Physics Quiz Software

Project Report

Contents:

* Analysis of Problem – Page 1
* Designing the Program – Page 4
* Testing – Page 9
* System Maintenance – Page 11

Appendices:

* 1. Initial Survey
  + 1.5. Survey Results
* 2. Source Code
* 3. Beta Testing Form
  + 3.5 Beta Feedback
* 4. User Manual
* 5. Feedback Survey

**Analysis of Problem**

The problem that I am addressing is the lack of online revision resources for AS Physics, especially interactive ones. At GCSE level, there are many revision tools in place, such as GCSE bitesize, but at A Level, I think there is an inadequate provision. Providing a tool will definitely help other taking AS Physics, and provide a more fun way to revise.

At the moment, the majority of physics revision materials consist of past papers. While this is a very effective method of revision, I feel that there is little to learn from this in terms of pointing out which areas need revision. This is why I would like to include in the tool a way of feeding back to the user what they can go over to improve their knowledge. In terms of content, I think that past exam questions are suitable, especially the multiple choice part of the paper. In order to use these questions effectively, I will need to create a database with all past paper questions in it, and have fields related to what topic they are from within the course and what could be revised if the user got that particular question wrong. The problem could be that the database is too large for the program to be distributed appropriately. This is something I would need to investigate.

The users would obviously be those taking AS Physics, however I could extend the program once completed and produce a version for A2 Physics or maybe even other subjects. This small user group means that I can have very specific features to accommodate to their needs. The program would need to be relatively easy for them to access both at home and at school, so I could implement it in a web page – maybe as part of the school website. The user interface would need to be very simple, as I do not know the level of competence with IT of the user. The critical part is the feedback. The user needs to know how they have done, and what they got wrong. This way they can focus their revision once they have used the tool, and possibly it could hold their scores, and show how they have improved.

I did a survey in the AS Physics class (Appendix 1), asking questions on what they would want from the revision tool. The most popular features were that there would be a mixture of different types of questions, however, there is no way that the program would be able to mark a worded question, as there are many ways of phrasing things. Therefore I will only be using multiple choice and calculation questions. They also preferred the questions to come from past papers, which means the inclusion of a database with all the questions on. The students also were very keen on having a form of feedback, so as well as showing their scores, I could include a database that updates every time the student uses the software with their scores, so they can access it and track their progress. The favoured distribution of the software was to be given to each student digitally for their home PC, possibly shared using a service such as dropbox or Google drive. The problem here is that the student would need to be competent at using that service and also, since I plan on developing the program in Java, they would need to be able to run a .jar file which requires an installation of the Java runtime environment. In the documentation I will have to include installation instructions for the Java runtime environment as well as instructions on how to run the program. A popular suggestion for additional features was a glossary with key terms and formulae for each topic. The feedback on how effective as a revision aid this would be was very positive, leading me to believe that this is a relevant and useful tool to be developing.

I have decided to create the following specification for the program based on the survey results:

* A quiz type program.
* Mixtures of multiple choice and calculation questions.
* Questions relating to topics chosen by the student.
* Feedback given to the students, in the form of a score and what the student needs to work on.
* Record of previous scores in each topic on a database.
* A formula sheet provided, possibly a pdf file stored in the program directory.
* Easy to use UI that can be displayed smoothly on any system.
* The ability to request and fetch the information from the database promptly.
* Installation instructions for JRE.

The main source of data will be the collection of Edexcel Physics AS Papers available on their website. I will store these in a database, in the form of 2 tables, Question and Answer. Papers that rely on diagrams and images will have to be omitted.

Here is a basic data flow diagram showing the structure the program will take:

Student

Selection of topic to revise

Program reads students selection

Database of questions and answers

Questions & answers

Request for questions and answers in the topic

Program stores questions & answers as variables and checks to see if the student is correct

Student

Questions

Answers

Student’s score, grade and areas to revise

Here is the Entity-Relationship diagram for my database. It is a simple database with 2 tables to avoid any errors and complications with the data.

