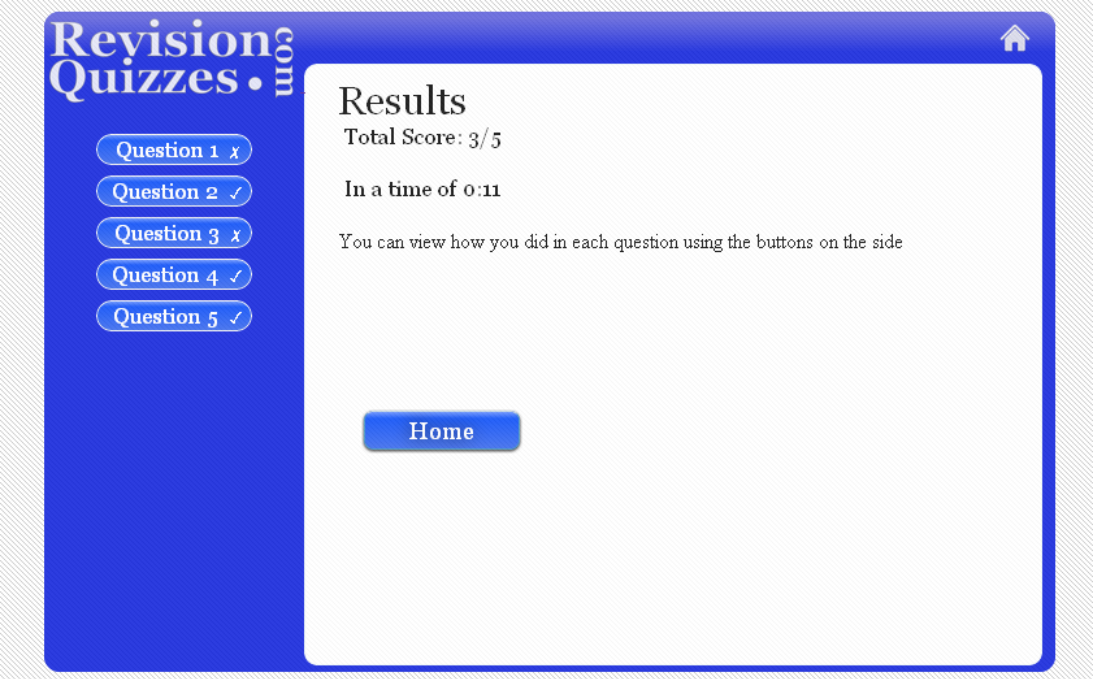

RQ Quiz
revisionquizzes.com/quiz.php?start=334
Revision Quizzes .com
Question 1
Question 2
The diagram below shows the demand and supply curves for sports equipment. Which one of the following could explain the shift of the supply curve from S1 to S2 ?
An improvement in the technology for making sports equipment
The granting of a subsidy to sports equipment producers
An increase in the price of a complementary good
An increase in wages in the industry
Cancel
Price
S2
S1
D
O
Quantity

Finishing a quiz

The quiz will finish automatically once all the questions are complete.

The finish screen will be displayed to the user, showing the time they completed it in and their score

The user will be able to click each of the bits of information to see more related to that.



Viewing your results

The user can view how they did on each question by clicking the question number on the side pain.

The question will be shown as before, if there is an answer explanation added for that question by the creator, then it will be displayed.

The user can either use the next button to navigate in order, or just click the link on the right. If they hover over a question number, the question will appear in a float box, to save time.

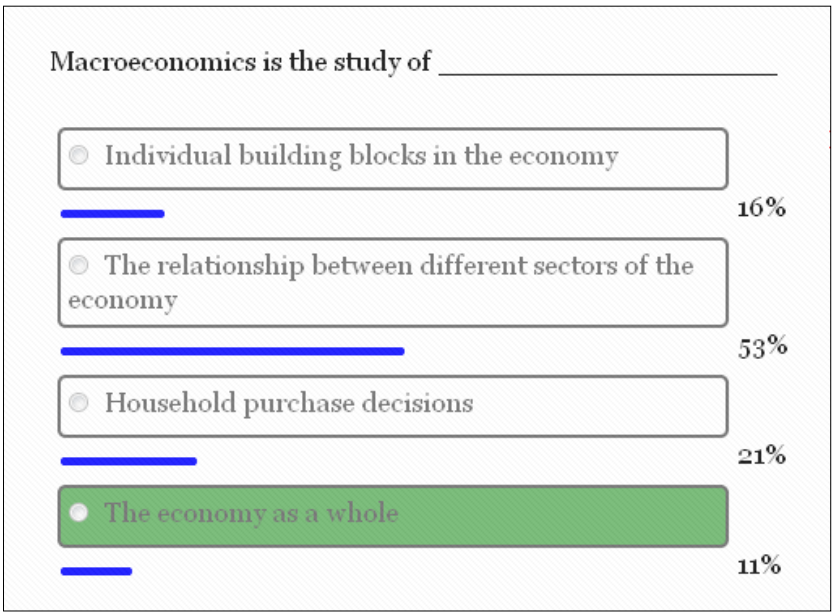
Viewing the averages

The user can compare their score with others who have done that quiz. This can be done by:

Clicking the *Show Stats* button which is on the right, and only visible when the quiz is complete

The answer blocks will slide down, and between each one a line chart will appear

This represents as a % what everyone else score on that question.



Creating a quiz

The program has the ability to allow users to create a quiz. This can be done by:

Click the *create a quiz* button which is on the lower left hand side

The *Create a quiz* screen should be displayed

Next the user must enter the necessary information into the create a quiz form. All the fields on the basic information tab are mandatory.

Quiz Name – this is the name of the quiz

Subject – this is the category that the quiz fits into

Level – this is the level that the quiz is at

The quiz will only be valid if the name – subject – level combination is unique.

Adding additional options to your quiz

RevisionQuizzes.com allows users to enter additional information about the quiz, to add functionality. To start click the tabs on the right.

To change whether the creators name is visible

Click the quiz settings tab on the right.

There is a check box saying *Show creators name*?

Once it is checked a text box will slide down

If you are logged in, your user name will already be in the text box

If not, you will have to type it in the input box.

Completing the quiz out of order. There is an option that will allow the user to do the quiz out of order, this will mean they can use the question buttons on the side to navigate through the questions. To allow users to do the quiz out of order:

Click the quiz settings tab

Check or uncheck the box that says allow questions to be done out of order.

Target time. A target time is the time that the user should aim to complete the quiz in under. To set a target time:

Go to the *Timing and Scores* tab

Click the Target time textbox, and the time picker will appear

Use the sliders to select the desired target time.

There is also a built in time limit function, which when set will not allow the user to do the quiz for longer than the set amount of time. To enable this function:

Click on the *Timing and Scores* tab

Change the radio button for time limit

Setting a pass mark, a pass mark can be set so the user will either pass the quiz, if they get above the pass mark, or fail if they get below the mark. To set a pass mark:

Click the timing and scores tab

In the text box where the label says pass mark, put in a number between 2 and the total number of questions.

The creator of the quiz can also change the privacy of the quiz; if the quiz is public then everyone can view and do the quiz. If the quiz is private, then only the creator and people in the creators list can view and do the quiz. To change the quiz privacy:

Go to the quiz privacy tab

Check the radio button you want.

Again if you hover over the radio buttons, a short explanation of what each does will appear

All these quiz settings can also be changed when you have finished creating the quiz

Adding a question

Once the quiz has been created the user can start adding questions. This can be done when the add question screen appears by:

Write the question in the first textbox labelled question

Add between 2 and 6 answers in the next text boxes

Adding text in one will cause the others to appear

Select the correct answer by checking the radio button in the right of each answer

Press next to proceed to the next question, or finish to finish the quiz creating process

Adding additional options to your question

The user can also add more options to each individual question, like they can the quiz. This can be done by using the tabs on the side.

To modify or add the question, answers and correct answer, use the questions and answers tab.

To change to type of question, from either multiple choice, long answer or drag and drop:

Click the type of question tab

It will slide down

Use the radio selectors to select the type of question

The fields on the question and answers tab will automatically change

If you switch the type of question back, your question will be stored

To add additional notes, which will be displayed during the running of your program:

Click the add additional notes tab on the right

Type the notes on the right

Follow the on screen instructions for more details

To add an image or diagram, which will be displayed at that question in the quiz

Click the image or diagram tab on the right

Click choose file, and use the file selector to choose your image

Click upload

To add an answer explanation, which will be displayed only when the question is completed:

Click the answer explanation tab on the right

Type your answer explanation in the box

Follow the on screen instructions for more details

Checking your question

There are a number of instant validation and methods of verification built in, to help increase the accuracy of the users questions

To check that all necessary information is entered, and meets the validation rules. On the lower right hand side or the create a question form, there is a number of bullet points saying what still needs to be done. As soon as it is done, the bullet point will disappear, when there are none left, suggestions on how the question can be improved will be displayed, and the user will be able to proceed.

