# Generative AI in Assessments: Before, During, and After

**Lee Clift**

Department of Computer and Information Science, University of Strathclyde
lee.clift@strath.ac.uk

**Olga Petrovska**

School of Mathematics and Computer Science, Swansea University
olga.petrovska@swansea.ac.uk

## Introduction

You have likely seen the impact of Generative AI (GenAI) on your career as an academic, whether through journals discussing GenAI authorship or students attempting to get away with using Large Language Models (LLMs) to write essays for them. This chapter discusses the way to harness the power of GenAI to make your life as an educator easier, particularly in the context of planning, conducting, and marking assignments. We will discuss how you can use GenAI to write and refine assignments and solutions, how it can be encouraged as an exploratory tool by students, and how you can use the tool to speed up marking large class sizes. All of the following tips are LLM-agnostic; you can use them with any model you prefer, such as ChatGPT[140], Claude[141], or Copilot[142].

### GenAI and LLM Basics

Throughout this chapter, we will go into detail regarding the use of GenAI and LLMs. Put simply, GenAI refers to computer systems capable of creating new content, such as text, code, or images, based on patterns they have learned from vast amounts of existing data. This allows them to answer questions and make new content for us, based on what already exists (such as information on the internet). LLMs are a specific type of GenAI focused on understanding and generating human language. In this chapter we use the terms GenAI and LLM interchangeably as all our examples are referring only to text-based interactions with GenAI.

It is important to highlight that our approach is LLM-agnostic. This means the techniques and prompts we discuss can be applied to whichever model you prefer, whether that is ChatGPT, Google Gemini[143], or Apple Intelligence[144]. While different models often have specific strengths, we have found them all to be highly capable of performing the assessment-related tasks we outline in the following sections. If at any point during this chapter you feel overwhelmed or confused, we invite you to pause, open up your LLM of choice (such as ChatGPT or Gemini), and simply experiment with it. Explore its outputs and try a few small tasks yourself. As with many practical skills, the best way to learn is through immersion and exploration — not just by reading about it here!

## Before Assessment

Imagine you need to design an assessment — whether it is a coursework, a lab task, or an exam. Depending on the level of involvement required, designing this assessment can be a time-consuming process. Between thinking of a fun task, writing it out, devising solutions, creating rubrics, and formatting, it quickly adds up. GenAI can help you streamline this process as it offers several time-savers. Let's look at each of those in detail.

### Using LLMs To Draft Multiple Choice Questions

Multiple choice questions (MCQs) are convenient in a variety of situations: you may want to quickly check content comprehension during a lecture, create a revision resource, or design a scalable assessment for a large cohort, which is free of grading bias. The idea behind MCQs is attributed to Wood and Thorndike and dates back to the 1920s (Engelhard 1988). One can design their own questions from scratch or refer to existing question banks. With the emergence of LLMs, educators have started exploring GenAI's capabilities to brainstorm ideas and accelerate the creation of MCQs. Some companies even developed custom-made solutions for MCQ generation and other teaching-related tasks, e.g. TeacherMatic[145]. Yet, these systems often require additional licenses that

[140] https://chatgpt.com/
[141] https://claude.ai/
[142] https://copilot.microsoft.com/
[143] https://gemini.google.com/
[144] https://www.apple.com/uk/apple-intelligence/
[145] https://teachermatic.com/generators-teaching/

your institution may not have. This, however, is not an issue and you still can generate MCQs using conventional general LLMs.

Tran et al. (2023) compared the capabilities of two OpenAI models to generate isomorphic MCQs, providing the Canterbury Question Bank[146] as a reference. Biancini, Ferrato, and Limongelli (2024) experimented with creating MCQs with LLMs, providing a prompt that addresses three basic components of MCQs: *stem* (context for the question), *key* (correct answer), and *distractors* (incorrect options). While the prompt they provide is specifically designed for history-related questions, it can still be used as a starting point for other subjects.