Answer

Question

In the case of a multiple choice question, I will store all answers in the Answers table, leading to an ER diagram as such:

Question

Answer

I think that overall, this tool will be successful and useful for Physics students, because often an interactive approach to revision is the most useful, especially with a feedback element. This project is definitely feasible, and I can see it being a well-used tool for future AS Students. It also has the potential to be expanded for other subjects, such as A2 Physics and other exam based courses.

**Designing the program**

Before beginning the development of the program, I need to design some algorithms and pseudo code.

Here is a basic flow chart outlining the program structure:

GUI Components

Process User input to request from Database

User chosen topic

Selected from menu

Process Data from Database into Questions on GUI Frame

Fetch appropriate questions and answers

Database

Multiple Choice

Checkboxgroup

Calculation or Multiple Choice?

Calculation

Text Field

Start

Is answer correct?

Check Answer

Yes

No

Show which questions were wrong and what to revise

Display total marks

No

Is this the last question?

This needs to be revised

Add marks to score

Yes

End

Some of the main modules of this program are the main menu, where the user chooses a topic to revise, the quiz, where the user is asked questions, and the feedback, where the program show the user’s score and where they need to improve.

In this program, some of the key algorithms are centred on the program interfacing with the database. This will be done using PHP incorporated in the java code. One such algorithm is getting the questions from the database according to topic. To represent the programs, they will have to be used as classes and methods, similarly to object oriented programming.

e.g.

Class Quiz

User inputs topic x

Method Test

Query: All questions with topic x

New Question = question1

New Question = question2

Etc.

Calculation:

Label = question

TextField = user’s answer

Multiple Choice:

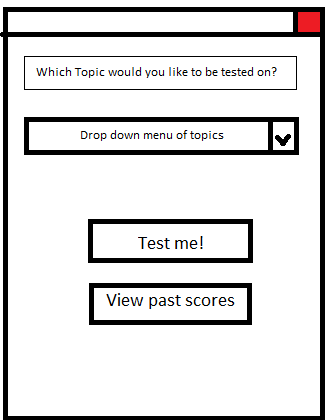
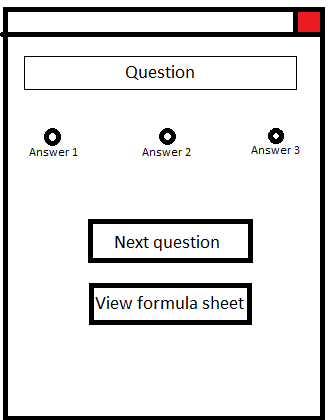
Label = question

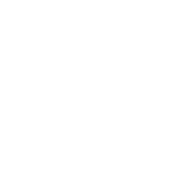
Checkboxgroup = user’s answer

It will return an answer to be checked with the mark scheme.

The user interface is an important part of the design as it enables the user to use the program efficiently and with ease.

Here are the initial designs for the UI.



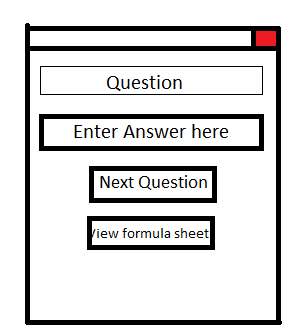


CheckboxGroup

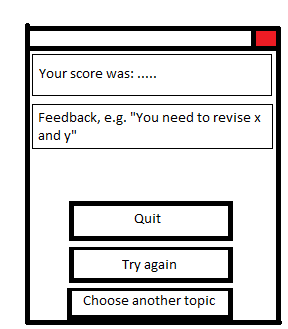
Will open a pdf with the formula sheet

Main menu, where the user chooses their topic

Multiple choice question layout



Calculation question layout



Goes back to menu

Will re-run the test

Closes application

Once the test has finished

One of the main parts of developing this program will be creating a database with all the questions in it. My database will consist of two tables, a questions table and an answers table.

