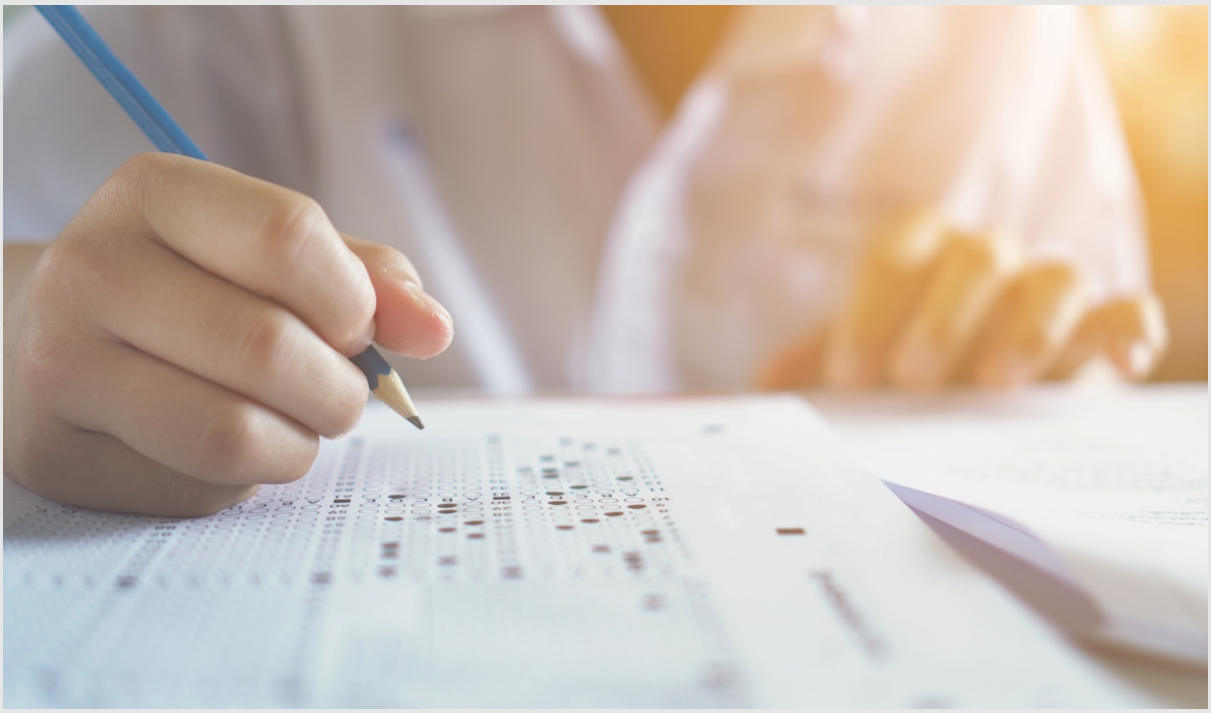
AQA A-Level Computer Science (7517) – Non-Exam Assessment (2023)

EXAM CREATION AND MANAGEMENT



Faizan Arshad

Candidate Number: 6018
Centre Number: 13327

Table of Contents

Analysis

Background to the project

Ilford County High School (ICHS) is a selective grammar school for boys located in Barkingside, East London. Established in 1901, ICHS educates 1260 students between the ages of 11 and 18, with students entering in years 7 and 12, and leaving in years 11 and 13. Admission into year 7 is conducted through an admissions test known as the 11+, which is administered by the London Borough of Redbridge. The school offers external exams at GCSE, AS and A-Level.

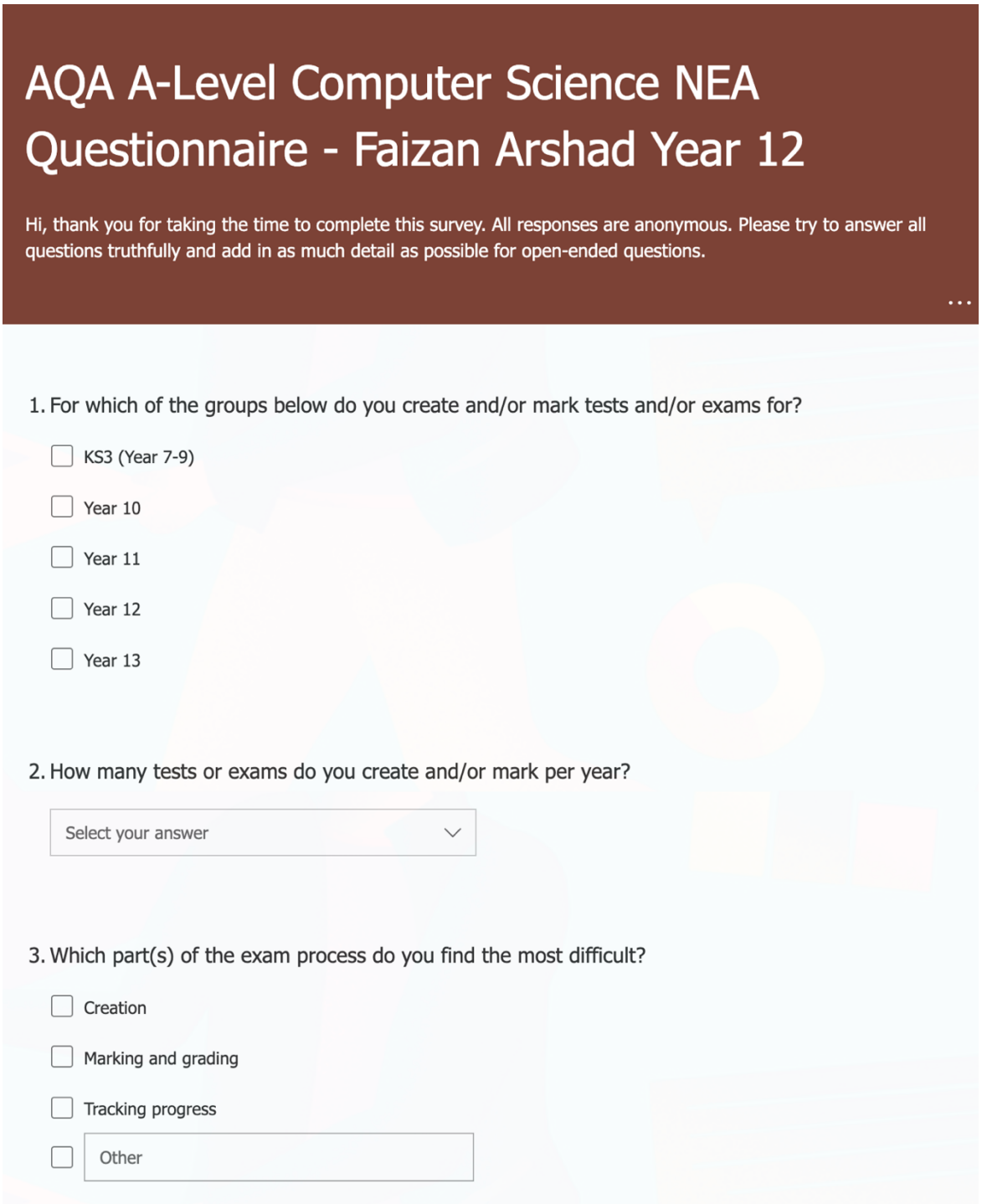
The ICHS Mathematics department is one of the largest departments in the school. With 7 specialist teachers that teach all students in the lower-school and majority of students in the sixth form, managing teaching and assessment is extremely important so that the department is productive. Assessment is conducted throughout the year for all year groups through common assessment tests (CATs) and mock exams. Questions are usually taken from past external exams, however this has led to students predicting questions and memorising mark schemes to get full marks, which is leading to students not revising properly and achieving lower grades when unfamiliar questions are placed in front of them. In addition, teachers have many tests and exams to mark and grade, and managing the workload is often difficult and can lead to fatigue, a drop in lesson quality and students receiving exam feedback later than expected.

Therefore, I have decided to create an application that will allow teachers in this department to easily and quickly manage tests and examinations. This will include the creation of exam papers using questions from multiple sources and exam boards, mark scheme collation, and saving marks into a database to allow teachers to easily track progress and generate current and future grades.

Research of prospective users

To collect more information about how my client's needs and wants, I need to do some research. I have decided to do this using a questionnaire and an interview, since this is the best way to collect analysable and informative data, in both a qualitative and quantitative format.

My questionnaire is hosted on Microsoft Forms and has a combination of both closed and open-ended questions. It is important that my research is relevant to my end user(s), and so all the respondents of the questionnaire are teachers at ICHS, with the majority being mathematics teachers. Full responses for the questionnaire can be viewed in Appendix 1.



AQA A-Level Computer Science NEA Questionnaire - Faizan Arshad Year 12

Hi, thank you for taking the time to complete this survey. All responses are anonymous. Please try to answer all questions truthfully and add in as much detail as possible for open-ended questions.

1. For which of the groups below do you create and/or mark tests and/or exams for?

- [ ] KS3 (Year 7-9)
- [ ] Year 10
- [ ] Year 11
- [ ] Year 12
- [ ] Year 13

2. How many tests or exams do you create and/or mark per year?

Select your answer

3. Which part(s) of the exam process do you find the most difficult?

- [ ] Creation
- [ ] Marking and grading
- [ ] Tracking progress
- [ ] Other

4. Imagine you are creating a bespoke software application to help you with exam management. Which part(s) of the exam process would you want this application to focus on?

- [ ] Creation
- [ ] Marking and grading
- [ ] Tracking progress
- [ ] Other

5. When creating tests and/or exams, which of these sources would you prefer to get questions from? (assume obtaining questions from all options are the same in terms of difficulty)

- ◯ The awarding exam board (including old specification questions)
- ◯ A range of exam boards and qualifications of the same level
- ◯ Both of these
- ◯ Other

6. What features would you like a bespoke exam creation tool to include? (give as much detail as possible)

Enter your answer

7. When marking exams, how do you prefer to store students' grades?

- ◯ Digitally
- ◯ On paper
- ◯ Other

8. What features would you like a bespoke exam marking tool to include? This includes the actual process of marking exams, storing the marks that students have achieved and generating grades. (give as much detail as possible)

Enter your answer

8. What features would you like a bespoke exam marking tool to include? This includes the actual process of marking exams, storing the marks that students have achieved and generating grades. (give as much detail as possible)

Enter your answer

9. What kind of insights would you find useful when tracking the progress of students in exams and tests?

- [ ] Colour coded tables
- [ ] Bar graphs
- [ ] Line graphs
- [ ] Comparisons to class average
- [ ] Comparisons to historical data
- [ ] Future grade predictions
- [ ] Other

10. What features would you like a bespoke exam tracking tool to include? (give as much detail as possible)

Enter your answer

11. Any other things that you want mention?

Enter your answer

Submit

Never give out your password. Report abuse

I have decided to base my interview around the current system that mathematics department uses to manage exams and find out what improvements could be made. I carefully selected the interview questions so that I can collect the most useful and informative data for my particular use case.
Interview questions:

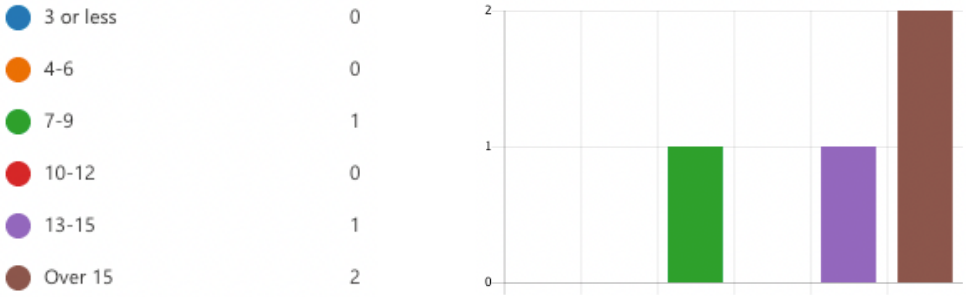
1. What is the current system used to create exams and tests?
2. Are there any issues with this system?
3. What would you like a tailor-made exam creation system to include?
4. Using questions from many exam boards and past papers encourages students to think about how they approach answering questions rather than memorising mark schemes and prevents students from cheating. Are you open to using a tool which has relevant questions from many exam boards? If so, what prevents you from doing so? If not, then why?
5. If a bespoke software application for exam creation was made, which education stage would it be most useful for: KS3, GCSE or A-Level?
6. What is the current system used to mark exams and tests?
7. Are there any issues with this system?
8. Are you open to storing students' marks and grades in a database of an easy-to-use software application (there will always be an option to export to Excel)?
9. What is the current system used to track the progress of students?
10. Are there any issues with this system?
11. Has the department ever considered using computer-based visualisations or AI in order to aid student tracking and predictions?
12. Finally, are there any standout features that you would deem necessary if you were to use a bespoke software application for exam management?

Questionnaire

1. For which groups below do you create and/or mark tests and/or exams for?

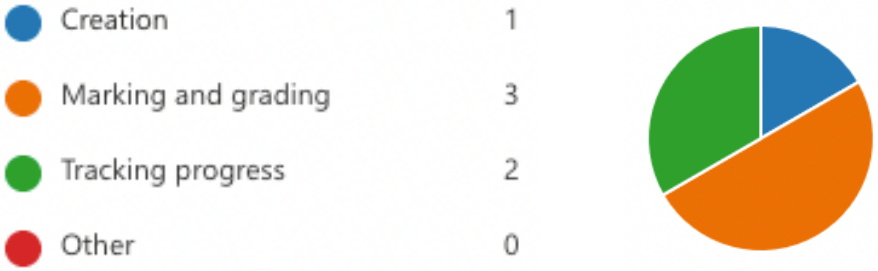
The purpose of this question was just for me to have an idea about the type teachers who are answering the questionaire and make sure that the responses are not biased. As seen above, the distribution of years is almost exactly evenly distributed, except for KS3, who do not sit external exams, so this will not affect the outcome of my results.

2. How many tests or exams do you create and/or mark per year?



This question, again, tells us about the type of teachers who answer the questionnaire. I want to survey teachers who have a lot of experience creating exams and know what kind of features would be helpful in an exam creation management solution. Majority of the teachers who answered this survey create and/or mark over 15 exams per year, with all creating and/or marking at least 7 exams. This is sufficient criteria to ensure that all the teachers who are answered are experienced in creating and/or marking exams.

3. Which part(s) of the exam process do you find the most difficult?



Overall, teachers found marking and grading the most difficult, followed by tracking progress and finally creation. This tells us that there are fundamental problems, difficulties, and flaws with the current system in all the aspects of the exam process. An all-inclusive bespoke software solution will help to address these difficulties and issues. This question suggests I should focus on the grading and tracking part of the solution, since creation is not really an issue. My software solution for exam creation will be unique in the fact that it will contain questions from multiple exam boards (see question 5) and so this will be a complement alongside existing software solutions, rather than a replacement.

4. Imagine you are creating a bespoke software application to help you with exam management. Which part(s) of the exam process would you want this application to focus on?

This is a more specific question to find out what part of the exam process a software solution would be most helpful for. All the answers are almost evenly distributed between

the parts of the exam process, with each part having at least 2 votes. This tells me that it would be useful to create a software solution that covers all three parts more generally, rather than focusing on one area. Tracking progress has slightly more votes, meaning it is important to have relevant visual insights to allow teachers to track the progress of their students.

5. When creating tests and/or exams, which of these sources would you prefer to get questions from? (assume obtaining questions from all options are the same in terms of difficulty)

This question helps to tell me what type of questions would be useful to include in an exam creation tool. All the respondents chose both questions from the awarding exam board and questions from a range of exam boards and qualifications. This mixture helps to strike the balance between questions that are in the style that students will face in the exam and questions that are still relevant content-wise but are presented in a different style and require a different way of thinking and/or applying the knowledge in a different way. I will make sure to include both types of questions in my final solution.

6. What features would you like a bespoke exam creation tool to include? (give as much detail as possible)

Responses

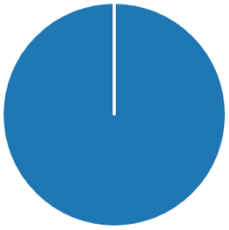
The grade for the overall paper, once the paper is completed
Facility to create exams from a database of questions at any given level
N/A
A range of different question (Multiple choice questions/practical style questions/short and long style questions and mark schemes. Examiner reports where students have made common mistakes

This was the first open-response question, and it is a very broad and open question, simply about what features would be useful in an exam creation tool. I wanted this question to be open to interpretation so that the respondents can think of their own ideas about what might be useful, potentially opening options that I previously did not consider. The first response talks about the grade of the paper. I will make sure that all papers generated will

have grade boundaries attached to them alongside the mark scheme and examiners' report. The second response suggests using a database of questions for multiple levels. This is a great way to be able to store and easily access questions from within my program, and I will make sure a relational database is used to store questions from multiple exam boards and multiple levels. The final response asks for a range of different types of questions, and examiners' reports to be available. As previously mentioned, every paper will be generated with an examiner's report. I will also attempt to allow the mark scheme and examiner report for each question to be viewable when previewed. I will also allow filters to be able to look for different types of questions.

7. When marking exams, how do you prefer to store students' grades?

Digitally	4
On paper	0
Other	0



This question was to find out more about how teachers store their marking and if a digitised solution would be suitable in their system. All the respondents said that they prefer to store their students' grades digitally and so I know that I can implement a storage solution into my program. I will need to use a multi-table database and suitably query this database so that teachers can both store and access the information from within the program.

8. What features would you like a bespoke exam marking tool to include? This includes the actual process of marking exams, storing the marks that students have achieved and generating grades (give as much detail as possible).

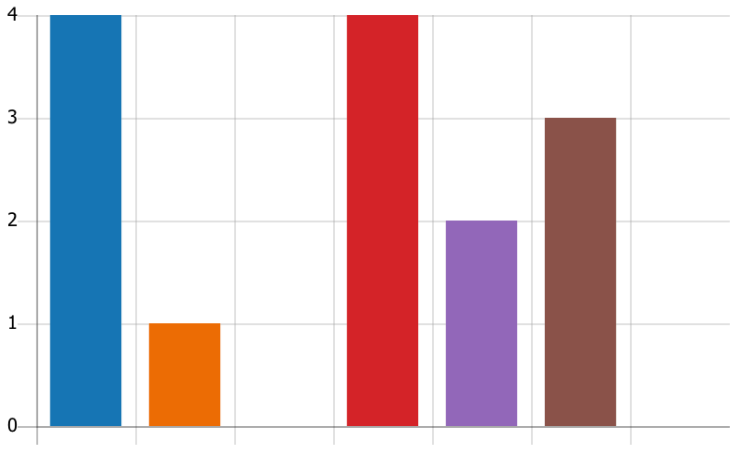
Responses

marks but also a Grade that they got on the paper feedback to students on each question.
1 - Database to store student results linked to a speadsheet for ease of analysis 2 - Ability to analyse a group's performance by question (i.e. which students scored maximum marks for any question/which students scored no marks for any question
Being able to annotate the students' working out
Spreadsheet holding scores/grades/generating targets for improvements

This was another open-ended question about the marking of exams and what features would be useful in a marking-oriented software solution. The first response talks about having marks and a grade viewable. I will make this possible by storing the marks in a database and querying the database. I will also have the grade boundaries for each level available so that they can be used to find the grade of the student. I will also use question-level analysis to be able to get feedback for students. The second response asks for the ability to export marks stored in the database onto a spreadsheet, which I will try to implement, as well as the ability to analyse the group's performance on a certain question, which I will again make possible by querying the database using parameters in the form of the question. The third response comments on the ability to annotate students' work, which is not relevant to my solution since it focuses on storing and analysing the marks. In hindsight, I should have made this clearer in the question. Finally, the last response further specifies that any exported spreadsheets should include the scores (marks) and the grade, as well as improvements where necessary. I will make sure to implement all of this.

9. What kind of insights would you find useful when tracking the progress of students in exams and tests?

Colour coded tables	4
Bar graphs	1
Line graphs	0
Comparisons to class average	4
Comparisons to historical data	2
Future grade predictions	3
Other	0



This is a more practical question asking what sorts of insights would be useful for tracking the progress of students. All respondents agreed that colour-coded tables and comparisons to the class average would be useful, so I am going to make sure that include these in the programmed solution. The next most voted were, in order, future grade predictions, comparisons to historical data and bar graphs. I am going to try to prioritise these aspects where possible.

10. What features would you like a bespoke exam tracking tool to include? (give as much detail as possible)

Responses
track progress and be able to suggest topics or work that has to be undertaken to improve.
Track student performance from Y7, colour coded
Be able to export data.
A system generating progress targets or steps to improve from question to question analysis

This was the final open-ended question about the tracking of exams and what features would be useful in a progress-oriented software solution. The first response talks about suggesting topics for improvement based on exam performance. This can be done by looking at what questions the student lost marks on and, by using the metadata for each question, recommending revision of the corresponding topics. I will make sure to include this in my solution. The second response talks about tracking student performance using colour coding. This has been discussed in the previous question (see question 9) and will be implemented. The third response asks for the ability to export data. Again, this has also been discussed in a previous question (see question 8). The final response comments on generating a target for improvement and question-level analysis, which again will be included.

11. Any other things that you want to mention?

There were no responses to this question

Interview

Comments in red

1. What is the current system used to create exams and tests?

"examWizard" software from Edexcel. *See pages 16-17*

2. Are there any issues with this system?

- Some questions are repeated
- It does not mark any work electronically

3. What would you like a tailor-made exam creation system to include?

- Questions linked with GCSE/A level grade and a GCSE/A level grade given for the overall paper according to the questions selected. *Also discussed in questionnaire*
- Students' work marked using the software *This goes beyond the scope of my project but could be used as a further addition*

4. Using questions from many exam boards and past papers encourages students to think about how they approach answering questions rather than memorising mark schemes and prevents students from cheating. Are you open to using a tool which has relevant questions from many exam boards? If so, what prevents you from doing so? If not, then why?

- Yes, I'll be happy to try the software that collates the questions from different boards
- It is expensive to buy software from different exam boards *This is one of the reasons why a software solution that has questions from various exam boards is necessary*

5. If a bespoke software application for exam creation was made, which education stage would it be most useful for: KS3, GCSE or A-Level?

A level, followed by GCSE. *I will focus on implementing A-Level questions and add some GCSE ones*

6. What is the current system used to mark exams and tests?

- We currently use Mathswatch for KS3 and KS4 to set homework and intervention work. The work set are mark electronically. We do not have anything that will mark exam. Mathswatch does not do A levels. *This is a different software that is used for homework and tasks done online, rather than an exam software*
- We can only build/amend a paper using "examWizard" *This has many cons (see page 17)*

7. Are there any issues with this system?

For Mathswatch sometimes, there is a certain precise way that the answer has to be written to get full marks. *Using proper mark schemes is important, just giving the answer is not sufficient*

8. Are you open to storing students' marks and grades in a database of an easy-to-use software application (there will always be an option to export to Excel)?

Yes.

9. What is the current system used to track the progress of students?

Excel. *Exporting marks to excel will allow the software to work seamlessly with the current system*

10. Are there any issues with this system?

Analysing data has to be done manually. Insights in the software will help resolve this

11. Has the department ever considered using computer-based visualisations or AI in order to aid student tracking and predictions?

No, but I am open to see what is available out there.

12. Finally, are there any standout features that you would deem necessary if you were to use a bespoke software application for exam management?

Questions should display:

- The exam board
- From which year it was taken and which question it was on the paper
- Mark pupils' work and give a grade
- Help teachers to quickly analyse the results and performance of students

Analysis

I have now gained some key insights into what kind of solution would be the most useful to teachers through my research. I have found out that exam creation, marking and tracking are all equally as important to teachers and an all-in-one bespoke solution for this purpose will be the most beneficial. I have also learnt more about the current solution used by the school and how my software solution will optimise this system and make the exam process simpler for teachers. For exam creation, a wide range of questions from different exam boards and specifications will be available and filtering of questions, as well as grade boundaries for the paper, are key features that should be implemented. When marking, saving the marks digitally is preferred, and marks should be easily accessible. In addition, analysis of student performance up to a question level and feedback generation should be

available. Finally, when tracking the progress of students, colour-coded tables and comparisons to the class average are the most useful insights. Suggestions of topics for revision or work to do is essential, and alongside this feedback should be helpful insights to allow for easy identification for top-level students in addition to those who may need some more guidance in the form of intervention or extra work/lessons.
All of this will be very helpful when forming my objectives and gives me a clearer picture of what my solution will look like and what it may entail. However, before I can get to this stage, I need to do some modelling. This requires me to analyse current existing solutions to help give me an idea of what my system will look like. I will then need to storyboard my idea to give it a clear UI design and help with the scripting of the pages. Finally, I will need to create a data-flow diagram so that I can identify how information currently flows around the system of exam management, and how this new system will be improving this information flow, as well as bringing in new data into the system.

Existing solutions

There are many software systems and solutions currently available to teachers that allow them to create and/or manage exams and test papers. I will be investigating further into the solutions offered by the 4 exam boards that offer GCSE, AS and A-Level qualifications in England, since these are the most relevant to my end user. Every exam board has their own software application that they offer to help teachers to create exams, and some offer more in-depth exam management alongside this, including analysis of marks and progress over multiple exams, and even allowing for report generation. I will be looking at each of the current solutions and evaluating the advantages and disadvantages of each, including what features will be useful to include in my solution.

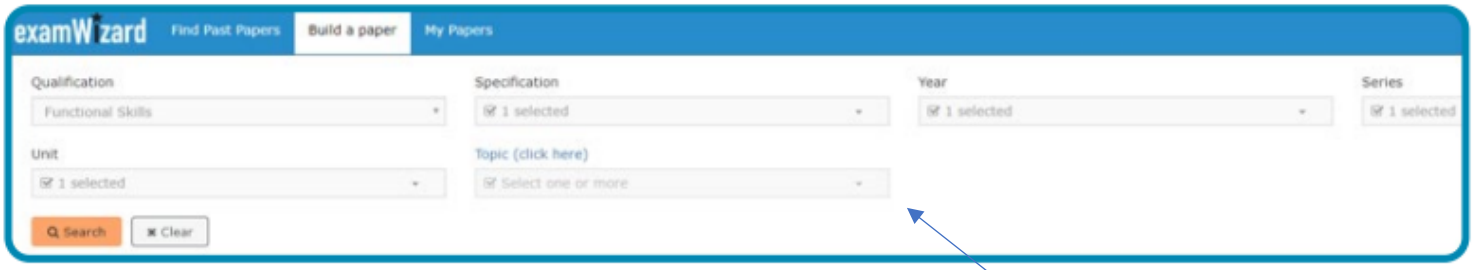
Edexcel

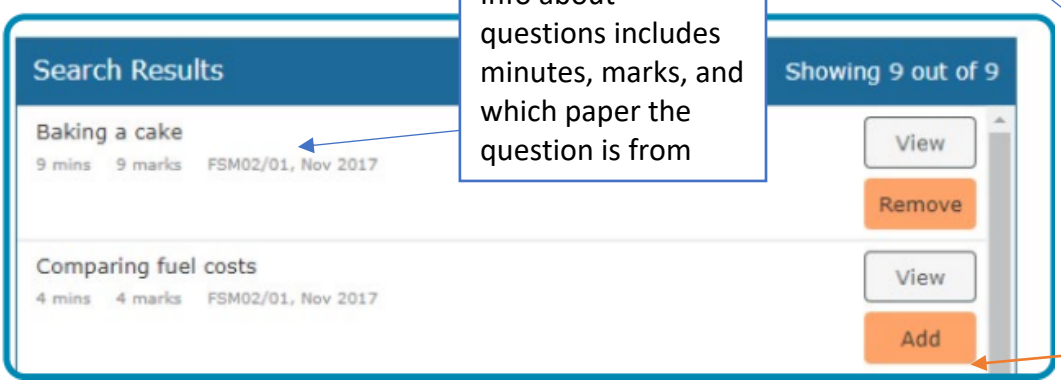
As of the time of writing, Edexcel is the exam board that the Ilford County Mathematics Department offer all external exams with, including GCSE, AS and A-Level. As a result, the school currently uses the software they offer to create some of the exams. Edexcel, being the only privately owned examination board in the UK, has the most extensive platform of resources for teachers to use.

Edexcel offers 2 different software solutions, “examWizard” and “ResultsPlus”.

examWizard

“examWizard is a free exam preparation tool containing a bank of past Edexcel exam questions, mark schemes and examiners’ reports for a range of GCSE, GCE, Functional Skills subjects & BTEC sectors.” (examwizard.co.uk). Created by Pearson plc, a British multinational publishing and education company, that has wholly owned Edexcel since 2005, the software aims to save time by enabling teachers to create their own mock exams, topic tests, homework, or revision activities in minutes, linking questions directly to their associated examiner reports and mark schemes. The software is available for all general qualification subjects that the exam board offers, as well as BTEC, Functional Skills and T Level qualifications.

