



A Level **Computer Science**

7517/C NEA

Report on the examination

Version: 1.0
June 2025

Further copies of this report are available from aqa.org.uk

AQA retains the copyright on all its publications. However, registered centres for AQA are permitted to copy material from this booklet for their own internal use, with the following important exception: AQA cannot give permission to centres to photocopy any material that is acknowledged to a third party even for internal use within the centre.

General Feedback

The second year of online submission allowed centres to submit their sample in what is hoped to be a much easier fashion. Centres are to be reminded that for each student the following should be submitted:

- Candidate Record Form – alongside the marks and signatures these forms can be used to identify resources used by a student. However, it is usually easier to identify sources in the actual project documentation especially if this is use of AI tools and the guidelines from JCQ must be followed. It is also to be noted that just referring to 'YouTube videos' is not enough detail when completing this form and this perhaps highlights the value of having a list in the actual documentation.
- Project Log – the AQA Project Log is used to provide the rationale for the marking, and it is a requirement to submit a form of marking rationale for each student in the sample.
- Project documentation – it is much easier for moderation if this is submitted as one PDF file.
- Testing videos – it is recommended to upload testing videos to the AQA sample submission portal rather than hosting on an external site such as YouTube. It is also recommended to compile into one video rather than submitting numerous short clips.

Each year whilst moderating it is impressive to see such a wide range of projects and we continue to see new ideas. This year moderation found projects making use of AI APIs and clever use of creating prompts and processing responses from these services as evidence of new skills being looked at by students.

There are many students who write about producing multi-user projects but then fail to think about the significance of where any database might be stored. It is common to see a well thought out client-server project making use of a local database and in many cases this would then not work for multiple users. It is also common to see an actual desktop application meant for multiple users implemented with a local database. If a student does this the networked database side could be scoped out in the analysis section and/or evaluated against carefully at the end with an improvement to the project to perhaps move to a remote database being considered.

Many students also look to produce 'apps' but then fail to actually produce a project that would work on a mobile device. This could be reflected on in the evaluation but for many students this consideration does not happen. It is also common for these mobile apps to have a login / registration process which probably is not required. However, some students do look at remote databases tools such as those systems provided by Google and produce apps that do work on a mobile device and make use of a remote database effectively.

A lot of students develop database driven projects, and these vary from very simplistic projects to quite complex data processing systems. A well-developed project will contain processing of data in some form as part of the core objectives. This could be summary reports, a kind of recommendation engine, tracking or invoicing to name a few but a small

number of projects do end up as just simple data entry and search systems. Students should also consider how a user would use a well-designed system. It is common to see usage of IDs for data entry when this is not really that appropriate in terms of how a user would work. It is also rare for students to consider carefully how deletions should work, with stronger students looking to delete all associated records from linked tables or utilising a way of archiving records.

It was pleasing to see a continuation in students exploring various libraries and environments. There is a useful opportunity for a student to perform research into these parts of their project in their analysis section. Whilst a few centres had students compare languages – so, for example, a centre might get all students to compare Python to C# and to VB, this is not really required. However, some research into a library and how it might be used for a specific project would be useful to see. This can then also lead into some initial modelling either in the research part or at the end of the analysis. For a few students this led to simple prototypes being built to demonstrate potential uses of libraries or APIs.

Analysis

The analysis section is the main focus for setting out the scope of the project and it is this scoping that students can sometimes fail to complete fully. Students can make good use of gathering requirements from potential users but often fail to scope out full details. Information, for example, as to how many books a library has, whether they have multiple copies of books, how many students are borrowing books, how many books can be borrowed at a time are a few questions that could be used to scope out a project yet many students will ask questions about the user interface and password reset procedures and not get into the details of the system to be developed.

Some centres clearly encourage students to identify libraries that will be used as part of their project or concepts such as the use of an API. This is good and those students who then also provide example usage can gain the higher level marks. For example, they might show an example API call and an example of the data returned by the API.

The initial modelling continues to be a variable part of the analysis section. Where students complete this well it clearly provides initial glimpses into the ideas and thoughts behind the development of the project. Centres should not think of the initial modelling as a 'one size fits all' and end up with DFDs or data dictionaries for every project. It was pleasing to see students produce prototypes where they tested out ideas for their project making use of simple library calls or an API. Students who just submit their code for the prototype do not really provide as much modelling as those who also show the output from their code. If appropriate, students could even submit a short video of their prototype to show it working.