For a question to be valid it must:

Have a question between 25 and 250 characters long

Have at least two answers, each between 1 and 150 characters long

Have a correct answer selected

If the user try's to proceed without their answer matching the criteria above, a dialog box will come up informing them what they need to be able to do before proceeding, and won't let them proceed.

To verify the question and answers are correct and what the user wants, there is a live preview tab, which will display the question, answers and correct answer as well as the additional details, answer explanation and image if they have been added.

Finishing creating a quiz

To finish creating a quiz the user must press the finish button. They will then be directed to the finish screen. From the finish screen they will be able to edit all the quiz information again. Also the quiz ID will be displayed, which can be used to go straight to a quiz.

Updating and editing a quiz

A quiz can be updated or edited only by the person who created it, if they are logged in, or an admin. To edit a quiz from the main page:

Click the quiz you would like to edit

When the begin screen is displayed, click the edit button on the lower right corner

You will be redirected to the edit screen

This will be the same as the create a quiz screen, only the quiz information will already be stored

Creating an account

RevisionQuizzes.com allows users to create an account, so that they can save their scores, edit there quizzes, and use the additional options. To create an account:

Click the login/ sign up

A bar will slide down

Click the sign up button on the right

A pop up box will appear

Fill in the fields as indicated

Press sign up

Logging in and out

To log in:

Click the login button

The login bar will slide down

Enter your user name and password

Press login

It will log you in, and redirect you back to the start page

To log out:

If you are logged in at the top of the screen it will say, logged in as [username]

Next to that there will be a button saying log out

Press log out

You will be redirected to the log out page, then back to the start page

You are no longer logged in

Viewing and amending your profile

To edit your profile:

If you are logged in at the top of the screen it will say, logged in as [username]

Next to that there will be a button saying profile

If you press profile you will be redirected to your profile

Because you are logged in, you will be able to edit and add more information

To preview your profile:

Go to your profile, as above

Click the preview tab on the left

To view someone else's profile:

Click there username where it appears on the begin screen

Logging in and using the teacher control panel

To use the teacher control panel, you must be logged on as a teacher. When creating an account, be sure to select teacher. If you are logged on as a teacher, next to the logged in as [username] button, there is log out, profile and teacher control panel link. Click the teacher control panel link, and you will be redirected to the teacher control panel page. There will then be on screen instructions to do various functions.

Logging in and using the admin control panel

Logging in and using the admin control panel is similar to using the teacher control panel. You must have an admin account to use this feature. When logged in, there will be a link to the admin control panel, which will redirect the user once clicked. A second password will be required to use the admin control panel.

Saving a quiz

Quizzes can be saved to the user's computer in a format that can be read, printed or edited offline. To save a quiz:

Click the quiz you would like to save from the start screen.

You will be redirected to the begin screen

From here, at the lower right hand side, there are some quiz options buttons

Click save

A pop out div will show

There will be a number of options

Click the option you would like, the download will start automatically

Printing a quiz

The user can also print a quiz, to be done on paper. This will be done by generating a PDF and opening the print menu. To print a quiz:

Click the quiz you would like to print

When redirected to the begin menu, there will be an option to print the quiz

Click print

The print menu will appear

Reporting a quiz

Quizzes can be reported by users for a number of reasons:

The quiz in inappropriate, or contains inappropriate content

The quiz is inaccurate, or contains incorrect information

The quiz is broken, or doesn't work as expected

To report a quiz:

Click the quiz, you will be redirected to the begin screen

On the lower right hand corner, there is a group of buttons

Click *Report Quiz*

A pop out div will appear

You will need to fill in the input fields

Then click send

Leaving feedback

The user can leave feedback of RevisionQuizzes.com. This is so that potential bugs can be found, and comments can be left. This will allow for the website to improve. To leave feedback:

Scroll down to the bottom of the start page

In small writing there is a hyper link saying *leave feedback*

Click it, and a pop up box will appear, where users can leave feedback

All fields are optional

Getting help

Users can get help on RevisionQuizzes.com in a number of ways. The documentation can be found at documentation.revisionquizzes.com. This will be a complete guide to everything related to the program. There is also on screen help, which will be made visible during the running of the program. For example while creating a quiz or adding a question the user can hover over any HTML element that they are unsure what to put in it, and a short sentence explaining what will go in that area will be made visible. Also the user can use the contact button which can be found at the bottom of the start screen near feedback. Similar to feedback, a pop out div will appear, and the user can enter their query, and optionally their contact details and send.

Degree of Success

To enter quiz information

Requirement	Passed	Degree of which it was met
Quiz Name	✓	This was fully met. It is essential that the user must enter a quiz name, and they will not be able to proceed otherwise
Level	✓	This was also met. It is a drop down menu as requested, the levels are sorted by key stages, and the ages will show on hover. If the level required is not available, the use can press other and type the required level into a textbox, which will have predicted text containing results from all the other levels in the database.
Subject	✓	This was also met, although not a drop down menu as the user initially suggested. This was because there were too many subjects to include, and a predictive text system was implemented instead, which contains results from the database, and is more dynamic.
Questions	✓	This was fully met. The user can enter between 1 or 100 questions per quiz, the minimum length of a question is 5 characters, and the maximum is 250. If 250 characters is not an adequate amount of space, the user can enter additional details which will be shown alongside the question.
Answers	✓	This was also met. The user specified that they would like to be able to add a variable amount of answers, and this feature was implemented successfully. They can enter between 2 and 6 answers, although these figures can be easily adjusted from code.
Correct Answer	✓	This was met; the user is able to easily select which answer they would like to be correct. They will not be able to proceed without selecting a correct answer, and it will be easy and convenient for the answer to be selected.
A place to insert an image or diagram	✓	This feature was also met. It is quick and easy for the user to upload an image or a diagram, and there are a large range of

		available data types and dimensions.

To be able to complete quizzes:

Requirement	**Passed**	**Degree of which it was met**
Loop through each question in order	✓	This was fully met, and was one of the first parts of the program to be implemented. The questions be default are shown in the order they are written
Show the user if they got each question right	✓	This requirement was also met, and users can view if they got each question right immediately after they have done it, providing they have JavaScript enabled
Questions laid out on a clear screen	✓	There is nothing else on the question screen other than the questions and possible answers, unless additional details or an image has been added.

Enable the user to search for quizzes:

Requirement	**Passed**	**Degree of which it was met**
Quick efficient way of searching	✓	This requirement was met, there is a built in instant search function which is quick, and the SQL queries make it efficient through appropriate sorting.
Key word searching	✓	This was also met. The user needs only to enter a keyword and quizzes will be shown according to weather they contain that word
Unique ID for each quiz	✓	The user will be given their quiz ID when they finish creating a quiz, or alternatively they can find it from the begin menu. It can then be entered into the enter a quiz ID form and they will be directed strait to it.

To keep scores of quizzes:

Requirement	Passed	Degree of which it was met
Record the users score while they do the quiz	✓	The users score is recorded in a session throughout the running of the program; data is amended to it as the user progresses.
Show total score at the end of the quiz	✓	The sum of the score array in the session is displayed to the user in the form of a percentage when they complete the quiz.
Show a breakdown of score at the end	✓	This requirement was met, although the score breakdown is not shown all on one page as originally suggested by the end user. It cluttered the page up, made it hard to understand and due to questions and answers varying in length the page was hard to design effectively.
Show averages	✓	Once the user has finished the quiz, they can view what answers everyone else selected, in a graphical form.
Record score for quiz	✓	The score for each answer is recorded in the database and used in calculations relating to averages.

Store quizzes:

Requirement	Passed	Degree of which it was met
A storage system to keep hold of all quizzes	✓	I have used a MySQL database which is hosted online with the other files. It contains many tables to best sort the quizzes.
Fast, efficient and effective.	✓	It is reasonably fast because it is hosted at the same location to the site, and it quite effective because of the structure of the tables and the use of relationships.
Can store all the necessary information	✓	There are separate tables to questions, answers, quizzes and users. There are the secondary tables for further information.
Store information securely and be	✓	All passwords and personal data will be md5 hashed and salted. This will make them safe. There is a cron job that

regularly backed up		automatically back up the MySQL database weekly.