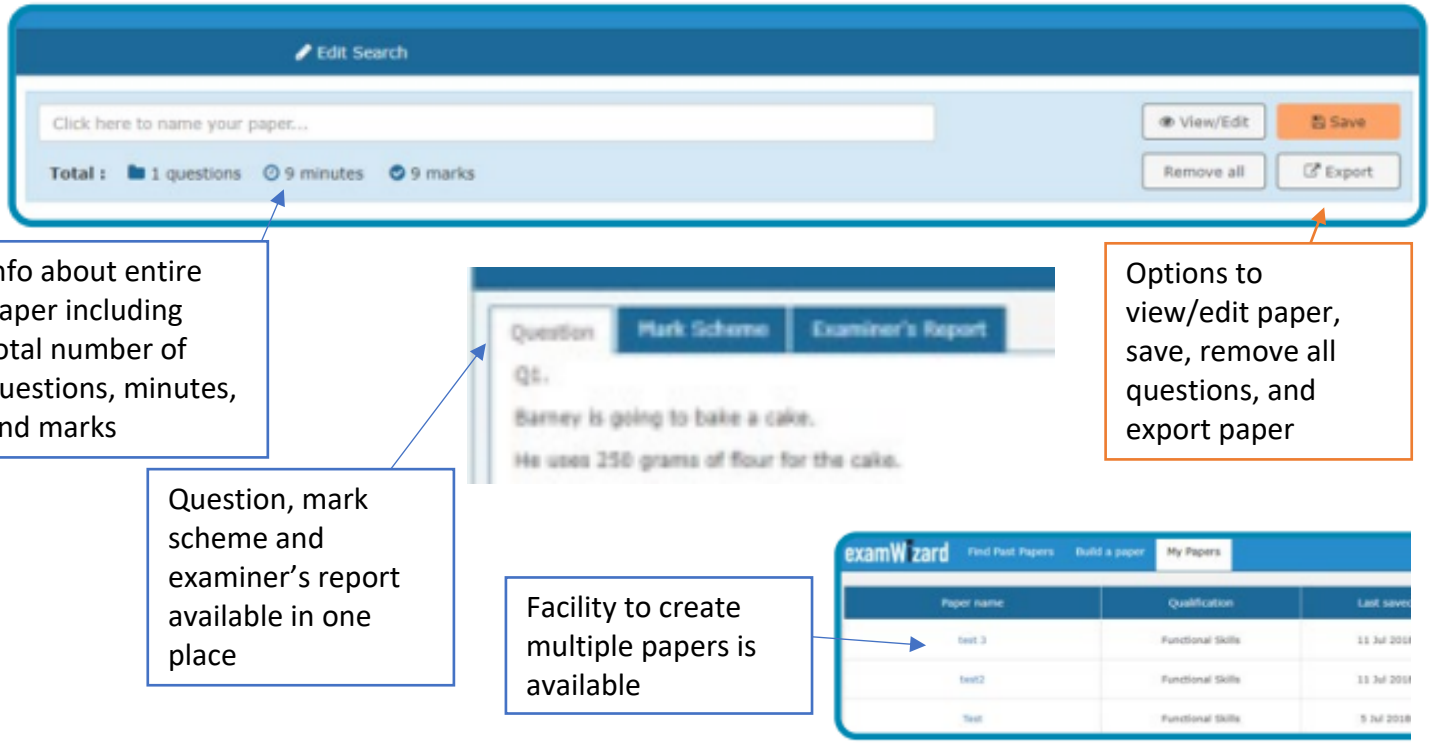




Info about questions includes minutes, marks, and which paper the question is from

There is a wide variety filters available when searching for questions

Large clear buttons to add, remove and view questions

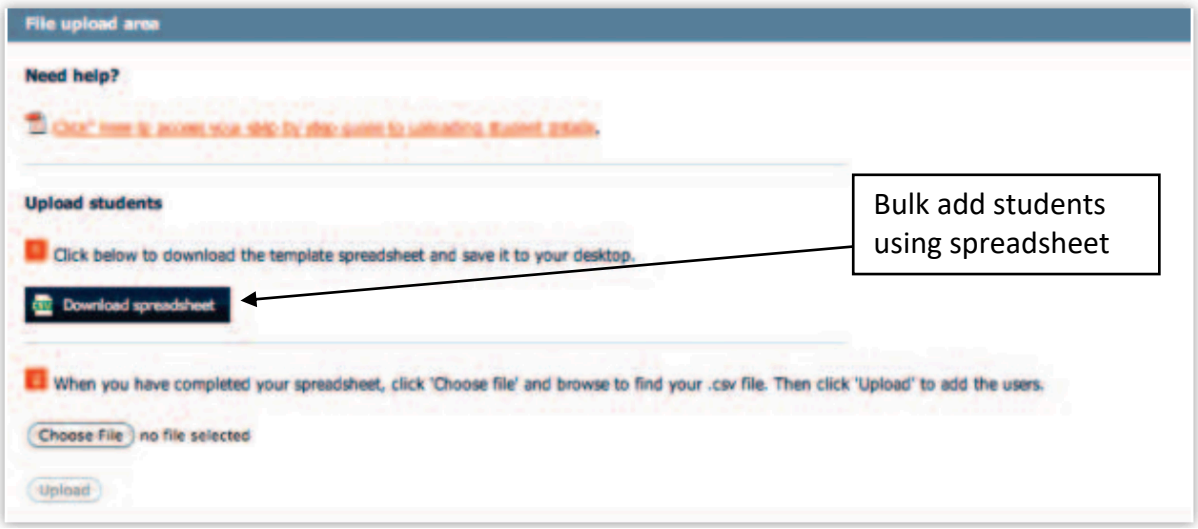


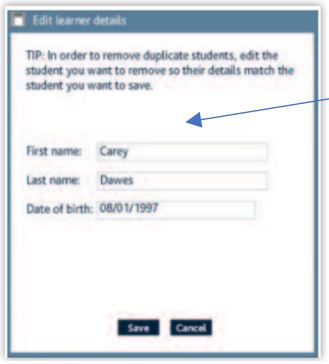
Advantages	Disadvantages
Questions available for various qualifications	Only Edexcel questions availables
Individual information for each question is available	No info about average student performance for each question
The question, mark scheme and examiner's report are all easily viewable	Access only available to teachers
Facility to create and save multiple papers	
Easy to use UI with large buttons	
A variety of filters available	

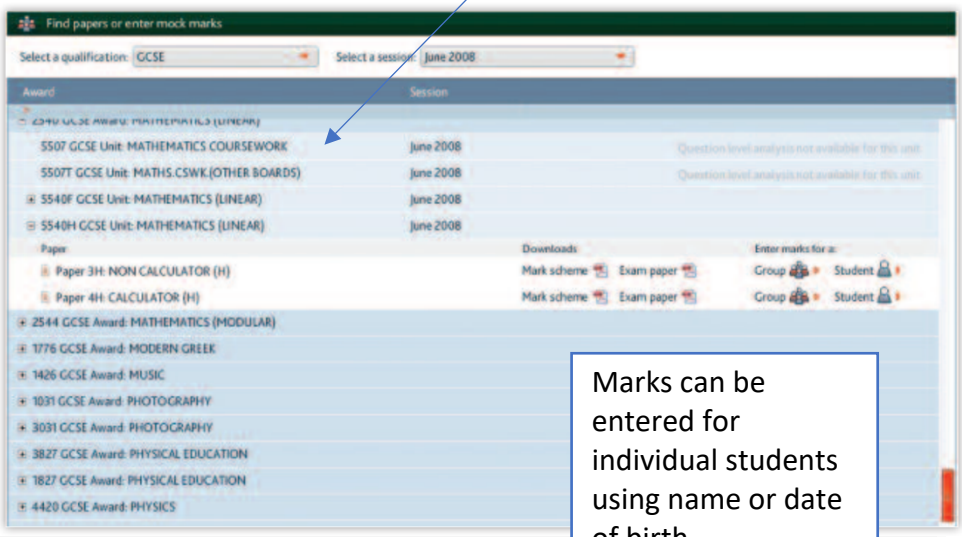
ResultsPlus

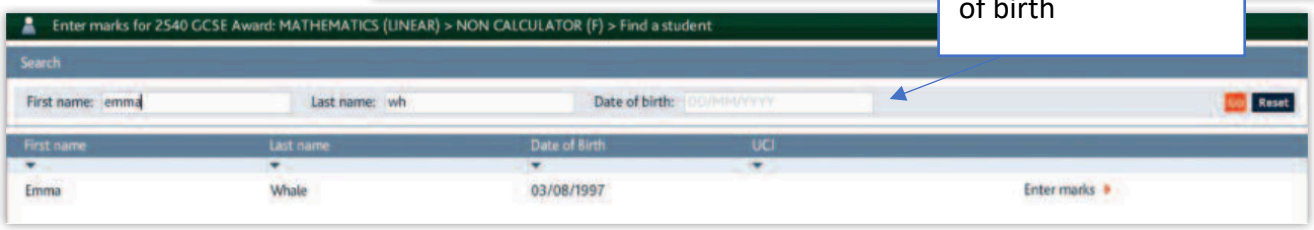
"ResultsPlus is a free online results analysis tool for teachers that gives you a detailed breakdown of your students' performance in Person Edexcel exams and BTEC external assessments."

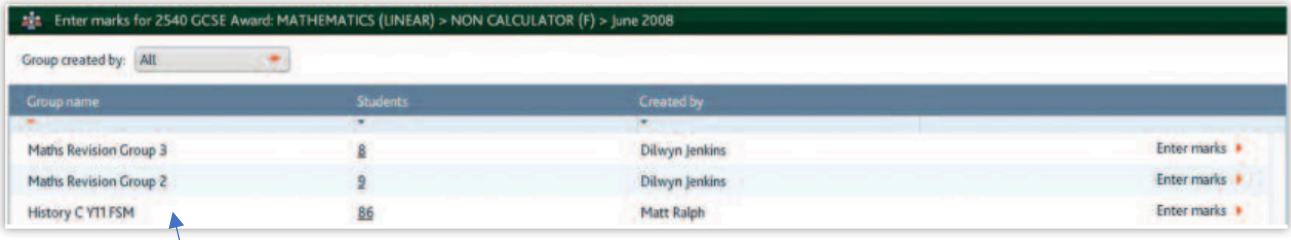
(qualifications.pearson.com/en/support/Services/ResultsPlus.html). This system provides teachers with a detailed analysis of their learner's performance. Teachers can identify potential topics, skills, and types of questions where students may need to develop their learning further. Additionally, teachers can see the actual scores of each exam question for a student, class, or group. This allows teachers to understand how their students' performance compares with the class and Edexcel national averages. Acquiring data to support effective learning and teaching approaches is easy. In order for the analysis to stay relevant to my task, I will focus on the "Mock Analysis Service" part of the software.



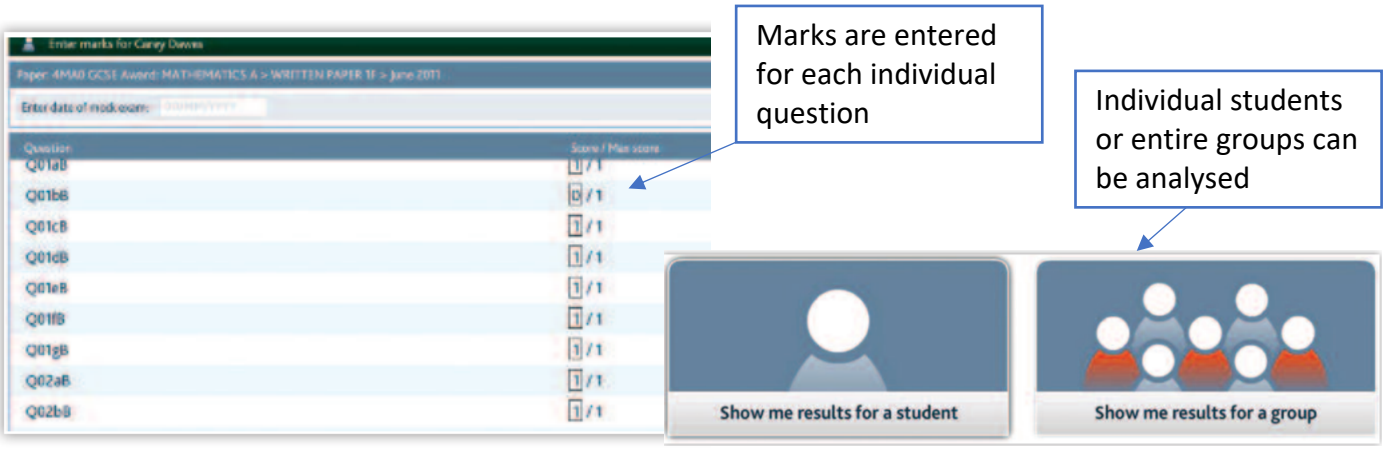


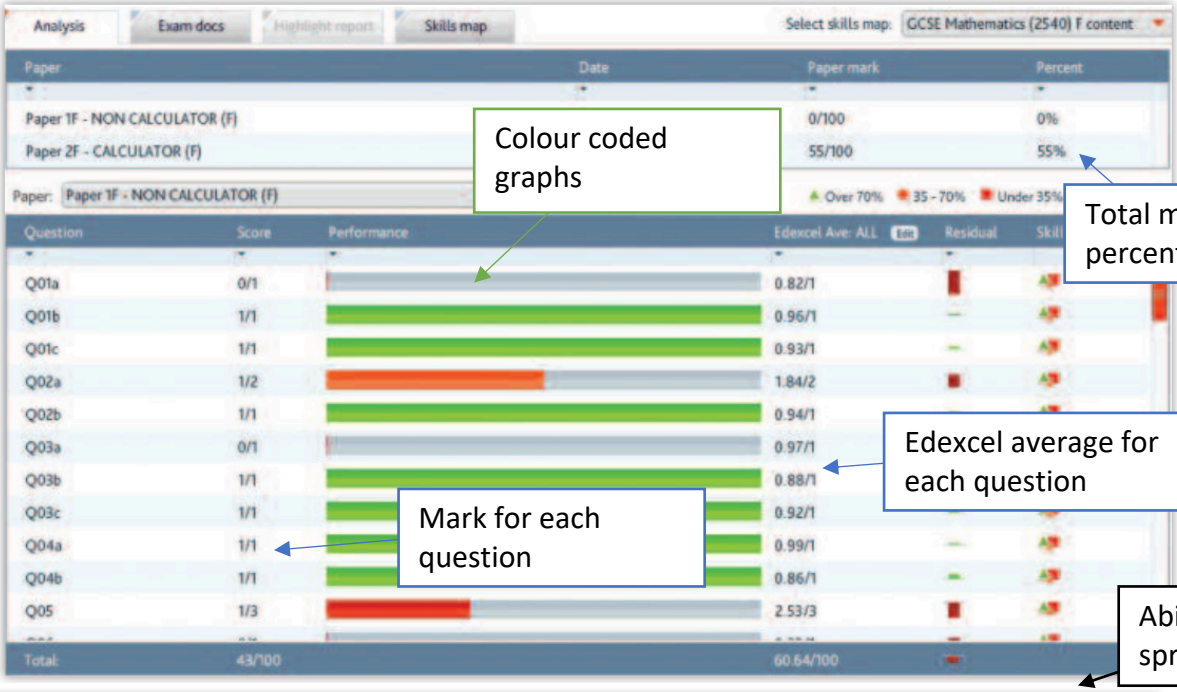


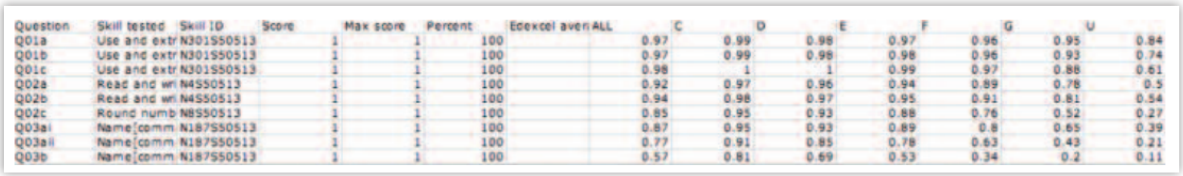


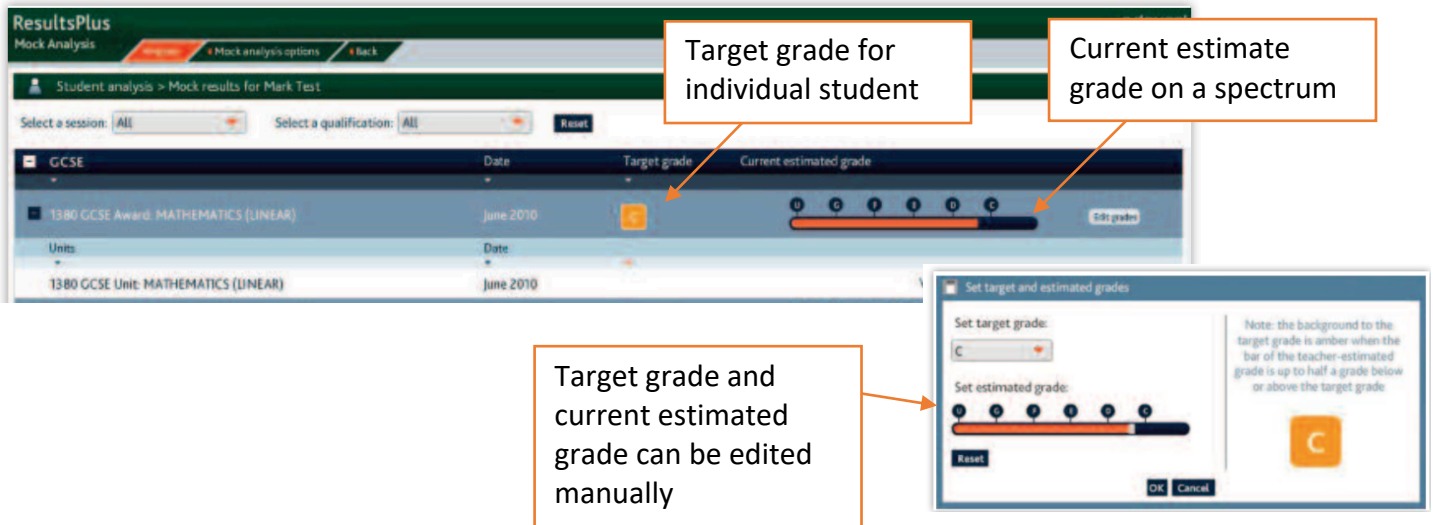


Marks can be entered for groups/classes


Marks are entered for each individual question
Individual students or entire groups can be analysed
Show me results for a student
Show me results for a group


Colour coded graphs
Total mark and percentage
Edexcel average for each question
Mark for each question
Ability to export to spreadsheet




Target grade for individual student
Current estimate grade on a spectrum
Target grade and current estimated grade can be edited manually

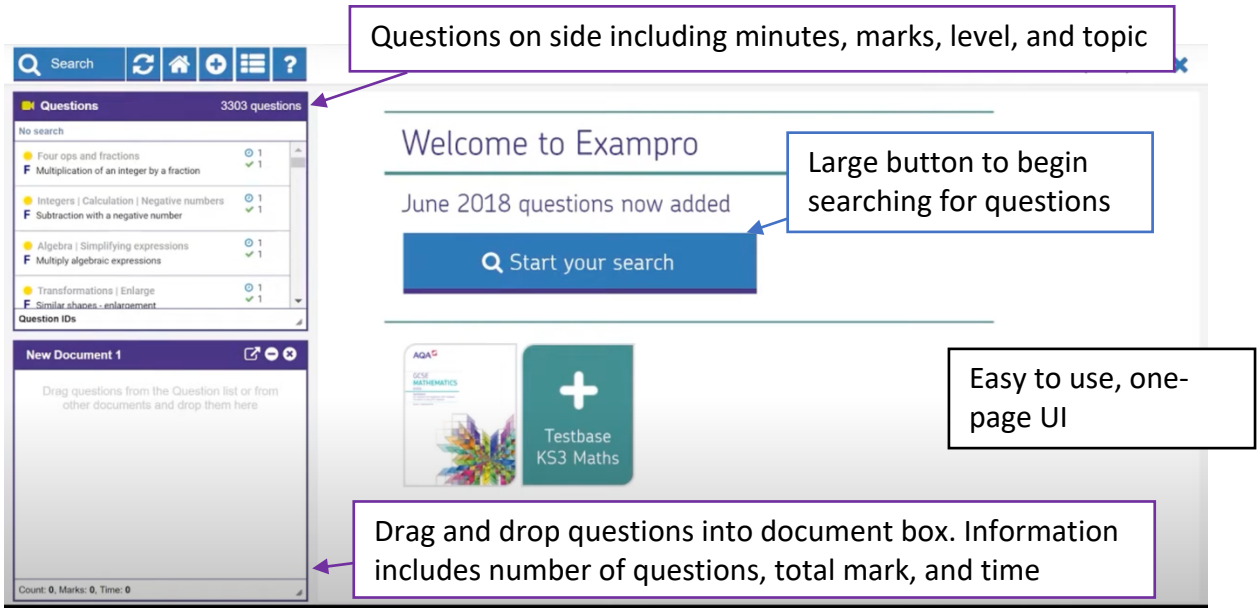
Advantages	Disadvantages
You can bulk add students	Only Edexcel past papers can be used
UI is intuitive to an extent	Papers can not be uploaded from examWizard
Colour coded graphs and target grades	Marks must be added for individual questions
Current estimated grade on a spectrum	Cannot bulk add marks
Edexcel average for each question available	Very wordy UI and analysis
Ability to export to spreadsheet	No facility to generate reports
Individual students or entire classes can be analysed	Very little use of graphs and tables

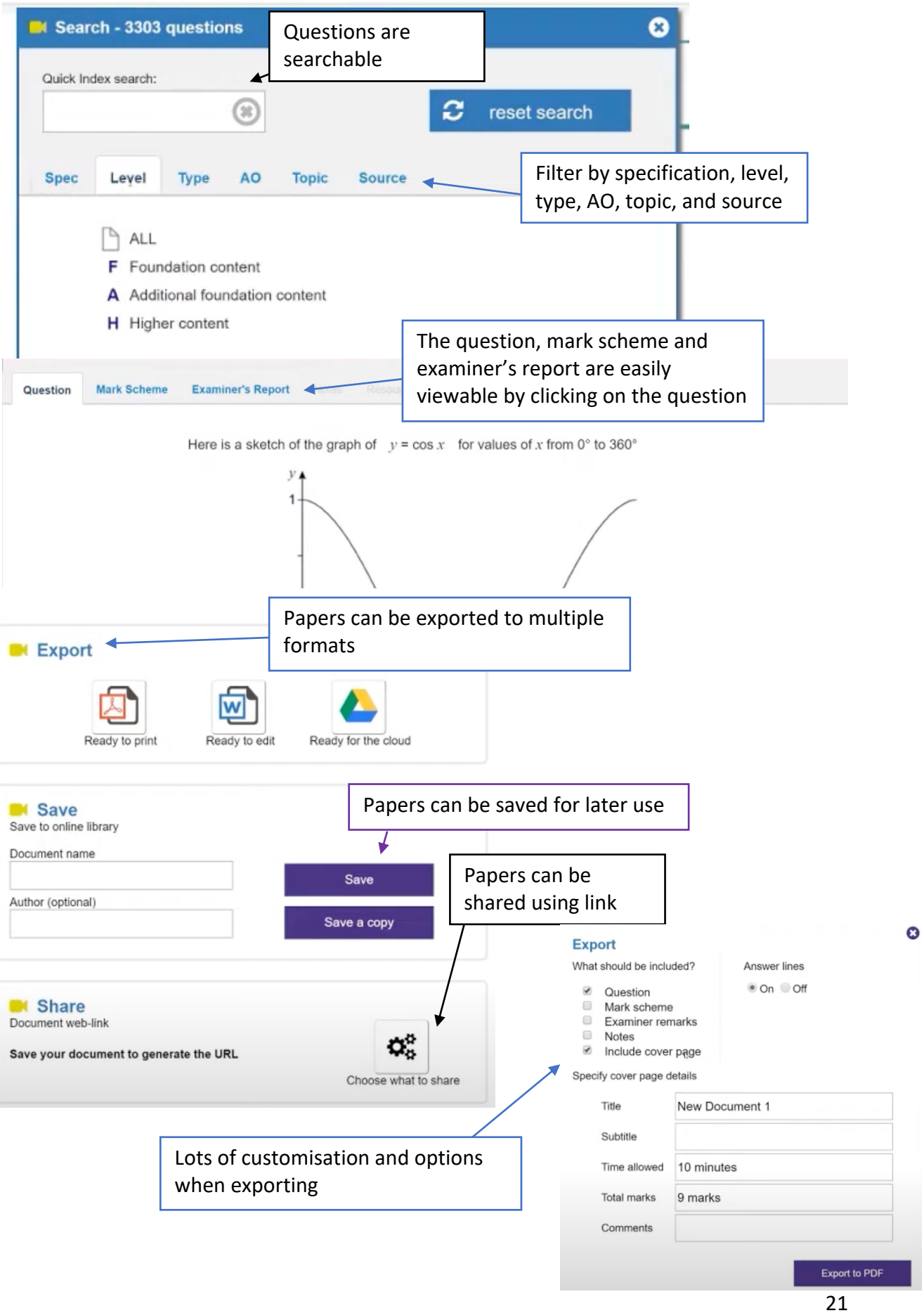
AQA

The majority of external exams offered at ICHS outside of the Mathematics department are offered with AQA. AQA operates in England, Wales and Northern Ireland and is the largest examination board for GCSEs and A-Levels in England. AQA has recently introduced computerized and digital marking, in addition to traditional marking of examinations, to increase efficiency and accuracy of the examination correction. AQA offers 2 different software solutions, “Exampro” and “MERiT”.

Exampro

Exampro is marketed by AQA as “The essential exam preparation tool for teachers” (exampro.co.uk). The software contains thousands of searchable KS3, AQA GCSE, Level 2, Level 3 and A-Level Maths questions, mark schemes and examiner comments. These questions can be used create shareable documents for all type of formative assessments including class activities, homework tasks and topic tests.

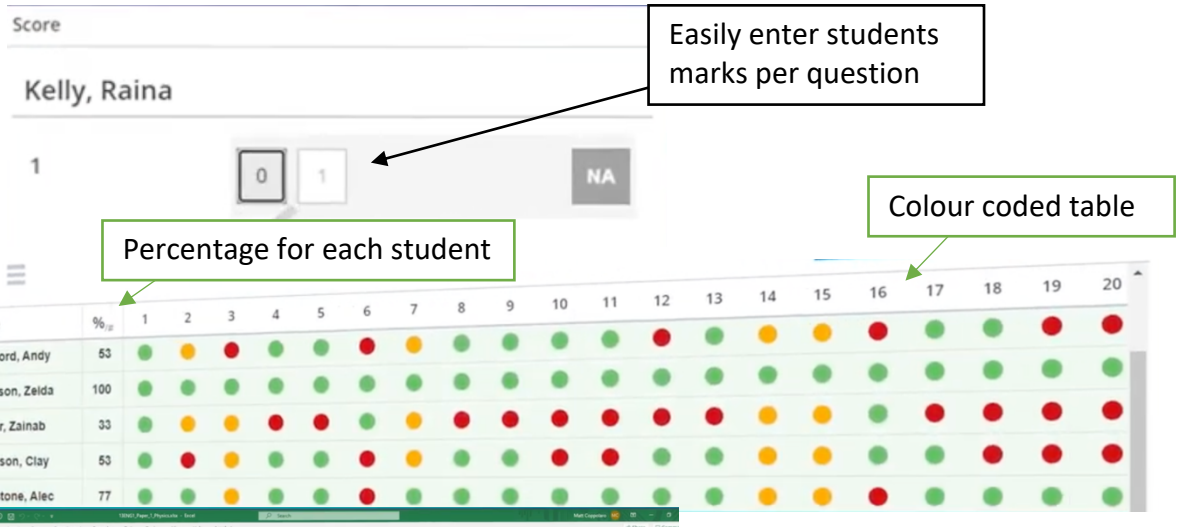


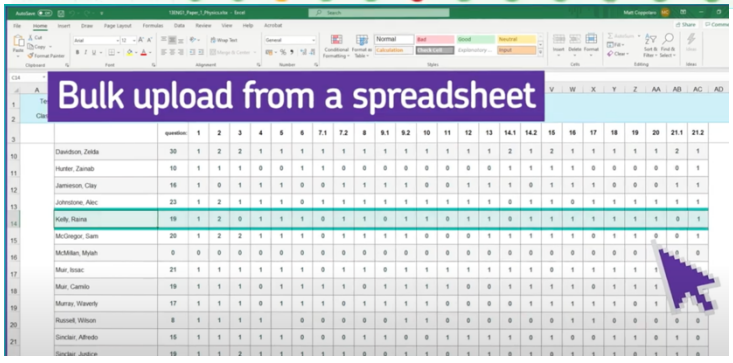

Search - 3303 questions
Questions are searchable
Quick Index search:
reset search
Spec
Level
Type
AO
Topic
Source
Filter by specification, level, type, AO, topic, and source
ALL
F Foundation content
A Additional foundation content
H Higher content
Question
Mark Scheme
Examiner's Report
The question, mark scheme and examiner's report are easily viewable by clicking on the question
Here is a sketch of the graph of y = cos x for values of x from 0° to 360°
Papers can be exported to multiple formats
Export
Ready to print
Ready to edit
Ready for the cloud
Save
Save to online library
Document name
Author (optional)
Save
Save a copy
Papers can be saved for later use
Papers can be shared using link
Share
Document web-link
Save your document to generate the URL
Choose what to share
Export
What should be included?
Question
Mark scheme
Examiner remarks
Notes
Include cover page
Answer lines
On
Off
Specify cover page details
Title
New Document 1
Subtitle
Time allowed
10 minutes
Total marks
9 marks
Comments
Export to PDF
Lots of customisation and options when exporting

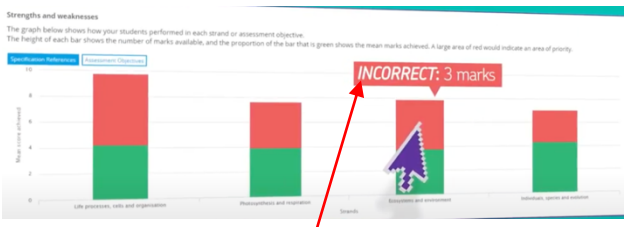
Advantages	Disadvantages
Very easy to use UI	Only AQA questions available
Drag and drop	No info about average student performance for each question
Questions are searchable and lots of filters	Access only available to teachers
Questions available for various qualifications	Pay to access
Individual information for each question available	
Question, mark scheme and examiner's report all easily viewable	
Facility to create, save and export multiple papers	
Lots of customisation when exporting	

MERiT

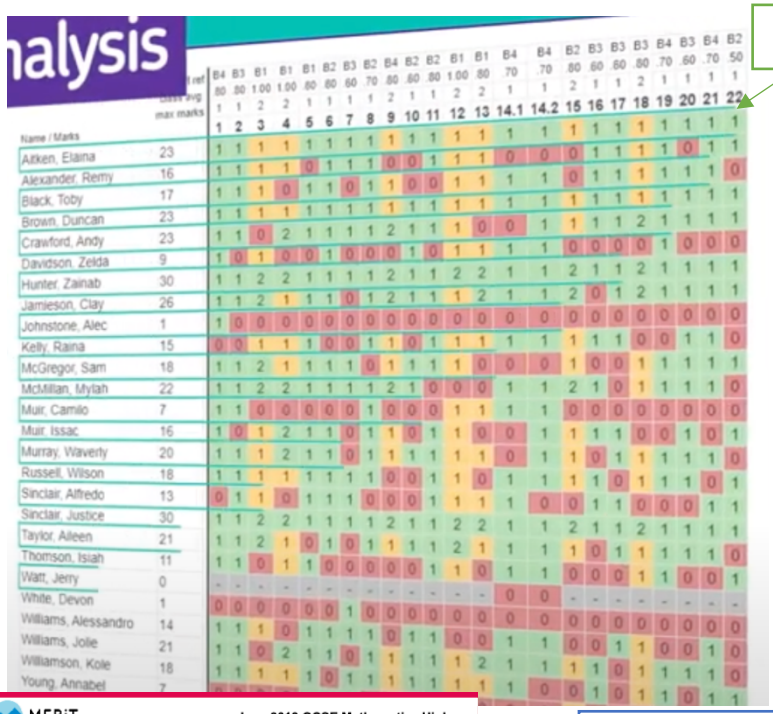
"MERiT is the reporting tool designed to give you detailed reports and a greater understanding of your students' performance." (exampro.co.uk/merit). MERiT allows teachers to compare the performance of their students with the population. Individual, class, and cohort reports can be generated which can help teachers gain a clear overview of their class by paper, AO and/or specification strand, compare their students' performance with national averages, and identify learning gaps to support planning for teaching and interventions.