## Practical Tip: Grounding Your Questions

To get the most accurate results, do not ask the LLM to generate questions from its general knowledge. Instead, provide it with your specific teaching materials, such as a lecture PDF, a PowerPoint, or a book chapter. This ensures the questions align exactly with what you have taught.

**Example: MCQ Generation**

**Prompt example:** Copy and paste the following prompt into your LLM after uploading your source material:

*You are an assistant who doesn't make mistakes. If a reference model is presented to you, you follow it perfectly without making errors. Create a university-level quiz for [year x] students studying [subject] based on the provided file.*

*You must strictly adhere to the following format without any errors:*
*[ Insert the question ]*
*a) [ Option A ]*
*b) [ Option B ]*
*c) [ Option C ]*
*d) [ Option D ]*

*Correct Answer: [Insert the letter corresponding to the correct answer, for example: 'a)']*

*Source: [Write the exact line or passage from the provided file where the information for this question can be found.].*

*Note that you are allowed to modify only the parts within brackets ([...]) in the format provided.*
*Ensure that all four options are distinct.*

**Example output:** Based on a lecture on Python programming, the LLM might produce:

```
Which keyword is used to
create a function in Python?
a) function
b) method
c) def
d) create

Correct Answer: c)


Source: "To define a function
in Python, we use the 'def'
keyword followed by the
function name." (Slide 14)
```

## Reflection: The Importance of Human Oversight

Crafting effective prompts that specify the desired format of GenAI output and refer to relevant subject content is crucial for drafting acceptable MCQs. Yet, human oversight is always recommended, especially if the MCQs refer to less common programming languages. Educators should be the final authority on the quality.

Interestingly, though, a comparative study of human-crafted MCQs in programming versus those generated by AI has shown that the latter were actually more aligned to the learning objectives (Doughty et al. 2024).

## Using GenAI To Write an Assessment Brief

Courseworks are usually rather involved assignments and require a detailed brief. The prompt "recipe" is similar to that of MCQs: provide a *context* (specify the subject, the learning objectives, the audience, and the level at which this subject is assessed), and determine a *format* (by either manually describing it or providing a sample document for reference). You can also use LLMs to help refine your prompt by asking them to serve as your guide — something some LLM's, e.g. GPT-5[147], already offer. Once you've got the brief, review and refine it with your human insight.

## Practical Tip: Providing Examples

You can use GenAI to generate an assignment by following these steps:

1. Determine and define the key aspects: the assignment type, the concepts you wish to test, the level of study, and a potential theme. If you do not have a theme, you can ask an LLM for a suggestion.
2. Provide some key teaching material you have used (either as PDFs, PPTs or code snippets) to help "ground" your assignment. When doing so, ensure that the LLM you are using does not use your content for training and does not make it publicly accessible. Educational institutions often have special licensing agreements with specific LLM providers to protect institutional data, so it is worth checking whether your university has such an agreement and which provider it covers.
3. Provide a previous example (such as last year's coursework) that an LLM can use to format your brief similarly to your previous assignments, so they all look uniform.

[146] https://web-cat.org/questionbank/

[147] https://openai.com/index/introducing-gpt-5/

4. Do not be afraid to ask for multiple suggestions or request changes. Review and refine the LLM output, and then follow institutional moderation guidelines.

**Example: Assessment Brief**

**Prompt example:** Copy and paste the following prompt after uploading your materials:

*Acting as a university module leader, draft a coursework assessment brief for [Level of Study] students in [Subject].*

*Use the provided [Teaching Materials] to ensure the task is relevant and use the [Previous Year's Brief] as a reference for the structure and tone.*

*The brief must include: A clear 'Task Description'. Specific 'Learning Outcomes' being assessed. 'Submission Requirements' (e.g., file formats, word counts). A 'Theme' for the project (If I haven't provided one, suggest three creative options).*

**Example output:** Based on a Mobile App Development module, the LLM might produce:

```
Task Description: Develop a
cross-platform mobile
application using Flutter.


Learning Outcomes:
Demonstrate proficiency in
asynchronous programming and
UI/UX consistency.


Suggested Theme: A "Campus
Sustainability Tracker" where
students map recycling
points.


Submission: A GitHub
repository link and a 5-
minute video demonstration.
```