Admin features:

Requirement	**Passed**	**Degree of which it was met**
Allow an administrator to log in and adjust settings from a user interface	✗	This requirement was not met. During the development it became clear that my end user would no longer need this feature mainly because all the tools that would have been available on the user interface are still available as ready-made packages such as PHPMyAdmin – for administrating the database, and the code is very easy to modify as all the variables containing key bits of data are on a separate page and are clearly commented on. There is however a link that will be made visible to admins directing them to a password protected page with links to the tools mentioned above. I will develop a more detailed admin control panel in the future.
Allow questions, answers and quizzes to be deleted	✓	Users can delete their quiz, if they are logged in and they created that quiz. Also administrators can modify or delete database value of the quiz.
Allow questions, answers and quizzes to be edited	✓	Users can edit their quiz, if they are logged in and they created that quiz. Also administrators edit database values of the quiz.

Good user interface:

Requirement	**Passed**	**Degree of which it was met**
A clear and easy to use user interface.	✓	The user interface does not contain much unnecessary objects or text, and is quite clear.

Simple installation routine and compatibility

Requirement	Passed	Degree of which it was met
Quick and easy to installation routine	✓	It will run on virtually any browser, so no installation is necessary, although to make full use of all the additional features a different browser may need to be installed. Instructions are included in the documentation. There is also a Chrome app which can be installed, this too is quick and very easy, it can be found on the Chrome web store.
Will work on computers with restricted permissions – like school computers	✓	This was met. Because it runs in the browser, providing that the computer has IE 8 or later installed it will work with full functionality.
Compatible with older computers	✓	Theoretically, yes it can run on any computer using XP or possibly earlier. Although the additional download of Google Chrome or a similar browser will be necessary. XP SP2 will not allow IE 7 or late to be installed.

Different quiz modes and settings

Requirement	Passed	Degree of which it was met
Questions in order, or out of order, and randomise	✓	The creator of the quiz can chose whether to allow the user to be able to do the questions out of order and jump between them, or to force them to do them in order, and disable the jump buttons. Also questions can be randomised, where they will appear in a random order, further to this answers inside questions can be randomised.
Show user the correct answer of question immediately	✓	The user can see whether they got the question right or not immediately after selecting an answer. It will appear green if correct, or red if incorrect. If they got it wrong the correct answer will also be highlighted to show the user.

Show score and breakdown at the end	✓	The user can see a full breakdown of their scores once they have finished the quiz.
Change how many times a quiz can be retaken	✗	This requirement was only partially met. Although the creator of the quiz can not control how many times a quiz can be retaken, they can control how many times the users score is recorded. It identifies and records users using their user account, IP and browser cookies.

Shortfalls

Although I fully met the majority of my initial requirements, some were either not met or adjusted slightly during user involvement over the development of the program.

The admin control panel

The main feature not met was the admin control panel. This was because the end user no longer though it necessary because of the tools already available. I will develop this feature in the future.

Overlapping object

Also there were some minor glitches relating to the screen resolution on smaller screens. This was found while a user was using the site from a 10 inch Android tablet. The search filters slightly overlap the side bar. Although this problem seems to only occur using Dolphin browser.

Browser incompatibility with older versions of IE

As of December 2011, 20% of people use Internet Explorer the default browser on Windows systems, and created by Microsoft (5% use IE9, 10% use IE8, 5% use older versions). My program works fully with full functionality on IE9, and has full functionality but less gradient fills or curved corners on IE 8. Although on older versions there are bugs and glitches that make it less usable. This is a fall down, because 5% of computer users won't be able to use RevisionQuizzes.com to its full potential. However I have set up a redirect, that will detect what browser they are using and if it is incompatible will send them to the browser compatibility page.

If I were remaking this program again, or developing a similar program in the future, I think I would improve it by having better structured and more efficient code. I would research available tools for doing this. I would probably incorporate more object-orientated PHP into my project, as this would allow for less to be written as classes can be reused throughout the project. I would also try and make my commenting more specific, because I found it hard at times carrying out dry-runs throughout my code as some parts were unclear as to what exactly it was doing.

Desirable extensions

Although my program met the most of the system requirements, there are some areas where if I had had more time, I would like to add additional features.

User high scores

I could have increased the incentive for pupils to voluntarily want to do public revision quizzes, by creating a high score system. This would combine their time and score, and would record their name and score for their first attempt.

User profile page

A page containing information that the user had added and had chosen to make public on their profile. It would also contain their scores and recently done quizzes, as well as all the quizzes they had written.

Charts and graphs to graphically display results when the user has finished the quiz

The results screen is quite empty, and I would like to create some charts and graphs that can graphically display the user's results, and the averages. They would be dynamic and be created using AJAX.

An improved user interface for some of the forms

The layout of some of the forms could be improved to make better use of space, be clearer, look better and allow for better compatibility. The quizzes on the start screen and the first screen of the create a quiz are two of the forms I think most need adjusting.

A tag system

When there are more quizzes on the system, searching by title may not display all relevant quizzes. I would like to create a tag system, which will store basic tags as a single word and the quiz ID in the database and allow for faster searching.

A quiz rating system

This would allow users to rate the quiz once, after they have completed it. It could be between 1 and 5 stars. The purpose of this is that then search results can show 5 start ratings first, then 4 star and so on. This will stop quizzes which are not so good being displayed at the top of the search results.

Improved instant search

One of the most important aspects of the program is that users should be able to browse and search for quizzes quickly. I incorporated an instant search to help enable this, although it is not comprehensive enough. It should also be able to instantly search using the advanced filters, and the did you mean

function should be instant. This will become harder to implement when there are more results in the database, as it could get slow, so I will rewrite it.

Different question types

Currently the only type of questions available are multiple choice, I would like to expand upon this and allow for long answers and drag and drop questions to be added.

End user involvement

I have interviewed my end user for the final time, to get their feedback. Below I have included the results from the interview relating to each user requirement and whether or not it was met.

To enter quiz information

Requirement	Passed	End user comments
Quiz Name	✓	Open choice for naming the quiz means they it can be adjusted to match the group taking the quiz
Level	✓	We discussed the age groups that could use the programme and some were added.
Subject	✓	Plenty of room for adding more subjects - very useful.
Questions	✓	
Answers	✓	
Correct Answer	✓	
A place to insert an image or diagram	✓	This provides an extension to the quiz.

To be able to complete quizzes:

Requirement	Passed	End user comments
Loop through each question in order	✓	
Show the user if they got each question right	✓	
Questions laid out on a clear screen	✓	Very clear for user.

Enable the user to search for quizzes:

Requirement	Passed	End user comments
Quick efficient way of searching	✓	
Key word searching	✓	New added instant search impressive
Unique ID for each quiz	✓	Programme recognises the users details so can not memorise and change answers.

To keep scores of quizzes:

Requirement	Passed	End user comments
Record the users score while they do the quiz	✓	
Show total score at the end of the quiz	✓	
Show a breakdown of score at the end	✓	
Show averages	✓	Users like to compare what they have done.
Record score for quiz	✓	

Store quizzes:

Requirement	Passed	End user comments
A storage system to keep hold of all quizzes	✓	
Fast, efficient and effective.	✓	
Can store all the necessary information	✓	
Store information securely and be regularly backed up	✓	Very good that questions can be added and stored.

Admin features:

Requirement	Passed	End user comments
Allow an administrator to log in and adjust settings from a user interface	✓	Gives leader/teacher more control.
Allow questions, answers and quizzes to be deleted	✓	
Allow questions, answers and quizzes to be edited	✓	Only by administrator or when logged in when you create it.

Good user interface:

Requirement	Passed	End user comments
A clear and easy to use user interface.	✓	

Simple installation routine and compatibility

Requirement	Passed	End user comments
Quick and easy to installation routine	✓	
Will work on computers with restricted permissions – like school computers	✓	
Compatible with older computers	✓	Very important as people have different grades of computers.

Different quiz modes and settings

Requirement	Passed	End user comments
Questions in order, or out of order, and randomise	✓	
Show user the correct answer of question immediately	✓	
Show score and breakdown at the end	✓	
Change how many times a quiz can be retaken	X	~~Could~~ Can't do at moment but maybe a moderation for the future

Into is a very impressive piece of work and I look forward to using the programme on a regular basis.

Confirmation of End user involvement

Signed End user [Mrs Munt] Jane's Munt.