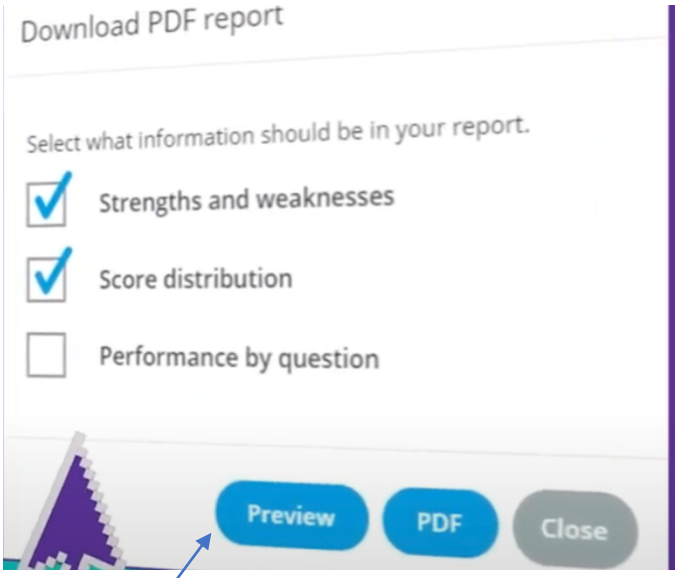




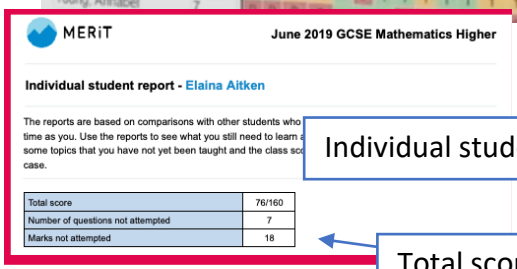
Class and student strengths and weaknesses viewable easily identifiable using bar graphs



Colour coded question level analysis

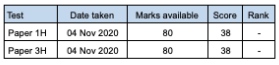


PDF reports can be downloaded

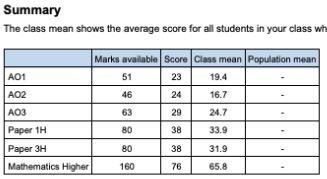


Individual student reports

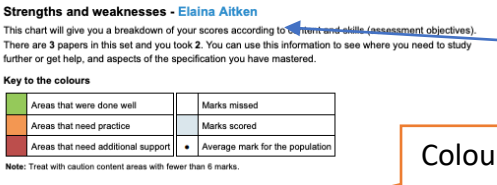
Total score and questions and marks not attempted



Paper level analysis

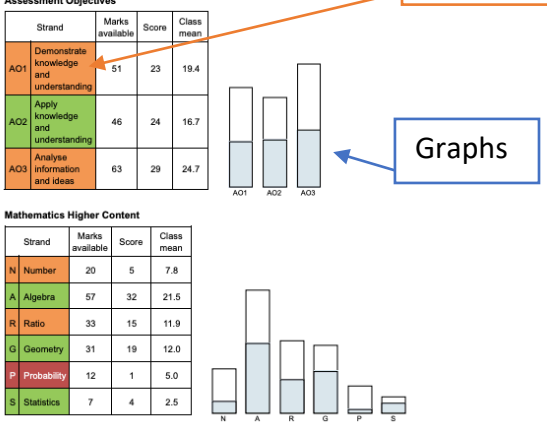


Comparison to rest of class and population



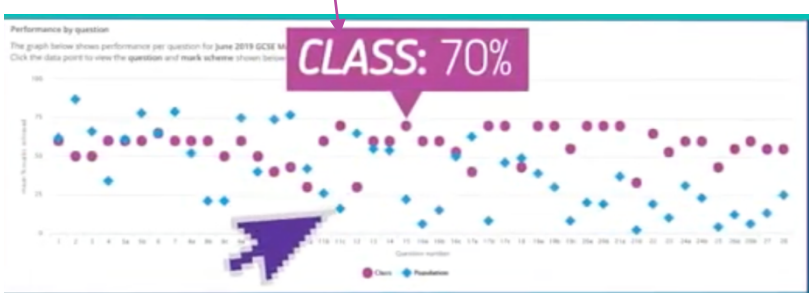
Strengths and weaknesses at an AO and content level

Colour coded



Graphs

Scatter graph of class vs rest of population at question level



Advantages	Disadvantages
Very easy to use UI	Only AQA questions can be used
Students marks per question can be entered easily through MERiT or in bulk using spreadsheet	Pay to access
Papers can be transferred over from exampro	No ability to add target grade or current estimated grade
Detailed analysis up to question level	No grade prediction specific features
Strengths and weaknesses	
Graphs to compare class to national average	
Ample use of graphs, charts and colur coding	
In-depth reports can be generated for individual students	

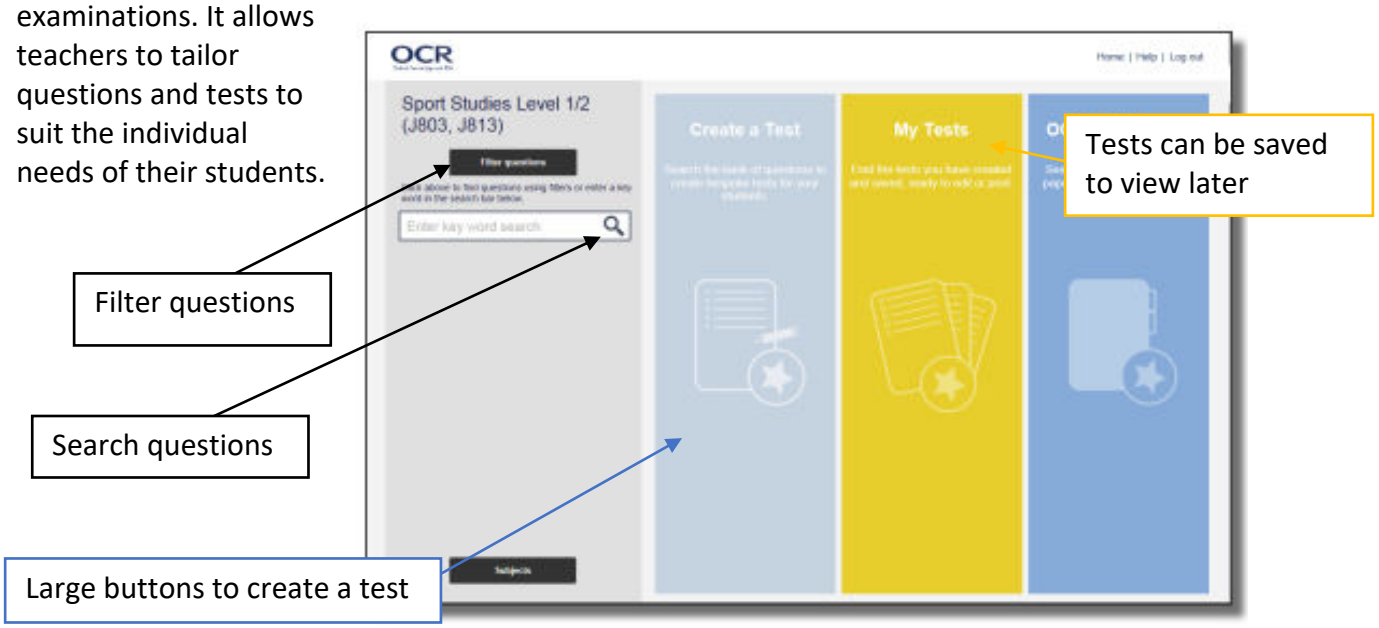
OCR

Oxford, Cambridge, and RSA examinations is an examination board that offers GCSE and A-Level qualifications in the UK. It is part of the University of Cambridge's "Cambridge Assessment". A limited number of external examinations offered at ICHS are offered with OCR, specifically Geography and PE GCSE and A-Level, Computer Science GCSE, and History A-level.

OCR offers an exam creation software "ExamBuilder". It does not offer a results analysis solution.

ExamBuilder

"ExamBuilder is our test-building platform for a wide range of OCR qualifications." (ocr.org.uk/qualifications/past-paper-finder/exambuilder). This software applications allows teachers to choose from a large bank of questions to build personalised tests and custom mark schemes, with the option to add custom cover pages to your tests to simulate real examinations. It allows teachers to tailor questions and tests to suit the individual needs of their students.



Many filters including Unit, Year, Difficulty and Assessment Objective

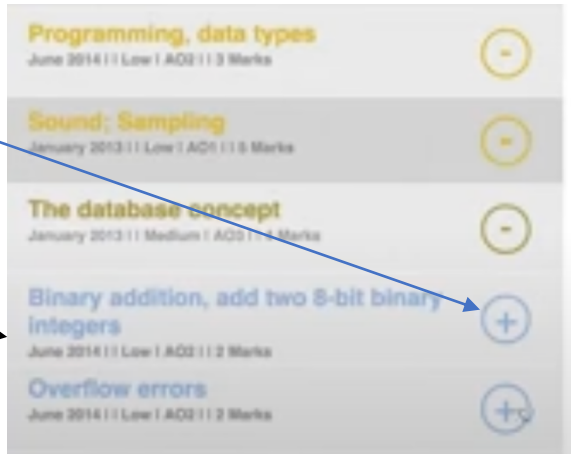
Question and mark scheme viewable (examiner's report attached to mark scheme)

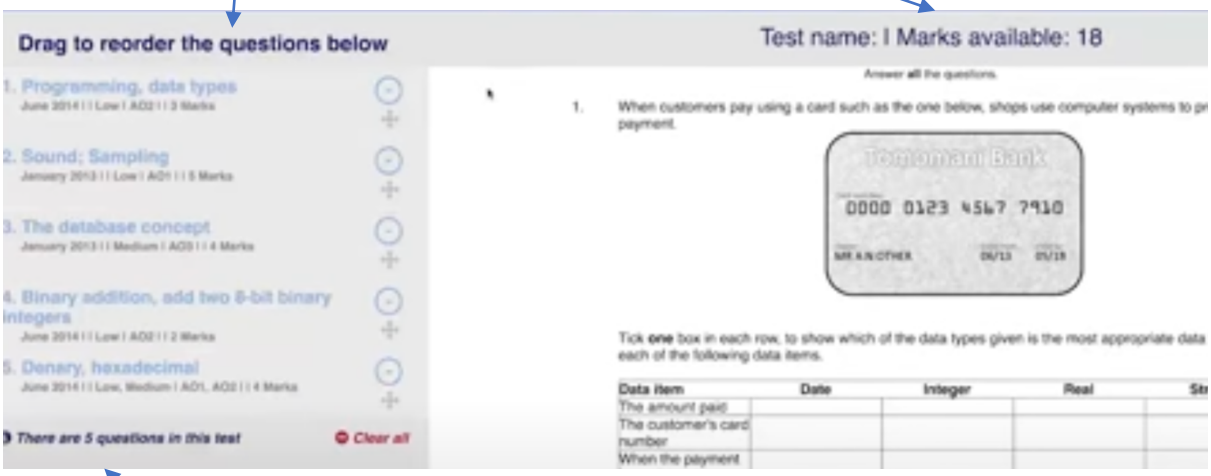
Large buttons to add or remove questions

Paper, difficulty, AO, and marks available per question

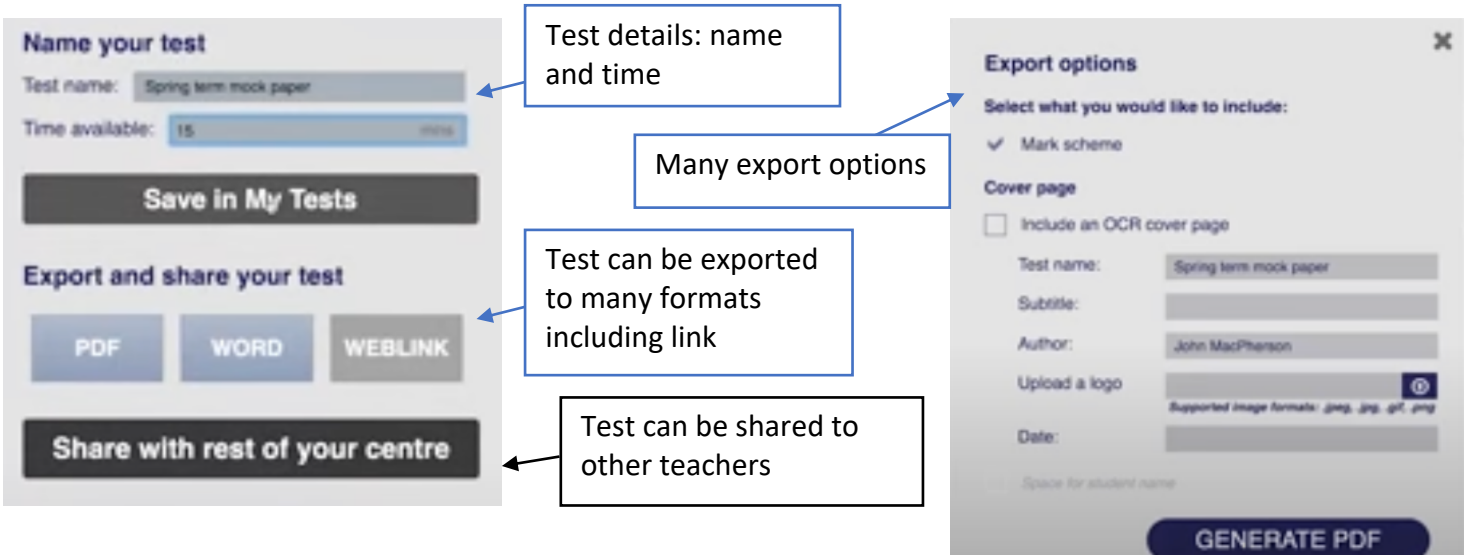
Drag and drop to reorder questions

Total marks





Total number of questions



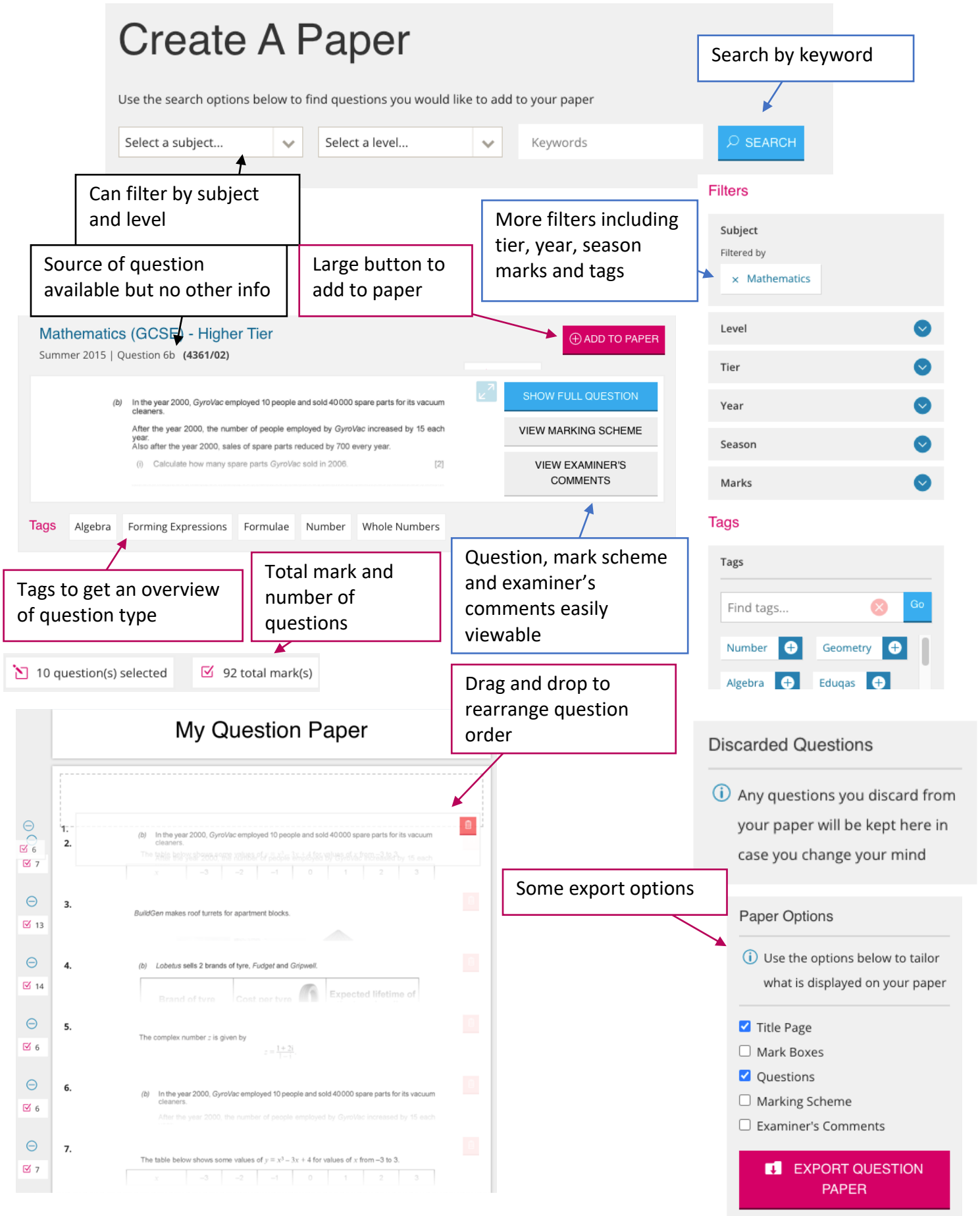
Advantages	Disadvantages
Clear design and uncluttered UI	Only OCR questions available
Simple navigation	No results analysis tool
Many filters and search option	Access only available to teachers
Question, mark scheme and examiner's report easily viewable	No info about average student performance for each question
Tests can be saved for later use	Questions only available for GCSE and A-Level
Many export options	
Tests can be shared to other teachers	

Eduqas

Eduqas is the English brand of the British examination board WJEC (Welsh Joint Education Committee), launched in 2015. Eduqas offers GCSE, AS and A-Level qualifications, as well as other Level 3 qualifications such as a Diploma/Certificate in Criminology or Medical Science to schools and colleges in England, Northern Ireland, Isle of Man, the Channel Islands, and the independent sector in Wales. ICHS does not currently offer any external examinations with Eduqas
Like OCR, Eduqas offers an exam creation software "Question Bank". It does not offer a results analysis solution.

Question Bank

"Question Bank is a free tool which allows you to create practice question papers from thousands of WJEC past paper questions." (eduqas.co.uk/en/home/question-bank). This tool helps teachers to find the questions they need, add them to a paper and export the paper with the accompanying mark scheme and examiner's comments as a PDF ready to use in the classroom. This tool is available for 4 GCSE subjects out of 24 offered by Eduqas and 7 AS/A-Level subjects out of 26. At the time of writing, WJEC is currently creating a new and improved Question Bank with greater functionality and more questions and subjects.


Create A Paper
Use the search options below to find questions you would like to add to your paper
Select a subject...
Select a level...
Keywords
SEARCH
Search by keyword
Can filter by subject and level
Source of question available but no other info
Large button to add to paper
More filters including tier, year, season marks and tags
Filters
Subject
Filtered by
Mathematics
Level
Tier
Year
Season
Marks
Mathematics (GCSE) - Higher Tier
Summer 2015 | Question 6b (4361/02)
ADD TO PAPER
SHOW FULL QUESTION
VIEW MARKING SCHEME
VIEW EXAMINER'S COMMENTS
Tags
Algebra
Forming Expressions
Formulae
Number
Whole Numbers
Tags to get an overview of question type
Total mark and number of questions
Question, mark scheme and examiner's comments easily viewable
Tags
Find tags...
Go
Number
Geometry
Algebra
Eduqas
10 question(s) selected
92 total mark(s)
Drag and drop to rearrange question order
My Question Paper
Discarded Questions
Any questions you discard from your paper will be kept here in case you change your mind
Some export options
Paper Options
Use the options below to tailor what is displayed on your paper
Title Page
Mark Boxes
Questions
Marking Scheme
Examiner's Comments
EXPORT QUESTION PAPER

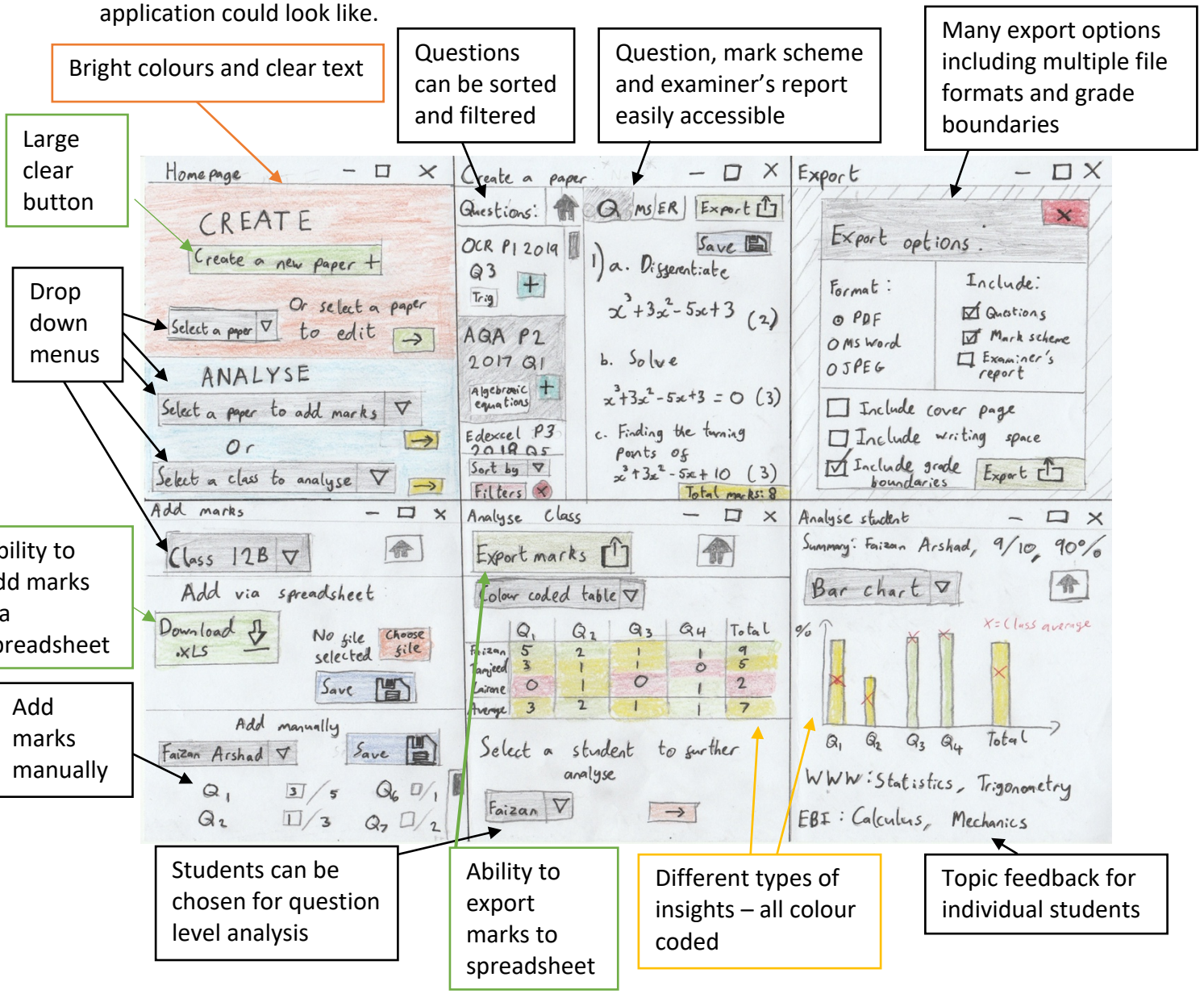
Advantages	Disadvantages
Simple design and uncluttered UI	Only Eduqas questions available
Easy to use	No results analysis tool
Limited filters and search option	Tests cannot be saved for later use
Question, mark scheme and examiner's report easily viewable	No info about average student performance for each question
Free for teachers and students to use	Questions only available for GCSE and A-Level
Some export options	Questions only available for a limited number of subjects
Tags for advanced filtering	No indication of minutes, AO, difficulty, etc

Analysis

Now that I have evaluated all the existing solutions offered by examination boards in England, I now understand what some of the good and bad points of an exam software solution are and what I want to include in my technical solution. I am going to make sure that the questions I have available are going to be from across multiple exam boards. Questions should have information available about them, such as marks, topic, source, difficulty and/or AO. Questions should be filterable and (if possible) searchable. Papers should be able to be saved for later use. Questions, mark schemes and examiner's reports should be easily viewable. There should be options to customise when exporting. The UI should be simple and easy to use. For the results analysis software, there should be detailed analysis up to a question level and coloured coded graphs, charts, and tables for analysis. Strengths and weaknesses of students and classes should be identifiable. Finally, it is advantageous if individual student reports can be generated.

Storyboard

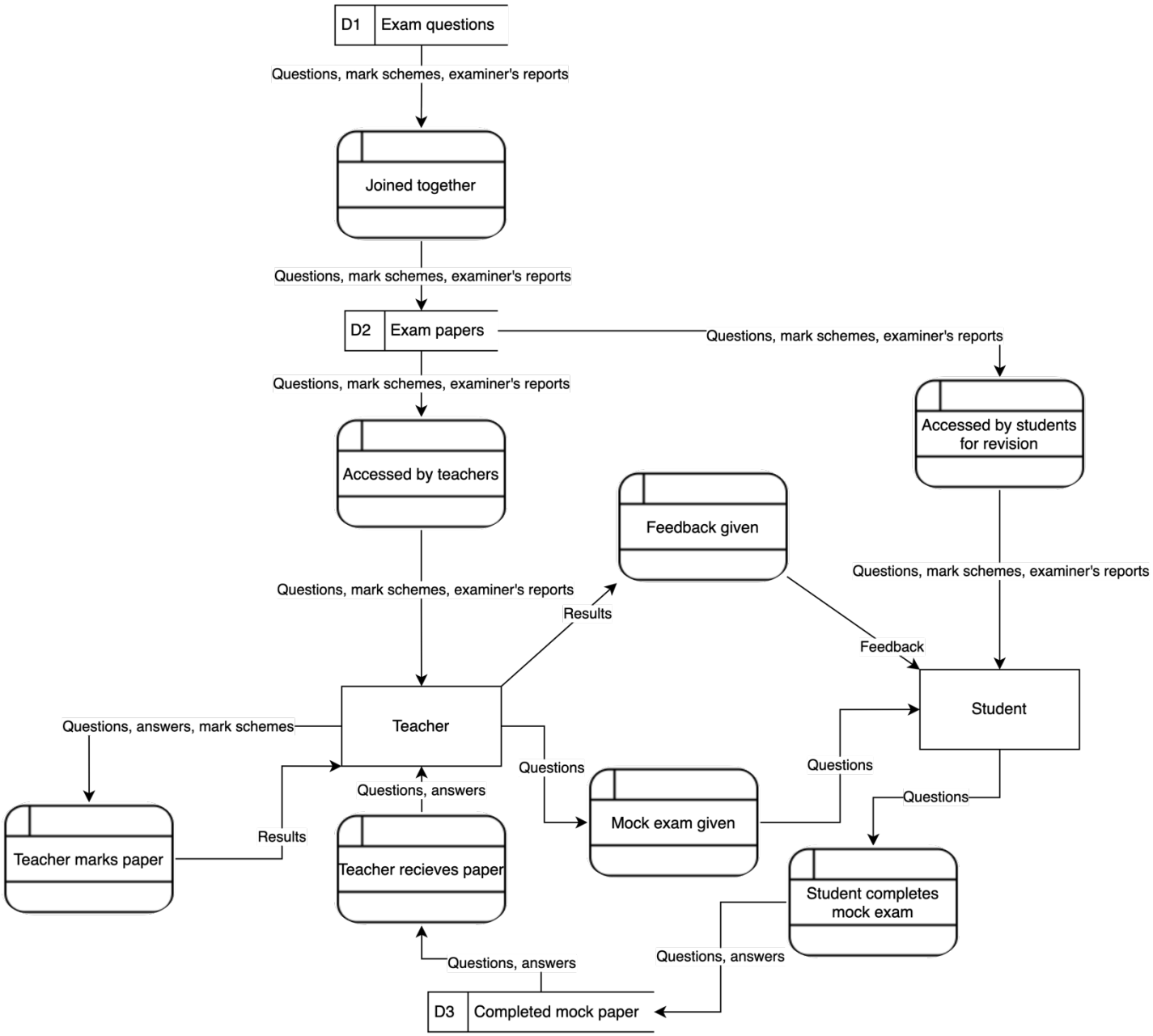
Now that I have researched some existing software solutions, and I have an idea about what I might want my solutions to look like, now I need to make it clear what my system, and specifically the UI, will entail so that I can focus on this going forward. I will be doing this using a storyboard as it is the best way to make a visual representation of what the application could look like.



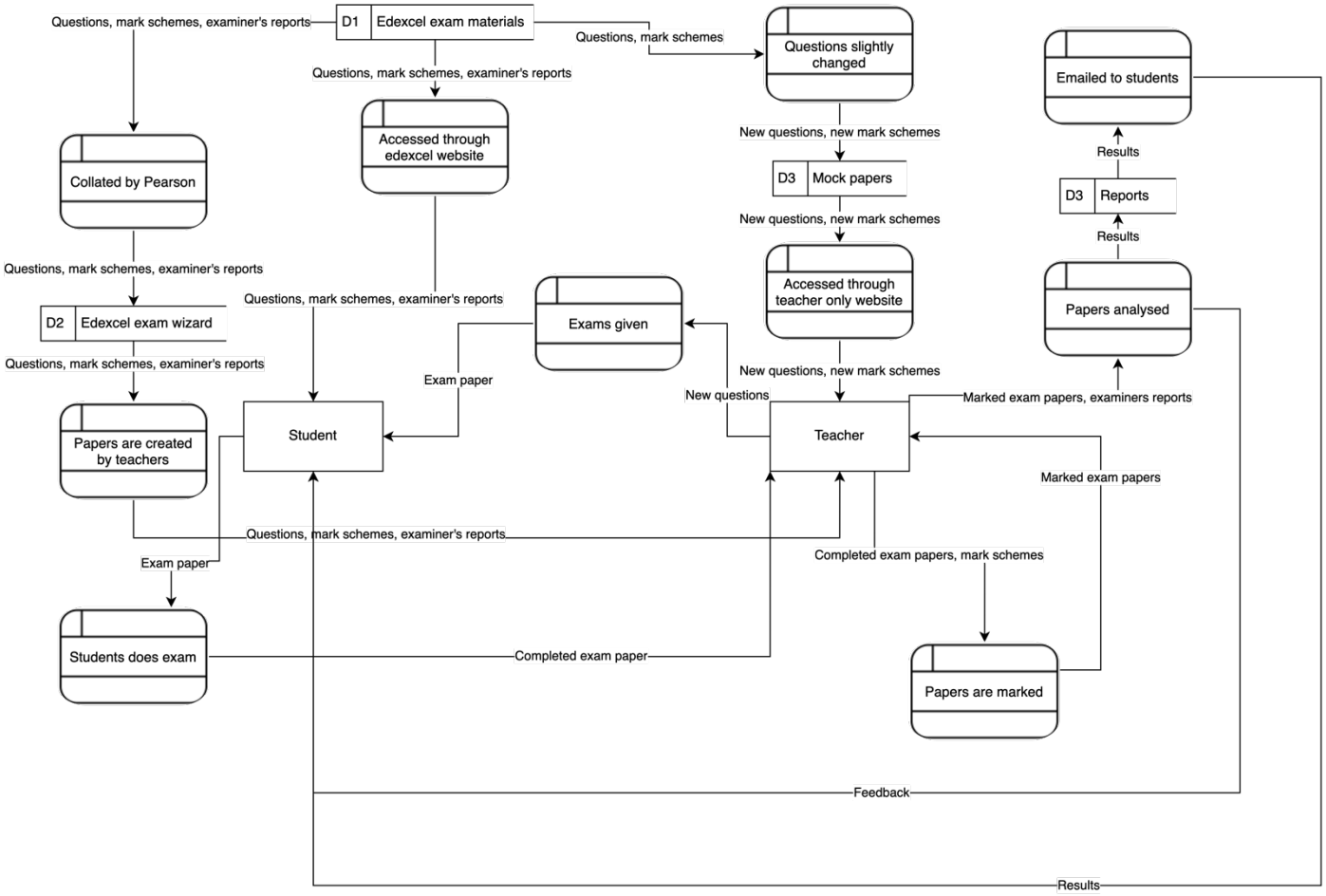
Data flow diagrams

A data flow diagram is essential for me to understand how information about exams currently flows around the system and how my system can fit in to help simplify this data flow while also not disrupting other parts of the system. Below are two data flow diagrams: a Level 0 DFD to show how the data is flowing on a surface level, and a Level 1 DFD to show this data flow in a more in-depth and thorough manner.

Level 0 Data Flow Diagram



Level 1 Data Flow Diagram



Objectives

I have established a background to my problem, researched the needs of my user, investigated some existing solutions, and modelled what my software solution is going to look like. Now I can set out the objectives that I will need to focus on and fulfil throughout the rest of this project. I need to make sure that my objectives are specific and measurable so that I can use these to aid the design and creation of my software and can use these to evaluate whether my program is successful. The objectives will be based on everything I have done so far, including the information collected through my research and my storyboard. I will number my objectives and create relevant sub-objectives so that it is easy for me to keep track of them.