Documented Design

This section continues to allow students to demonstrate their knowledge and understanding. The aim is for the student to provide information to help another developer create code for the project, but it is sometimes clear that students might not know that this is the expectation and instead spend time just explaining code rather than designing a solution.

Those students who can identify the principles behind how their project works and express these clearly can score very well in this section. Expressing how a project works may include various forms of diagrams where appropriate. As mentioned last year, when providing user interface design work there is ample opportunity to also explain how the output is to be created – ie the algorithms necessary to get the output onto the screen. It is also a shame to still see students separating out the design into chunks such as ‘example SQL statements’ which in AQA documentation is really just a statement that we would expect example SQL statements to appear across the design. So, when doing a user interface design for a reporting element of a project at the same time the SQL needed and the algorithms to produce the output could all be placed together.

The inclusion of flowcharts can really help explain complex processes. However, the student who only includes flowcharts for login / registration / account management and then fails to include them for the core functionality of their project is not using flowcharts well.

It continues to be the case that students can fail to explore the complex parts of their project and choose to just focus on easier bits such as login and registration. So, for example, a project that makes extensive use of an external API should provide exemplar calls to this API, example returned data and then discuss how this data will be processed but this level of detail is often missing. It is to be assumed that a developer looking at the design would be able to create registration / login procedures but would need to know how to utilise any API and/or libraries being used by the project.

Skills and Completeness

These sections continue to generally be well assessed by centres. There is still a concern about the use of ‘borrowed’ code, be this from tutorials, third-party repositories or generated by AI tools and centres must encourage students to clearly distinguish between code that has been taken from another source and the code written by themselves. Good practice, where appropriate and feasible, can be to run a ‘viva’ with students where they can be asked to explain their project code and also perhaps demonstrate it, to help with any completeness mark being awarded.

There are many students who paste in screenshots of their code and this is not the best way to present it. Students can just copy and paste their code into their documentation or make use of a ‘code to PDF’ tool that could then be submitted.

There are also centres who appear to encourage students to create a code diary or to show their code with explanations. For the NEA, it is preferred for code explanations to be in the design section with the student discussing at a higher level how the code works rather than explaining the written code line-by-line. A diary of code iterations or line-by-line explanations rarely gains credit in the NEA and often sits alongside a short and weak design section. The time invested in the diary or explaining code line-by-line would be better spent in the documented design section.

Whilst a copy of the full code is needed this really only needs supporting with perhaps some cover page(s) where the student can pick out technical skills and provide a page number as to where these can be found in the code.

Completeness is about both whether a student is completing their objectives set in the analysis section and what might be expected of an A-level student to complete for that particular kind of project. A student who completes all of their objectives might not be able to secure full marks in completeness if their objectives are not at an appropriate level.

Testing

Video testing is now the preferred way for providing evidence for a project and for the majority of students this proves to be completed very well. It would be preferable for the submission to be one compiled video rather than many videos (sometimes upwards of thirty). It continues to be the case that a commentary over the video adds significant value. For some students, a video is sometimes just a demo of their project rather than what might be expected for formal testing.

A submitted video should be supported by a test plan with timestamps for the video. For most students, a video walkthrough of their project can also provide evidence of objectives being met but there might be a requirement to repeat small segments with different data. It is also important for a student to show that they are checking the output produced and this can be completed in a variety of ways. For database projects it is common to see evidence of the database being changed. For networked projects it can be useful to see multiple clients opened and whilst this can be on one device it is perhaps more convincing to see multiple devices operating at the same time. Sometimes checking against another solution can also be appropriate – testing a chess engine against an online chess player, testing a calculator against another calculator, or just manually checking calculations for a project such as a SUVAT simulation.

It is pleasing to see a group of students look to move to formally include unit testing strategies as part of their submission.

Evaluation

The evaluation section can be completed very well by those students who carefully consider their project and how it matches the ideas put across in their analysis section.

It is very useful for a student to start with an 'overall evaluation' where they can consider honestly their project against the ideas put across in their analysis section.

Many students fail to really evaluate their objectives but just provide a met / not met tick sheet. A good student will take time to reflect on the key objectives and consider whether they have been met in an appropriate way, highlighting weaknesses that might exist.

User feedback can vary in quality. Better feedback from a user allows the student to judge against the objectives of the project and provide honest reflection whereas it can be common for weaker evaluations just to have very general comments from a user.

When considering improvements, it is pleasing to see students take time to provide thoughts about how some of these could be implemented and this does help students move into the top level for an evaluation.

Mark Ranges and Award of Grades

Grade boundaries and cumulative percentage grades are available on the [Results Statistics](#) page of the AQA Website.