Signed Developer [Alicia Sykes] Alicia Sykes

Appendix

Cascading Style Sheets

All styles:

```
1.
    /*=====GENERAL=====*/
2. div.{ }
3. * {margin: 0;
4.  padding: 0;
5.  }
6.
7. body{
8.   background-image:url('../img/bg.gif');
9.   height:100%;
10.  width: 100%;
11.  padding: 0;
12.  margin: 0;
13.  overflow: hidden;
14.}
15.
16.
17.
18. /*=====BIG OBJECTS=====*/
19.
20.
21.div.main{
22.  margin-left: 180px;
23.  margin-top: 20px;
24.  margin-right: 8px;
25.  margin-bottom: 5px;
26.  height: 93%;
27.  width: auto;
28.  background-color: white;
29.  padding-left: 25px;
30.  padding-right: 25px;
31.  padding-top: 10px;
32.  border-style: solid;
33.  border-color: #EFF7FA;
34. border-width: 1px;
35.  border-radius: 10px;
36.  overflow: auto;
37.  overflow-x: hidden;
38.       }
39.
40.div.back{
41.  height: 90%;
42.  width: 80%;
43.  margin-left: 10%;
```

```
  border-radius: 12px;
  position:fixed;
  top:10px;
  padding-left: 20px;
  padding-top: 20px;
  padding-right: 2px;
  padding-bottom:2px;
  z-index: -1;
  opacity: 0.85;
  }


div.questionBox{
  width: 400px;
  min-height: 25px;
  border-color: black;
  border-style: solid;
  border-radius: 5px;
  border-width: 2px;
  padding: 5px;
  margin: auto;
  margin-top: 10px;
  cursor: pointer;
  background-color: white;
  opacity: 0.6;
  filter:alpha(opacity=60);
    -webkit-transition: opacity 1s linear;
    overflow: auto;
}

div.questionBox:hover{
    opacity: 0.9;
    filter:alpha(opacity=90);
    }

div.questionBoxLM{
  width: 400px;
  min-height: 25px;
  border-color: black;
  border-style: solid;
  border-radius: 5px;
  border-width: 2px;
  padding: 5px;
  margin: auto;
  margin-top: 10px;
  margin-left: 10px;
  background-color: white;
  opacity: 0.6;
  cursor: pointer;
  filter:alpha(opacity=60);
    -webkit-transition: opacity 1s linear;
}
```

```
96.
97. div.questionBoxLM:hover{
98.   opacity: 0.9;
99.   filter:alpha(opacity=90);
100.          }
101.
102.
103.      div.form{
104.        border-radius: 10px;
105.        margin-left: 20px;
106.        margin-top: 20px;
107.        width: 700px;
108.        height: 400px;
109.        z-index: 1;
110.              }
111.
112.      div.innerForm{
113.        width: 500px;
114.        height: 200px;
115.        padding-top: 15px;
116.        padding-bottom: 15px;
117.        margin-top: 10px;
118.        margin-botton: 10px;
119.        background-color: #F0FAFF;
120.       position: absolute;
121.              }
122.
123.      div.sideForm{
124.        width: 200px;
125.        height: 300px;
126.        display:block;
127.        float: right;
128.        padding-top: 100px;
129.        background-color: #66A3FF;
130.       border-top-right-radius: 10px;
131.        border-bottom-right-radius: 10px;
132.              }
133.
134.       div.formQuestion{
135.        border-radius: 10px;
136.        margin-top: -10px;
137.        margin-left: -25px;
138.        min-width: 790px;
139.        height: 100%;
140.              }
141.
142.
143.      div.formTabs{
144.        width: 200px;
145.        height: 30px;
146.        display:block;
147.        float: right;
```

```
148.         margin-bottom: 5px;
149.         cursor: pointer;
150.         background-color: #7dbbfa;
151.              }
152.
153.       div.formTabs:hover{
154.         background-color: #b9dafa;
155.              }
156.
157.       div.sideFormQuestion{
158.         width: 200px;
159.         height: 84%;
160.         display:block;
161.         float: right;
162.         margin-top: -1px;
163.         margin-right: -13px;
164.         padding-top: 10%;
165.         background-color: #66A3FF;
166.        border-top-right-radius: 10px;
167.         border-bottom-right-radius: 10px;
168.             }
169.
170.       div.innerFormQu{
171.         padding-top: 15px;
172.         padding-bottom: 15px;
173.         margin-top: 10px;
174.         margin-bottom: 10px;
175.         position: absolute;
176.         width: 550px;
177.         padding-left: 3px;
178.         position: absolute;
179.             }
180.
181.       div.FormQuestion{
182.         width: 200px;
183.         height: 84%;
184.         display:block;
185.         float: right;
186.         padding-top: 10%;
187.         background-color: #66A3FF;
188.        border-top-right-radius: 10px;
189.         border-bottom-right-radius: 10px;
190.              }
191.
192.
193.
194.       div.typeQuestion{
195.         width: 200px;
196.         height: 150px;
197.         display: block;
198.         float: right;
199.         margin-bottom: 5px;
```

```
200.        cursor: pointer;
201.        background-color: #7dbbfa;
202.       margin-bottom: 5px;
203.              }
204.
205.      /*=====SMALL OBJECTS=====*/
206.
207.      .quizBlock{
208.        border-radius: 5px;
209.        border-width: 1px;
210.        float: left;
211.        width: 150px;
212.        height: 50px;
213.        margin-left: 16px;
214.        margin-right: 16px;
215.        margin-top: 10px;
216.        margin-bottom: 10px;
217.        display: inline-block;
218.        text-shadow:2 2 25px black ;
219.        cursor: pointer;
220.
221.          -moz-border-radius: 10px;
222.         -webkit-border-radius: 10px;
223.         border-radius: 7px;
224.
225.         border: 1px solid #368DBE;
226.        border-top: 1px solid #c3d6df;
227.        text-shadow: 0.5px 0.5px 0.5px white;
228.         -moz-box-shadow: 0 1px 3px black;
229.         -webkit-box-shadow: 0 1px 3px black;
230.         box-shadow: 0 1px 3px black;
231.
232.          }
233.
234.      .quizBlock:active {
235.        -moz-box-shadow: 0 2px 6px black;
236.        -webkit-box-shadow: 0 2px 6px black;
237.          }
238.
239.
240.      .browse{
241.        max-width: 98%;
242.        min-width: 96%;
243.        max-height: 350px;
244.        margin-left:20px;
245.        margin-right:25px;
246.        margin-bottom: 25px;
247.        border-top-right-radius: 5px;
248.        border-bottom-right-radius: 5px;
249.        border-bottom-left-radius: 5px;
250.        overflow: auto;
251.        padding-top: 20px;
```

```
252.      }
253.
254.      .tab{
255.          width: 125px;
256.          height: 20px;
257.          margin-left:20px;
258.          border-top-left-radius: 5px;
259.          border-top-right-radius: 5px;
260.
261.      }
262.
263.      .ext{
264.
265.          min-height: 100px;
266.          position: fixed;
267.          top: 150px;
268.          margin-left:480px;
269.          margin-right: 120px;
270.          border-radius: 8px;
271.      }
272.
273.      .filters{
274.          width: 75px;
275.          height: 25px;
276.          padding: 1px;
277.          margin-left: -8px;
278.          text-shadow: 1px 1px 1px white;
279.
280.
281.      }
282.
283.      .filtersExpand{
284.          width: 160px;
285.          height: auto;
286.          padding: 10px;
287.          margin-left: -9px;
288.          margin-top: 10px;
289.
290.           -moz-border-radius: 10px;
291.           -webkit-border-radius: 10px;
292.           border-radius: 7px;
293.           border: 1px solid #368DBE;
294.          border-top: 1px solid #c3d6df;
295.          -moz-box-shadow: 0 1px 3px black;
296.           -webkit-box-shadow: 0 1px 3px black;
297.           box-shadow: 0 1px 3px black;
298.
299.      }
300.
301.      .filtersCollapse{
302.          width: 160px;
303.          height: 15px;
```

```
304.          padding: 10px;
305.          margin-left: -9px;
306.          overflow: hidden;
307.           -moz-border-radius: 10px;
308.           -webkit-border-radius: 10px;
309.           border-radius: 7px;
310.           border: 1px solid #368DBE;
311.          border-top: 1px solid #c3d6df;
312.          -moz-box-shadow: 0 1px 3px black;
313.           -webkit-box-shadow: 0 1px 3px black;
314.           box-shadow: 0 1px 3px black;
315.
316.      }
317.
318.      .filtersTab{
319.         border-top-left-radius: 5px;
320.         border-top-right-radius: 5px;
321.         border-top-color: #d2d8fc;