1. My software solution will allow the user to create exams, add marks from exams, and analyse class performance
    a. All 3 of these features will be easily accessible from the menu, within one button press after opening the application (not including any drop-down menus)
    b. The menu will be accessible in one click from each of these sections
2. Questions included in the exam creation tool will be sourced from a range of exam boards and qualifications.
    a. There will be at least 2 questions available from Edexcel papers at A-Level
    b. There will be at least 1 question available from every English exam board specification (AQA, Edexcel, OCR [A and B], Eduqas) for GCSE and A-Level
    c. Questions will be filterable by exam board specification and qualification
    d. Exam board specification and qualification will be easily viewable from question information
3. Paper creation will be intuitive and as easy to use as Edexcel examWizard
    a. Questions will be viewable in a list and will display the question details
    b. Number of marks, number of minutes and paper information will be available from the question information
    c. Once a question has been clicked, the question, mark scheme and examiner's report will be viewable with one click
    d. Questions will be filterable by topic, year and paper, and sortable by number of marks and number of minutes.
    e. Total mark of the paper will be viewable upon opening the paper without any clicks
    f. Questions can be added and removed from a paper in one click
    g. Papers should be saveable and editable from within the software
4. There will be a range of export options for the papers created
    a. The export menu should be accessible with one click from the creation tool
    b. Papers will be exportable to at least 2 file formats (preferably PDF and DOCX)
    c. There will be options to decide whether the user would like to export the questions, mark scheme and/or examiner's report
    d. There will be the ability to export grade boundaries alongside everything else
    e. There will be at least one other export option, such as a cover page or writing space
    f. All export options will be selectable and deselect able in one click

5. It should be easy and intuitive to add and view the marks of students from exams/papers from within the software
    a. There will be an option to type in the marks for each student for each question from within the software
    b. For each question, the total mark should be shown and the software should not allow the user to enter a mark above the available total
    c. Marks for each student for each question must be viewable from within the software
6. The software will have the ability to analyse the performance of students and classes
    a. Classes and students will be analysable at a paper level and a question level
    b. There will be at least 2 different types of insights available from within the software, including a table and some sort of visual chart
    c. All insights will be colour coded
    d. Insights will show comparisons to a class average
    e. Individual feedback for students will be viewable including at least 1 topic that went well and 1 topic for improvement (except when 0% or 100% was achieved, or the whole paper was based on one single topic)

Documented Design

Overview

My program will begin on a home screen. This screen can be accessed from all the other screens through a dedicated home button, which will save any changes when selected also. The home screen is split up into two distinct sections, named CREATE and ANALYSE. The CREATE section allows the user to create or edit exam papers. There are two options, either to create a new paper, which when selected will ask the user for the name of the paper and a list of all the current available papers, so that the user can select which paper to edit. In the ANALYSE section, there are also two options, either to select a paper that has previously been created to add the marks of students or a class can be selected to analyse their performance on an exam.

Paper creation – once a paper has been selected from the home screen, or a new paper has been created and named, this will lead to the paper creation UI. On the side, there is a scroll bar that shows all the available questions and some information about each question, such as the exam board, paper, and topic of each question. There is also a plus sign button on each question, which when pressed, will add the question to the paper. This will then turn into a minus sign, which will remove the question when pressed. This action can also be undone, using an undo button that works by saving all previous adds/removes into a stack. All questions added will be saved in a queue. These questions can be sorted by different categories using a drop-down below the scroll bar questions. Also below the menu is a filters button, which will prompt the user to add any filters that they need so that only questions that are required are shown. There is also a clear all option to clear all filters. On the right of the scroll bar is the display window, which allows the user to view the actual question, the mark scheme and the examiner's report. There will also be a save button, to save the paper, and an export button, which leads to the export menu.

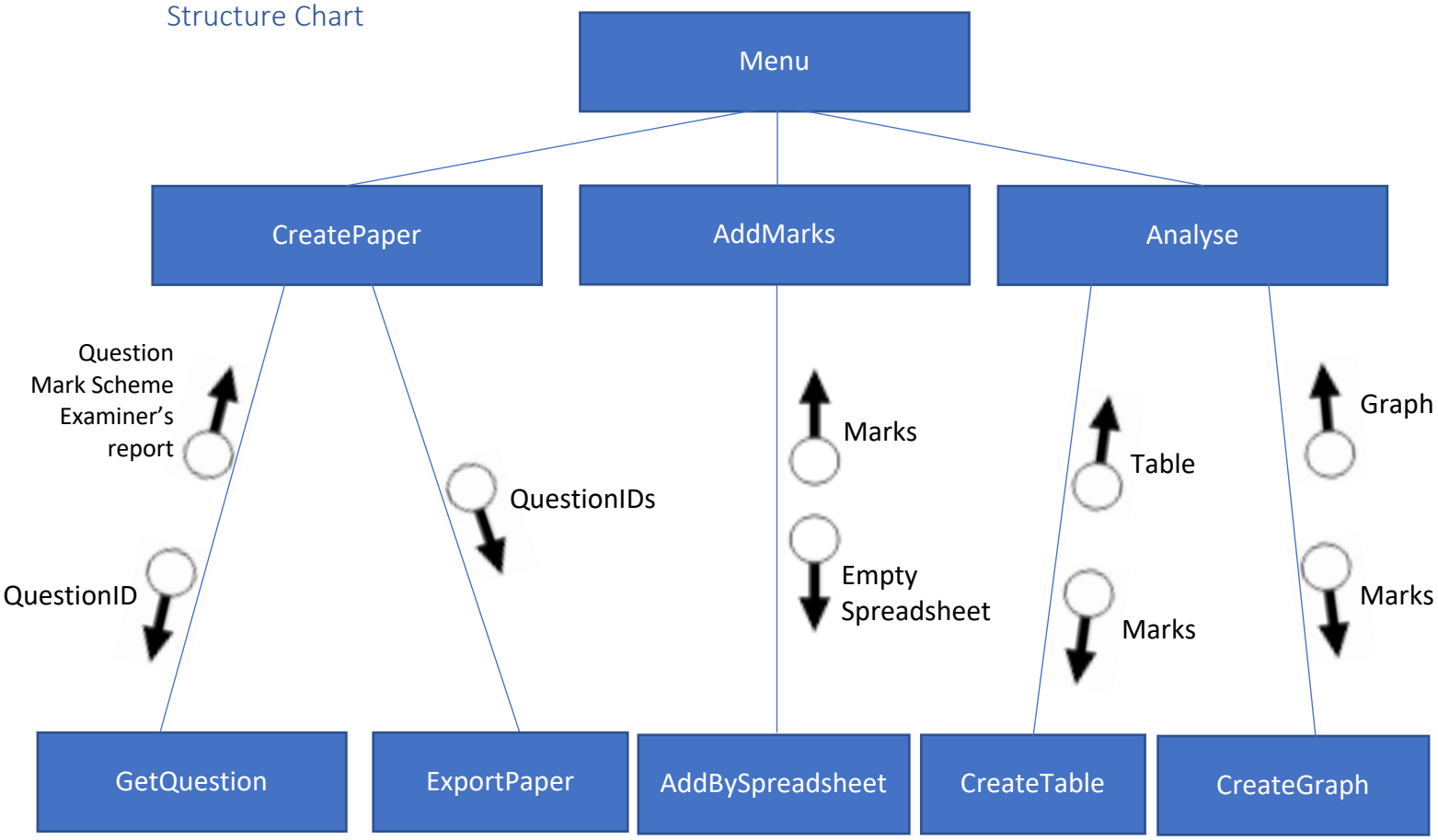
The export menu allows the user to export the paper. The user will have to choose which file format to export the paper to and can decide the format of the paper from many options, such as a cover page, and whether the user would like the questions, mark schemes and/or examiner's reports to be included. Once done, the questions are dequeued onto the file and the file is saved.

If a paper is selected in the home screen ANALYSE section, this will allow the user to add the marks of their students into the software, and consequently the database. Firstly, a class must be selected, or a new one must be created. Students can also be added to classes from this screen.

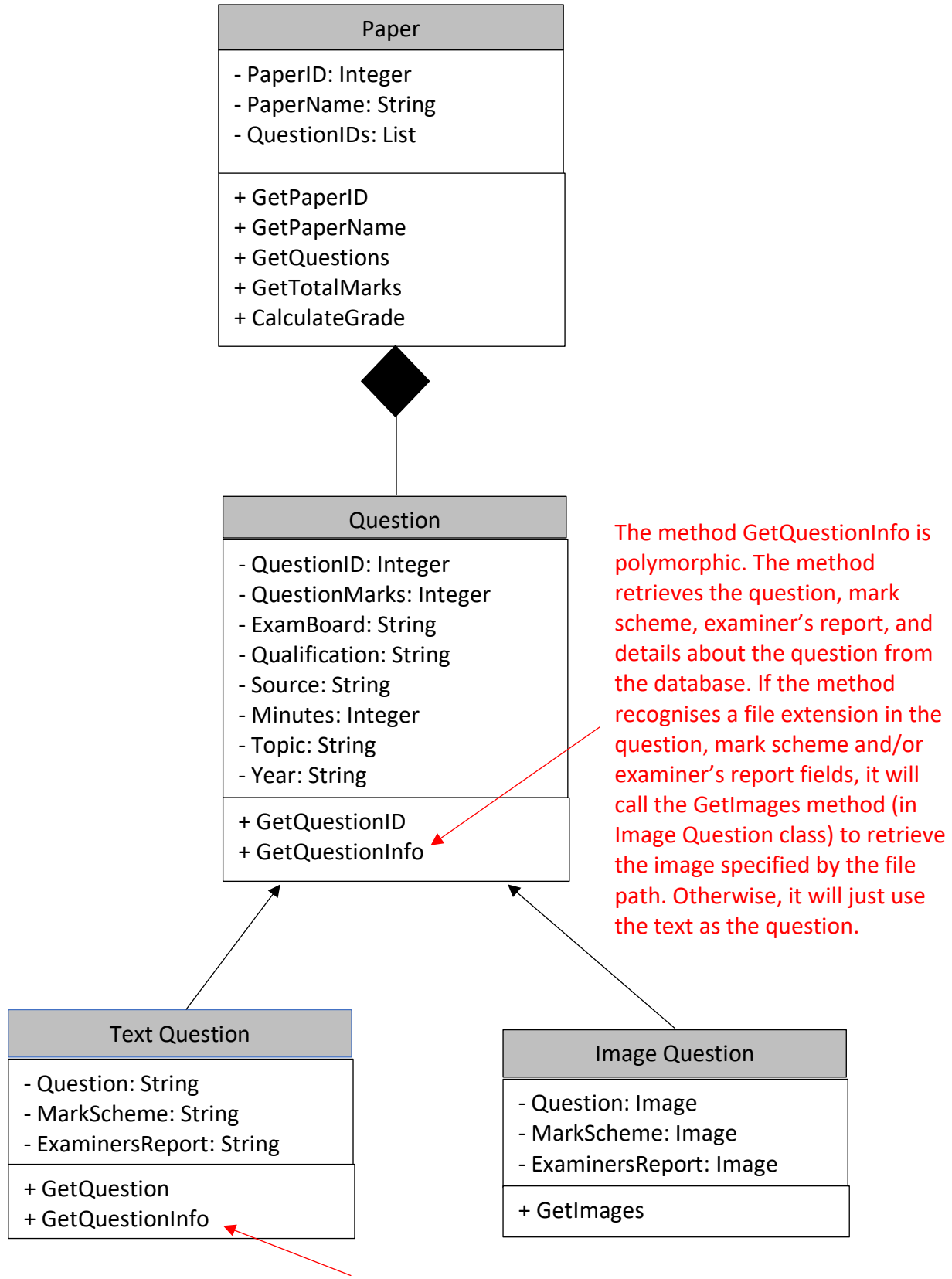
Once the marks are added, the class can be analysed by selecting the class on the home screen. Below this, there is a drop-down menu which allows the user to view several visual insights into the performance of the class, such as a colour-coded table, or a graph. At the bottom of the screen, there is an option to analyse a single student, by choosing a student from the class using a drop-down.
Once a student is chosen, at the top of the screen is a summary of their performance. Below this, there are visual insights available, just like in the analyse class screen. Finally at the

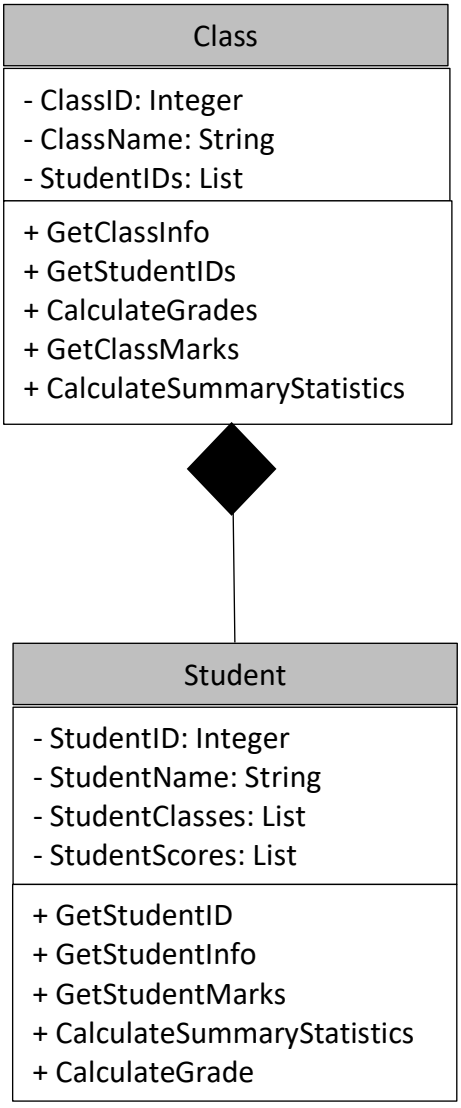
bottom of the page is topic-based feedback for the student in the format of a WWW (what went well) and an EBI (even better if).

Structure Chart



Class Diagram




Class
- ClassID: Integer
- ClassName: String
- StudentIDs: List
+ GetClassInfo
+ GetStudentIDs
+ CalculateGrades
+ GetClassMarks
+ CalculateSummaryStatistics
Student
- StudentID: Integer
- StudentName: String
- StudentClasses: List
- StudentScores: List
+ GetStudentID
+ GetStudentInfo
+ GetStudentMarks
+ CalculateSummaryStatistics
+ CalculateGrade

Algorithms

Merge Sort

On the paper creation interface, there will an option that will allow the user to sort questions by different criteria, such as by the number of marks or the number of minutes. To do this, I will be using a merge sort. A merge sort works by splitting the unordered list into two halves called sublists. The algorithm repeatedly splits the sublists into smaller sublists and then merges the sublist together which is when the items are ordered. This merge sort will also be used to calculate the median and quartiles for my summary statistics (see page 45).

The reason I have chosen to use a merge sort over other alternative sorts is that it is relatively efficient. In terms of time complexity, the list is divided log(n) times, so the time complexity of splitting is O(logn), and during the merging process every item is examined in turn and so in the worst-case scenario, there will be n comparisons to merge. Combining these two parts means that merge sort has a time complexity of O(nlogn). Although this could be better, alternatives are mostly worse. For example, the time complexity of bubble sort is $O(n^2)$. This would take far too long. I will be sorting a medium number of questions, much more than a small amount, but not enough to be classified as Big Data, so merge sort helps to strike the balance between simplicity and efficiency.

Merge sort however is not that efficient in terms of space complexity, as it requires space for the right and left sublists, and so has a space complexity of O(n). This is not very efficient in comparison to something like a bubble sort which has a space complexity of O(1). However, since I will not be dealing with a significant amount of data, I don't think this will have too much effect on my program, and the benefit gained from the highly efficient time complexity outweighs this disadvantage.

Before the items are sorted, I will be replacing each question in the list with another list inside of it, containing two items, the question, and the marks/minutes of the question. This allows for easy access when sorting. For example, when sorting by minutes:

questions = [question1, question2, question3]

becomes

questions = [[question1, question1.minutes], [question2, question2.minutes], [question3, question3.minutes]]

This can be done by a very simple for loop structure:

```
FOR x IN RANGE(LENGTH(questions))
        questions[x] = [questions[x], questions[x].minutes]
```

Similarly, to convert back once sorted:

FOR x IN RANGE(LENGTH(questions))

```
    questions[x] = questions[x][0]
```

The merge sort works using 2 procedures, 1 to split the items (merge_sort) and 1 to merge them back together (merge). The merge sort utilises recursion to manage all the sublists:

```
PROCEDURE merge_sort(questions)

    IF LENGTH(questions) < 2
        RETURN
    ELSE
        midpoint = (LENGTH(questions) – 1) // 2
        left_half = questions[0:midpoint]
        right_half = questions[midpoint+1:LENGTH(questions)-1]

        merge_sort(left_half)
        merge_sort(right_half)

        merge(questions, left_half, right_half)

PROCEDURE merge(merged, list_1, list_2)

    index_1 = 0
    index_2 = 0
    index_merged = 0

    WHILE index_1 < LENGTH(list_1) AND index_2 < LEN(list_2)

        IF list_1[index_1][1] < list_2[index_2][1]
            merged[index_merged] = list_1[index_1]
            index_1 = index_1 + 1
        ELSE
            merged[index_merged] = list_2[index_2]
            index_2 = index_2 + 1
        index_merged = index_merged + 1

    WHILE index_1 < LENGTH(list_1)
        merged[index_merged] = list_1[index_1]
        index_1 = index_1 + 1
        index_merged = index_merged + 1


    WHILE index_2 < LENGTH(list_2)
        merged[index_merged] = list_2[index_2]
        index_2 = index_2 + 1
        index_merged = index_merged + 1
```

Stack Operations

When creating papers, the user will periodically add questions to the paper, storing the questions in an array. The questions are stored in an array so that they can easily be added and removed. The order in the array will not matter as each question will have a priority that indicates the order, and this will be manipulated when exporting by using a priority queue (see page 43). However, there is going to be a feature where you can undo previous actions and to be able to implement this, I need to use a stack.

A stack is an abstract data type that operates using a last-in, first-out structure. It holds an ordered, linear sequence of items. Every time a change is made, such as adding or removing questions, or editing the order, the change is saved on the stack. If the user wants to undo this change, the previous change can just be retrieved from the top of the stack and then the change can be reversed.

The reason I chose to use a stack is that it is the best data structure for this use case. Operating using a last-in, first-out methodology allows for an easy and effective way to consistently get the previous actions. To contrast, a queue, which operates using a first in, first out methodology would not be suitable since rather than retrieving the last action it will retrieve the first action, which is not what we want in this case. Apart from this, there are not many alternatives, if any, to this structure and using the alternative would be extremely inefficient with respect to both space and time complexity and so would significantly slow down the program.

My stack will be implemented using a dynamic list and a pointer that indicates the position of the top of the stack. I will use the python built-in .append() function to be able to increase the size of the list when it is full, but all actions will be added to the list using the top pointer.

```
CLASS Undo_Stack:
    PROCEDURE Constructor(Self):
        Self.stack = [“ ”]
        Self.top = 0

    FUNCTION CheckIfEmpty(Self):
        IF Self.top == 0 AND Self.stack[Self.top] IS “ “:
            RETURN True
        ELSE:
            RETURN False

    FUNCTION CheckIfFull(Self):
        IF Self.stack[length(Self.stack)-1] IS NOT “ “:
            RETURN True
        ELSE:
            RETURN False

    PROCEDURE Add(Self, Item):
        Self.stack[Self.top] = item
```

```
        IF Self.CheckIfFull() IS True:
            Self.stack.append(" ")
        Self.top = Self.top + 1

    FUNCTION Remove(Self):
        IF Self.CheckIfEmpty() IS TRUE:
            Return "Empty"
        ELSE:
            Temp = Self.stack[Self.top]
            Self.stack[Self.top] = " "
            Self.top = Self.top – 1
            RETURN Temp
```

Exporting a paper

To export a paper, a short algorithm must be run. First, the questions need to be arranged in the correct order. Next, a PDF must be created with the questions added to the PDF. Finally, the PDF needs to be exported.

To do the first step, where the questions need to be arranged in the correct order, a priority queue will be used. The queue will be implemented using a list and 2 pointers pointing to the front and the back of the queue. Each question in the list will have a priority associated with it. This priority is in the form of a number, that indicates the question number (e.g., the first question has a priority of 1, the second has a priority of 2, etc.). Each question in the list will be enqueued onto the priority queue and once all the questions are in, they will be ordered in reverse order. This is what is needed since the next step will use recursion to add the questions to the PDF. Below is a representation of how the queue will look.

Pqueue = [[question3, 3], [question2, 2], [question1, 1]]
Front = 0
Back = 2

This use of a priority queue has a time complexity of O(nlogn). An alternative to using this implementation would be to use a sorting algorithm to order the questions within the list. One example could be using a bubble sort. However, this would be extremely inefficient since it has a time complexity of $O(n^2)$. Another option could be using a merge sort, which also has a time complexity of O(nlogn). However, using a priority queue is superior to these when considering space complexity, since bubble sort requires a temporary variable to be able to make the swaps and merge sort splits up the list into multiple sublists which is very inefficient.

```
CLASS Pqueue:
    PROCEDURE Constructor(Self):
        Self.queue = []
        Self.front = 0
        Self.back = -1

    PROCEDURE Enqueue(Self, Item, Priority):
        Self.queue[Self.back+1] = " "
        Temp = Self.back
        Self.back = Self.back + 1
        Found = FALSE
        WHILE Temp IS NOT < Self.front AND Found IS FALSE:
            IF Self.queue[Temp][1] < Priority:
                Self.queue[Temp+1] = Self.queue[Temp]
                Temp = Temp – 1
            ELSE:
                Found = TRUE
        Self.queue[Temp+1] = [Item, Priority]

    FUNCTION Dequeue(Self):
```

```
        Self.front = Self.front + 1
        RETURN Self.queue[Self.front-1]
```

Once the questions are sorted in the priority queue, a recursion algorithm will be used to add the questions to the PDF. A recursive algorithm is one that is defined in terms of itself. This means that, when the steps of the algorithm are written down, one of the steps refers to the same algorithm by name. When implemented, the subroutine will include one or more calls to the original algorithm. I will use recursion to load up the questions each within one instance of a subroutine, then go through the system stack to insert the questions into the PDF in order. To create the PDF, I will use an external module called "pyfpdf: FPDF for Python" (see bibliography for more information). For simplicity, I have abstracted away the details of using this module and instead explained what will happen using comments (denoted by a # at the start of the line and in *red text*).

```
CLASS CreatePDF:
    PROCEDURE Constructor(Self, Pqueue):
        Self.Pqueue = Pqueue
        PDF = # Create PDF
        # Insert code here to create a cover page

    PROCEDURE AddQuestion(Self):
        Question = Self.Pqueue.Dequeue()
        IF Self.Pqueue.Front IS NOT > Self.Pqueue.Back:
            Self.AddQuestion()
        PDF.ADD_PAGE()
        IF Question.Type IS "text":
            # Add text question to PDF
        ELSE:
            # Add image question to PDF
```

There are many reasons why this use of recursion is beneficial. For example, fewer lines of code allow the function to be easier to read. In addition, using recursion is a more natural way to process data and utilise the priority queue instead of using something like a for or while loop. There are some disadvantages, such as using more memory since there is a need to store multiple stack frames and it could slower since there is a need to manage stack operations. However, since it is unlikely that a paper will be made with more than 20 questions, this is unlikely to affect the program in a substantial way.

Summary statistics

My final algorithm comes in during the analysis stage, and it is the calculation of summary statistics. When comparing students/classes, it is very useful to have access to summary statistics of scores. Things that come in useful include calculating percentages, as well as measures of central tendency and variation. All calculations are taken from the Pearson Edexcel Level 3 Advanced GCE in Mathematics (A-Level Maths) specification, specifically page 31. A link to the specification can be found in the bibliography and screenshots of the relevant part of the specification can be found below and in Appendix 2.

2.3	**Interpret measures of central tendency and variation, extending to standard deviation.**	**Data may be discrete, continuous, grouped or ungrouped. Understanding and use of coding.** **Measures of central tendency: mean, median, mode.** **Measures of variation: variance, standard deviation, range and interpercentile ranges.** **Use of linear interpolation to calculate percentiles from grouped data is expected.**
	Be able to calculate standard deviation, including from summary statistics.	**Students should be able to use the statistic x** $$S_{xx} = \sum(x-\bar{x})^2 = \sum x^2 - \frac{(\sum x)^2}{n}$$ **Use of standard deviation =** $\sqrt{\frac{S_{xx}}{n}}$ **(or equivalent) is expected but the use of** $S = \sqrt{\frac{S_{xx}}{n-1}}$ **(as used on spreadsheets) will be accepted.**

I have split up the calculations into 3 separate functions, one for calculating a percentage, one for calculating mean, standard deviation, and variance, and one for all the rest. This is so that statistics that are used more often can be a separate method so that it is not necessary to calculate every statistic when it is not needed. The later of the two functions will return the results as a dictionary, so that it is easy to identify which statistic is which. The percentage function takes in a count and a total as parameter, while the other functions take in a list of numbers.

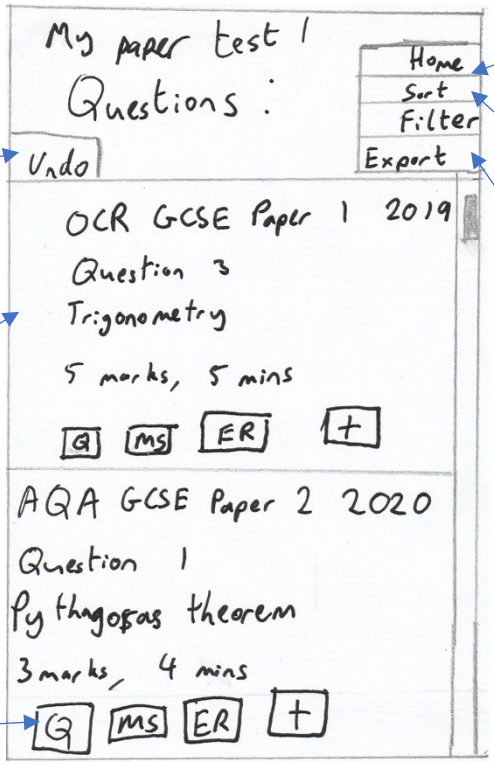
```
FUNCTION Percentage(Count, Total):
    RETURN Count/Total*100
```

```
FUNCTION MainStats(List):
    Dict = {
        “Mean”: 0,
        “Variance”: 0,
        “StandardDeviation”: 0
    }
    Total = 0
    SquareTotal = 0
    FOR x IN List:
        Total = Total + x
        SquareTotal = SquareTotal + x^2
    Dict[“Mean”] = Total / LENGTH(List)
    Dict[“Variance”] = (SquareTotal / LENGTH(List)) – (Dict[“Mean”])^2
    Dict[“StandardDeviation”] = sqrt(Dict[“Variance”])
    RETURN Dict

FUNCTION OtherStats(List):
    Dict = {
                "Mode" : 0
                "Median" : 0
                “Q1”: 0
                “Q3”: 0
                “Range”: 0
                “IQRange”: 0
                “IPRange”: 0
    }
    Count = []
    FOR x IN List:
        Found = FALSE
        FOR y IN RANGE(LENGTH(Count)):
            IF Count[y][0] IS x:
                Found = TRUE
                Count[y][1] = Count[y][1] + 1
        IF Found IS FALSE:
            Count.append([x, 1])
    merge_sort(Count) # See page 39
    Dict[“Mode”] = Count[0][0]
    SortedList = []
    FOR x in List:
        SortedList.append([0, x])
    merge_sort(SortedList) # See page 39
    FOR x IN RANGE(LENGTH(SortedList)):
        SortedList[x] = SortedList[x][1]
    Dict["Q1"] = SortedList[INT(LENGTH(SortedList)*0.25 + 0.5) – 1]
    Dict[“Median”] = SortedList[INT(LENGTH(SortedList)*0.5 + 0.5) – 1]
    Dict[“Q3”] = SortedList[INT(LENGTH(SortedList)*0.75 + 0.5) – 1]
    Dict[“Range”] = SortedList[LENGTH(SortedList) – 1] – SortedList[0]
```

```
    Dict["IQRange"] = Dict["Q3"] – Dict["Q1"]
    Dict["IPRange"] = SortedList[INT(LENGTH(SortedList)*0.9 + 0.5) – 1] -
SortedList[INT(LENGTH(SortedList)*0.1 + 0.5) – 1]
    RETURN Dict
```

User interface



Easy to find buttons at the top of the page

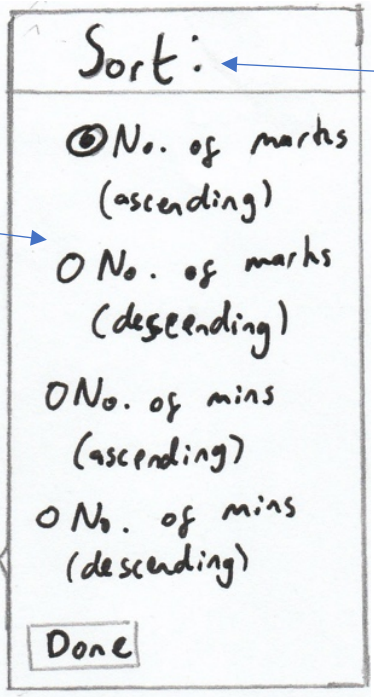
Undo button – see page 41

Sort using merge sort – see page 39

Export easily accessible – Objective 4a (more info on algorithm on page 43)

Questions in a scrollbar

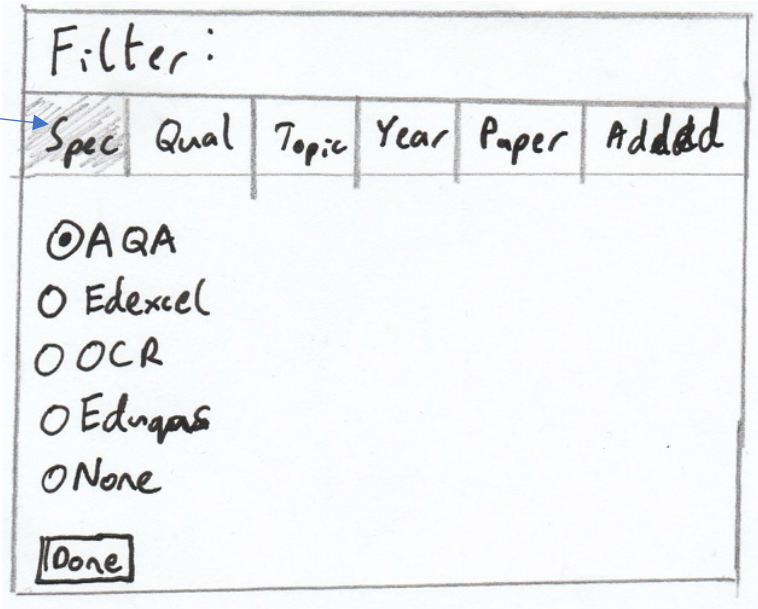
Question, mark scheme and examiner's report viewable – Objective 3c

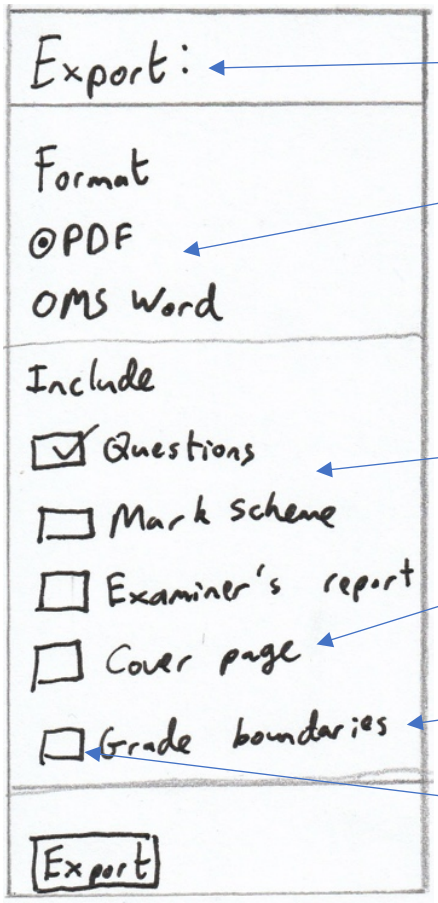


Sort using merge sort – see page 39

Sorting options of marks and minutes – Objective 3d

Filter options – Objective 3d





Export – see page 43

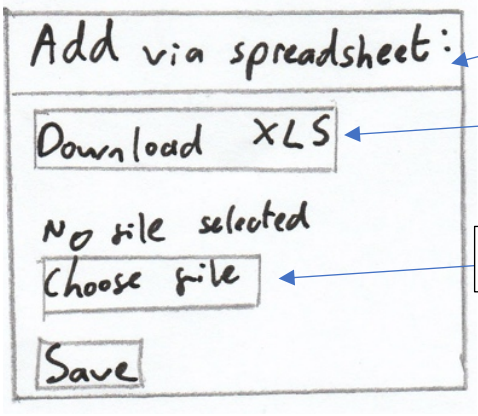
Two export formats of PDF and Microsoft Word – Objective 4b