Here is what the database will be like:

Question Answer

|  |
| --- |
| QuestionID |
| QuestionNo |
| QuestionSubNo |
| PaperDate |
| QuestionContent |
| QuestionTopic |
| QuestionMarks |
| QuestionType |

|  |
| --- |
| AnswerID |
| AnswerContent |
| AnswerOption1 |
| AnswerOption2 |
| AnswerOption3 |
| AnswerEquation |
| AnswerUnit |

An example of an SQL query that will be used is:

SELECT QuestionContent, FROM Question

WHERE QuestionTopic = “Mechanics”

This will select the suitable questions for that topic. Then, 10 would be chosen at random to use in the quiz.

When testing the software, I will need to have two stages of testing. The first will be white-box testing, where I will ensure every possible input is made into the program.

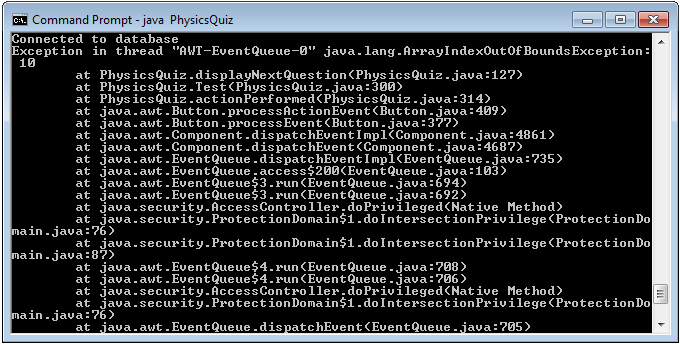
The second will be beta testing. I will give copies of the program to some users who will use and record their experience so I can make some final changes.

In terms of security, this software does not hold any sensitive data so does not need that type of security. However, there will need to be something to stop the user changing the code of the program, simply so that it can perform its task even if the user attempts to tamper with it. The way I will stop this is to compile the program into a .jar file. This should prevent the user from accessing the code of the program.

Testing

To test the program, I am going to put in various possible inputs to the program and see if the outputs match what is expected. I will also list any possible fixes, and whether or not I have implemented them.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Expected Output/Response** | **Actual Output/Response** | **Possible fix** | **Fix implemented?** |
| Any topic, all wrong answers | Score of 0, scores file reads 0. | Expected |  |  |
| Any topic, Calculation questions have equations in wrong format | Marked as incorrect | Expected | Explain correct answer syntax in text field so user avoids this. | Yes |
| Any topic, leave all answers blank | All Calculation questions marked incorrect. Some Multiple Choice questions are correct. | Expected |  |  |
| Any topic, all correct answers | Given 10 points. Score saved in text file as 10. | Expected |  |  |
| Do not change name field | Name stored as “Enter your name” | Expected |  |  |
| Blank name field | Name stored as blank | Expected | If loop to save name as “Student” | Yes |
| Run test, back to menu, click test again | Runs test again. | “ArrayIndexOutOfBoundsException”  (See screenshot below) | Reset variable questionNo to 0 upon test starting | Yes |



Now that this testing stage is complete, I have distributed copies of the program to be beta tested by some of my peers. They will give feedback (Appendix 3) on any problems they encountered and any improvements that they feel need to be made.

The feedback from the Beta Testers suggested a few changes that could be made to improve the usability of the program. Unfortunately, only 1 of the 5 testers provided feedback (Appendix 3.5). The first suggestion was that there should be a button to go back to the menu once you have started the test. This is easily implemented and has been added to the program. Another issue encountered was that often text would go off the screen, and resizing the window caused overlapping issues. Unfortunately, I was unable to find a way to fix this issue, and it remains to be fixed. Another main issue with the program is that there are only more than 10 questions on Mechanics in the database, so other topics were unable to be quizzed. This is due to having not enough time to finish the database before the deadline.

Having done these tests, I am confident that the program is ready, as all inputs work as they should do. I am now ready to distribute the program to the end users.

**System Maintenance**

System Overview:

The system is made up of 2 main parts. The first is the executable jar file called *PhysicsQuiz*. This is what the user will be interacting with. The second is the web server application. This will run a server locally on the host PC that has the database needed for the program.