322.        border-left-color: #d2d8fc;
323.        border-right-color: #d2d8fc;
324.        background-color: white;
325.          border-color: #d2d8fc;
326.        width: 100px;
327.         height: 25px;
328.      }
329.
330.      .side{
331.          width: 200px;
332.          position:fixed;
333.
334.
335.      }
336.
337.      .ansBlock{
338.          width: 600px;
339.          padding-top: 12px;
340.          padding-bottom: 0px;
341.          margin-bottom: 12px;
342.          border-radius: 12px;
343.      }
344.
345.      .selectedAns{
346.          width: 600px;
347.          padding-top: 12px;
348.          padding-bottom: 12px;
349.          border-radius: 12px;
350.          background-color: #6b8bff;
351.      }
352.
353.      .validationCheck{
354.          padding: 5px;
355.          border-radius: 5px;
```

```
356.          border-style: solid;
357.          border-width: 2px;
358.          border-color: #fad73a;
359.         background-color: #ffffa1;
360.         z-index: 0;
361.          width: 180px;
362.          display:block;
363.          float: right;
364.          margin-right: 4px;
365.          margin-left: 4px;
366.      }
367.
368.      .validationCheck:hover{
369.       opacity: 0.95;}
370.
371.
372.
373.      .stats{
374.          max-width: 400px;
375.          height: 5px;
376.          border-radius: 5px;
377.          margin: 12px;
378.          background-color: blue;
379.          display: none;
380.      }
381.
382.      .statsPercentage{
383.          left:650px;
384.          position:absolute;
385.          display: none;
386.      }
387.
388.      .quizOptions{
389.          position: relative;
390.          float: right;
391.          top: 5%;
392.      }
393.
394.      .homeFilters{
395.          padding-top: 25px;
396.          padding-bottom: 10px;
397.          padding-left: 20px;
398.          padding-right: 20px;;
399.          border-width: 1px;
400.
401.
402.           border-bottom: 1px solid #a187cf;
403.          -moz-box-shadow: 0 1px 3px #cdb4f7;
404.          -webkit-box-shadow: 0 1px 3px #cdb4f7;
405.          box-shadow: 0 1px 2px #cdb4f7;
406.
407.      }
```

```
408.
409.      .logButton{
410.      display:inline;
411.      cursor: pointer;
412.      font-size: 12px;
413.      }
414.
415.      .loginBlock{
416.      margin-left: 180px;
417.      margin-top: 20px;
418.      margin-right: 8px;
419.      margin-bottom: -10px;
420.      height: 25px;
421.      padding-left: 18px;
422.      padding-right: 10px;
423.      padding-top: 6px;
424.      padding-bottom: 5px;
425.      background-color: white;
426.      display: none;
427.      border-radius: 5px;
428.      overflow: auto;
429.      }
430.
431.      .createAccountWrapper{
432.          position: absolute;
433.          top: 20%;
434.          left: 35%;
435.          height: 350px;
436.          width: 450px;
437.          background-color: white;
438.          z-index: 1000;
439.          border-radius: 10px;
440.      }
441.
442.      .title{
443.         width: 450px;
444.         height: 50px;
445.         padding-top: 5px;
446.         padding-bottom: 5px;
447.         border-top-left-radius: 10px;
448.         border-top-right-radius: 10px;
449.      }
450.
451.      .dim{
452.          position: absolute;
453.          top: 0px;
454.          left: 0px;
455.          right: 0px;
456.          bottom: 0px;
457.          height: 100%;
458.          width: 100%;
459.
```

```
460.          opacity: 0.6;
461.          filter: alpha(opacity =60);
462.          background-color: black;
463.      }
464.
465.      .logInTB{
466.          width: 100px;
467.      }
468.
469.      .scoreName{
470.          width: 60px;
471.          border-radius: 6px;
472.      }
473.
474.      .addScore{
475.          padding-top: 5px;
476.          padding-bottom: 5px;
477.      }
478.
479.      #scoreWrapper{
480.         background-color: #4747FF;
481.         border-radius: 10px;
482.          padding: 11px;
483.          width: 300px;
484.
485.      }
486.
487.      .scoreTable
488.      {
489.        padding: 0;
490.        margin: 0;
491.        border-collapse: collapse;
492.        border: 1px solid #333;
493.       font-family: "Trebuchet MS", Verdana, Arial, Helvetica, sans-serif;
494.        font-size: 0.9em;
495.        color: #000;
496.       background: white;
497.        width: 300px;
498.
499.      }
500.
501.
502.      .scoreTable th, .scoreTable td
503.      {
504.        border: 1px dotted #666;
505.       padding: 0.5em;
506.        text-align: left;
507.        color: #0000CC;
508.      }
509.
510.      .scoreTable th[scope=col]
511.      {
```

```
512.         color: #000;
513.        background-color: #8fadcc;
514.        text-transform: uppercase;
515.         font-size: 0.9em;
516.         border-bottom: 2px solid #333;
517.        border-right: 2px solid #333;
518.       }
519.
520.       .scoreTable th+th[scope=col]
521.       {
522.         color: #fff;
523.        background-color: #7d98b3;
524.        border-right: 1px dotted #666;
525.       }
526.
527.       .scoreTable th[scope=row]
528.       {
529.         background-color: #b8cfe5;
530.        border-right: 2px solid #333;
531.       }
532.
533.       .scoreTable tr.alt th, .scoreTable tr.alt td
534.       {
535.         color: #2a4763;
536.       }
537.
538.       .scoreTable tr:hover th[scope=row], .scoreTable tr:hover td
539.       {
540.         background-color: #0000CC;
541.        color: #fff;
542.       }
543.
544.        /*=====FILL GRADIENTS=====*/
545.
546.       .fill{
547.           background: #e5e5e5;
548.       background: -moz-linear-gradient(top,  #e5e5e5 1%, #b3b0c1 100%);
549.       background: -webkit-gradient(linear, left top, left bottom, color-
   stop(1%,#e5e5e5), color-stop(100%,#b3b0c1));
550.       background: -webkit-linear-gradient(top,  #e5e5e5 1%,#b3b0c1 100%);
551.       background: -o-linear-gradient(top,  #e5e5e5 1%,#b3b0c1 100%);
552.       background: -ms-linear-gradient(top,  #e5e5e5 1%,#b3b0c1 100%);
553.       background: linear-gradient(top,  #e5e5e5 1%,#b3b0c1 100%);
554.       filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#e5e5e5', e
   ndColorstr='#b3b0c1',GradientType=0 );
555.       }
556.       .form{
557.           background: #cee2ec;
558.          background: -moz-linear-gradient(top,  #cee2ec 0%, #b4d2e4 49%, #cee2ec
   99%);
559.       background: -webkit-gradient(linear, left top, left bottom, color-
   stop(0%,#cee2ec), color-stop(49%,#b4d2e4), color-stop(99%,#cee2ec));
```

```
560.      background: -webkit-linear-gradient(top,  #cee2ec 0%,#b4d2e4 49%,#cee2ec 99%);
561.      background: -o-linear-gradient(top,  #cee2ec 0%,#b4d2e4 49%,#cee2ec 99%);
562.      background: -ms-linear-gradient(top,  #cee2ec 0%,#b4d2e4 49%,#cee2ec 99%);
563.      background: linear-gradient(top,  #cee2ec 0%,#b4d2e4 49%,#cee2ec 99%);
564.      filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#cee2ec', e
   ndColorstr='#cee2ec',GradientType=0 );
565.      }
566.
567.      .formTabsFill{
568.          background: #bbddff;
569.
570.      background: -moz-linear-gradient(top,  #bbddff 0%, #afd5fb 44%, #9ecaf6 100%);
571.      background: -webkit-gradient(linear, left top, left bottom, color-
   stop(0%,#bbddff), color-stop(44%,#afd5fb), color-stop(100%,#9ecaf6));
572.      background: -webkit-linear-gradient(top,  #bbddff 0%,#afd5fb 44%,#9ecaf6
   100%);
573.      background: -o-linear-gradient(top,  #bbddff 0%,#afd5fb 44%,#9ecaf6 100%);
574.      background: -ms-linear-gradient(top,  #bbddff 0%,#afd5fb 44%,#9ecaf6 100%);
575.      background: linear-gradient(top,  #bbddff 0%,#afd5fb 44%,#9ecaf6 100%);
576.      filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#bbddff', e
   ndColorstr='#9ecaf6',GradientType=0 );
577.
578.      }
579.
580.      .formTabsFill:hover{
581.      background: #d7ebff;
582.      background: -moz-linear-gradient(top,  #d7ebff 0%, #cfe6fd 44%, #c6e0fa 100%);