Options to export questions, mark scheme and/or examiner's report – Objective 4c

Option to export cover page – Objective 4e

Option to export grade boundaries – Objective 4d

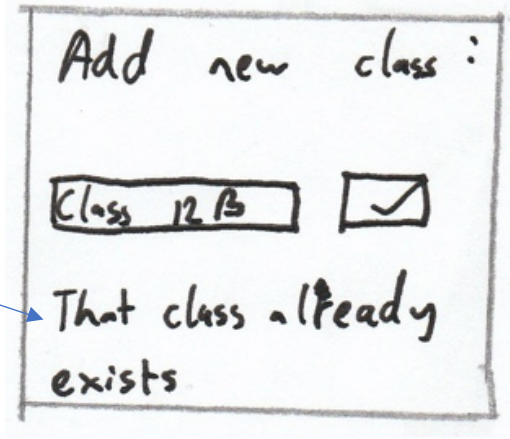
All options are easily selectable and deselect able – Objective 4f



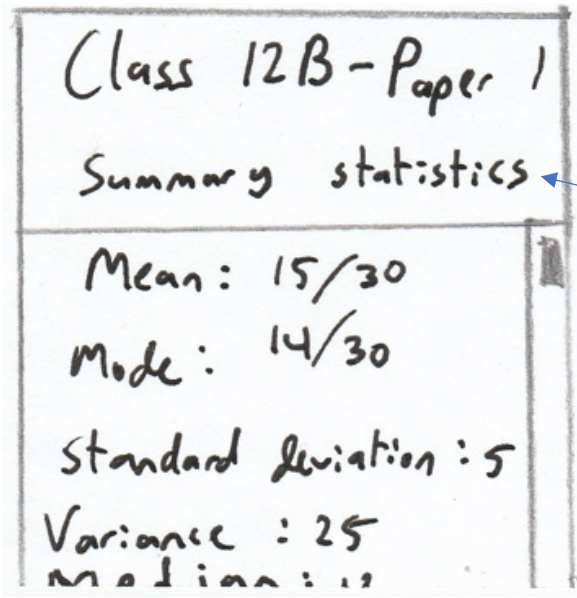
Add marks via spreadsheet

Large button to download XLS template

Upload file

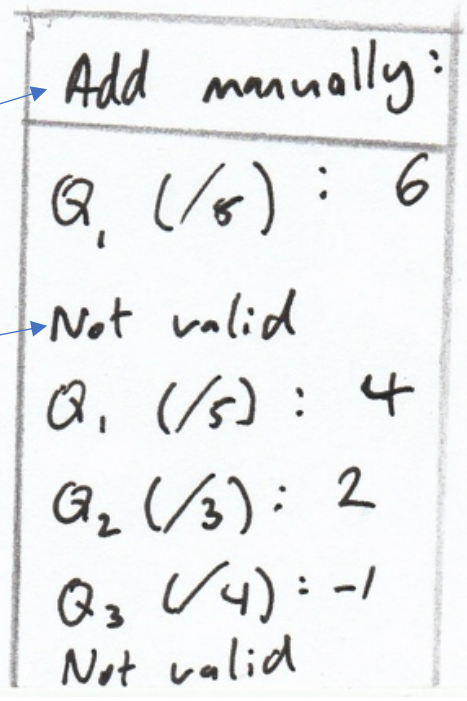


Data validation when attempting to add pre-existing class



Summary statistics – see page 45

Add marks manually – Objective 5a



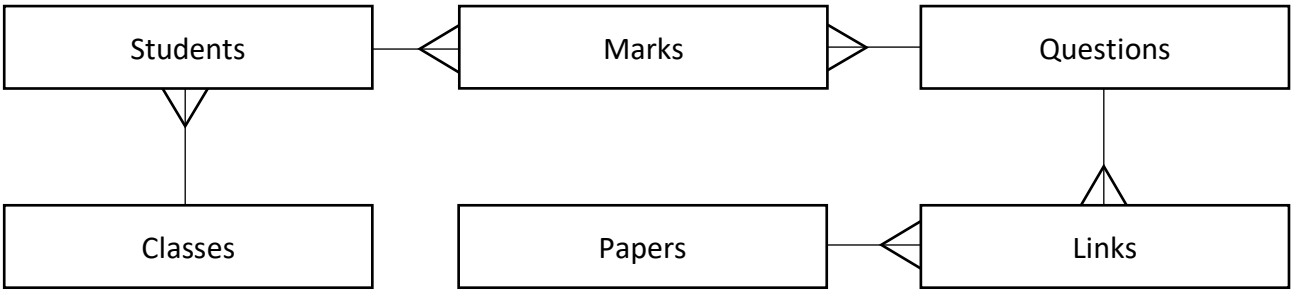
Data validation when invalid marks are added – Objective 5b

Database design

There is a lot of information that needs to be stored about the questions, papers, students, classes, and marks. I will be using a relational database to store this data, alongside SQL to query the database, allowing me to retrieve, add and remove data from the database. I will be using the sqlite3 module in python to allow me to run SQL queries from within the software.

Using a database is the most efficient way to store the data and quickly and easily access it. An alternative solution would have been to use text files, which would be cluttered and messy and be very hard to edit manually. An advantage of using a database is not only can you work with it using SQL, but there are also many open-source software that you can install to easily change, manipulate, add, and remove data from the database in a user-friendly GUI format. The software I will be using is called "DB Browser for SQLite", and it can be downloaded at this link: https://sqlitebrowser.org/

Below is an entity relationship diagram and entity descriptions that show how the database will be organised:



Classes (ClassID, ClassName)
Links (LinksID, PaperID, QuestionID, QuestionOrder)
Marks (StudentID, PaperID, QuestionID, MarksAchieved)
Papers (PaperID, PaperName)
Questions (QuestionID, QuestionUnit, QuestionTopic, QuestionMarks, QuestionMins, ExamBoard, Qualification, Year, QuestionPaper, Question, MarkScheme, ExaminersReport)
Students (StudentID, FirstName, Surname, ClassID)

