# Government of South Australia LogoSACE Board Logo2024 Digital Technologies Subject Assessment Advice

Overview

This subject assessment advice, based on the 2024 assessment cycle, gives an overview of how students performed in their school and external assessments in relation to the learning requirements, assessment design criteria, and performance standards set out in the relevant subject outline. It provides information and advice regarding the assessment types, the application of the performance standards in school and external assessments, and the quality of student performance.

Across the assessment types for this subject, students can present their responses in oral or multimodal form, where 6 minutes is equivalent to 1000 words. Students should not speed-up the recording of their videos to condense more content into the maximum time limit.

From 2024, if a video is flagged by markers/moderators as impacted by speed, schools will be requested to provide a transcript and markers/moderators will be advised to mark/moderate based on the evidence in the transcript, only considering evidence up to the maximum word limit (e.g. up to 2000 words for AT3).

If the speed of the recording makes the speech incomprehensible, it affects the accuracy of transcriptions and it also impacts the ability of markers/moderators to find evidence of student achievement against the performance standards.

The Subject Renewal program has introduced changes for many subjects in 2025; these changes are detailed in the change log at the front of each subject outline. When reviewing the 2024 subject assessment advice, it is important to consider any updates to this subject to ensure the feedback in this document remains accurate.

# School Assessment

Teachers can improve the moderation process and the online process by:

* ensuring student submissions are oral/multimodal presentations only, as per the subject outline.
* ensuring that only approved video files (See ‘Accepted File Formats on the SACE website’) or PPT files are uploaded, i.e. no Zip files containing code or digital solutions
* avoiding large PowerPoints with embedded videos when a screen recording of the presentation and submitting as a single video file is possible, and can achieve the same result
* avoiding uploading unnecessary content which is not part of the presentation or designer’s statement, such as PDFs or Word documents containing large files of code as the student should have already recorded a multimodal presentation
* supporting student’s use of their natural voice in presentation recordings: developing presentation skills is a part of the course experience. Allow students to use computer generated voice overs only if genuinely necessary (such as a Special Provisions arrangement). Computer generated voice overs can be difficult to understand, and does not allow expression to be heard.

Assessment Type 1: Project Skills Task

Project skills should enable students to create solutions of interest to them. As part of the collaborative task, it is critical that students can showcase their own individual contribution to the project.

Teachers can elicit more successful responses by:

* in the set of tasks, ensuring that there is at least one assessed performance standard from each of the design criteria (‘computational thinking’, ‘development and evaluation’ and ‘research and ethics’)
* providing tasks that allow students to solve problems in unique ways
* ensuring students use an appropriate object-oriented, general purpose programming language, which is crucial for demonstrating complex coding concepts. Please refer to the ‘Digital Technologies Technical Glossary’ on the subject mini site for more information.

The more successful responses commonly:

* showed a wide range of computational thinking skills including flowcharts, pseudocode, and class diagrams with complexity
* used flowcharts or tables to explain abstraction
* clearly showed how their computational thinking skills (flowcharts, pseudocode, class diagrams) were then applied to their code, i.e. use of functions, objects etc
* showed detailed steps in breaking down an original abstract idea into a potential solution
* used a general-purpose, object-oriented programming language that included complex branching, loops/nested loops, and use of libraries as well as arrays/lists
* showcased complex programming, which includes code that the student had developed and not merely “adapted”, and used live coding to showcase their deep understanding of the code, and the ways they troubleshooted issues that arose
* utilised complex datasets and using tools like spreadsheets or JSON
* were able to organise and analyse data, converting raw data into meaningful information and then supporting their findings with other research
* was clear what role that student took for the collaborative task, and their contribution to the task was explained with clear evidence of how they collaborated (for example, GANTT charts and meeting minutes)
* ensured the collaborative tasks were suitable for collaboration where parts could be allocated to members that allowed all members to demonstrate all performance standards
* clearly showed modifications and improvements in the iterative project development task between iterations, in response to feedback from clients/users as well as developmental milestones
* used multiple stakeholder feedback to critically evaluate the effectiveness of the Digital Solution / prototype
* evaluated the effectiveness of the digital solution (including both the negatives and positives) versus just evaluating their performance in the task
* highlighted and discussed innovative features.

The less successful responses commonly:

* did not include core computational thinking concepts in the structure charts, such as looping and selections, instead showing navigations
* did not express complexities in algorithmic design, the flowcharts were more in the form of block diagrams
* showed limited or no computational thinking skills
* did not use complex datasets, which limited the ability to perform effective analysis
* only outlined the advantages and disadvantages for the ethics task, without discussing implications and potential solutions
* implied iterative development, but no evidence of testing, modification and feedback was evident. Had limited documentation and explanation of how they planned, tested and refined their solution
* used template code without adding unique modification or solutions, typically by following tutorials rather than creating a unique solution
* only displayed the code on screen without explanation, missing the opportunity to show understanding
* used coding in the programming task and iterative project development task that were the same, which limited the student’s ability to show more advanced coding concepts in the second task
* used basic input and output for digital solutions, with no manipulation of data or complex calculations, so code was very simplistic
* gave an overview of how the program worked as a whole without discussing how individual features were developed to achieve specific aims
* used a visual block coding platform (such as Makecode-Micro:Bit); which does not allow for demonstration of using a general purpose programming language
* did not revisit the initial problem or inquiry, therefore not discussing potential innovation
* over-emphasised on how the student felt about the project, rather than to what extent the digital solution addressed the initial problem.

Assessment Type 2: Collaborative Project

The collaborative project must identify a client and outline the problem, with a showcase of the iterative work undertaken. The presentation should evaluate the group’s work and clearly showcase the individual student contribution to the overall project.