## Reflection: The "human-in-the-Loop" Workflow

While GenAI is excellent at formatting and brainstorming themes, it lacks the context of your specific cohort's progress or institutional regulations. Some LLMs (such as GPT-4o[148] or Claude 3.5[149]) can even act as a guide, asking you clarifying questions to refine the brief further. Once the output is generated, always review it against your institutional moderation guidelines to ensure it remains a valid and fair instrument of assessment. Feel free to ask it questions, or to make modifications against your own style or institutional policies, such as late work submission policy.

## Using GenAI To Design and Format a Marking Rubric

Once you have your brief written, you can get it to assist in writing a rubric. By stating the number of marks you want to award and some key learning objectives, you can quite easily create a marking rubric with discrimination between grades — you can even request this in various formats, such as a table or a standalone document.

## Practical Tip: Building Step By Step

It is good practice to design your rubric in a step-by-step fashion. Here is a quick algorithm to do this:

1. Determine the key aspects that you want to evaluate — these will form the dimensions in your marking rubric. At this stage, you may prompt an LLM to provide suggestions based on specific learning objectives.
2. For each dimension, decide on the number of points it should be worth, making sure that they add up to a full grade across all dimensions.
3. For each dimension, prompt the LLM to break it down, depending on the number of points allocated for this dimension. Specify how fine-grained your rubric should be. Consider whether you want to base it on grade boundaries, the number of points, or work with ranges.
4. Review and refine the LLM output, then add it to the marking rubric.

**Example: Marking Rubric**

**Prompt example:** After uploading your Assessment Brief, use the following prompt:

*Based on the provided assessment brief, create a marking rubric in a table format.*

*Use the following dimensions: [Insert Dimensions, e.g., 'Technical Execution' and 'Critical Reflection'].*

*Allocate [Insert Points] to each dimension. For each, provide detailed descriptors for the following grade boundaries: [Insert Boundaries, e.g., 70+, 60-69, 50-59, 40-49, and <40]. Ensure the descriptors show clear progression and discrimination between the levels.*

**Example output:** For a 'Technical Execution' dimension (100 points), the LLM might produce:

```
Grade Descriptor:


70+ (Distinction)
Flawless execution. Code is
optimized, fully commented,
and follows all best
practices.


60-69 (Merit)
High-quality execution with
minor inefficiencies. Logic
is sound and mostly
commented.


40-49 (Pass)
Functional but unoptimized.
Basic requirements are met
with some structural errors.
```

## Reflection: Refining the Rubric

While an LLM is excellent at generating "scaffolded" text, it often relies on generic adjectives like "good" or "very good." Review the output to ensure that a "70+" in your rubric specifically describes what a top-tier student in your field looks like. Additionally, you may not use terms like "good" and opt to use percentages or grade boundaries instead. Once refined, the LLM can also reformat this rubric into a standalone document or a table ready to be pasted into your Virtual Learning Environment (VLE).

[148] https://openai.com/index/hello-gpt-4o/

[149] https://www.anthropic.com/news/claude-3-5-sonnet

### Using GenAI To Develop a Model Solution

Finally, we can request a model solution that incorporates both of these components. This will vary significantly depending on the assignment type; it works best for tests and less well for essays. A typical rule of thumb is that the larger the assessment, the worse the model answer will be. Regardless, these model answers can be used to quickly estimate what a solution may look like — we can even request multiple solutions from different ability levels! This will help us understand the typical LLM style for solving the problem and provide a better idea of when students overrely on GenAI output.

This can also serve an additional purpose — to test an LLM's capability to solve your coursework. This can be a good way to see how "LLM-proof" your assignment is, especially if the use of GenAI is prohibited. This can aid in either redeveloping aspects of an assignment or in developing an understanding of how an LLM may assist students in completing this specific assignment.