Technical solution

```
### Faizan Arshad
### AQA A-Level Computer Science (7517) NEA 2023
### Candidate number: 6018
### Centre number: 13327

# SQLite is used to query the database that is used to store the details about
questions, papers, students, and marks
# This includes selecting data, removing data, and inserting the data to and from
the database
import sqlite3
from sqlite3 import Error

# Sys is used for accessing files (.txt and .png files)
# This is needed to allow the program to be able to access the files that contain
the questions, mark schemes and examiner's reports
import sys

# Pillow is used to view PNG images
# This allows the user to view image questions from within the program by opening a
separate window that displays the question/mark schemes/examiner's report
from PIL import Image

# fpdf is used to export papers into PDFs
# These pdfs are created within the createPDF class
from fpdf import FPDF

# Used to convert pdf files to docx files
# When a user wants to export a paper into a docx file, the program first creates a
pdf file, converts this to docx, and the deletes the pdf file
from pdf2docx import Converter

# Used to remove unwanted files
# This is specifically used to delete the pdf files in the process described above
import os

# Used to print out tables
# Pandas dataframe allows me to show data in a table when analysing marks in an
aesthetically pleasing and easy to view format
import pandas as pd

# Used to copy data structures
# This is specifically used to deep copy the priority queue of questions within a
paper into 3 separate, unlinked priority queues to use to export the questions,
mark schemes and examiner's reports
# This makes sure that when an item is dequeued from one queue, the other queues
are not affected
import copy

# Used for summary statistics
# Specifically used to obtain the ability to square root a value when calculating
standard deviation
```

```
import math

# This is used to create and display the graphs in the analyse section
# This includes bar charts when analysing a student and scatter graphs when
analysing a class
import matplotlib.pyplot as plt

# This class contains the main part of the program, and an object is created at the
start of the program
# It only contains the constructor, which is where all the code for the menu is
located
class menu:
    def __init__(self):
        self.option = 1
        # Loops until the user decides to exit
        while self.option != 0:
            try:
                self.option = int(input('''MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: '''))
            except:
                print()
                print('That is not a valid option (Error code 2)')
                print()
            else:
                # Creating a new paper
                if self.option == 1:
                    name = None
                    while name == None:
                        name = input("What would you like to name this paper? ")
                        # Checks if the name already exists
                        if name in data.getNames():
                            print('A paper with that name already exists')
                            print()
                            name = None
                        # Checks if name entered is an empty string and therefore
not valid
                        if name == '' or name == ' ':
                            print('That is not a valid name')
```

```
                            name = None
                    # If the name does not already exist, an empty paper object is
created, added to the database and the edit screen is loaded to allow the user to
begin adding questions to the empty paper
                    current_paper = paper(name, data.generateID())
                    current_paper.addPaper()
                    current_paper.editPaper()
                    print()
                # Editing an existing paper
                elif self.option == 2:
                    # Gets the names of all the available papers from the database
                    papers = data.getPapers()
                    option = None
                    # Choosing which paper to edit
                    while option is None:
                        try:
                            print()
                            print('Please choose a paper:')
                            print()
                            options = []
                            count = 0
                            # Prints list of names of all the papers
                            for x in papers:
                                count += 1
                                print(str(count) + '. ' + x[1])
                                options.append(x[0])
                            print('0. Go back')
                            print()
                            option = int(input(''))
                            # Checks if user entered a valid option
                            if option > 0 and option < count + 1:
                                # PaperID of the selected paper is retrieved
                                ID = options[option-1]
                                # Paper object is created for the selected paper
                                current_paper = paper(papers[option-1][1], ID)
                                # Paper object is populated with questions in the
paper by retrieving questions from the database
                                current_paper.initialiseQuestions()
                                # Edit screen is loaded
                                current_paper.editPaper()
                            elif option != 0:
                                print('That is not a valid option')
                                option = None
                        except ValueError:
                            print()
                            print('That is not a valid option (Error code 3)')
                            option = None
                    print()
                # Removing a paper
                elif self.option == 3:
                    # Gets the names of all the available papers from the database
```

```
                    papers = data.getPapers()
                    option = None
                    # Choosing which paper to remove
                    while option is None:
                        try:
                            print()
                            print('Please choose a paper to remove:')
                            print()
                            num = 0
                            # Prints list of names of all the papers
                            for x in papers:
                                print(str(num + 1) + '. ' + x[1])
                                num += 1
                            print('0. Go back')
                            print()
                            option = int(input(''))
                            # Checks if user entered a valid option
                            if option > 0 and option <= num:
                                # Prompts user for confirmation before continuing
to remove the paper
                                print()
                                print('Are you sure you would like to remove this
paper?')
                                confirmation = input('Type "YES" to confirm: ')
                                # Must enter "YES" to start the deletion process
                                if confirmation == 'YES':
                                    # PaperID of the paper to remove is retrieved
                                    ID = papers[option-1][0]
                                    # Paper details removed from database
                                    data.removePaper(ID)
                                    # Database is saved
                                    data.save()
                                    # Paper is removed from papers list
                                    papers.remove(papers[option-1])
                                    # Message to indicate that the deletion process
has completed and that the paper has been removed
                                    print('Removed')
                                # If required confirmation is not given, the
deletion process is cancelled, and the paper is not removed
                                else:
                                    print('Deletion cancelled')
                                print()
                            elif option != 0:
                                print('That is not a valid option')
                                option = None
                        except ValueError:
                            print()
                            print('That is not a valid option')
                            option = None
                    print()
                # Adding/editing marks
```

```
        elif self.option == 4:
            # Gets all available classes from database
            classes = data.getClasses()
            print()
            # Choosing a class to add/edit marks from
            IDs = []
            count = 1
            for x in classes:
                print(str(count) + '. ' + x[1])
                count += 1
                IDs.append(x[0])
            count -= 1
            print('0. Go back')
            print()
            choice = ''
            while choice == '':
                try:
                    choice = int(input('Enter an option: '))
                except:
                    print('That is not an option')
                    print()
                else:
                    if choice < 0 or choice > count:
                        print('That is not an option')
                        print()
                        choice = ''
                    elif choice != 0:
                        # Getting the class ID
                        choice = IDs[choice-1]
                        print()
                        # Getting all the students in the class
                        students = data.getStudents(choice)
                        # Choosing a student
                        count = 1
                        IDs = []
                        for x in students:
                            print(str(count) + '. ' + x[1] + ' ' + x[2])
                            IDs.append(x[0])
                            count += 1
                        count -= 1
                        print('0. Go back')
                        print()
                        choice = ''
                        while choice == '':
                            try:
                                choice = int(input('Enter an option: '))
                            except:
                                print()
                                print('That is not an option')
                            else:
                                if choice < 0 or choice > count:
```

```
                print()
                print('That is not an option')
                choice = ''
            elif choice != 0:
                # Getting the student ID
                studentID = IDs[choice-1]
                # Getting all the available papers
                papers = data.getPapers()
                # Choosing a paper
                paperID = None
                while paperID is None:
                    try:
                        print()
                        print('Please choose a paper:')
                        print()
                        IDs = []
                        count = 0
                        for x in papers:
                            count += 1
                            print(str(count) + '. ' +
x[1])
                            IDs.append(x[0])
                        print('0. Go back')
                        paperID = int(input(''))
                        if paperID < 0 or paperID >
count:
                            print('That is not a valid
option')
                            paperID = None
                        elif paperID != 0:
                            # Getting the paper ID
                            paperID = IDs[paperID-1]
                            print()
                            # Removing all pre-existing
marks for that student and paper
                            data.removeMarks(studentID,
paperID)
                            questionNo = 0
                            # Looping through each
question
                            for x in
data.getQuestionsinPaper(paperID):
                                questionNo += 1
                                print('Question ' +
str(questionNo))
                                print('Out of ' +
str(x[1]))
                                # Adding the marks
                                while True:
                                    try:
```

```
                                                    marks =
int(input('Marks achieved: '))
                                                except:
                                                    print('That is
not a valid option')
                                                    print()
                                                else:
                                                    # Marks cannot
be less than 0 or more than the total mark for the question
                                                    if marks < 0 or
marks > x[1]:
                                                        print('That
is not a valid option')
                                                        print()
                                                    # Adding the
marks to the database
                                                    else:
data.addMarks(studentID, paperID, x[0], marks)
                                                        print()
                                                        break
                                            data.save()
                                            print('Marks added')
                                    except:
                                        print()
                                        print('That is not a valid
option')
                                        option = None
                    print()
                # Analysing
                elif self.option == 5:
                    # Initialising the analyse class
                    analyse = analysedata()
                    # Choosing to analyse by class or student
                    print()
                    print('Please select an option:')
                    print('1. Analyse by class')
                    print('2. Analyse by student')
                    print('0. Go back')
                    option1 = None
                    while option1 != 1 and option1 != 2 and option1 != 0:
                        try:
                            print()
                            option1 = int(input(''))
                        except:
                            print('That is not a valid option')
                        else:
                            if option1 != 1 and option1 != 2 and  option1 != 0:
                                print('That is not a valid option')
                    # Choosing to analyse by paper or question
                    if option1 == 1 or option1 == 2:
```

```
                        print()
                        print('Please select an option:')
                        print('1. Analyse by paper')
                        print('2. Analyse by question')
                        print('0. Go back')
                        print()
                        option2 = None
                        while option2 != 1 and option2 != 2 and option2 != 0:
                            try:
                                option2 = int(input(''))
                            except:
                                print('That is not a valid option')
                                print()
                            else:
                                if option2 != 1 and option2 != 2 and  option2 != 0:
                                    print('That is not a valid option')
                        # Choosing to display a table, graph, or summary statistics
                        if option2 == 1 or option2 == 2:
                            print()
                            print('Please select an option:')
                            print('1. Table')
                            print('2. Graph')
                            print('3. Summary statistics')
                            print('0. Go back')
                            print()
                            option3 = None
                            while option3 != 1 and option3 != 2 and option3 != 3
and option3 != 0:
                                try:
                                    option3 = int(input(''))
                                except:
                                    print('That is not a valid option')
                                    print()
                                else:
                                    if option3 != 1 and option3 != 2 and option3 !=
3 and  option3 != 0:
                                        print('That is not a valid option')
                            if option3 == 1 or option3 == 2 or option3 == 3:
                                # Choosing a class
                                classes = data.getClasses()
                                print()
                                print('Please select a class:')
                                IDs = []
                                count = 1
                                for x in classes:
                                    print(str(count) + '. ' + x[1])
                                    count += 1
                                    IDs.append(x[0])
                                count -= 1
                                print('0. Go back')
                                print()
```

```
                                selectedclass = ''
                                while selectedclass == '':
                                    try:
                                        selectedclass = int(input('Enter an option:
'))
                                    except:
                                        print('That is not an option')
                                    else:
                                        if selectedclass < 0 or selectedclass >
count:
                                            print('That is not an option')
                                            selectedclass = ''
                                        elif selectedclass != 0:
                                            selectedclass = IDs[selectedclass-1]
                                            # Analyse by class
                                            if option1 == 1:
                                                # Analyse by class and paper
                                                if option2 == 1:
                                                    # Retrieving the data
                                                    paperlist, percentages =
analyse.classandpaper(selectedclass)
                                                    if len(percentages) == 0:
                                                        print('There are no marks
available')
                                                    else:
                                                        # Printing a table
                                                        if option3 == 1:
                                                            outputtable =
table([percentages], paperlist, ['Average percentage'])
                                                            placeholder =
input('Press enter to close')
                                                        # Making a bar graph
                                                        elif option3 == 2:
                                                            c = []
                                                            for x in percentages:
                                                                if x <= 20:
                                                                    c.append('red')
                                                                elif x <= 40:

c.append('orange')
                                                                elif x <= 60:

c.append('yellow')
                                                                elif x <= 80:

c.append('lightgreen')
                                                                else:

c.append('darkgreen')
                                                            plt.bar(paperlist,
height = percentages, color = c)
```

```
                    plt.ylabel('Average
percentage')
                    plt.show()
                # Summary statistics
                else:
                    print('Press enter to
go to next page')
                    print()
                    dict =
ss.mainstats(percentages)
                    placeholder = input()
                    dict =
ss.otherstats(percentages)
                    placeholder = input()
            # Analyse by class and question
            else:
                # Getting all papers
                papers = data.getPapers()
                paperid = None
                # Choosing a paper to analyse
                while paperid is None:
                    try:
                        paperIDs = []
                        count = 0
                        print()
                        print('Please choose a
paper:')
                        print()
                        for x in papers:
                            count += 1
                            print(str(count) +
'. ' + x[1])
                            paperIDs.append(x[0])
                        print('0. Go back')
                        print()
                        paperid =
int(input(''))
                    except ValueError:
                        print()
                        print('That is not a
valid option (Error code 1)')
                        paperid = None
                    else:
                        if paperid < 0 or
paperid > count:
                            print('That is not
a valid option')
                        elif paperid != 0:
                            # Getting the paper
ID
```

```
paperid =
paperIDs[paperid-1]
# Retrieving the
data
qlist, studentlist,
percentages = analyse.classandquestion(selectedclass, paperid)
if len(percentages)
== 0:
    print('There
are no marks available')
else:
    # Printing a
table
    if option3 ==
1:
        outputtable
= table(percentages, studentlist, qlist)
        placeholder
= input('Press enter to close')
    # Making a
scatter graph
    elif option3 ==
2:
        for index
in range(len(percentages[0])):
            y = []
            for x
in percentages:
                x =
y.append(x[index])
qlist
plt.scatter(x, y)
plt.ylabel('Percentage')
            plt.show()
    # Summary
statistics
    else:
print('Press enter to go to next page')
        print()
        ls = []
        for x in
percentages:
            for y
in x:
ls.append(y)
```

```
                                                                    dict =
ss.mainstats(ls)
                                                                    placeholder
= input()
                                                                    dict =
ss.otherstats(ls)
                                                                    placeholder
= input()
                                                # Analyse by student
                                                else:
                                                    # Getting the students in the class
                                                    students =
data.getStudents(selectedclass)
                                                    # Choosing a student
                                                    count = 1
                                                    IDs = []
                                                    for x in students:
                                                        print(str(count) + '. ' + x[1]
+ ' ' + x[2])
                                                        count += 1
                                                        IDs.append(x[0])
                                                    count -= 1
                                                    print('0. Go back')
                                                    print()
                                                    selectedstudent = ''
                                                    while selectedstudent == '':
                                                        try:
                                                            selectedstudent =
int(input('Enter an option: '))
                                                        except:
                                                            print('That is not an
option')
                                                        else:
                                                            if selectedstudent < 0 or
selectedstudent > count:
                                                                print('That is not an
option')
                                                                selectedstudent = ''
                                                            elif selectedstudent != 0:
                                                                # Getting the student
ID
                                                                selectedstudent =
IDs[selectedstudent-1]
                                                                # Analyse by student
and paper
                                                                if option2 == 1:
                                                                    # Retrieving the
data
                                                                    paperlist,
percentages = analyse.studentandpaper(selectedstudent)
```

```
if len(percentages)
== 0:
    print('There
are no marks available')

# Printing a table
if option3 == 1:
    outputtable =
table([percentages], paperlist, ['Percentage achieved'])
    placeholder =
input('Press enter to close')
# Making a bar
graph
elif option3 == 2:
    c = []
    for x in
percentages:
        if x <= 20:
c.append('red')
        elif x <=
40:
c.append('orange')
        elif x <=
60:
c.append('yellow')
        elif x <=
80:
c.append('lightgreen')
        else:
c.append('darkgreen')
plt.bar(paperlist, height = percentages, color = c)
plt.ylabel('Percentage score')
    plt.show()
# Summary
statistics
else:
    print('Press
enter to go to next page')
    print()
    dict =
ss.mainstats(percentages)
    placeholder =
input()
    dict =
ss.otherstats(percentages)
```

```
                                                                placeholder =
input()
                                                            # Analyse by student
and question
                                                            else:
                                                                # Getting all the
papers
                                                                papers =
data.getPapers()
                                                                paperid = None
                                                                # Choosing a paper
to analyse
                                                                while paperid is
None:
                                                                    try:
                                                                        print()
print('Please choose a paper:')
                                                                        print()
                                                                        for x in
papers:
print(str(x[0]) + '. ' + x[1])
                                                                        print()
                                                                        paperid =
int(input(''))
                                                                    except
ValueError:
                                                                        print()
                                                                        print('That
is not a valid option (Error code 4)')
                                                                        paperid =
None
                                                                # Retrieving the
data
                                                                percentages, www,
ebi, qlist = analyse.studentandquestion(selectedstudent, paperid)
                                                                if len(percentages)
== 0:
                                                                    print('There
are no marks available')
                                                                else:
                                                                    # Printing a
table
                                                                    if option3 ==
1:
                                                                        outputtable
= table([percentages], qlist, ['Percentage achieved'])
                                                                        placeholder
= input('Press enter to close')
```

```
                                                    # Making a bar
graph
                                                    elif option3 ==
2:
                                                        c = []
                                                        for x in
percentages:
                                                            if x <=
20:
c.append('red')
                                                            elif x
<= 40:
c.append('orange')
                                                            elif x
<= 60:
c.append('yellow')
                                                            elif x
<= 80:
c.append('lightgreen')
                                                            else:
c.append('darkgreen')
plt.bar(qlist, height = percentages, color = c)
plt.ylabel('Percentage score')
                                                        plt.show()
                                                    # Summary
statistics
                                                    else:
print('Press enter to go to next page')
                                                        print()
                                                        dict =
ss.mainstats(percentages)
                                                        placeholder
= input()
                                                        dict =
ss.otherstats(percentages)
                                                        placeholder
= input()
                                                        # Printing
the WWW and EBI
                                                        # If WWW =
EBI, then none are generated
                                                        if www ==
ebi:
```

```
print('A WWW and EBI could not be generated')
                                                                    else:

print('WWW: ' + www)
                                                                    print('EBI:
' + ebi)
                                                                    placeholder
= input()
                    print()
                # Add/remove a class/student
                elif self.option == 6:
                    while True:
                        try:
                            # Choosing an option
                            print('1. Add a student')
                            print('2. Remove a student')
                            print('3. Add a class')
                            print('4. Remove a class')
                            print('0. Go back')
                            print()
                            option = int(input('Please choose an option: '))
                            print()
                            # Adding a student
                            if option == 1:
                                data.addStudent()
                            # Removing a student
                            elif option == 2:
                                data.removeStudent()
                            # Adding a class
                            elif option == 3:
                                data.addClass()
                            # Removing a class
                            elif option == 4:
                                data.removeClass()
                            elif option != 0:
                                continue
                            break
                        except:
                            print('That is not a valid option')
                            print()
                    print()
                elif self.option != 0:
                    print('That is not a valid option (Error code 5)')
                    print()

# Class to manage the database
# For program to work database must be stored in the same folder as the program and
must be named "database.db"
class database:
    # Initiates the SQLite database by attempting to establish a connection
```

```
    def __init__(self, path):
        self.path = path
        self.connection = None
        try:
            self.connection = sqlite3.connect(path)
        except Error as e:
            print(f"The error '{e}' occurred")

    # This method allows us to run SQL queries on the database
    def execute_read_query(self, query):
        # The cursor allows us to input the SQL queries
        cursor = self.connection.cursor()
        result = None
        # Attempts to execute the query and return the result
        try:
            cursor.execute(query)
            result = cursor.fetchall()
            return result
        except Error as e:
            print(f"The error '{e}' occurred")

    # Saves the database by committing all changes
    def save(self):
        self.connection.commit()
        # Connection to the database is re-established after committing changes
        self.__init__(self.path)

    # Returns all papers sorted by ID
    def getPapers(self):
        return self.execute_read_query('SELECT * FROM Papers ORDER BY PaperID')

    # Adds a new paper into the database
    def addPaper(self, name, ID):
        self.execute_read_query('INSERT INTO Papers VALUES (' + str(ID) + ', "' +
name + '")')

    # Removes a paper from the database
    # This includes removing all records
    def removePaper(self,ID):
        self.execute_read_query("DELETE FROM Papers WHERE PaperID=" + str(ID))
        self.execute_read_query("DELETE FROM Links WHERE PaperID=" + str(ID))
        self.execute_read_query("DELETE FROM Marks WHERE PaperID=" + str(ID))

    # Returns the lowest unused PaperID
    def generateID(self):
        currentID = 0
        while currentID in [i[0] for i in self.execute_read_query('SELECT PaperID
FROM Papers')]:
            currentID += 1
        return currentID
```

```
    # Returns names of all papers
    def getNames(self):
        return [i[0] for i in self.execute_read_query('SELECT PaperName FROM
Papers')]

    # Returns all questions using filters
    def getQuestions(self, unit, topic, board, qualification, year, paper,
orderby):
        return self.execute_read_query("SELECT * FROM Questions WHERE QuestionUnit
LIKE '" + unit + "' AND QuestionTopic LIKE '" + topic + "' AND ExamBoard LIKE '" +
board + "' AND Qualification LIKE '" + qualification + "' AND Year LIKE '" +
str(year) + "' AND QuestionPaper LIKE '" + paper + "' ORDER BY " + orderby)

    def getQuestionsinPaper(self, ID):
        return self.execute_read_query("SELECT Questions.QuestionID, QuestionMarks
FROM Questions, Links WHERE Questions.QuestionID=Links.QuestionID AND PaperID=" +
str(ID) + " ORDER BY QuestionOrder")

    # Returns all question units
    def getUnits(self):
        return [i[0] for i in self.execute_read_query('SELECT DISTINCT QuestionUnit
FROM Questions')]

    # Returns all topics in a specified question unit
    def getTopics(self, unit):
        return [i[0] for i in self.execute_read_query("SELECT DISTINCT
QuestionTopic FROM Questions WHERE QuestionUnit LIKE '" + unit + "'")]

    # Returns all exam boards for which there is a question from the specific topic
and/or unit
    def getBoards(self, unit, topic):
        return [i[0] for i in self.execute_read_query("SELECT DISTINCT ExamBoard
FROM Questions WHERE QuestionUnit LIKE '" + unit + "' AND QuestionTopic LIKE '" +
topic + "'")]

    # Returns all qualifications for which there is a question from the specific
topic and/or unit and/or exam board
    def getQualifications(self, unit, topic, board):
        return [i[0] for i in self.execute_read_query("SELECT DISTINCT
Qualification FROM Questions WHERE QuestionUnit LIKE '" + unit + "' AND
QuestionTopic LIKE '" + topic + "' AND ExamBoard LIKE '" + board + "'")]

    # Returns all years for which there is a question from the specific topic
and/or unit and/or exam board and/or qualification
    def getYears(self, unit, topic, board, qualification):
        return [i[0] for i in self.execute_read_query("SELECT DISTINCT Year FROM
Questions WHERE QuestionUnit LIKE '" + unit + "' AND QuestionTopic LIKE '" + topic
+ "' AND ExamBoard LIKE '" + board + "' AND Qualification LIKE '" + qualification +
"'")]
```

```
    # Returns all exam papers for which there is a question from the specific topic
and/or unit and/or exam board and/or qualification and/or year
    def getQuestionPapers(self, unit, topic, board, qualification, year):
        return [i[0] for i in self.execute_read_query("SELECT DISTINCT
QuestionPaper FROM Questions WHERE QuestionUnit LIKE '" + unit + "' AND
QuestionTopic LIKE '" + topic + "' AND ExamBoard LIKE '" + board + "' AND
Qualification LIKE '" + qualification + "' AND Year LIKE '" + str(year) + "'")]

    # Returns all classes
    def getClasses(self):
        return [self.execute_read_query("SELECT * FROM Classes")][0]

    # Returns all students where they are in the class specified by the given class
ID
    def getStudents(self, classid):
        return [self.execute_read_query("SELECT * FROM Students WHERE ClassID=" +
str(classid))][0]

    # Deletes all marks where the mark belongs to the student and paper specified
by given student ID and paper ID respectively
    def removeMarks(self, studentid, paperid):
        self.execute_read_query('DELETE FROM Marks WHERE StudentID=' +
str(studentid) + ' AND PaperID=' + str(paperid))

    # Adds a record into the marks table that specifies the mark achieved by a
particular student for a particular question in a particular paper
    def addMarks(self, studentid, paperid, questionid, marks):
        self.execute_read_query('INSERT INTO Marks VALUES (' + str(studentid) + ',
' + str(paperid) + ', ' + str(questionid) + ', ' + str(marks) +')')

    # Returns the position of a question within a paper
    def getQuestionOrder(self, paperid, questionid):
        return self.execute_read_query("SELECT QuestionOrder FROM Links WHERE
PaperID=" + str(paperid) + " AND QuestionID=" + str(questionid))

    # Adds a student into the student table
    # First the user chooses what class to add the student too
    # Then they can enter the first name and surname of the student
    # Finally, the student is added to the database
    def addStudent(self):
        classes = self.getClasses()
        IDs = []
        count = 0
        for x in classes:
            count += 1
            print(str(count) + '. ' + x[1])
            IDs.append(x[0])
        print('0. Go back')
        print()
        choice = ''
        while choice == '':
```

```
            try:
                choice = int(input('Enter an option: '))
            except:
                print('That is not an option')
            else:
                if choice < 0 or choice > count:
                    print('That is not an option')
                    choice = ''
                elif choice != 0:
                    choice = IDs[choice-1]
                    firstname = input('First name: ')
                    surname = input('Surname: ')
                    self.execute_read_query("INSERT INTO Students (FirstName,
Surname, ClassID) VALUES ('" + firstname + "','" + surname + "'," + str(choice) +
")")
                    self.save()

    # Returns a list of all students
    # This list contains the student ID, first name and surname for each student
    def _getAllStudents(self):
        students = self.execute_read_query('SELECT StudentID, FirstName, Surname
FROM Students')
        newstudents = []
        for x in students:
            temp = []
            for y in x:
                temp.append(y)
            newstudents.append(temp)
        return newstudents

    # Removes a student
    # First the user chooses a student to remove
    # Then the user confirms if they really want to remove the student
    # If confirmed then all details of the student are removed from the marks and
student tables
    def removeStudent(self):
        while True:
            try:
                students = self._getAllStudents()
                nums = []
                count = 0
                for x in students:
                    count += 1
                    nums.append(x[0])
                    print(str(count) + '. ' + x[1] + ' ' + x[2])
                print('0. Go back')
                print()
                student = int(input('Please select an option: '))
            except:
                print('That is not a valid option')
                print()
```

```
            else:
                if student > 0 and student < count + 1:
                    student = nums[student-1]
                    print()
                    print('Are you sure you want to remove this student?')
                    print('This will remove all record of marks from this student')
                    print()
                    confirmation = input('To confirm deletion, please type YES: ')
                    if confirmation == 'YES':
                        self.execute_read_query('DELETE FROM Marks WHERE
StudentID=' + str(student))
                        self.execute_read_query('DELETE FROM Students WHERE
StudentID=' + str(student))
                        self.save()
                        print('Deleted')
                    else:
                        print('Deletion cancelled')
                    break
                elif student != 0:
                    print('That is not a valid option')
                    print()
                else:
                    break

    # Adding a class of the name that the user inputs
    def addClass(self):
        classname = input('Enter a name for the class: ')
        self.execute_read_query("INSERT INTO Classes (ClassName) VALUES ('" +
classname + "')")
        self.save()
        print('Added')

    # Removes a class
    # First the user chooses a class to remove
    # Then the user confirms if they really want to remove the class
    # If confirmed, all the marks of the students of the class are removed from the
marks table
    # Then all the students in that class are removed from the student table
    # Finally, the class is removed from the classes table
    def removeClass(self):
        classes = self.getClasses()
        for x in range(len(classes)):
            temp = []
            for y in classes[x]:
                temp.append(y)
            classes[x] = temp
        while True:
            try:
                count = 1
                classIDs = []
                for x in classes:
```

```
                    classIDs.append(x[0])
                    print(str(count) + '. ' + x[1])
                    count += 1
                count
                print('0. Go back')
                print()
                selectedclass = int(input('Please choose an option: '))
                break
            except:
                print('That is not a valid option')
                print()
        if selectedclass > 0 and selectedclass < count + 1:
            selectedclass = classIDs[selectedclass-1]
            print('Are you sure you would like to remove this class')
            print('Removing the class will delete all students associated with it
and their record of marks')
            print()
            confirmation = input('Enter YES to confirm deletion: ')
            if confirmation == 'YES':
                students = self.getStudents(selectedclass)
                for x in students:
                    self.execute_read_query('DELETE FROM Marks WHERE StudentID=' +
str(x[0]))
                self.execute_read_query('DELETE FROM Students WHERE ClassID=' +
str(selectedclass))
                self.execute_read_query('DELETE FROM Classes WHERE ClassID=' +
str(selectedclass))
                self.save()
                print('Deleted')
            else:
                print('Deletion cancelled')
        elif selectedclass != 0:
            print('That is not a valid option')
            print()
            self.removeClass()

# This represents the paper that the user creates/edits (not the exam paper from
where the questions are sourced)
class paper:
    def __init__ (self, name, ID):
        self._name = name
        self._ID = ID
        # List stores all questions in the paper
        self._questions = []
        # This stack stores any changed made to the paper, allowing for an undo
feature
        self._changes = stack()

    # Returns the ID of the paper
    def getID(self):
        return self._ID
```

```
    # Queries the database for all the questions that are within the specific paper
    def initialiseQuestions(self):
            list = data.execute_read_query("SELECT Questions.QuestionID,
QuestionUnit, QuestionTopic, QuestionMarks, QuestionMins, ExamBoard, Qualification,
Year, QuestionPaper, Question, MarkScheme, ExaminersReport FROM Questions, Links
WHERE Links.QuestionID = Questions.QuestionID AND Links.PaperID=" + str(self._ID) +
" ORDER BY Links.QuestionOrder")
            print()
            print('Questions in paper:')
            totalmark = 0
            totalmins = 0
            for x in range(len(list)):
                templist = []
                for y in list[x]:
                    templist.append(y)
                if templist[3] == 1:
                    templist[3] = '1 mark'
                else:
                    templist[3] = str(templist[3]) + ' marks'
                if templist[4] == 1:
                    templist[4] = '1 min'
                else:
                    templist[4] = str(templist[4]) + ' mins'
                print(str(x+1) + '.', templist[1:9])
                totalmark += list[x][3]
                totalmins += list[x][4]
            print()
            print('Total mark - ' + str(totalmark))
            print('Total mins - ' + str(totalmins))
            for x in list:
                # Adds the question into the list as either a text question if the
question is stored as a txt file (check if last letter is t) or an image question
if it is not
                if x[9][len(x[9])-1] == 't':
                    self._questions.append(textQuestion(x[0],x[1], x[2], x[3],
x[4], x[5], x[6], x[7], x[8], x[9], x[10], x[11]))
                else:
                    self._questions.append(imageQuestion(x[0],x[1], x[2], x[3],
x[4], x[5], x[6], x[7], x[8], x[9], x[10], x[11]))

    # Adds new paper to database and saves
    def addPaper(self):
        data.addPaper(self._name, self._ID)
        data.save()

    # This allows the user to edit a filter
    def filters(self, list, filtertype):
        while True:
            print()
            print('Filter by ' + filtertype + ':')
```

```
            count = 1
            # Displays all available options for the filer
            for x in list:
                print(str(count) + '. ' + str(x))
                count += 1
            # Option to not filter that specific metric
            print('0. Do not filter by ' + filtertype)
            try:
                option = int(input('Please choose an option: '))
                # Wildcard is used if option to not filter is chosen
                if option == 0:
                    return '%'
                # Returns relevant syntax for the filter to work in the SQL query
                else:
                    return list[option-1]
            except:
                print()
                print('That is not a valid option (Error code 6)')

    # This is the main paper editing menu screen
    # Initially, all filters are turned off and questions are sorted by marks
    def editPaper(self, unit = '%', topic = '%', board = '%', qualification = '%',
year = '%', paper = '%', orderby = 'QuestionMarks'):
        option = None
        while option != 0:
            # Getting all questions
            questions = data.getQuestions(unit, topic, board, qualification, year,
paper, orderby)
            # Formatting questions to be displayed
            question_list = []
            for x in questions:
                x = list(x)
                question_list.append(x)
            for x in question_list:
                if x[3] == 1:
                    x[3] = '1 mark'
                else:
                    x[3] = str(x[3]) + ' marks'
                if x[4] == 1:
                    x[4] = '1 min'
                else:
                    x[4] = str(x[4]) + ' mins'
            while True:
                try:
                    # Choosing an option
                    print()
                    print("Choose a question:")
                    print()
                    num = 1
                    # All questions displayed
                    for x in question_list:
```

```
                        if num < 10:
                            print(str(num) + '.  ' + str(x[1:9]))
                        else:
                            print(str(num) + '. ' + str(x[1:9]))
                        num += 1
                    # Other options are displayed
                    print(str(len(question_list) + 1) + '. Undo previous change')
                    print(str(len(question_list) + 2) + '. Edit filters and sort')
                    print(str(len(question_list) + 3) + '. Change the order of
questions')
                    print(str(len(question_list) + 4) + '. Export paper')
                    print('0. Save and quit')
                    print()
                    option = int(input('Please select an option: '))
                    if option < 0 or option > (len(question_list) + 4):
                        print('That is not a valid option (Error code 8)')
                    else:
                        break
                except ValueError:
                    print('That is not a valid option (Error code 7)')
            # Save paper
            if option == 0:
                # Deleting all existing data about the paper
                data.execute_read_query('DELETE FROM Links WHERE PaperID=' +
str(self._ID))
                counter = 1
                # Adding in new info about the paper
                for x in self._questions:
                    data.execute_read_query('INSERT INTO Links (PaperID,
QuestionID, QuestionOrder) VALUES (' + str(self._ID) + ', ' + str(x.getID()) + ', '
+ str(counter) + ')')
                    counter += 1
                # Saving the database
                data.save()
                print()
            # Undo change using stack
            elif option == len(question_list) + 1:
                # Popping latest change from the stack
                change = self._changes.pop()
                # No changes
                if change == False:
                    print('There are no changes to undo')
                # Re-adding a question
                elif change[0] == 'r':
                    self._questions.append(change[1])
                    print('Question re-added')
                # Removing a question
                elif change[0] == 'a':
                    self._questions.remove(change[1])
                    print('Question removed')
                # Unswapping 2 questions
```

```
                elif change[0] == 's':
                    self._questions[change[1]], self._questions[change[2]] =
self._questions[change[2]], self._questions[change[1]]
                    print('Unswapped')
            # Edit filters and sort
            elif option == len(question_list) + 2:
                # Asking the user to choose what filters they want
                unit = self.filters(data.getUnits(), 'unit')
                topic = self.filters(data.getTopics(unit), 'topic')
                board = self.filters(data.getBoards(unit, topic), 'exam board')
                qualification = self.filters(data.getQualifications(unit, topic,
board), 'qualification')
                year = self.filters(data.getYears(unit, topic, board,
qualification), 'year')
                paper = self.filters(data.getQuestionPapers(unit, topic, board,
qualification, year), 'paper')
                # Choosing to sort by minutes or marks
                option = 0
                while option != 1 and option != 2:
                    print()
                    print('Sort by:')
                    print('1. Marks')
                    print('2. Minutes')
                    try:
                        option = int(input('Please select an option: '))
                    except ValueError:
                        print()
                        print('That is not a valid option (Error code 10)')
                    else:
                        if option == 1:
                            orderby = 'QuestionMarks'
                        elif option == 2:
                            orderby = 'QuestionMins'
                        else:
                            print('That is not a valid option (Error code 9)')
            # Change the order of questions
            elif option == len(question_list) + 3:
                print()
                # Getting the questions in the paper
                templist = []
                for x in self._questions:
                    templist.append(x)
                # Questions cannot be swapped if 1 or no questions are in the paper
                if len(templist) < 2:
                    print('Please add more questions to the paper before swapping')
                else:
                    while True:
                        # Choosing a question to swap
                        print('Choose a question to swap:')
                        possible_options = []
                        for x in templist:
```

```
                            print(str(self._questions.index(x)+1) + '. ',
list(x.getInfo())[1])
                            possible_options.append(self._questions.index(x)+1)
                        print()
                        try:
                            option = int(input('Please select an option: '))
                        except ValueError:
                            print('That is not a valid option (Error code 12)')
                            print()
                        if option not in possible_options:
                            print('That is not a valid option (Error code 11)')
                            print()
                        else:
                            break
                    question1 = option-1
                    templist.remove(templist[option-1])
                    print()
                    while True:
                        # Choosing second question to swap with
                        print('Choose another question to swap with:')
                        possible_options = []
                        for x in templist:
                            print(str(self._questions.index(x)+1) + '. ',
list(x.getInfo())[1])
                            possible_options.append(self._questions.index(x)+1)
                        print()
                        try:
                            option = int(input('Please select an option: '))
                        except ValueError:
                            print('That is not a valid option (Error code 14)')
                            print()
                        if option not in possible_options:
                            print('That is not a valid option (Error code 13)')
                            print()
                        else:
                            break
                    question2 = option-1
                    # Swapping the questions
                    self._questions[question1], self._questions[question2] =
self._questions[question2], self._questions[question1]
                    print('Swapped')
                    # Pushing the swap onto the stack
                    self._changes.push(['s', question1, question2])
            # Export paper
            elif option == len(question_list) + 4:
                # Saving the paper before exporting
                data.execute_read_query('DELETE FROM Links WHERE PaperID=' +
str(self._ID))
                counter = 1
                for x in self._questions:
```

```
                    data.execute_read_query('INSERT INTO Links (PaperID,
QuestionID, QuestionOrder) VALUES (' + str(self._ID) + ', ' + str(x.getID()) + ', '
+ str(counter) + ')')
                    counter += 1
                data.save()
                # Can only export if there is at least 1 question in the paper
                while len(self._questions)>0:
                    # Choosing a file format
                    print()
                    print('Choose a file format:')
                    print('1. PDF')
                    print('2. DOCX')
                    print('0. Cancel export')
                    print()
                    try:
                        option = int(input('Please select an option: '))
                    except ValueError:
                        print('That is not a valid option (Error code 15)')
                    else:
                        # Export to PDF
                        # Even if DOCX is chosen, the paper is exported to a PDF
initially
                        if option == 1 or option == 2:
                            # Choosing if user wants a cover page
                            cover_page = ''
                            while cover_page != 'Y' and cover_page != 'y' and
cover_page != 'N' and cover_page != 'n':
                                cover_page = input('Would you like a cover page
(Y/N): ')
                                if cover_page != 'Y' and cover_page != 'y' and
cover_page != 'N' and cover_page != 'n':
                                    print('That is not a valid option')
                            # Setting up the priority queue
                            q = pqueue(len(self._questions))
                            for x in self._questions:
                                questionorder = data.getQuestionOrder(self._ID,
x.getID())
                                questionorder = questionorder[0][0]
                                q.enqueue(x, questionorder)
                            # Choosing if user wants mark schemes
                            ms = ''
                            while ms != 'Y' and ms != 'y' and ms != 'N' and ms !=
'n':
                                ms = input('Would you like the mark scheme (Y/N):
')
                                if ms != 'Y' and ms != 'y' and ms != 'N' and ms !=
'n':
                                    print('That is not a valid option')
                            # Choosing if user wants grade boundaries
                            gb = ''
```

```
                            while gb != 'Y' and gb != 'y' and gb != 'N' and gb !=
'n':
                                gb = input('Would you like grade boundaries (Y/N):
')
                                if gb != 'Y' and gb != 'y' and gb != 'N' and gb !=
'n':
                                    print('That is not a valid option')
                            # Choosing if user wants examiner's reports
                            er = ''
                            while er != 'Y' and er != 'y' and er != 'N' and er !=
'n':
                                er = input("Would you like the examiner's report
(Y/N): ")
                                if er != 'Y' and er != 'y' and er != 'N' and er !=
'n':
                                    print('That is not a valid option')
                            # Exporting the PDF
                            print('Exporting...')
                            pdf = createPDF(q, cover_page, ms, gb, er, self._name)
                            if option == 1:
                                print('Exported')
                            # If DOCX format is chosen, convert PDF file to DOCX
then delete PDF file
                            else:
                                # Converting to DOCX
                                pdf_file = sys.path[0] + '/' + self._name + '.pdf'
                                docx_file = sys.path[0] + '/' + self._name +
'.docx'
                                cv = Converter(pdf_file)
                                cv.convert(docx_file)
                                cv.close()
                                # Deleting the PDF
                                os.remove(self._name + '.pdf')
                                print('Exported')
                            break
                        elif option == 0:
                            break
                        else:
                            print('That is not a valid option (Error code 16)')
                        print()
                if len(self._questions) == 0:
                    print('There are no questions to export!')
            # Opening question screen
            else:
                option -= 1
                # Getting the file path of the question
                path = question_list[option][9]
                # Recognising whether the question is a text or image question and
setting up the object
                if path[len(path)-1] == 't':
```

```
                    current_question = textQuestion(question_list[option][0],
question_list[option][1], question_list[option][2], question_list[option][3],
question_list[option][4], question_list[option][5], question_list[option][6],
question_list[option][7], question_list[option][8], question_list[option][9],
question_list[option][10], question_list[option][11])
                else:
                    current_question = imageQuestion(question_list[option][0],
question_list[option][1], question_list[option][2], question_list[option][3],
question_list[option][4], question_list[option][5], question_list[option][6],
question_list[option][7], question_list[option][8], question_list[option][9],
question_list[option][10], question_list[option][11])
                while True:
                    try:
                        # Choosing an option
                        print()
                        print('1. View Question')
                        print('2. View Mark Scheme')
                        print("3. View Examiner's report")
                        check = False
                        # Checking if question is currently in the paper
                        for x in self._questions:
                            if x.getID() == current_question.getID():
                                check = True
                        if check:
                            print('4. Remove from paper')
                        else:
                            print('4. Add to paper')
                        print('0. Go back')
                        print()
                        option = int(input('Please select an option: '))
                    except ValueError:
                        print('That is not a valid option (Error code 17)')
                    else:
                        # Viewing the question
                        if option == 1:
                            current_question.viewQuestion()
                        # Viewing the mark scheme
                        elif option == 2:
                            current_question.viewMarkScheme()
                        # Viewing the examiner's report
                        elif option == 3:
                            current_question.viewExaminersReport()
                        # Adding/Removing the question
                        elif option == 4:
                            check = False
                            # Checking if the question is in the paper
                            for x in self._questions:
                                # If question is found in the paper, then it is
removed
                                if x.getID() == current_question.getID():
                                    check = True
```

```
                                    self._questions.remove(x)
                                    print('Removed')
                                    self._changes.push(['r', current_question])
                            # If question is not in the paper, it is added
                            if not check:
                                self._questions.append(current_question)
                                print('Added')
                                self._changes.push(['a', current_question])
                            break
                        elif option == 0:
                            self.editPaper(unit, topic, board, qualification, year,
paper, orderby)
                            break
                        else:
                            print('That is not a valid option (Error code 18)')
        # Resetting the stack before exiting
        if option == 0:
            self._changes = stack()


# This class represents the individual questions that are stored within a paper
class question:
    # Initialises all the information stored about a question
    def __init__(self, ID, unit, topic, marks, mins, board, qual, year, paper,
path, mspath, erpath):
        self._ID = ID
        self._unit = unit
        self._topic = topic
        self._marks = marks
        self._mins = mins
        self._board = board
        self._qual = qual
        self._year = year
        self._paper = paper
        self._path = path
        self._mspath = mspath
        self._erpath = erpath

    # Returns the ID of the question
    def getID(self):
        return self._ID

    # Returns the ID of the question, as well as an array containing the info about
the question
    def getInfo(self):
        return self._ID, [self._unit, self._topic, self._board, self._qual,
self._year, self._paper]

    # Returns the file path of the question within the questions folder
    def getPath(self):
        return self._path
```

```
    # Returns the file path of the mark scheme within the questions folder
    def getMSPath(self):
        return self._mspath

    # Returns the file path of the examiner's report within the questions folder
    def getERPath(self):
        return self._erpath

# This subclass represents a question that is stored using text files
class textQuestion(question):
    def __init__(self, ID, unit, topic, marks, mins, board, qual, year, paper,
path, mspath, erpath):
        super().__init__(ID, unit, topic, marks, mins, board, qual, year, paper,
path, mspath, erpath)

    # Opens the text file where the question is stored and prints out the contents
    def viewQuestion(self):
        print()
        with open(sys.path[0] + '/Questions/' + self._path, "r") as f:
            print(f.read())

    # Opens the text file where the mark scheme is stored and prints out the
contents
    def viewMarkScheme(self):
        print()
        with open(sys.path[0] + '/Questions/' + self._mspath, "r") as f:
            print(f.read())

    # Opens the text file where the examiner's report is stored and prints out the
contents
    # If there is no examiner's report available, it will print out a suitable
message to say so
    def viewExaminersReport(self):
        print()
        if self._erpath == 'N/A':
            print("Examiner's report not available for this question")
        else:
            with open(sys.path[0] + '/Questions/' + self._erpath, "r") as f:
                print(f.read())

# This subclass represents a question that is stored using image files
class imageQuestion(question):
    def __init__(self, ID, unit, topic, marks, mins, board, qual, year, paper,
path, mspath, erpath):
        super().__init__(ID, unit, topic, marks, mins, board, qual, year, paper,
path, mspath, erpath)

    # Opens the image file where the question is stored and displays it on the
screen
    def viewQuestion(self):
```

```
        im = Image.open(sys.path[0] + '/Questions/' + self._path)
        im.show()

    # Opens the image file where the mark scheme is stored and displays it on the
screen
    def viewMarkScheme(self):
        im = Image.open(sys.path[0] + '/Questions/' + self._mspath)
        im.show()

    # Opens the image file where the examiner's report is stored and displays it on
the screen
    # If there is no examiner's report available, it will print out a suitable
message to say so
    def viewExaminersReport(self):
        if self._erpath == 'N/A':
            print("Examiner's report not available for this question")
        else:
            im = Image.open(sys.path[0] + '/Questions/' + self._erpath)
            im.show()

# The stack is used to facilitate the undo function when creating papers
class stack:
    def __init__(self):
        # The stack is of a fixed length, but if the stack is full, it can
facilitate making it longer before pushing the next item onto the stack
        self._stack = [' ',' ',' ',' ',' ',' ',' ',' ']
        # This pointer indicates the top of the stack
        self._top = 0

    # Pushing an item onto the stack
    def push(self, item):
        # The item is pushed onto the stack at the position indicated by the top
pointer
        self._stack[self._top] = item
        # The top pointer is incremented by 1
        self._top += 1
        # If the stack is full, an empty string is appened to the list to make
space to push the next item onto the stack
        if self._top == len(self._stack):
            self._stack.append(' ')

    # Popping an item off the stack
    def pop(self):
        # If the stack is empty, no item is popped off
        if self._top == 0:
            return False
        # Item stored in temporary variable
        item = self._stack[self._top - 1]
        # Item in stack changed to an empty string
        self._stack[self._top - 1] = ' '
        # Pointer is decremented by 1
```

```
        self._top -= 1
        # Item in temporary variable is returned
        return item

# The priority queue is used when exporting a paper
# The priorities of the questions represent their position within the paper
# The priority queue allows the questions to be in order when exporting the paper
class pqueue:
    def __init__ (self, numofqs):
        self._length = numofqs
        # This list represents the priority queue
        # The number of sublists is the number of questions
        # The first item in the sublist is the question and the second is the
priority (or in this case, the position)
        self._queue = [['',0] for x in range (self._length)]
        # This pointer represents the front of the queue
        self._front = 0
        # This pointer represents the back of the queue
        self._back = 0

    # Returns the length of the queue (in this case this is the same as the number
of questions)
    def getLength(self):
        return self._length

    # Enqueuing an item
    def enqueue (self, question, priority):
        current = self._back
        # If the queue has no items, the question is enqueued at the front (which
is also the back in this case)
        if current == 0:
            self._queue[0] = [question, priority]
        else:
            # Finding the position where the priority of the question in front is
higher than the priority of the question being enqueued
            while self._queue[current-1][1] < priority and current != 0:
                self._queue[current] = self._queue[current-1]
                current -= 1
            # Item is enqueued when correct position is found
            self._queue[current] = [question, priority]
        # Back pointer is incremented by 1
        self._back += 1

    # Dequeuing an item
    def dequeue(self):
        # Question at front of queue is stored in a temporary variable
        temp = self._queue[self._front]
        # Front of the queue is returns to an empty list
        self._queue[self._front] = ['',0]
        # Front pointer is incremented by 1
        self._front += 1
```

```
        # Returns the question stored in the temporary variable
        return temp[0]

    # This returns whether the queue is empty
    # If the queue is not empty, True is returned to indicate that there are more
questions to dequeue
    # Otherwise, False is returned
    def recursion(self):
        if self._front < self._back:
            return True
        return False

# This class is used to export a paper to a PDF
class createPDF:
    # The constructor is what creates the PDF
    def __init__ (self, pq, coverpage, ms, gb, er, name):
        self._name = name
        # 3 copies of the priority queue are made so that the questions, mark
schemes and examiner's reports can be exported separately
        self._questionq = copy.deepcopy(pq)
        self._msq = copy.deepcopy(pq)
        self._erq = copy.deepcopy(pq)
        self._pdf = FPDF()
        # A cover page is added if the user would like one
        if coverpage == 'Y' or coverpage == 'y':
            self._pdf.add_page()
            self._pdf.set_font('Arial', 'B', 16)
            self._pdf.cell(190, 266, self._name, 1, 0, 'C')
        self._pdf.set_font('Arial', '', 14)
        # Questions will always be added
        self._addQuestion()
        # Mark scheme, grade boundaries and examiner's reports will only be added
if the user asks for them
        if ms == 'Y' or ms == 'y':
            self._addMS()
        if gb == 'Y' or gb == 'y':
            self._addGB()
        if er == 'Y' or er == 'y':
            self._addER()
        # PDF is exported to the same directory as this python file
        # The name of the PDF is "the name of the paper".pdf
        self._pdf.output(self._name+'.pdf', 'F')

    # This method adds questions to the paper
    def _addQuestion(self):
        q = self._questionq.dequeue()
        # Recursion is used to export every question
        # The base case is when there are no more questions in the paper, so the
first question is added
        if self._questionq.recursion() == True:
            self._addQuestion()
```

```
        self._pdf.add_page()
        # Making sure text and image questions are dealt with appropriately
        if 'txt' in q.getPath():
            with open(sys.path[0] + '/Questions/' + q.getPath(), "r") as f:
                self._pdf.write(5, f.read())
        else:
            self._pdf.image(sys.path[0] + '/Questions/' + q.getPath(), w = 190)

    # This method adds mark schemes to the paper
    def _addMS(self):
        ms = self._msq.dequeue()
        # Recursion is used like in the addQuestion method
        if self._msq.recursion() == True:
            self._addMS()
        if ms.getMSPath() != 'N/A':
            self._pdf.add_page()
            # Making sure text and image mark schemes are dealt with appropriately
            if 'txt' in ms.getMSPath():
                with open(sys.path[0] + '/Questions/' + ms.getMSPath(), "r") as f:
                    self._pdf.write(5, f.read())
            else:
                self._pdf.image(sys.path[0] + '/Questions/' + ms.getMSPath(), w =
190)

    # This method adds grade boundaries to the paper
    def _addGB(self):
        self._pdf.add_page()
        self._pdf.multi_cell(txt='Grade Boundaries:', w = 0, h = 10)
        self._pdf.multi_cell(txt='GCSE (Higher)', w = 0, h = 10)
        self._pdf.multi_cell(txt='9 - 82.5%', w = 0, h = 10)
        self._pdf.multi_cell(txt='8 - 69.58%', w = 0, h = 10)
        self._pdf.multi_cell(txt='7 - 57.08%', w = 0, h = 10)
        self._pdf.multi_cell(txt='6 - 45%', w = 0, h = 10)
        self._pdf.multi_cell(txt='5 - 33.33%', w = 0, h = 10)
        self._pdf.multi_cell(txt='4 - 21.67%', w = 0, h = 10)
        self._pdf.multi_cell(txt='3 - 15.83%', w = 0, h = 10)
        self._pdf.multi_cell(txt='U - 0', w = 0, h = 10)
        self._pdf.multi_cell(txt='A-Level', w = 0, h = 10)
        self._pdf.multi_cell(txt='A* - 72.33%', w = 0, h = 10)
        self._pdf.multi_cell(txt='A - 55%', w = 0, h = 10)
        self._pdf.multi_cell(txt='B - 44.67%', w = 0, h = 10)
        self._pdf.multi_cell(txt='C - 34.33%', w = 0, h = 10)
        self._pdf.multi_cell(txt='D - 24.33%', w = 0, h = 10)
        self._pdf.multi_cell(txt='E - 14.33%', w = 0, h = 10)
        self._pdf.multi_cell(txt='U - 0', w = 0, h = 10)

    # This method adds examiner's reports to the paper
    def _addER(self):
        er = self._erq.dequeue()
        # Recursion is used like in the addQuestion method
        if self._erq.recursion() == True:
```

```
            self._addER()
        self._pdf.add_page()
        if er.getERPath() != 'N/A':
            # Making sure text and image examiner's reports are dealt with
appropriately
            if 'txt' in er.getERPath():
                with open(sys.path[0] + '/Questions/' + er.getERPath(), "r") as f:
                    self._pdf.write(5, f.read())
            else:
                self._pdf.image(sys.path[0] + '/Questions/' + er.getERPath(), w =
190)
        # If there is no examiner's report available, an appropriate message is
added instead
        else:
            self._pdf.write(5, "This question has no examiner's report available")

# This class is used to take data and display it in a table format
class table:
    def __init__ (self, values, columnheadings, rowheadings):
        # Putting the data into a pandas DataFrame
        df = pd.DataFrame(values, columns = columnheadings, index = rowheadings)
        # Printing the DataFrame as a table
        print(df)
        print()

# This merge sort is used to sort raw data when processing the algorithm to
calculate summary statistics
class mergesort:
    def __init__(self):
        self._placeholder = 0

    def merge_sort(self, items):
        # Base case for recursion
        # The recursion will stop when the list has been divided into single items
        if len(items) <= 1:
            return items
        else:
            # Calculate the midpoint index
            midpoint = (len(items)-1) // 2
            # Create left half list
            left_half = items[0:midpoint+1]
            # Create right half list
            right_half = items[midpoint+1:len(items)]

            # Recursive call on left half
            left_half = self.merge_sort(left_half)
            # Recursive call on right half
            right_half = self.merge_sort(right_half)

            # Call method to merge both halves
            merged_items = self.merge(left_half, right_half)
```

```
            # Returns the sorted list
            return merged_items

    # Method used to merge sublists
    def merge(self, left, right):
        # New list for merging the items
        merged = []
        # This pointer indicates the left current position
        index_left = 0
        # This pointer indicates the right current position
        index_right = 0

        # While there are still items to merge
        while index_left < len(left) and index_right < len(right):

            # Find the lowest of the two items being compared and add it to the new
list
            if left[index_left] < right[index_right]:
                merged.append(left[index_left])
                index_left += 1
            else:
                merged.append(right[index_right])
                index_right += 1

        # Add to the merged list any remaining data from left list
        while index_left < len(left):
            merged.append(left[index_left])
            index_left += 1

        # Add to the merged list any remaining data from right list
        while index_right < len(right):
            merged.append(right[index_right])
            index_right += 1

        # Returns the merged list
        return merged

# Class to calculate the summary statistics
class summarystatistics:
    def __init__ (self):
        self._placeholder = 0

    # Returns the percentage that the count is of the total
    def percentage(self, total, count):
        return count/total*100

    # This algorithm calculates the 3 main summary statistics:
    # Mean, variance and standard deviation
    def mainstats(self, ls):
        # A dictionary is used to store each statistic
```

```
        dict = {
            'Mean': 0,
            'Variance': 0,
            'StandardDeviation' : 0
        }
        # Iterating through the list to find the sum of all items and the sum of
the squares of all the items
        total = 0
        squaretotal = 0
        for x in ls:
            total += x
            squaretotal += x**2
        # All values are rounded to 2 decimal places for simplicity
        # The mean is calculated by dividing the sum by the number of items
        dict['Mean'] = round(total / len(ls),2)
        # The variance is calculated as the mean of the squared take away the
square of the mean
        dict['Variance'] = round((squaretotal / len(ls)) - ((dict['Mean'])**2),2)
        # The standard deviation is the square root of the variance
        dict['StandardDeviation'] = round(math.sqrt(dict['Variance']),2)
        print('Mean = ' + str(dict['Mean']) + '%')
        print('Variance = ' + str(dict['Variance']) + '%')
        print('Standard Deviation = ' + str(dict['StandardDeviation']) + '%')

    # This algorithm calculates all the other summary statistics
    def otherstats(self, ls):
        dict = {
            'Mode': 0,
            'Median': 0,
            'Q1': 0,
            'Q3': 0,
            'Range': 0,
            'IQRange': 0,
            'IPRange': 0
        }
        # Makes a list with all the items and the number of times the items appear
        count = []
        for x in ls:
            found = False
            for y in range(len(count)):
                if count[y][0] == x:
                    found = True
                    count[y][1] += 1
            if found == False:
                count.append([x,1])
        sorter = mergesort()
        # Formats the list so it can be merge sorted by putting the number of times
an item appears at the start of the respective sublist
        for x in count:
            x.reverse()
        count = sorter.merge_sort(count)
```

```
        count.reverse()
        # All values are rounded to 2 decimal places for simplicity
        # The mode is the most common value
        dict['Mode'] = round(count[0][1],2)
        sortedlist = []
        for x in ls:
            sortedlist.append([0,x])
        # Sorting the values in order by the actual value now
        sortedlist = sorter.merge_sort(sortedlist)
        for x in range(len(sortedlist)):
            sortedlist[x] = sortedlist[x][1]
        # First quartile is the value at the 1/4 position of the entire list
        dict['Q1'] = round(sortedlist[int(len(sortedlist)*0.25 + 0.5)-1],2)
        # Median is the value at the middle (1/2 position) of the list
        dict['Median'] = round(sortedlist[int(len(sortedlist)*0.5 + 0.5)-1],2)
        # Third quartile is the value at the 3/4 position of the entire list
        dict['Q3'] = round(sortedlist[int(len(sortedlist)*0.75 + 0.5)-1],2)
        # The range is the largest value take away the smallest value
        dict['Range'] = round(sortedlist[len(sortedlist) - 1] - sortedlist[0],2)
        # The inter-quartile range is the third quartile take away the first
quartile
        dict['IQRange'] = round(dict['Q3'] - dict['Q1'],2)
        # The inter-percentile range is the value at the 9/10 position of the list
take away the value at the 1/10 position
        dict['IPRange'] = round(sortedlist[int(len(sortedlist)*0.9 + 0.5)-1] -
sortedlist[int(len(sortedlist)*0.1 + 0.5)-1],2)
        print('Mode = ' + str(dict['Mode']) + '%')
        print('Median = ' + str(dict['Median']) + '%')
        print('Q1 = ' + str(dict['Q1']) + '%')
        print('Q3 = ' + str(dict['Q3']) + '%')
        print('Range = ' + str(dict['Range']) + '%')
        print('Interquartile Range = ' + str(dict['IQRange']) + '%')
        print('Interpercentile Range = ' + str(dict['IPRange']) + '%')

# This class is used to retrieve and return the data required when analysing
class analysedata:
    def __init__ (self):
        self._placeholder = 0

    # Analysing by student and question
    def studentandquestion(self, studentid, paperid):
        ls = data.execute_read_query("SELECT MarksAchieved, QuestionMarks,
QuestionTopic FROM Marks, Questions, Links WHERE Marks.QuestionID =
Links.QuestionID AND Marks.PaperID = Links.PaperID AND Marks.QuestionID =
Questions.QuestionID AND Marks.StudentID=" + str(studentid) + " AND Marks.PaperID="
+ str(paperid) + " ORDER BY QuestionOrder")
        for x in range(len(ls)):
            temp = []
            for y in ls[x]:
                temp.append(y)
            ls[x] = temp
```

```
        # Making two lists that represent the percentage achieved for each topic
        percentages = []
        topics = []
        for x in ls:
            percentages.append(round(ss.percentage(x[1],x[0]),2))
            topics.append(x[2])
        # If no data then empty strings are returned
        if len(percentages) == 0:
            biggesttopic = ''
            smallesttopic = ''
            qlist = ''
        else:
            # Working out what the WWW and EBI are
            biggestpercentage = percentages[0]
            biggesttopic = topics[0]
            smallestpercentage = percentages[0]
            smallesttopic = topics[0]
            # Making a list of questions and percentage achieved
            qlist = []
            for x in range(len(percentages)):
                qlist.append('Q' + str(x+1))
                if percentages[x] > biggestpercentage:
                    biggestpercentage = percentages[x]
                    biggesttopic = topics[x]
                if percentages[x] < smallestpercentage:
                    smallestpercentage = percentages[x]
                    smallesttopic = topics[x]
        return percentages, biggesttopic, smallesttopic, qlist

    # Analysing by student and paper
    def studentandpaper(self, studentid):
        ls = data.execute_read_query("SELECT PaperName, MarksAchieved,
QuestionMarks FROM Marks, Papers, Students, Questions WHERE Papers.PaperID =
Marks.PaperID AND Students.StudentID = Marks.StudentID AND Questions.QuestionID =
Marks.QuestionID AND Students.StudentID=" + str(studentid) + " ORDER BY
Marks.PaperID")
        for x in range(len(ls)):
            temp = []
            for y in ls[x]:
                temp.append(y)
            ls[x] = temp
        for x in range(len(ls)):
            try:
                while True:
                    if ls[x][0] == ls[x+1][0]:
                        ls[x][1] += ls[x+1][1]
                        ls[x][2] += ls[x+1][2]
                        ls.remove(ls[x+1])
                    else:
                        break
            except:
```

```
                continue
        # Making two lists that represent the percentage achieved for each paper
        papers = []
        percentages = []
        for x in ls:
            papers.append(x[0])
            percentages.append(ss.percentage(x[2],x[1]))
        return papers, percentages

    # Analysing by class and question
    def classandquestion(self,classid, paperid):
        ls = data.execute_read_query("SELECT Marks.StudentID, MarksAchieved,
QuestionMarks, FirstName, Surname FROM Marks, Questions, Students, Links WHERE
Questions.QuestionID = Links.QuestionID AND Marks.QuestionID = Questions.QuestionID
AND Students.StudentID = Marks.StudentID AND Students.ClassID=" + str(classid) + "
AND Links.PaperID=" + str(paperid) + " AND Marks.PaperID=" + str(paperid) + " ORDER
BY Marks.StudentID, QuestionOrder")
        # If no data then empty lists are returned
        if len(ls) == 0:
            qlist = []
            studentlist = []
            percentages = []
        else:
            for x in range(len(ls)):
                temp = []
                for y in ls[x]:
                    temp.append(y)
                ls[x] = temp
            check = ls[0][0]
            numofqs = 1
            for x in range(1,len(ls)):
                if ls[x][0] == check:
                    numofqs += 1
                else:
                    break
            # Making a list of questions, students, and percentages
            qlist = []
            for x in range(1,numofqs+1):
                qlist.append('Q' + str(x))
            studentlist = []
            for x in range(0,len(ls),numofqs):
                studentlist.append(ls[x][3] + ' ' + ls[x][4])
            for x in range(len(ls)):
                ls[x][1] = round(ss.percentage(ls[x][2],ls[x][1]),2)
            percentages = []
            for x in range(numofqs):
                temp = []
                for y in range(x,len(ls),numofqs):
                    temp.append(ls[y][1])
                percentages.append(temp)
        return qlist, studentlist, percentages
```

```
    # Analysing by class and paper
    def classandpaper(self,classid):
        ls = data.execute_read_query("SELECT Marks.StudentID, PaperName,
MarksAchieved, QuestionMarks FROM Marks, Papers, Students, Questions WHERE
Papers.PaperID = Marks.PaperID AND Students.StudentID = Marks.StudentID AND
Questions.QuestionID = Marks.QuestionID AND ClassID=" + str(classid) + " ORDER BY
Marks.PaperID")
        for x in range(len(ls)):
            temp = []
            for y in ls[x]:
                temp.append(y)
            ls[x] = temp
        done = False
        while done == False:
            done = True
            for x in range(len(ls)):
                for y in range(len(ls)):
                    if x != y:
                        if ls[x][0:2] == ls[y][0:2]:
                            ls[x][2] += ls[y][2]
                            ls[x][3] += ls[y][3]
                            ls.remove(ls[y])
                            done = False
                            break
                if done == False:
                    break
        for x in ls:
            x.remove(x[0])
            x[1] = round(ss.percentage(x[2],x[1]),2)
            x.remove(x[2])
        mergingcomplete = False
        while mergingcomplete == False:
            mergingcomplete = True
            try:
                for x in range(len(ls)-1):
                    if ls[x][0] == ls[x+1][0]:
                        total = ls[x][1]
                        count = 1
                        while True:
                            try:
                                if ls[x][0] == ls[x+1][0]:
                                    total += ls[x+1][1]
                                    count += 1
                                    ls.remove(ls[x+1])
                                else:
                                    break
                            except:
                                break
                        ls[x][1] = round(total/count,2)
            except:
```

```
                continue
        # Making two lists that represent the percentage achieved for each paper
        papers = []
        percentages = []
        for x in ls:
            papers.append(x[0])
            percentages.append(x[1])
        return papers, percentages

# Setting up summary statistics and database objects ready to be used
ss = summarystatistics()
data = database('database.db')
# Starting the program by running the menu
new_menu = menu()
```