583.      background: -webkit-gradient(linear, left top, left bottom, color-
   stop(0%,#d7ebff), color-stop(44%,#cfe6fd), color-stop(100%,#c6e0fa));
584.      background: -webkit-linear-gradient(top,  #d7ebff 0%,#cfe6fd 44%,#c6e0fa
   100%);
585.      background: -o-linear-gradient(top,  #d7ebff 0%,#cfe6fd 44%,#c6e0fa 100%);
586.      background: -ms-linear-gradient(top,  #d7ebff 0%,#cfe6fd 44%,#c6e0fa 100%);
587.      background: linear-gradient(top,  #d7ebff 0%,#cfe6fd 44%,#c6e0fa 100%);
588.      filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#d7ebff', e
   ndColorstr='#c6e0fa',GradientType=0 );
589.          }
590.
591.          .back{
592.      background: #595cdb;
593.      background: -moz-linear-gradient(top,  #595cdb 1%, #081adb 6%);
594.      background: -webkit-gradient(linear, left top, left bottom, color-
   stop(1%,#595cdb), color-stop(6%,#081adb));
595.      background: -webkit-linear-gradient(top,  #595cdb 1%,#081adb 6%);
596.      background: -o-linear-gradient(top,  #595cdb 1%,#081adb 6%);
597.      background: -ms-linear-gradient(top,  #595cdb 1%,#081adb 6%);
598.      background: linear-gradient(top,  #595cdb 1%,#081adb 6%);
599.      background-color: #182bd9;
600.          }
601.
602.          .tab{
603.      background: #cedcff;
```

```
604.
605.
606.
607.          }
608.
609.          .browse{
610.      background: #cedcff;
611.      background: -moz-linear-gradient(top,  #cedcff 0%, #f4f7ff 11%, #f2f3f7 100%);
612.      background: -webkit-gradient(linear, left top, left bottom, color-
    stop(0%,#cedcff), color-stop(11%,#f4f7ff), color-stop(100%,#f2f3f7));
613.      background: -webkit-linear-gradient(top,  #cedcff 0%,#f4f7ff 11%,#f2f3f7
    100%);
614.      background: -o-linear-gradient(top,  #cedcff 0%,#f4f7ff 11%,#f2f3f7 100%);
615.      background: -ms-linear-gradient(top,  #cedcff 0%,#f4f7ff 11%,#f2f3f7 100%);
616.      background: linear-gradient(top,  #cedcff 0%,#f4f7ff 11%,#f2f3f7 100%);
617.      background-color: #cedcff;
618.
619.
620.          }
621.
622.          .quizBlock{
623.      background: #5484ff;
624.      background: -moz-linear-gradient(top,  #5484ff 2%, #0046f9 18%, #3163ed 100%);
625.      background: -webkit-gradient(linear, left top, left bottom, color-
    stop(2%,#5484ff), color-stop(18%,#0046f9), color-stop(100%,#3163ed));
626.      background: -webkit-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed
    100%);
627.      background: -o-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
628.      background: -ms-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
629.      background: linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
630.      background-color: #182bd9;
631.
632.          }
633.
634.          .quizBlock:hover{
635.      opacity: 0.6;
636.      filter:alpha(opacity=60);
637.        -webkit-transition: opacity 0.1s linear; }
638.
639.          .button{
640.      background: #5484ff;
641.      background: -moz-linear-gradient(top,  #5484ff 2%, #0046f9 18%, #3163ed 100%);
642.      background: -webkit-gradient(linear, left top, left bottom, color-
    stop(2%,#5484ff), color-stop(18%,#0046f9), color-stop(100%,#3163ed));
643.      background: -webkit-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed
    100%);
644.      background: -o-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
645.      background: -ms-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
646.      background: linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
647.      background-color: #5484ff;
648.
649.           -moz-border-radius: 10px;
```

```
650.            -webkit-border-radius: 10px;
651.            border-radius: 7px;
652.
653.            border: 1px solid #368DBE;
654.           border-top: 1px solid #c3d6df;
655.
656.            text-shadow: 1px 1px 1px white;
657.
658.            -moz-box-shadow: 0 1px 3px black;
659.            -webkit-box-shadow: 0 1px 3px black;
660.            box-shadow: 0 1px 3px black;
661.           }
662.
663.           .button:hover {
664.      background: #7a8eff;
665.      background: -moz-linear-gradient(top,  #7a8eff 0%, #6175f9 45%, #405fed 100%);
666.      background: -webkit-gradient(linear, left top, left bottom, color-
   stop(0%,#7a8eff), color-stop(45%,#6175f9), color-stop(100%,#405fed));
667.      background: -webkit-linear-gradient(top,  #7a8eff 0%,#6175f9 45%,#405fed
   100%);
668.      background: -o-linear-gradient(top,  #7a8eff 0%,#6175f9 45%,#405fed 100%);
669.      background: -ms-linear-gradient(top,  #7a8eff 0%,#6175f9 45%,#405fed 100%);
670.      background: linear-gradient(top,  #7a8eff 0%,#6175f9 45%,#405fed 100%);
671.      background-color: #7a8eff;
672.      }
673.
674.       .buttonLogin{
675.      background: #5484ff;
676.      background: -moz-linear-gradient(top,  #5484ff 2%, #0046f9 18%, #3163ed 100%);
677.      background: -webkit-gradient(linear, left top, left bottom, color-
   stop(2%,#5484ff), color-stop(18%,#0046f9), color-stop(100%,#3163ed));
678.      background: -webkit-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed
   100%);
679.      background: -o-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
680.      background: -ms-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
681.      background: linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
682.      filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#5484ff', e
   ndColorstr='#3163ed',GradientType=0 );
683.
684.            -moz-border-radius: 10px;
685.            -webkit-border-radius: 10px;
686.            border-radius: 7px;
687.
688.            border: 1px solid #368DBE;
689.           border-top: 1px solid #c3d6df;
690.
691.            text-shadow: 1px 1px 1px white;
692.
693.            -moz-box-shadow: 0 1px 3px black;
694.            -webkit-box-shadow: 0 1px 3px black;
695.            box-shadow: 0 1px 3px black;
696.           }
```

```
697.
698.          .buttonLogin:hover {
699.      background: #7a8eff;
700.      background: -moz-linear-gradient(top,  #7a8eff 0%, #6175f9 45%, #405fed 100%);
701.      background: -webkit-gradient(linear, left top, left bottom, color-
   stop(0%,#7a8eff), color-stop(45%,#6175f9), color-stop(100%,#405fed));
702.      background: -webkit-linear-gradient(top,  #7a8eff 0%,#6175f9 45%,#405fed
   100%);
703.      background: -o-linear-gradient(top,  #7a8eff 0%,#6175f9 45%,#405fed 100%);
704.      background: -ms-linear-gradient(top,  #7a8eff 0%,#6175f9 45%,#405fed 100%);
705.      background: linear-gradient(top,  #7a8eff 0%,#6175f9 45%,#405fed 100%);
706.      filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#7a8eff', e
   ndColorstr='#405fed',GradientType=0 );
707.          }
708.          .questionList{
709.      background: #5484ff;
710.      background: -moz-linear-gradient(top,  #5484ff 2%, #0046f9 18%, #3163ed 100%);
711.      background: -webkit-gradient(linear, left top, left bottom, color-
   stop(2%,#5484ff), color-stop(18%,#0046f9), color-stop(100%,#3163ed));
712.      background: -webkit-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed
   100%);
713.      background: -o-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
714.      background: -ms-linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
715.      background: linear-gradient(top,  #5484ff 2%,#0046f9 18%,#3163ed 100%);
716.      background-color: #5484ff;
717.      }
718.
719.              .questionList:hover {
720.                  opacity: 0.6;
721.      filter:alpha(opacity=60);
722.        -webkit-transition: opacity 1s linear;
723.        color:white;
724.
725.       }
726.
727.                .title{background: #060263;
728.      background: -moz-linear-gradient(top,  #060263 0%, #07037f 3%, #1e27a0 15%,
   #0514b7 28%, #0b05b7 57%, #2b06d3 90%, #040e9b 97%, #030d7a 99%);
729.      background: -webkit-gradient(linear, left top, left bottom, color-
   stop(0%,#060263), color-stop(3%,#07037f), color-stop(15%,#1e27a0), color-
   stop(28%,#0514b7), color-stop(57%,#0b05b7), color-stop(90%,#2b06d3), color-