The AT2 Collaborative Project requires two multimodal submissions per student. This has been further clarified in the 2025 Subject Adjustments, which can be found in the subject outline.

The first, with a total time of 5 minutes per student, should be a live recorded presentation of the student, to a client or users, where they explain the digital solution (product, prototype, or proof of concept), including showing their computational thinking, their project evaluation and their role in and contribution to the project, supported by evidence.

The second multimodal submission, up to 3 minutes in length, should be a short walk-through video of the code, and a demonstration of the working final solution.

Teachers can elicit more successful responses by:

* ensuring students are individually filmed presenting the project. This is often best done in front of a projector or display screen
* ensuring students have a client for their problem
* ensuring students submit two files as stated above.

*The more successful responses commonly:*

* had a clear client and showed evidence of working with this client for feedback and demonstrated the solution to the client
* utilised a problem that was big enough and complex enough to allow each member to show evidence of each performance standard
* included strong evidence of their own contributions and their role in the project
* showed computational thinking skills in more than one way (e.g. flowcharts with pseudocode and class diagrams) and then clearly showed how this was then applied to their code, i.e. use of functions, objects etc.
* had two clear multimodal submissions, the 5min presentation which included evidence of computational thinking, working with the client, iterative project development and group work; and a 2-3min video showing the solution working and highlighting main features in the code that student produced including innovative features
* used an object-oriented programming language, beginning with a problem that was complex to start with, so that the coding showed complex structures
* the iterative project development approach was applied, showcasing strong problem-solving skills with clear evidence of effective communication with clients, and refinement of digital solutions based on feedback.

The less successful responses commonly:

* had no presentation which was evident to a client or otherwise
* was not clear which part of the solution the individual student had created
* had too much focus on analysing data, and not on creating and coding a digital solution
* the digital solution was not big enough to split evenly amongst students in the group and therefore not all performance standards could be met by each student
* contained limited or no evidence of collaboration
* had no or limited computational thinking skills evident
* students used tools like ChatGPT to generate code without demonstrating a deep understanding of its functionality
* no innovative features were highlighted
* the problem was provided by the teacher, not necessarily of interest to the students, and hence did not employ a real client’s situation.

# External Assessment

The investigation needs to focus on solving a problem of interest, with the development process clearly shown. A client (real or fictitious) is not necessary for AT3. Students should be encouraged to identify a problem of interest to solve.

Reminder, please take note of de-identifying student materials for AT3 and not including assessment marking or grades.

Teachers can elicit more successful responses by:

* ensuring students look to solve a problem of interest. The problem identified should be clearly articulated at the beginning of the presentation and deconstructed throughout the presentation
* ensuring students submit a Designer’s Statement as well as their main presentation.

Assessment Type 3: Investigation

The more successful responses commonly:

* students developed the Designer’s Statement as a presentation, with the focus on the project, where they provide their Evaluation
* designer’s statement showed reflecting on the success of all the features, outlining how it met the client’s needs, and considered future features and versions of the software.
* chose a complex problem that was of interest to them and was not already available as a tutorial. This enabled the student to show evidence of computational thinking and applied programming concepts
* clearly showed computational thinking, including abstraction by using Class diagrams, flowcharts, pseudocode as well as mock-up designs of their solutions
* had planned algorithm designs and then explained as well as showed how they were then converted into code
* iterative development was shown by breaking up the development stage into specific developmental milestones. Within each milestone they showed testing, error fixing, feedback from others and how they made changes based on that feedback
* highlighted where programming had been used to solve complex problems, not just routine problems, using an appropriate general-purpose, object-oriented programming language
* incorporated core programming techniques such as selections and iterations, paired with data structures such as arrays/lists
* broke the program down and explicitly stated core features of the program
* evaluated the success of the project based on core goals or features that were set, and how well the implementation of them went, paired with insightful upgrades for future versions
* clearly highlighted innovative features and explained why there were innovative
* referred back to original scope of the project within the presentation
* demonstrated the final product with appropriate voice over, after a full development cycle process was shown
* addressed all appropriate performance standards.

The less successful responses commonly:

* did not show evidence of all performance standards, in particular little evidence of computational thinking (design mock-ups, flowcharts, pseudocode, class diagrams) or the evidence was not detailed
* problems were not complex to start with, so students did not have the opportunity to show detailed and complex code
* evidence of iterative development consisted only of the iterations of their solution and maybe a table showing user testing
* did not include a designer’s statement
* no evidence of the actual code or how it works, was a digital solution only
* more evidence on the visual design rather than computational thinking
* evidence of code with no explanation
* included structure charts that did not include selections or iterations, instead were navigational
* included long ethical outlines and discussions that did not contribute to performance standards
* did not set clear goals or features, resulting in weaker abstraction as well as weaker evaluations
* did not outline changes between iterations, such as new planning or directions based on the previous iterations
* used template code for the solution, or incorporated template code without explaining changes and adaptations
* used ChatGPT to generate code without adequately refactoring in context and explaining its functionality
* did not mention innovative features
* coding was very simplistic with simple loops, conditional statements, and functions
* read code line by line, rather than demonstrating their understanding of the code
* did not show the final product working.

General

Submissions must be oral/multimodal presentations; information in word documents may be referred to within a presentation discussion but are not suitable for submission on their own.

Developing a student’s ability to demonstrate computational thinking, in addition to their project skills, is crucial; many of the less successful responses do not feature any evidence of Computational Thinking.

Please ensure that the LAP accurately reflects the performance standards assessed on each task.

Please make note of the subject adjustments which apply to Digital Technologies, which can be found in the current version of the subject outline.