Testing

Video overview

To be added

Objectives

Objective 1

My software solution will allow the user to create exams, add marks from exams, and analyse class performance

a. All 3 of these features will be easily accessible from the menu, within one button press after opening the application (not including any drop-down menus)
b. The menu will be accessible in one click from each of these sections

Test No	Type	Data	Expected	Actual	
1 – Creating a new paper from menu	Normal	1	Will ask to add name of new paper	Fig 1 The program successfully asked for a name of the new paper	✓
	Erroneous	1A	Error message then allows user to try again	Fig 2 Not allowing 1A as an input	✓
2 – Editing an existing paper from menu	Normal	2	Will display a list of existing papers and ask user to choose one	Fig 3 Successfully allowing the user to choose a paper	✓
	Erroneous	2#	Error message then allows user to try again	Fig 4 Not allowing 2# as an option	✓
3 – Adding marks to a paper from menu	Normal	4	Will allow user to choose a class to add marks to	Fig 5 Allowing the user to choose a class	✓
	Erroneous	4.	Error message then allows user to try again	Fig 6 Not allowing 4. as an option	✓

<table>
<tr><td rowspan="2">4 – Analysing performance from menu</td><td>Normal</td><td>5</td><td>Allows user to analyse performance</td><td>Fig 7
Allowing the student to choose an analyse option</td><td>✓</td></tr>
<tr><td>Erroneous</td><td>50</td><td>Error message then allows user to try again</td><td>Fig 8
Not allowing 50 as an option</td><td>✓</td></tr>
<tr><td rowspan="5">5 – Going to menu from paper editing screen</td><td>Normal</td><td>2,0,0</td><td>Menu opens</td><td>https://youtu.be/QKkBderbZWk
Allowing the user to choose a paper and access the edit menu, then exiting the menu</td><td>✓</td></tr>
<tr><td>Erroneous</td><td>2,0,0.</td><td>Error message then allows user to try again</td><td>https://youtu.be/Z5xYdeteR40
Allows the user to choose a paper and access the edit menu, but invalid input in edit screen crashes the program. This is because I jumbled up the order of the while loop and the try-except.</td><td>X</td></tr>
<tr><td colspan="5">Incorrect code:
<pre>try:
   while True:…
except ValueError:
   print('That is not a valid option (Error code 7)')</pre></td></tr>
<tr><td colspan="5">Corrected code:
<pre>while True:
   try:…
   except ValueError:
      print('That is not a valid option (Error code 7)')</pre></td></tr>
<tr><td>Erroneous</td><td>2,0,0.</td><td>Error message then allows user to try again</td><td>https://youtu.be/SVAvSEx31Q4
Same as previous but this time relevant error message is displayed when invalid input is given</td><td>✓</td></tr>
<tr><td>6 – Going to menu from adding</td><td>Normal</td><td>4,0,4,1,0</td><td>Goes back to menu from both class</td><td>https://youtu.be/HefE0lrYeo0
Allows user to go back to menu after accessing class</td><td>✓</td></tr>
</table>

<table>
<tr><td rowspan="5">marks screen</td><td></td><td></td><td>selection and student selection screens</td><td>selection screen. When a class is selected, can still go back to menu in student selection screen</td><td></td></tr>
<tr><td>Erroneous</td><td>4,0/,0,4,1,9,0</td><td>Errors in entering 0 will display error message and allow user to input something valid, after which entering 0 returns to menu as usual</td><td>https://youtu.be/J06Hfq4oIzo
Not allowing invalid input in class and student selection screens</td><td>✓</td></tr>
<tr><td rowspan="4">7 – Going to menu from analysing screen</td><td>Normal</td><td>5,0,5,1,0,5,1,1,0</td><td>Goes back to menu from all sections</td><td>https://youtu.be/vMJh8KXsLuk
Can go back to menu when choosing both class/student and paper/question</td><td>✓</td></tr>
<tr><td>Erroneous</td><td>5,0 0,0</td><td>Error message then allows user to try again</td><td>https://youtu.be/0lutNBqRp-4
An invalid input when choosing to analyse by class/student crashes the program. This is because I have not validated the input.</td><td>X</td></tr>
<tr><td colspan="5">Incorrect code:
<pre>option1 = int(input(''))</pre></td></tr>
<tr><td colspan="5">Corrected code:
<pre>option1 = None
while option1 != 1 and option1 != 2 and option1 != 0:
    try:
        print()
        option1 = int(input(''))
    except:
        print('That is not a valid option')
    else:
        if option1 != 1 and option1 != 2 and option1 != 0:
            print('That is not a valid option')</pre></td></tr>
</table>

		5,0 0,0,5,1,0),0,5,1,1, o,0	Error message then allows user to try again	https://youtu.be/ufM0pGrVR78 Relevant error message displayed if invalid input entered in all analysing option screens	✓

Fig 1

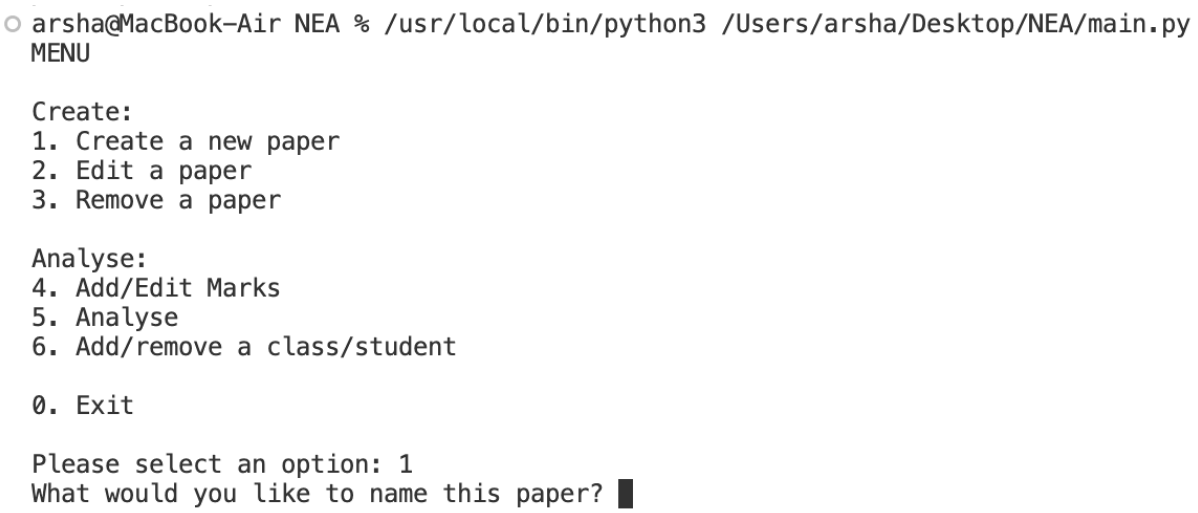


Fig 2

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 1A

That is not a valid option (Error code 2)

MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option:
```

Fig 3

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 2

Please choose a paper:

0. Paper0
1. Paper1
2. Paper2
3. test
4. TEST1
5. siuuuuuuu
```

Fig 4

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 2#

That is not a valid option (Error code 2)

MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option:
```

Fig 5

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 4
1. Lairone's class
2. My class
0. Go back

Enter an option:
```

Fig 6

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 4.

That is not a valid option (Error code 2)

MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option:
```

Fig 7

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 5

Please select an option:
1. Analyse by class
2. Analyse by student
0. Go back
```

Fig 8

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 50
That is not a valid option (Error code 5)

MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option:
```

Objective 2

Questions included in the exam creation tool will be sourced from a range of exam boards and qualifications.

a. There will be at least 2 questions available from Edexcel papers at A-Level
b. There will be at least 1 question available from every English exam board specification (AQA, Edexcel, OCR [A and B], Eduqas) for GCSE and A-Level
c. Questions will be filterable by exam board specification and qualification
d. Exam board specification and qualification will be easily viewable from question information

Test No	Type	Data	Expected	Actual	
1 – Filter questions by exam board	Normal	4	Only questions from Eduqas to be shown	https://youtu.be/fMJZt-bYFcw The program allows the user to choose filters after which only Eduqas questions are shown	✓
	Erroneous	77	Error message then allows user to try again	https://youtu.be/qXbgOSm9CQY Not allowing an invalid input when choosing an exam board	✓
	Boundary	1	Only questions from Edexcel B to be shown	https://youtu.be/RLlR5frsV7A The program allows the user to choose filters after which only Edexcel B questions are shown	✓
2 – Filter questions by qualification	Normal	2	Only questions from AS & A-Level to be shown	https://youtu.be/cgvzNR9of6o The program allows the user to choose filters after which only AS & A-Level questions are shown	✓
	Erroneous	4k	Error message then allows user to try again	https://youtu.be/1VXAhLqe2L8 Not allowing an invalid input when choosing a qualification	✓
	Boundary	1	Only questions from GCSE to be shown	https://youtu.be/WUiq2ndexVc The program allows the user to choose filters after which only GCSE questions are shown	✓

Objective 3

Paper creation will be intuitive and as easy to use as Edexcel examWizard

a. Questions will be viewable in a list and will display the question details
b. Number of marks, number of minutes and paper information will be available from the question information
c. Once a question has been clicked, the question, mark scheme and examiner's report will be viewable with one click
d. Questions will be filterable by topic, year, and paper, and sortable by number of marks and number of minutes.
e. Total mark of the paper will be viewable upon opening the paper without any clicks
f. Questions can be added and removed from a paper in one click
g. Papers should be saveable and editable from within the software

<table>
<tr><th>Test No</th><th>Type</th><th>Data</th><th>Expected</th><th>Actual</th><th></th></tr>
<tr><td rowspan="2">1 – Viewing a text question</td><td>Normal</td><td>1</td><td>Question printed into terminal</td><td>https://youtu.be/TI3PKz_n_j0
When chosen, the question is printed into the terminal</td><td>✓</td></tr>
<tr><td>Erroneous</td><td>11</td><td>Error message then allows user to try again</td><td>https://youtu.be/qn_kydE4u_s
Not allowing an invalid input when choosing what to do with the selected question</td><td>✓</td></tr>
<tr><td rowspan="4">2 – Viewing an image question</td><td>Normal</td><td>1</td><td>Window opens that displays question</td><td>https://youtu.be/hmniucrc3gY
When chosen, the question is displayed in an external window that appears on the screen</td><td>✓</td></tr>
<tr><td>Erroneous</td><td>1*</td><td>Error message then allows user to try again</td><td>https://youtu.be/FOEABQ-mCwQ
When invalid input is entered, the program still attempts to show the question, causing confusion about what file to display. This is because I used an if outside of the try-except, so if there is an error, it still attempts to use the if statements.</td><td>X</td></tr>
<tr><td colspan="5">Incorrect code:
<pre>try:…
except ValueError:…
if option == 1:…
elif option == 2:…
elif option == 3:…
elif option == 4:…
elif option == 0:…
else:…</pre></td></tr>
<tr><td colspan="5">Corrected code:
<pre>try:…
except ValueError:…</pre></td></tr>
</table>

	`else:`
	`  if option == 1:…`
	`  elif option == 2:…`
	`  elif option == 3:…`
	`  elif option == 4:…`
	`  elif option == 0:…`
	`  else:…`				

	Erroneous	1*	Error message then allows user to try again	https://youtu.be/QtYCGLRMGmU When invalid input is entered, the program displays a relevant error message and does not attempt to display any files	✓
3 – Viewing a text question mark scheme	Normal	2	Mark scheme printed into terminal	https://youtu.be/nWhgwTCeCyQ When chosen, the mark scheme is printed into the terminal	✓
	Erroneous	2(	Error message then allows user to try again	https://youtu.be/0yi4SLjaLXo Not allowing an invalid input when choosing what to do with the selected question	✓
4 – Viewing an image question mark scheme	Normal	2	Window opens that displays mark scheme	https://youtu.be/XWqwqBDvn0U When chosen, the mark scheme is displayed in an external window that appears on the screen	✓
	Erroneous	21	Error message then allows user to try again	https://youtu.be/_ojbpAnFxIk When invalid input is entered, the program displays a relevant error message and does not attempt to display any files	✓
5 – Viewing a text question examiner's report	Normal	3	Examiner's comments printed into terminal	https://youtu.be/kPnATc14Ut0 When chosen, the examiner's report is printed into the terminal	✓
	Erroneous	3$	Error message then allows user to try again	https://youtu.be/SRLGtZS49BE Not allowing an invalid input when choosing what to do with the selected question	✓
6 – Viewing an image question examiner's report	Normal	3	Window opens that displays examiner's comments	https://youtu.be/7qhP4XiTjCI When chosen, the examiner's report is displayed in an external window that appears on the screen	✓

	Erroneous	30	Error message then allows user to try again	https://youtu.be/4J1LGLfycqY When invalid input is entered, the program displays a relevant error message and does not attempt to display any files	✓
7 – Attempting to view an examiner's report where one isn't available	Normal	3	Message informing user that an examiner's report is not available	https://youtu.be/aSgyey0WYaU When no report is available, a message is displayed in the terminal, informing the user that there is no examiner's report available for the question	✓
8 – Filter questions by topic	Normal	3	Only questions about proof to be shown	https://youtu.be/L7KGA0DVopg Filters are applied and proof questions are correctly shown	✓
	Erroneous	Proof	Error message then allows user to try again	https://youtu.be/rusQsN6OOdM When an invalid input is entered, a message is correctly displayed, and the user can retry choosing a topic	✓
	Boundary	1	Only questions about compound interest to be shown	https://youtu.be/Ss8Px5rYD8o Filters are applied and compound interest questions are correctly shown	✓
9 – Filter questions by year	Normal	3	Only questions from 2019 to be shown	https://youtu.be/kTMyWP11Ctc Filters are applied and only 2019 questions are correctly shown	✓
	Erroneous	2020	Error message then allows user to try again	https://youtu.be/cdXPLj8InSI When an invalid input is entered, a message is correctly displayed, and the user can retry choosing a year	✓
	Boundary	1	Only questions from 2011 to be shown	https://youtu.be/ahRkvhyrndo Filters are applied and only 2011 questions are correctly shown	✓
	Normal	4	Only questions	https://youtu.be/Vhki9QQVeJY	✓

10 – Filter questions by paper			from non-calculator paper to shown	Filters are applied and only non-calculator questions are correctly shown	
	Erroneous	Pure	Error message then allows user to try again	https://youtu.be/bGqPa_dy3VQ When an invalid input is entered, a message is correctly displayed, and the user can retry choosing a paper	✓
	Boundary	8	Only questions from Unit B paper to be shown	https://youtu.be/D1ABuXLEOxk Filters are applied and only unit B questions are correctly shown	✓
11 – Sort questions by marks	Normal	1	Questions will be sorted by marks	https://youtu.be/GD1v6rNSYg8 Sort is applied and questions are correctly sorted by marks	✓
	Erroneous	Marks	Error message then allows user to try again	https://youtu.be/A-Hkf9xy-X4 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing a sort	✓
12 – Sort questions by minutes	Normal	2	Questions will be sorted by minutes	https://youtu.be/9o4ZWOXpNFo Sort is applied and questions are correctly sorted by minutes	✓
	Erroneous	21	Error message then allows user to try again	https://youtu.be/8yd2V71aPK0 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing a sort	✓
13 – Viewing total mark	Normal	0	Shows total mark for Paper0	Fig 9 Total mark is shown when paper is opened	✓
14 – Adding a question	Normal	4	Question is added to paper	https://youtu.be/hB8XO4c8y44 Question is successfully added to paper, as seen in database	✓
	Erroneous	Add	Error message then allows user to try again	https://youtu.be/FZDPJ9K_nF4 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
15 – Removing a question	Normal	4	Question removed from paper	https://youtu.be/jxv8WKTPisg Question is successfully removed from paper, as seen in database	✓

	Erroneous	4£	Error message then allows user to try again	https://youtu.be/MTh7CgMbxyI When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
16 – Saving a paper	Normal	0	All changes saved to database and main menu is displayed	See clip for test 14 normal (https://youtu.be/tsCCoQdo7Ko) or test 15 normal (https://youtu.be/jxv8WKTPisg)	✓

Fig 9

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 2

Please choose a paper:

0. Paper0
1. Paper1
2. Paper2
3. test
4. TEST1
5. siuuuuuuu

0

Questions in paper:
1. ('Algebra', 'Notation, vocabulary and manipulation', 1, 1, 'Eduqas', 'GCSE', 2020, 'Non-Calc', '9.txt', '10.txt', '11.txt')
2. ('Proof', 'Proof', 5, 6, 'OCR A', 'AS & A-Level', 2019, 'Pure & Stats', '6.png', '7.png', '8.png')
3. ('Number', 'Compound interest', 2, 3, 'Edexcel B', 'GCSE', 2011, 'Statistics', '0.txt', '1.txt', '2.txt')
4. ('Number', 'Decimals', 2, 2, 'Edexcel', 'GCSE', 'N/A', 'N/A', '3.png', '4.png', '5.png')

Total mark - 10
Total mins - 12
```

Objective 4

There will be a range of export options for the papers created

a. The export menu should be accessible with one click from the creation tool
b. Papers will be exportable to at least 2 file formats (preferably PDF and DOCX)
c. There will be options to decide whether the user would like to export the questions, mark scheme and/or examiner's report
d. There will be an option to export grade boundaries alongside everything else
e. There will be at least one other export option, such as a cover page or writing space
f. All export options will be selectable and deselect able in one click

Test No	Type	Data	Expected	Actual	
1 – Accessing export menu from creation menu	Normal	24	First of export options is shown, which allows the user to choose a file format to export to	https://youtu.be/3ojhT6yb7PI Successfully displays first export option when asked to export	✓
	Erroneous	Export	Error message then allows user to try again	https://youtu.be/Az4lbMHTDTg When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
2 – Exporting to PDF	Normal	1	A PDF document will be created in the same directory as the original file	https://youtu.be/xwbTeEXvejI A PDF document is successfully created	✓
	Erroneous	PDF	Error message then allows user to try again	https://youtu.be/BLm_-7Z2bdc When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
3 – Exporting to DOCX	Normal	2	A DOCX document will be created in the same directory as	https://youtu.be/3FcncqOCBHk A DOCX document is successfully created	✓

			the original file		
	Erroneous	2.	Error message then allows user to try again	https://youtu.be/jb7umbrqrhk When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
4 – Exporting questions only	Normal	N,N,N,N	A document will be created in the same directory as the original file that contains only questions	https://youtu.be/y6YdDrEz6Jw A paper is successfully created that only contains questions	✓
	Erroneous	No	Error message then allows user to try again	https://youtu.be/WHlX06a6UNI When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
5 – Exporting questions and mark scheme	Normal	n,Y,n,n	A document will be created in the same directory as the original file that contains questions and mark scheme	https://youtu.be/IQEatOy3UIg A paper is successfully created that contains questions and a mark scheme	✓
6 – Exporting questions and examiner's report	Normal	N,n,N,Y	A document will be created in the same directory as the original file that contains questions and examiner's report	https://youtu.be/P1eL9fXJMCo A paper is successfully created that contains questions and an examiner's report	✓

7 – Exporting questions, mark scheme and examiner's report	Normal	N,y,N,y	A document will be created in the same directory as the original file that contains questions, mark scheme and examiner's report	https://youtu.be/ikWS_gG2WPA A paper is successfully created that contains questions, a mark scheme and an examiner's report	✓
8 – Exporting questions, mark scheme, examiner's report, and grade boundaries	Normal	N,Y,Y,Y	A document will be created in the same directory as the original file that contains questions, mark scheme and examiner's report and grade boundaries	https://youtu.be/hYsKhnAh3bc A paper is successfully created that contains questions, a mark scheme, an examiner's report, and grade boundaries	✓
9 – Exporting questions with a cover page	Normal	Y,N,N,N	A document will be created in the same directory as the original file that contains questions and a cover page	https://youtu.be/GeQBvsNKnRg A paper is successfully created that contains questions and a cover page	✓
	Erroneous	YES	Error message then allows user to try again	https://youtu.be/-h9qCp8ISvk When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓

Objective 5

It should be easy and intuitive to add and view the marks of students from exams/papers from within the software

a. There will be an option to type in the marks for each student for each question from within the software
b. For each question, the total mark should be shown and the software should not allow the user to enter a mark above the available total
c. Marks for each student for each question must be viewable from within the software

Test No	Type	Data	Expected	Actual	
1 – Manually entering marks	Normal	1	Allows the user to enter the marks for each question in the paper	https://youtu.be/v9SGbGxOKIs Marks successfully added, as seen in database	✓
	Erroneous	2	Error message then allows user to try again	https://youtu.be/-qtR786HcGg When an invalid input is entered, a message is correctly displayed, and the user can retry entering the mark for the question	✓
	Boundary	0	Allows the user to enter the marks for each question in the paper	https://youtu.be/3Q7Y_9XoysI Marks successfully added, as seen in database	✓
2 – Viewing marks	Normal	5,1,2,1,1,4	Displays a table of all marks for students who have taken the paper	https://youtu.be/PHoruRYvwTU Table is displayed showing the marks for each question of the students that have taken the paper	✓
	Erroneous	GCSE	Error message then allows user to try again	https://youtu.be/S3CPUBCcyIA When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓

Objective 6

The software will have the ability to analyse the performance of students and classes

a. Classes and students will be analysable at a paper level and a question level
b. There will be at least 2 different types of insights available from within the software, including a table and some sort of visual chart
c. All insights will be colour coded
d. Insights will show comparisons to a class average
e. Individual feedback for students will be viewable including at least 1 topic that went well and 1 topic for improvement (except when 0% or 100% was achieved, or the whole paper was based on one single topic)

Test No	Type	Data	Expected	Actual	
1 – Analysing a class at paper level	Normal	5,1,1,1,1	A table is shown that displays the class average for the papers taken by the selected class	Fig 10 Table is correctly displayed showing the average performance of year 11 on the 2 papers they have taken	✓
	Erroneous	Analyse by class	Error message then allows user to try again	Fig 11 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
2 – Analysing a class at question level	Normal	5,1,2,2,1,1	Scatter graph is shown displaying the marks for each question in the selected paper for each student in the selected class	https://youtu.be/Hzp50m5fiZw Scatter graph is correctly displayed, showing the performance of each student in year 11 for each question in the GCSE paper	✓
	Erroneous	Analyse by question	Error message then allows user to try again	Fig 12 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
3 – Analysin	Normal	5,2,1,3,1,1	Summary statistics are	https://youtu.be/grXoQd8xNOI	✓

g a student at paper level			shown for the selected student across all papers completed by that student	Summary statistics are correctly displayed, showing the performance of Kye Greer for all the papers he has taken	
	Erroneous	2.	Error message then allows user to try again	Fig 13 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
4 – Analysing a student at question level	Normal	5,2,2,1,1,1,0	A table is shown that displays the marks achieved by the student of each question in the selected paper	https://youtu.be/bTJli4unaeg Table is correctly displayed showing the performance of Kye Greer for each question in the GCSE paper	✓
	Erroneous	2.	Error message then allows user to try again	Fig 14 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
5 – Insight using a table	Normal	5,1,1,1,1	A table is shown that displays the class average for the papers taken by the selected class	See image for test 1 normal (Fig 10)	✓
	Erroneous	Table	Error message then allows user to try again	Fig 15 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
6 - Insight using a chart	Normal	5,1,2,2,1,1	Scatter graph is shown displaying	See image for test 1 normal (Fig 10)	✓

			the marks for each question in the selected paper for each student in the selected class		
	Erroneous	2.	Error message then allows user to try again	Fig 16 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
8 – Viewing a class average	Normal	5,1,1,1,1	Scatter graph is shown displaying the marks for each question in the selected paper for each student in the selected class	See clip for test 2 normal (h)	✓
	Erroneous	¢	Error message then allows user to try again	Fig 17 When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓
9 – Student feedback	Normal	5,2,2,3,1,1,0	Summary statistics are shown and at the end a WWW and EBI is displayed based of the performance of the selected student on the selected paper	https://youtu.be/V8ezoTH7taQ Summary statistics and subsequently WWW and EBI are displayed based on the performance of Kye Greer on the GCSE paper	✓

	Erroneous	§	Error message then allows user to try again	https://youtu.be/GgWKvXA-fOM When an invalid input is entered, a message is correctly displayed, and the user can retry choosing an option	✓

Fig 10

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 5

Please select an option:
1. Analyse by class
2. Analyse by student
0. Go back

1

Please select an option:
1. Analyse by paper
2. Analyse by question
0. Go back

1

Please select an option:
1. Table
2. Graph
3. Summary statistics
0. Go back

1

Please select a class:
1. Year 11
2. Year 12
3. Year 13
0. Go back

Enter an option: 1
                      GCSE  Adding marks
Average percentage  50.37          50.0

Press enter to close
```

Fig 11

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 5

Please select an option:
1. Analyse by class
2. Analyse by student
0. Go back

Analyse by class
That is not a valid option

1

Please select an option:
1. Analyse by paper
2. Analyse by question
0. Go back
```