### Practical Tip: Combining Tools

When asking for a model solution from an LLM, using both an assignment brief and a rubric can aid in getting better results. By using the rubric as a reference point, you can be specific about the model answer you are requesting, such as an answer that would receive 40% versus 60%, and quantify the differences.

**Example: Model Solutions**

**Example prompt:** Input your Brief and Rubric first, then use this prompt:

*Act as a student attempting the provided assessment. Using the [Assessment Brief] and the [Marking Rubric] as your guide, generate two separate model solutions:*

*A 'High Distinction' (80%+) response that follows every criterion perfectly.*

*A 'Pass' (45-50%) response that meets the basic requirements but lacks depth in [Insert Specific Dimension, e.g., 'Analysis' or 'Optimization'].*

**Example output:** For a programming task, the LLM might produce:

```
High Distinction: [Code which includes full error handling, modular code, and a detailed README.]


Pass: [Code which runs and completes the basic task, but lacks comments, uses "hard-coded" values, and has no error catching.]
```

### Reflection: The "LLM-Proofing" Test

A general rule of thumb is that the larger and more open-ended the assessment, the more the model answer's quality will degrade. However, by generating these solutions, you can determine if your assignment is too easily solved by a single prompt. If an LLM provides a perfect 80%+ answer instantly, you may want to redevelop certain aspects, such as adding a reflective component or a specific real-world constraint, to ensure the assignment remains a true test of student skill.

## During Assessments

While all the above uses of GenAI revolved around helping you to write an assignment, we should not forget that using LLMs effectively is becoming a desired skill (Petrovska, Clift, and Moller 2023). Hence, we need to prepare students for the real-world challenges, where businesses are proactively integrating LLMs into their processes. How do we do this? By offering them more authentic assignments that encourage them to explore different aspects of LLMs, students can learn to use these tools as helpers and not all-knowing oracles.

In our experience, students generally enjoy exploring GenAI as part of an assessment activity, provided it is not mandatory. Some students are ethically opposed to using GenAI; therefore, offering it as an optional assistive tool allows those who wish to engage with it to do so, without requiring participation from those who prefer not to use it. In cases where GenAI usage is required (such as reflective exercises — covered in the upcoming sections), providing examples for students to reflect without having to use the tool themselves can help mitigate any student concerns.

### Tasks Around Interaction With GenAI and Critical Evaluation of Its Output

This can be anything from asking students to compare their own code with GenAI output or asking GenAI to provide several versions of a problem solution and encouraging students to determine which one is the best (Petrovska et al. 2024). Alternatively, students can be given a set of solutions — some human-written and some generated by GenAI — without being told their origin (Petrovska, Pearsall, and Clift 2025), and asked to identify the best solution and explain the drawbacks of the others. Such hands-on experience and exposure to GenAI's imperfections should, hopefully, build a more conscientious approach to using these tools.

## Reflective Components in Programming Assignments

Historically, programming assignments have been evaluated based on whether the software could pass test cases and whether the code quality met a certain standard. From our informal discussions with other academics who teach programming, many begin to reconsider this approach. Understanding the code and being able to explain it becomes a more accurate criterion for assessing skill acquisition. If this resonates with you, consider incorporating various reflective components into your assessments, outside of typical code comprehension and understanding. Ask yourself: are students aware of the ethical and social implications of using GenAI? Are they aware and concerned about copyright and other legal implications? Are they observant and conscious about the impact of GenAI on their learning? Programming assignments do not have to be purely about programming, so feel free to add some extra tasks in the form of a reflective log, a video, or any other format that you find appropriate.

## Allowing Unlimited (But Documented) Use of GenAI

In one of our experimental studies (Clift and Petrovska 2025), we have allowed students to use GenAI without restriction for their mobile application development projects. A word of warning: this type of exposure is not recommended for novice programmers, as their metacognitive skills and self-efficacy may not be well developed. They may not have acquired the required fundamental skills to fully appreciate and understand the benefits GenAI can offer or recognise its pitfalls. When applied to experienced programmers (for example, final-year degree students), the inclusion of, and unlimited access to, these tools can provide a constructivist learning environment in which students explore how the tools work and how to integrate them effectively into their own development processes. Results show that allowing GenAI usage during assessments leads to larger, more ambitious solutions being attained, while also encouraging students to learn about GenAI tools and their capabilities (Clift and Petrovska 2025).