   stop(97%,#040e9b), color-stop(99%,#030d7a));
730.      background: -webkit-linear-gradient(top,  #060263 0%,#07037f 3%,#1e27a0
   15%,#0514b7 28%,#0b05b7 57%,#2b06d3 90%,#040e9b 97%,#030d7a 99%);
731.      background: -o-linear-gradient(top,  #060263 0%,#07037f 3%,#1e27a0 15%,#0514b7
   28%,#0b05b7 57%,#2b06d3 90%,#040e9b 97%,#030d7a 99%);
732.      background: -ms-linear-gradient(top,  #060263 0%,#07037f 3%,#1e27a0
   15%,#0514b7 28%,#0b05b7 57%,#2b06d3 90%,#040e9b 97%,#030d7a 99%);
733.      background: linear-gradient(top,  #060263 0%,#07037f 3%,#1e27a0 15%,#0514b7
   28%,#0b05b7 57%,#2b06d3 90%,#040e9b 97%,#030d7a 99%);
734.      background-color: #2A16AB;
735.
```

```
736.      }
737.
738.      .createAccountWrapper{
739.      background: #708fff;
740.      background: -moz-linear-gradient(top,  #708fff 13%, #ffffff 23%, #ffffff 96%,
   #dde3ff 98%, #708fff 100%);
741.      background: -webkit-gradient(linear, left top, left bottom, color-
   stop(13%,#708fff), color-stop(23%,#ffffff), color-stop(96%,#ffffff), color-
   stop(98%,#dde3ff), color-stop(100%,#708fff));
742.      background: -webkit-linear-gradient(top,  #708fff 13%,#ffffff 23%,#ffffff
   96%,#dde3ff 98%,#708fff 100%);
743.      background: -o-linear-gradient(top,  #708fff 13%,#ffffff 23%,#ffffff
   96%,#dde3ff 98%,#708fff 100%);
744.      background: -ms-linear-gradient(top,  #708fff 13%,#ffffff 23%,#ffffff
   96%,#dde3ff 98%,#708fff 100%);
745.      background: linear-gradient(top,  #708fff 13%,#ffffff 23%,#ffffff 96%,#dde3ff
   98%,#708fff 100%);
746.      background-color: #708fff;
747.
748.
749.
750.      }
751.
752.       /*=====BUTTONS=====*/
753.
754.
755.      .button {
756.           margin: 20px;
757.           min-width: 120px;
758.           width: auto;
759.           height: 30px;
760.           line-height: 30px;
761.           color: white;
762.           text-shadow:0 0 20px black ;
763.           font-size: 18px;
764.           font-family: Georgia, Verdana, Tahoma;
765.           display: block;
766.           text-align: center;
767.           position: relative;
768.           display: inline;
769.           cursor: pointer;
770.      }
771.
772.
773.          .button:active {
774.              -moz-box-shadow: 0 2px 6px black;
775.              -webkit-box-shadow: 0 2px 6px black;
776.          }
777.
778.
779.       .buttonLogin {
780.           width: 70px;
```

```
781.           margin-left: 5px;
782.           color: white;
783.           text-shadow:0 0 20px black ;
784.           font-size: 18px;
785.           font-family: Georgia, Verdana, Tahoma;
786.           text-align: center;
787.           display: inline;
788.           cursor: pointer;
789.
790.       }
791.
792.
793.           .questionList {
794.               width: 120px;
795.               border-radius: 15px;
796.               padding: 1px;
797.               border-style: solid;
798.               border-width: 1px;
799.               color: white;
800.               margin-top: 8px;
801.               margin-left: 20px;
802.               cursor: pointer;
803.
804.                   }
805.
806.            .questionList:active {
807.               -moz-box-shadow: 0 2px 6px black;
808.               -webkit-box-shadow: 0 2px 6px black;
809.           }
810.
811.            .questionListCurrent {
812.       width: 120px;
813.       border-radius: 15px;
814.       background-color: #;
815.       padding: 1px;
816.       border-style: none;
817.       margin-top: 8px;
818.       margin-left: 20px;
819.
820.       background: #bbddff;
821.       background: -moz-linear-gradient(top,  #bbddff 0%, #afd5fb 44%, #9ecaf6 100%);
822.       background: -webkit-gradient(linear, left top, left bottom, color-
    stop(0%,#bbddff), color-stop(44%,#afd5fb), color-stop(100%,#9ecaf6));
823.       background: -webkit-linear-gradient(top,  #bbddff 0%,#afd5fb 44%,#9ecaf6
    100%);
824.       background: -o-linear-gradient(top,  #bbddff 0%,#afd5fb 44%,#9ecaf6 100%);
825.       background: -ms-linear-gradient(top,  #bbddff 0%,#afd5fb 44%,#9ecaf6 100%);
826.       background: linear-gradient(top,  #bbddff 0%,#afd5fb 44%,#9ecaf6 100%);
827.       background-color: #bbddff;
828.
829.                   }
830.           .questionListCurrent:hover {
```

```
831.      opacity: 0.6;
832.      filter:alpha(opacity=60);
833.        -webkit-transition: opacity 1s linear;
834.                }
835.
836.
837.          .questionListCurrent:active {
838.              -moz-box-shadow: 0 2px 6px black;
839.              -webkit-box-shadow: 0 2px 6px black;
840.          }
841.
842.          .divButton{
843.           padding-left: 16px;
844.           padding-right: 16px;
845.           padding-top: 8px;
846.           padding-bottom: 2px;
847.           cursor: pointer;
848.
849.          }
850.       /*=====OTHER INPUTS=====*/
851.       .dropDown{
852.        border-radius: 5px;
853.        margin-left: 5px;
854.        margin-left: 5px;
855.
856.       }
857.
858.
859.
860.       /*=====TEXTBOXES=====*/
861.
862.      .textbox{
863.          display: inline;
864.              border-radius: 8px;
865.              }
866.
867.      .search{ background-image: url(img/mag.png);
868.              background-repeat: no-repeat;
869.              background-position:4%;
870.              padding-left: 20px;
871.              width: 140px;
872.              color: grey;
873.              margin-bottom: 20px;}
874.
875.      .level{
876.          padding-left: 20px;
877.          width: 140px;
878.          color: grey;
879.          margin-bottom: 20px;}
880.
881.
882.
```

```
883.      .textboxQuestions{
884.          display: inline-block;
885.              margin-left: 50px;
886.              padding: 4px;
887.              width: 400px;
888.              background: #E6E6FA;
889.              background: -moz-linear-gradient(top,   #E6E6FA, #FFFFFF);
890.             background: -webkit-gradient(linear,
891.                      left top, left bottom, from(#E6E6FA), to(#FFFFFF));
892.             filter: progid:DXImageTransform.Microsoft.Gradient(
893.
   StartColorStr='#E6E6FA', EndColorStr='#FFFFFF', GradientType=0);
894.
895.      }
896.
897.
898.
899.
900.       /*=====FONTS=====*/
901.
902.      p.titleFont {
903.          font-family:  Georgia, Verdana, Tahoma;
904.          font-size:45px;
905.          text-align:center;}
906.
907.      p.subFont {
908.          font-family:  Georgia, Verdana, Tahoma;
909.          font-size:17px;
910.          text-align:left;
911.          display: inline;
912.          padding: 5px;
913.           }
914.
915.      p.subFontSm {
916.          font-family:  Georgia, Verdana, Tahoma;
917.          font-size:12px;
918.          text-align:left;
919.          display:inline;
920.          text-decoration: none;
921.          }
922.
923.      p.subFontMe {
924.          font-family:  Georgia, Verdana, Tahoma;
925.          font-size:12px;
926.          text-align:left;
927.          display:inline;
928.          text-decoration: underline;
929.          color: white;
930.          }
931.
932.      p.subFontLa {
933.          font-family:  Georgia, Verdana, Tahoma;
```

```
934.          font-size:30px;
935.          text-align:left;
936.          display:inline;
937.          text-decoration: none;
938.          }
939.
940.      p.subFontWh {
941.          font-family:  Georgia, Verdana, Tahoma;
942.          font-size:12px;
943.          text-align:left;
944.          display:inline;
945.          text-decoration: none;
946.          color: white;
947.          }
948.
949.      p.subFontGr {
950.          font-family:  Georgia, Verdana, Tahoma;
951.          color: grey;
952.          font-size:17px;
953.          text-align:left;
954.          display: inline;
955.          padding: 5px;
956.           }
957.
958.      p.subFontP {
959.          font-family:  Georgia, Verdana, Tahoma;
960.          font-size:15px;
961.          text-align:left;
962.          padding-left: 15px;
963.          }
964.
965.           p.subFontMW{