Fig 12

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 5

Please select an option:
1. Analyse by class
2. Analyse by student
0. Go back

1

Please select an option:
1. Analyse by paper
2. Analyse by question
0. Go back

Analyse by question
That is not a valid option

0

MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option:
```

Fig 13

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 5

Please select an option:
1. Analyse by class
2. Analyse by student
0. Go back

2.
That is not a valid option

0

MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option:
```

Fig 14

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 5

Please select an option:
1. Analyse by class
2. Analyse by student
0. Go back

2

Please select an option:
1. Analyse by paper
2. Analyse by question
0. Go back

2.
That is not a valid option

0

MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option:
```

Fig 15

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 5

Please select an option:
1. Analyse by class
2. Analyse by student
0. Go back

1

Please select an option:
1. Analyse by paper
2. Analyse by question
0. Go back

1

Please select an option:
1. Table
2. Graph
3. Summary statistics
0. Go back

Table
That is not a valid option

0

MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
```

Fig 16

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 5

Please select an option:
1. Analyse by class
2. Analyse by student
0. Go back

1

Please select an option:
1. Analyse by paper
2. Analyse by question
0. Go back

2

Please select an option:
1. Table
2. Graph
3. Summary statistics
0. Go back

2.
That is not a valid option

0

MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
```

Fig 17

```
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
MENU

Create:
1. Create a new paper
2. Edit a paper
3. Remove a paper

Analyse:
4. Add/Edit Marks
5. Analyse
6. Add/remove a class/student

0. Exit

Please select an option: 5

Please select an option:
1. Analyse by class
2. Analyse by student
0. Go back

1

Please select an option:
1. Analyse by paper
2. Analyse by question
0. Go back

1

Please select an option:
1. Table
2. Graph
3. Summary statistics
0. Go back

1

Please select a class:
1. Year 11
2. Year 12
3. Year 13
0. Go back

Enter an option: ¢
That is not an option
Enter an option:
```

Algorithms

Merge sort

Below is an example of how the list [3,1,4 2] is expected to be sorted by the algorithm

questions	midpoint	left_half	right_half	merged	list_1	list_2	index_1	index_2	index_merged
[3,1,4,2]	1	[3,1]	[4,2]						
See green table									
		[1,3]							
See black table									
			[2,4]	[3,1,4,2]	[1,3]	[2,4]	0	0	0
				[1,1,4,2]			1		1
				[1,2,4,2]				1	2
				[1,2,3,2]			2		3
				[1,2,3,4]				2	4
[1,2,3,4]									

questions	midpoint	left_half	right_half	merged	list_1	list_2	index_1	index_2	index_merged
[3,1]	0	[3]	[1]	[3,1]	[3]	[1]	0	0	0
				[1,1]				1	1
				[1,3]			1		2
[1,3]									

questions	midpoint	left_half	right_half	merged	list_1	list_2	index_1	index_2	index_merged
[4,2]	0	[4]	[2]	[4,2]	[4]	[2]	0	0	0
				[2,2]				1	1
				[2,4]			1		2
[2,4]									

Expected result = [1,2,3,4]

Actual result:

```
1879    sort = mergesort()
1880    print(sort.merge_sort([3,1,4,2]))
```

PROBLEMS 54 OUTPUT DEBUG CONSOLE TERMINAL

```
/usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
/Users/arsha/.zshrc:1: not an identifier: [
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
[1, 2, 3, 4]
arsha@MacBook-Air NEA %
```

Summary statistics

Below is an example of how the main summary statistics are calculated for the list [1,2,3,4,5,6,7,8,9]

List	Mean	Variance	StandardDeviation	Total	SquareTotal	x
[1,2,3,4,5,6,7,8,9]	0	0	0	0	0	1
				1	1	2
				3	5	3
				6	14	4
				10	30	5
				15	55	6
				21	91	7
				28	140	8
				36	204	9
				45	285	
	5	6.66…	2.581988897			

Expected result:

- Mean = 5
- Variance = 6.6666666666….
- Standard deviation = 2.581988897

Actual result:

```
1874    ss = summarystatistics()
1875    # data = database('database.db')
1876    # # Starting the program by running the menu
1877    # new_menu = menu()
1878
1879    ss.mainstats([1,2,3,4,5,6,7,8,9])
```

```
PROBLEMS 42   OUTPUT   DEBUG CONSOLE   TERMINAL

/usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
/Users/arsha/.zshrc:1: not an identifier: [
arsha@MacBook-Air NEA % /usr/local/bin/python3 /Users/arsha/Desktop/NEA/main.py
Mean = 5.0%
Variance = 6.67%
Standard Deviation = 2.58%
arsha@MacBook-Air NEA %
```

Evaluation

Throughout my evaluation, I will be referring to feedback collected both from classmates during the early stages of my code and from my client on the final code. The full feedback notes can be viewed in the appendix (appendix 3-5)

Below is a list of bugs that were found during feedback sessions and how they were fixed:

- If an apostrophe (‘) was entered when adding a class, an error would occur
    - This was fixed by adding simple error handling to the input
- In a scenario where there are no questions in a paper and the user chooses to swap questions, the program would be stuck in a loop of “choosing a question to swap”
    - This was fixed by checking if there are at least 2 questions in the paper before allowing the user to swap questions
- In a scenario where there are no questions in a paper and the user chooses to export the paper, the program would attempt to export the paper
    - This was fixed by simply checking if there is at least 1 question in the paper before allowing the user to export it.

Other than the above and the bugs found during formal testing, there were numerous other smaller bugs that were found outside of testing but not notable enough to be documented. These include forgetting to add a colon after an if or while, or forgetting to add brackets after the name of a function/procedure during a subroutine call. These were all fixed during production, and as of submitting this report there are no know bugs within the program.

Objective 1

My software solution will allow the user to create exams, add marks from exams, and analyse class performance

a. All 3 of these features will be easily accessible from the menu, within one button press after opening the application (not including any drop-down menus)
b. The menu will be accessible in one click from each of these sections

I have fully met the conditions of this objective.

The software solution allows the user to create exams, add marks from exams, and analyse class performance. All 3 of these features are accessible from the menu within one button press after opening the application, as can be seen in all my testing evidence. There is also a "go back" option in every menu. The only exception to this is when a user input is being taken as a reference value, such as when entering the name of a new paper or choosing whether or not to add a cover page when exporting a paper.

During my feedback, I was told positive things about the menu system by every user. This included that the menu was clear, understandable, user-friendly, well-presented, and easy to use. In particular, my client praised the number system for navigating, as this is a lot easier than typing the name of an option. I was informed that there might be a slight learning-curve, especially for teachers who are older or less technically inclined, and that learning how to use the software could be time-consuming. However, once learnt, it is easy for anyone to use. The menu was also praised for its defensive design, avoiding crashes if an invalid input is entered at any point during the execution of the program.

One improvement I was suggested by my client was to implement a GUI for the program, rather than a CLI. As can be seen in my design section, this was originally my intention. However, I dropped this idea since I felt it fell outside the scope of the project and due to time-constraints, I wanted to prioritise getting the more important parts of the program done over implementing a GUI. If I was to implement a GUI in the future, this could be done in two ways. One would be to use an external library that would allow me to create a native GUI, such as Tkinter or PySimpleGui The other would be to create a web-based GUI that could be run in a browser, using an external web framework such as Django or Flask, alongside HTML, CSS, and JavaScript for the front-end part of the website.

Objective 2

Questions included in the exam creation tool will be sourced from a range of exam boards and qualifications.

a. There will be at least 2 questions available from Edexcel papers at A-Level
b. There will be at least 1 question available from every English exam board specification (AQA, Edexcel, OCR [A and B], Eduqas) for GCSE and A-Level
c. Questions will be filterable by exam board specification and qualification
d. Exam board specification and qualification will be easily viewable from question information

I have fully met the conditions of this objective.

For the final build of my program, there are 20 questions available, of which 6 are from Edexcel papers (2 from GCSE and 4 from A-Level), 5 are from AQA papers (1 from GCSE and 4 from A-Level), 5 are from OCR papers (1 from GCSE, 4 from A-Level, 3 from OCR A and 2 from OCR B), and 4 are from Eduqas GCSE papers (Eduqas does not offer AS or A-level Mathematics). In the edit paper screen, there is a filter option where you can filter by exam board specification and qualification, among other criteria. Additionally, in the list of questions, the exam board specification and qualification are visible.

During my feedback, I was told that the list of questions is quite difficult to read, as it is a massive lump of text on the screen with not much spacing or uniformity. I experimented with the order of the information in the list, but this can only do very little to improve readability. I was given some feedback from my client on how I can improve the readability of the question list. This included using a table format, with marks separate from the rest of the information and calculator and non-calculator questions separated, as well as using topic sections to make it easy to find the questions you want. Below is an example of how this might look if this was to be implemented:

Algebra

Calculator

OPTION	SPECIFICATION	QUALIFICATION	MINUTES	MARKS
1	AQA	GCSE	3	3
2	Edexcel	A-Level	10	9
3	OCR A	AS & A-Level	7	6

Non-calculator

OPTION	SPECIFICATION	QUALIFICATION	MINUTES	MARKS
4	Eduqas	GCSE	5	5
5	AQA	AS & A-Level	15	14
6	OCR B	A-Level	10	12

Projectile motion…

Objective 3

Paper creation will be intuitive and as easy to use as Edexcel examWizard

a. Questions will be viewable in a list and will display the question details
b. Number of marks, number of minutes and paper information will be available from the question information
c. Once a question has been clicked, the question, mark scheme and examiner's report will be viewable with one click
d. Questions will be filterable by topic, year, and paper, and sortable by number of marks and number of minutes.
e. Total mark of the paper will be viewable upon opening the paper without any clicks
f. Questions can be added and removed from a paper in one click
g. Papers should be saveable and editable from within the software

I have fully met the conditions of this objective.

Questions are viewable in a list and question details are displayed, including number of marks and minutes, and paper information. Once a question is clicked, it takes 1 click to view the question, mark scheme or examiner's report. Questions can be filtered by topic, year, and paper, and can be sorted by number of marks and minutes. Upon opening the paper, an overview is shown, displaying the questions currently in the paper, along with the total mark and total minutes for the paper. Questions can be added and removed from a paper in one click, and the paper can be saved, as well as edited, from within the software itself.

See previous page for feedback on the formatting of the question list.

Feedback specifically for this objective was generally positive. I was told that the undo feature was useful, and the client liked the fact that you could change the order of the questions.

One specific improvement suggested by my client was to show a URL that links to the entire examiner's report. I discussed with my client why this wouldn't be viable, such as exam boards taking down the examiner's report from their website or migrating them to different URLs, and the question number not matching up with the one used in the user-created paper, meaning the user would not know what question to look at. Despite this, I am still taking on this feedback as a potential improvement further down the line.

Objective 4

There will be a range of export options for the papers created

a. The export menu should be accessible with one click from the creation tool
b. Papers will be exportable to at least 2 file formats (preferably PDF and DOCX)
c. There will be options to decide whether the user would like to export the questions, mark scheme and/or examiner's report
d. There will be an option to export grade boundaries alongside everything else
e. There will be at least one other export option, such as a cover page or writing space
f. All export options will be selectable and deselect able in one click

I have fully met the conditions of this objective.

The export menu is accessible with one click from the creation menu. Papers can be export to PDF and DOCX file formats. The user can decide whether they would like to export the mark scheme and/or the examiner's report alongside the questions. Grade boundaries can also be exported with the paper. The user can choose whether to have a cover page. All export options can be selected (or not selected) within one click.

The general consensus was that exporting a paper is easy and clear. There was no difficulty in using this feature and it was well-designed to be easy to use. My client particularly liked the fact that the mark scheme could be exported in the same document as the paper. They also liked that the paper can be exported to DOCX format, making both the cover page and the paper itself editable, allowing teachers to modify the paper as they wish.

There were a few improvements that were suggested for the exporting section of the program. The first was adding a simple field on the cover page that allows the student to write their name and date on the paper before the test begins. This could be easily added by turning the cell on the cover page into a multicell and adding new lines for the student to write the information. Another suggestion was to add question numbers for each question. This could be easily added by putting the question number at the top of the page before placing the text/image question onto the page. The final suggestion would be to separate calculator and non-calculator questions so that the teacher can set separate timings for when the student is allowed to use a calculator and when they are not. This would be a bit more difficult and would require manipulating the priority queue, to give either calculator or non-calculator questions a higher priority. Below is an example algorithm written in pseudocode.

```
calcCheck(priority, calc):
        if calc == True:
                return priority + 100
        else:
                return priority
```

Objective 5

It should be easy and intuitive to add and view the marks of students from exams/papers from within the software

a. There will be an option to type in the marks for each student for each question from within the software
b. For each question, the total mark should be shown and the software should not allow the user to enter a mark above the available total
c. Marks for each student for each question must be viewable from within the software

I have fully met the conditions of this objective.

The user can quickly and easily type in the marks for each student for each question within a paper from within the software. The software makes sure the mark is valid so that a negative mark or a mark above the total mark for the question is not possible. Marks are viewable from within the analyse section, although this is in the form of a percentage rather than the raw mark, and this may be considered difficult to find for a user who is not familiar with the software.

This is the section of the program that could be improved the most. One key functionality that the client wants to see in the future, is uploading marks from a spreadsheet. This is something that I initially wanted to implement, as can be seen in my design, but was dropped due to being outside of the scope for my project and being extremely time-consuming to implement. If I was to make one addition to the program, this would be the most beneficial feature to add. It would need to be clear how the marks must be formatted in the spreadsheet, so to make sure that the python program is able to retrieve the data from the spreadsheet with no issues. The client also said that they would like to see the marks table colour-coded and would like functionality to analyse marks directly from an imported spreadsheet.

Objective 6

The software will have the ability to analyse the performance of students and classes

a. Classes and students will be analysable at a paper level and a question level
b. There will be at least 2 different types of insights available from within the software, including a table and some sort of visual chart
c. All insights will be colour coded
d. Insights will show comparisons to a class average
e. Individual feedback for students will be viewable including at least 1 topic that went well and 1 topic for improvement (except when 0% or 100% was achieved, or the whole paper was based on one single topic)

I have partially met the conditions of this objective.

The user can analyse by class and student at a paper and question level. Insights available include a table, a scatter graph or bar chart, and summary statistics. The scatter and bar graphs are colour coded, but the table is not. When analysing by class, a class average is used. Within the summary statistics section when analysing by student and question, a WWW and EBI is generated for the student.

My client wanted to be able to view student's marks within the format of a colour-coded table. This was unfortunately impossible using the method I am currently using to output a table by printing a pandas dataframe. To produce a colour coded table, I would need to use an alternative method, such as using the tabulate and colorama external libraries.

Despite this criticism, overall, my client was happy with the insights available. I was told that the analysis functionality was very intuitive and insightful. My client liked that the bar and scatter graphs were colour coded, and that a WWW and EBI could be generated for each student. I had one issue with the scatter graphs since on matplotlib it is not clear when points are overlapping. This isn't a major issue since it is still possible to have a general view of how the class has performed and allows you to see when no students have achieved 0% or 100%. However, it would be useful to have a count or clear visualisation when scatter points overlap. One improvement suggested by my client for the long-term would be to allow for report generating, especially where all the WWWs and EBIs for a single student can be viewed. This would allow the teacher to see if a student consistently struggles on a particular topic, and this can lead to focussed support on that topic for that student.

Concluding statement

In conclusion, I can confidently say that this project has been successful and that I have created a software solution that solves the core issue of not having a centralised program where teachers are able to create papers, with questions from multiple exam boards, and analyse students' performance.

I met all but one of the objectives that I set out after analysing the problem and I created an easy-to-use program that can now be used and potentially improved by the ICHS Mathematics department.

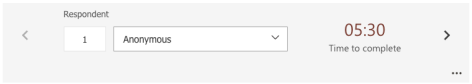
There are no known bugs in the program and the program is designed to prevent crashes. Overall, my client liked how easy it is to use the program, and the flexibility that is available when exporting papers and analysing marks. Some key improvements for future versions of the program include making the UI graphical, allowing importing marks using a spreadsheet, and making tables colour coded.

Bibliography

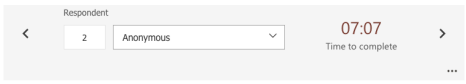
- aqa.org.uk (AQA logo)
- ichs.org.uk (Ilford County High School logo and image)
- en.wikipedia.org (Information about Ilford County High School and examination boards)
- ichs.org.uk (Information about Ilford County High School)
- qualifications.pearson.com and examwizard.co.uk (Information for existing solutions from Edexcel)
- exampro.co.uk (Information for existing solutions from AQA)
- ocr.org.uk (Information for OCR ExamBuilder)
- eduqas.co.uk and questionbank.wjec.co.uk (Information for Eduqas Question Bank)
- qualifications.pearson.com/content/dam/pdf/A%20Level/Mathematics/2017/specification-and-sample-assesment/a-level-l3-mathematics-specification-issue4.pdf (Pearson Edexcel Level 3 Advanced GCE in Mathematics Specification)
- isaaccomputerscience.org/concepts/dsa_search_merge (Information about merge sort)
- pyfpdf.readthedocs.io (Guidance on how to use pyfpdf module to create PDFs)
- realpython.com/python-sql-libraries/ (Initial setup of sqlite3 in python)
- stackoverflow.com/questions/1413540/showing-an-image-from-console-in-python (Guidance on how to display images in python using pillow)
- dothinking.github.io/pdf2docx/quickstart.convert.html (Guidance on how to convert pdf to docx using pdf2docx)
- learnpython.com/blog/print-table-in-python/ (Guidance on how to display a table using pandas dataframe)
- w3schools.com/python/matplotlib_intro.asp (Guidance on how to create bar charts and scatter graphs using matplotlib)

Appendix

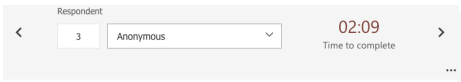
Appendix 1: Questionnaire responses



1. For which of the groups below do you create and/or mark tests and/or exams for?
   - [x] KS3 (Year 7-9)
   - [x] Year 10
   - [x] Year 11
   - [x] Year 12
   - [x] Year 13
2. How many tests or exams do you create and/or mark per year?
   Over 15
3. Which part(s) of the exam process do you find the most difficult?
   - [ ] Creation
   - [ ] Marking and grading
   - [x] Tracking progress
   - [ ] Other
4. Imagine you are creating a bespoke software application to help you with exam management. Which part(s) of the exam process would you want this application to focus on?
   - [x] Creation
   - [ ] Marking and grading
   - [x] Tracking progress
   - [ ] Other
5. When creating tests and/or exams, which of these sources would you prefer to get questions from? (assume obtaining questions from all options are the same in terms of difficulty)
   - ( ) The awarding exam board (including old specification questions)
   - ( ) A range of exam boards and qualifications of the same level
   - (•) Both of these
   - ( ) Other
6. What features would you like a bespoke exam creation tool to include? (give as much detail as possible)
   The grade for the overall paper, once the paper is completed
7. When marking exams, how do you prefer to store students' grades?
   - (•) Digitally
   - ( ) On paper
   - ( ) Other
8. What features would you like a bespoke exam marking tool to include? This includes the actual process of marking exams, storing the marks that students have achieved and generating grades (give as much detail as possible).
   marks but also a Grade that they got on the paper feedback to students on each question.
9. What kind of insights would you find useful when tracking the progress of students in exams and tests?
   - [x] Colour coded tables
   - [ ] Bar graphs
   - [ ] Line graphs
   - [x] Comparisons to class average
   - [ ] Comparisons to historical data
   - [x] Future grade predictions
   - [ ] Other
10. What features would you like a bespoke exam tracking tool to include? (give as much detail as possible)
    track progress and be able to suggest topics or work that has to be undertaken to improve.
11. Any other things that you want mention?



1. For which of the groups below do you create and/or mark tests and/or exams for?
   - [x] KS3 (Year 7-9)
   - [ ] Year 10
   - [x] Year 11
   - [x] Year 12
   - [x] Year 13
2. How many tests or exams do you create and/or mark per year?
   Over 15
3. Which part(s) of the exam process do you find the most difficult?
   - [ ] Creation
   - [x] Marking and grading
   - [ ] Tracking progress
   - [ ] Other
4. Imagine you are creating a bespoke software application to help you with exam management. Which part(s) of the exam process would you want this application to focus on?
   - [ ] Creation
   - [ ] Marking and grading
   - [x] Tracking progress
   - [ ] Other
5. When creating tests and/or exams, which of these sources would you prefer to get questions from? (assume obtaining questions from all options are the same in terms of difficulty)
   - ( ) The awarding exam board (including old specification questions)
   - ( ) A range of exam boards and qualifications of the same level
   - (•) Both of these
   - ( ) Other
6. What features would you like a bespoke exam creation tool to include? (give as much detail as possible)
   Facility to create exams from a database of questions at any given level
7. When marking exams, how do you prefer to store students' grades?
   - (•) Digitally
   - ( ) On paper
   - ( ) Other
8. What features would you like a bespoke exam marking tool to include? This includes the actual process of marking exams, storing the marks that students have achieved and generating grades (give as much detail as possible).
   1 - Database to store student results linked to a speadsheet for ease of analysis 2 - Ability to analyse a group's performance by question (i.e. which students scored maximum marks for any question/which students scored no marks for any question
9. What kind of insights would you find useful when tracking the progress of students in exams and tests?
   - [x] Colour coded tables
   - [ ] Bar graphs
   - [ ] Line graphs
   - [x] Comparisons to class average
   - [ ] Comparisons to historical data
   - [x] Future grade predictions
   - [ ] Other
10. What features would you like a bespoke exam tracking tool to include? (give as much detail as possible)
    Track student performance from Y7, colour coded
11. Any other things that you want mention?



1. For which of the groups below do you create and/or mark tests and/or exams for?

- [x] KS3 (Year 7-9)
- [x] Year 10
- [ ] Year 11
- [ ] Year 12
- [ ] Year 13

2. How many tests or exams do you create and/or mark per year?

13-15

3. Which part(s) of the exam process do you find the most difficult?

- [ ] Creation
- [x] Marking and grading
- [x] Tracking progress
- [ ] Other

4. Imagine you are creating a bespoke software application to help you with exam management. Which part(s) of the exam process would you want this application to focus on?

- [ ] Creation
- [x] Marking and grading
- [ ] Tracking progress
- [ ] Other

5. When creating tests and/or exams, which of these sources would you prefer to get questions from? (assume obtaining questions from all options are the same in terms of difficulty)

- ( ) The awarding exam board (including old specification questions)
- ( ) A range of exam boards and qualifications of the same level
- (•) Both of these
- ( ) Other

6. What features would you like a bespoke exam creation tool to include? (give as much detail as possible)

N/A

7. When marking exams, how do you prefer to store students' grades?

- (•) Digitally
- ( ) On paper
- ( ) Other

8. What features would you like a bespoke exam marking tool to include? This includes the actual process of marking exams, storing the marks that students have achieved and generating grades (give as much detail as possible).

Being able to annotate the students' working out

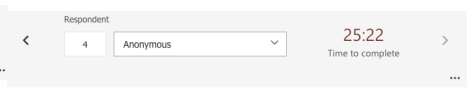
9. What kind of insights would you find useful when tracking the progress of students in exams and tests?

- [x] Colour coded tables
- [ ] Bar graphs
- [ ] Line graphs
- [x] Comparisons to class average
- [x] Comparisons to historical data
- [ ] Future grade predictions
- [ ] Other

10. What features would you like a bespoke exam tracking tool to include? (give as much detail as possible)

Be able to export data.

11. Any other things that you want mention?



1. For which of the groups below do you create and/or mark tests and/or exams for?

- [x] KS3 (Year 7-9)
- [x] Year 10
- [x] Year 11
- [x] Year 12
- [x] Year 13

2. How many tests or exams do you create and/or mark per year?

7-9

3. Which part(s) of the exam process do you find the most difficult?

- [x] Creation
- [x] Marking and grading
- [ ] Tracking progress
- [ ] Other

4. Imagine you are creating a bespoke software application to help you with exam management. Which part(s) of the exam process would you want this application to focus on?

- [x] Creation
- [x] Marking and grading
- [x] Tracking progress
- [ ] Other

5. When creating tests and/or exams, which of these sources would you prefer to get questions from? (assume obtaining questions from all options are the same in terms of difficulty)

- ( ) The awarding exam board (including old specification questions)
- ( ) A range of exam boards and qualifications of the same level
- (•) Both of these
- ( ) Other

6. What features would you like a bespoke exam creation tool to include? (give as much detail as possible)

A range of different question (Multiple choice questions/practical style questions/short and long style questions and mark schemes. Examiner reports where students have made common mistakes

7. When marking exams, how do you prefer to store students' grades?

- (•) Digitally
- ( ) On paper
- ( ) Other

8. What features would you like a bespoke exam marking tool to include? This includes the actual process of marking exams, storing the marks that students have achieved and generating grades (give as much detail as possible).

Spreadsheet holding scores/grades/generating targets for improvements

9. What kind of insights would you find useful when tracking the progress of students in exams and tests?

- [x] Colour coded tables
- [x] Bar graphs
- [ ] Line graphs
- [x] Comparisons to class average
- [x] Comparisons to historical data
- [x] Future grade predictions
- [ ] Other

10. What features would you like a bespoke exam tracking tool to include? (give as much detail as possible)

A system generating progress targets or steps to improve from question to question analysis

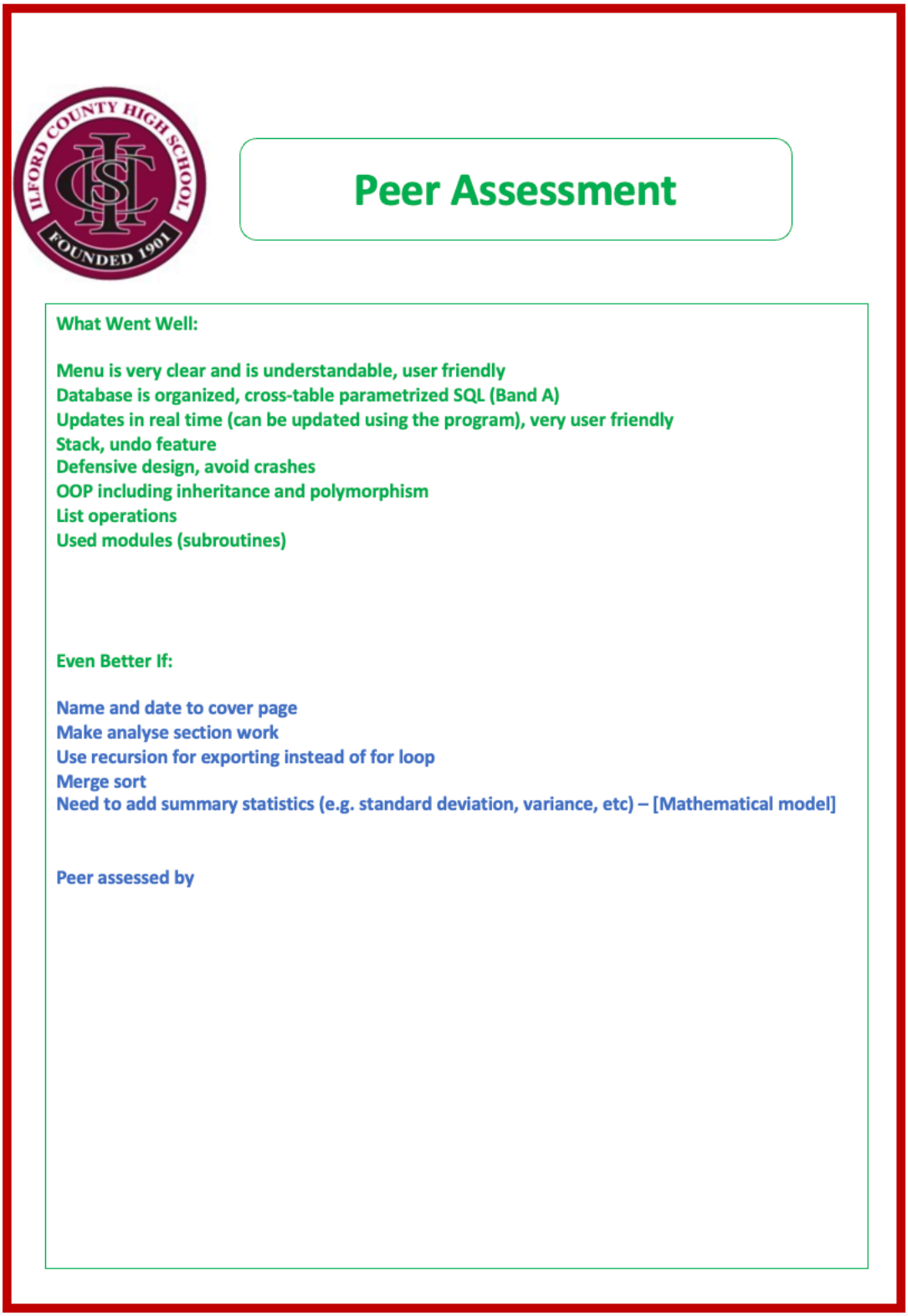
11. Any other things that you want mention?

None

Appendix 2: Pearson Edexcel Level 3 Advanced GCE in Mathematics Specification screenshot

2.3	**Interpret measures of central tendency and variation, extending to standard deviation.** **Be able to calculate standard deviation, including from summary statistics.**	**Data may be discrete, continuous, grouped or ungrouped. Understanding and use of coding.** **Measures of central tendency: mean, median, mode.** **Measures of variation: variance, standard deviation, range and interpercentile ranges.** **Use of linear interpolation to calculate percentiles from grouped data is expected.** **Students should be able to use the statistic x** $$S_{xx} = \sum(x-\bar{x})^2 = \sum x^2 - \frac{(\sum x)^2}{n}$$ **Use of standard deviation =** $\sqrt{\frac{S_{xx}}{n}}$ **(or equivalent) is expected but the use of** $S = \sqrt{\frac{S_{xx}}{n-1}}$ **(as used on spreadsheets) will be accepted.**

Appendix 3: Alpha build feedback from classmate 1



ILFORD COUNTY HIGH SCHOOL
FOUNDED 1901

Peer Assessment

What Went Well:

Menu is very clear and is understandable, user friendly
Database is organized, cross-table parametrized SQL (Band A)
Updates in real time (can be updated using the program), very user friendly
Stack, undo feature
Defensive design, avoid crashes
OOP including inheritance and polymorphism
List operations
Used modules (subroutines)

Even Better If:

Name and date to cover page
Make analyse section work
Use recursion for exporting instead of for loop
Merge sort
Need to add summary statistics (e.g. standard deviation, variance, etc) – [Mathematical model]

Peer assessed by

Appendix 4: Beta build feedback from classmate 2

WWW:

- Very well presented and easy to navigate
- Analysis functionality is very intuitive and insightful

EBI:

- Format questions so that it is easier for the user to read
- If you are adding a class and the user enters an ‘ the program will return an error - fixed
- If you chose to swap questions when there are no questions in the paper then the program will keep you stuck in a loop of “choosing a question to swap” - fixed
- Could make it so that for the analytics the user can choose what statistics they want to see (I.E look at mean, look at variance)
- Fix exporting when no questions in paper - fixed

Appendix 5: Final build feedback from the client

1. How easy is it to navigate the menu system?
- Easy since number based
- Easy to learn but time-consuming
- Once learnt it's easy to use
- GUI would be much easier
2. How easy is it to find the questions you are looking for?
- Separate the marks from the rest of the information
- Topic sections
- As a table format
- Link to examiner's report
- Separating calculator and non-calc
3. How easy is it to create the paper overall?
- Changing order is useful
4. How easy is it to export a paper?
- Easy to export
- Editable paper and front cover
- Mark scheme available on same paper
- Separating calculator and non calc
- Question numbers on the paper
5. How easy is it to add/edit marks of students?
- Importing spreadsheet would be useful
- Colour coded
- Analysing from imported spreadsheet
6. How easy is it to analyse students' performance?
- Colour-coded student mark table
- Dots overlapping in the graph
- WWW and EBI
- Report generating – one page for WWW and EBI
7. Do you have any other comments, feedback or improvements?
- Making it look nicer
- Year of the paper
- Questions
- Colour coded table
- Adding marks is useful