Language:

The software has been programmed in Java 7, and the database is a MySQL database.

Third Party Software:

The web server used is *UsbWebServer*, developed by Border-IT. The database file is located in WebServer\mysql\data\pastpapers. To access and edit the database, simply run the usbwebserver.exe and click on phpmyadmin. This will open up a page in your default web browser where you can log in to the database and edit the data, if necessary.

In the .jar file, alongside the main classes are the classes from mysql-connector-java-5.1.34-bin.jar, which are the driver classes for the jdbc specific to mysql. These ensure that the program can connect to the database.

Structure of the code:

The code consists of 3 java class files. The main file is *PhysicsQuiz*.*class*, and this has 8 main methods. The first, *PhysicsQuiz* is simply the start, and only serves to run the *mainMenu* method which is what displays the menu the user first sees, and links to all the other methods through the *actionPerformed* method. The *actionPerformed* method will look at what button is being pressed and respond appropriately.

The methods *Test*, *getTopicQuestions* and *getTopicAnswers* each get the questions and answers from the database and store them in arrays. *displayNextQuestion* is what takes the questions from the array and displays them for the user to see, and moves to the next when the user clicks next. Answers and Results are what make up the feedback par toff the software. *Answers* compares the user’s answers to the answers from the database and will total a score. *Results* displays the results and the incorrect answers.

List of variables:

This is the list of variable names and their purpose in the code:

*questionNo* – An integer used for identifying questions in the array.

*score* – The integer that indicates how many questions the user got correct.

*random* – A random integer used to select questions at random for the test.

*multirandom ­*– A random integer used to randomly arrange the multiple choice answers.

*username* – A string that the user inputs for their score to be saved under in the text file.

*content –* A string that is made up of the content of the question.

*marks* – A string made up of the number of marks the question is worth.

*type* – A string that determines the type of the question, either multiple choice or calculation.

*id* – An integer that is unique to each question and corresponding multiple choice answer. It is used to select specific ones easily.

*answer1, answer2, answer3* – 3 Strings that are all alternate answers ot a multiple choice question.

**Evaluation**

When analysing the initial problem, I came up with a number of requirements that this software had to have. I will be evaluation whether each one was met and if not, why.

The first was that it would be a quiz program. This was my primary focus when thinking about the User Interface, and it has been met. The second was that it would have a mixture of calculation and multiple choice questions. While I have met this, I have not fully filled each question topic in the database (as the beta testing pointed out), and there is potential to add more questions to it in future. I also stated that the student needed to be able to choose a topic to be quizzed on. This is the main feature of the menu in the software. The fourth requirement was that feedback was given to the student, in the form of a score and what to revise. The software displays a score and which questions were incorrectly answered, but does not have a specific revision focus other than the topic. I had stated that previous scores should be recorded on a database. I felt that a database was a little too complex for simply storing the scores, so they are now recorded in a text file according to topic. Another requirement was a formula sheet attached to the software. I have provided this in a .pdf in the folder the software is sent out in. The UI has been designed so that the buttons are as big as possible and it is layed out appropriately. One issue is that sometimes longer questions will be too long to display on the window. I have not found a fix for this. The provision of the UsbWebServer allows the data from the database to be stored and requested quickly and it is sufficient for the low number of users that there are. My final requirement was that there were installation instructions for JRE. This is included in the attached User Manual.

An important aspect of the program to consider is usability. While there are a few minor problems such as text going off the screen, a competent user would be able to use this software easily. Since my end users are all A Level Physics students, I am confident that they will be able to use this software easily, and that it will help them sufficiently.

One issue with this software is that it will be difficult to update. To do this, the whole .jar package would need to be recompiled, and the new one replace the old one. While new .jar files can easily be sent out to users, it is up to them to be able to successfully replace it with the old one.

To find out is I need to send an update, I will give a survey to the users to fill out and send back, with any issues they have, so I can see if it is necessary to update it.