966.          font-family:  Georgia, Verdana, Tahoma;
967.          font-size:20px;
968.          text-align:center;
969.          display: inline;
970.          padding: 5px;
971.          color: white;
972.           }
973.
974.           p.subFontV {
975.          font-family:  Georgia, Verdana, Tahoma;
976.          font-size:14px;
977.          text-align:left;
978.          display: inline;
979.          padding: 5px;
980.           }
981.
982.      p.subFontTiny {
983.          font-family:  Georgia, Verdana, Tahoma;
984.          font-size:9px;
985.          text-align:right;
```

```
986.          display: inline;
987.          padding: 2px;
988.          color: blue;
989.          cursor: pointer;
990.          float:right;
991.           }
992.
993.           p.subFontTiny:hover{
994.               text-decoration: underline;
995.           }
996.
997.
998.       p.subFontScores {
999.          font-family:  Georgia, Verdana, Tahoma;
1000.         font-size:27px;
1001.         text-align:centre;
1002.         display: inline;
1003.         padding: 5px;
1004.         padding-bottom: 7px;
1005.         color: white;
1006.         font-weight: bold;
1007.         padding-left: 65px;
1008.         text-shadow:0px 2px 3px #000552;
1009.
1010.          }
1011.
1012.     p.questionListP {
1013.         font-family:  Georgia, Verdana, Tahoma;
1014.         color: white;
1015.         font-size:17px;
1016.         text-align:left;
1017.         display: inline;
1018.         padding: 5px;
1019.          }
1020.     p.questionListP:hover {
1021.         font-family:  Georgia, Verdana, Tahoma;
1022.         color: snow;
1023.         font-size:17px;
1024.         text-align:left;
1025.         display: inline;
1026.         padding: 5px;
1027.          }
1028.
1029.
1030.     p.correct {
1031.         font-family:  Georgia, Verdana, Tahoma;
1032.         font-size:15px;
1033.         text-align:left;
1034.         display: inline;
1035.         font-weight: bold;
1036.         color: green;
1037.         padding: 5px;
```

```
1038.          }
1039.
1040.     p.incorrect {
1041.         font-family:  Georgia, Verdana, Tahoma;
1042.         font-size:15px;
1043.         text-align:left;
1044.         display: inline;
1045.         font-weight: bold;
1046.         color: red;
1047.         padding: 5px;
1048.          }
1049.
1050.     p.subFontCA{
1051.         font-family:  Georgia, Verdana, Tahoma;
1052.         font-size:40px;
1053.         text-align:center;
1054.         display: inline;
1055.         padding: 5px;
1056.         color: #d4d6ff;
1057.        text-shadow:0px 2px 3px #000552;
1058.        }
1059.
1060.     p.subFontSU {
1061.         font-family:  Georgia, Verdana, Tahoma;
1062.         font-size:17px;
1063.         text-align:left;
1064.         display: inline;
1065.         padding: 5px;
1066.          }
1067.
1068.          .smallGreyText{
1069.           font-family:  Georgia, Verdana, Tahoma;
1070.              font-size:10px;
1071.              text-align: center;
1072.              display: inline;
1073.              color: grey;
1074.              bottom: 5px;
1075.              margin-left: 5px;
1076.              margin-right: 5px;
1077.              position: relative;
1078.              cursor: pointer;
1079.
1080.          }
1081.          .smallGreyText:hover{
1082.              text-decoration: underline;
1083.              bottom: 6px;
1084.              color: black;
1085.              -webkit-transition: color 1s linear;
1086.          }
1087.
1088.       /*=====SCROLLERS=====*/
1089.
```

```
1090.     ::-webkit-scrollbar {
1091.         width: 12px;
1092.     }
1093.
1094.     ::-webkit-scrollbar-track {
1095.         -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.3);
1096.         -webkit-border-radius: 10px;
1097.         border-radius: 10px;
1098.     }
1099.
1100.     ::-webkit-scrollbar-thumb {
1101.         -webkit-border-radius: 10px;
1102.         border-radius: 10px;
1103.         background: #0400ff;
1104.        -webkit-box-shadow: inset 0 0 6px rgba(0,0,0,0.5);
1105.     }
1106.     ::-webkit-scrollbar-thumb:window-inactive {
1107.         background: #0400ff;
1108.     }
1109.
1110.
1111.
1112.
1113.
1114.
1115.
1116.
1117.      /*=====APPRISE=====*/
1118.
1119.          .appriseOverlay
1120.         {
1121.         position:fixed;
1122.         top:0;
1123.         left:0;
1124.         background:rgba(0, 0, 0, 0.3);
1125.         display:none;
1126.         }
1127.     .appriseOuter
1128.         {
1129.         background:#eee;
1130.         border:1px solid #fff;
1131.         box-shadow:0px 3px 7px #333;
1132.         -moz-box-shadow:0px 3px 7px #333;
1133.         -webkit-box-shadow:0px 3px 7px #333;
1134.         -moz-border-radius:4px;
1135.         -webkit-border-radius:4px;
1136.         border-radius:4px;
1137.         -khtml-border-radius:4px;
1138.         position:absolute;
1139.         z-index:99999999;
1140.         min-width:200px;
1141.         min-height:50px;
```

```
1142.          max-width:75%;
1143.          position:fixed;
1144.          display:none;
1145.          }
1146.      .appriseInner
1147.          {
1148.          padding:20px;
1149.          color:#333;
1150.          text-shadow:0px 1px 0px #fff;
1151.          }
1152.      .appriseInner button
1153.          {
1154.          border:1px solid #bbb;
1155.          -moz-border-radius:3px;
1156.          -webkit-border-radius:3px;
1157.          border-radius:3px;
1158.          -khtml-border-radius:3px;
1159.          background: -moz-linear-gradient(100% 100% 90deg, #eee, #d5d5d5);
1160.          background: -webkit-gradient(linear, 0% 0%, 0% 100%, from(#eee),
    to(#d5d5d5));
1161.          background: -webkit-linear-gradient(#eee, #d5d5d5);
1162.         background: -o-linear-gradient(#eee, #d5d5d5);
1163.          color:#232d3d;
1164.          font-size:12px;
1165.          font-weight:bold;
1166.          padding:4px 10px;
1167.          margin:0 3px;
1168.          text-shadow:0px 1px 0px #fff;
1169.          cursor:pointer;
1170.          box-shadow:0px 1px 2px #ccc;
1171.          -moz-box-shadow:0px 1px 2px #ccc;
1172.          -webkit-box-shadow:0px 1px 2px #ccc;
1173.          }
1174.      .appriseInner button:hover
1175.          {
1176.          color:#d85054;
1177.          }
1178.      .aButtons, .aInput
1179.          {
1180.          margin:20px 10px 0px 10px;
1181.          text-align:center;
1182.          }
1183.      .aTextbox
1184.          {
1185.          border:1px solid #aaa;
1186.          -moz-border-radius:4px;
1187.          -webkit-border-radius:4px;
1188.          border-radius:4px;
1189.          -khtml-border-radius:4px;
1190.          box-shadow:0px 1px 0px #fff;
1191.          -moz-box-shadow:0px 1px 0px #fff;
1192.          -webkit-box-shadow:0px 1px 0px #fff;
```

```
1193.          width:180px;
1194.          font-size:12px;
1195.          font-weight:bold;
1196.          padding:5px 10px;
1197.          }
1198.
1199.
1200.
1201.      .ui-timepicker-div .ui-widget-header { margin-bottom: 8px; }
1202.      .ui-timepicker-div dl { text-align: left; }
1203.      .ui-timepicker-div dl dt { height: 25px; margin-bottom: -25px; }
1204.      .ui-timepicker-div dl dd { margin: 0 10px 10px 65px; }
1205.      .ui-timepicker-div td { font-size: 90%; }
1206.      .ui-tpicker-grid-
   label { background: none; border: none; margin: 0; padding: 0; }
1207.
```

Bibliography

Below is a list of all sources of information which I used in my program or was not all my own work:

- jQuery library – I included this library to make my JavaScript quicker to write and more functionable.
- apprise library – I included this library to output messages to the user
- mag.png was coppyed from a free clip art website and modified
- Gradient fills – I used a gradient generator to create the cross-browser gradient fills, because it was considerably quicker and easier.
- I had help on part of start.php and logIn.php