## Using GenAI as an Instant Feedback Tool

Alongside the explicit use of GenAI during a summative assessment, there is also potential to integrate GenAI into a more freeform, formative assessment process. This would allow students to explore and utilise GenAI during activities such as lab sessions, which are lower-risk and more casual than a typical summative assessment. One key example of this is CS50[150] — Harvard University's Intro to Computer Science class. The class is run both in-person and remotely, and is currently the most popular Intro to CS class in the world. CS50 introduced a GenAI model trained off their own resources called CS50 Duck[151] (Liu et al. 2024). CS50 Duck allows students, either remotely or in person, to seek instant help on problem sheets during formative assessments, while retaining the metacognitive benefits from reinforcing theory through problem sessions. This is achieved through the style of answers provided by CS50 Duck, including guards that prevent students getting a direct answer, instead helping them work towards the answer by asking questions, similar to those of a lab demonstrator or teacher.

## Practical Tip: Prompting for an AI Tutor

Naturally, not all of us have access to the resources of CS50 or Harvard, and therefore, it can not be expected that we can build a system like CS50 Duck. However, by utilising an existing LLM, we can create something similar, through clever uses of prompts and by providing our taught content. This would allow us to provide an LLM to our students, enabling them to use it to solve problems rather than have it solve them.

**Example prompt**

*Act as a Tutor for a computer science student.*

*I am working on [Topic/Problem]. My current logic is: [Insert Student Logic/Code].*

*Your Rules: Do NOT give me the correct code or the direct answer. Ask me a guiding question that helps me identify the flaw in my own logic. If I am stuck, give me a small hint related to [Insert Specific Concept or topic].*

[150] https://pll.harvard.edu/course/cs50-introduction-computer-science

[151] https://marketplace.visualstudio.com/items?itemName=CS50.ddb50

## After Assessments: Writing Feedback

Finally, we have received the students' summative assignments, and one of the most dreaded tasks is now here: marking! We do not endorse using GenAI to mark for you, although some educators have explored such possibilities (Banihashem et al. 2024) and several ed-tech companies are testing feedback generation capabilities of their GenAI solutions. One needs to understand that certain ethical and copyright concerns arise from submitting students' work to an LLM. Hence, we encourage continuing to mark the usual way, but utilising GenAI to help you create feedback that is detailed and constructive. With larger classes, providing all students with well-written, detailed feedback is a significant challenge and a time-consuming task. Here is a way to speed up the process while still keeping your feedback customised for each student.

### Practical Tip: The "Feedback Persona"

Using GenAI, you can create a "Feedback Persona" simply by "teaching" a standard LLM your personal style.

**The Setup:** Open a new chat session in your LLM of choice. Paste 3–5 examples of your own past, high-quality feedback.

**The Instruction:** Provide your LLM with a description and some examples of your feedback style.

> This is my feedback style. It is encouraging but rigorous. I will provide you with bulleted marking notes, and I want you to turn them into two coherent paragraphs of feedback addressed to the student.

**The Workflow:** For each student, simply paste your shorthand notes. This keeps the feedback customised to the student's actual work while saving you the "blank page" fatigue of drafting 100+ individual paragraphs of feedback.

**Example: Generating Feedback**

**Example prompt:** Once you have established your style in the chat, use this prompt for each submission:

*Using the feedback style I previously provided, transform these marking notes into a constructive feedback response for a student.*

*Marking Notes:*

*[Note 1: e.g., Great use of Python libraries, very efficient.]*

*[Note 2: e.g., Missed the requirement for error handling on line 42.]*

*[Note 3: e.g., Next time, try to modularize the functions.] ...*

*Ensure the tone remains [Insert Tone, e.g., Professional and encouraging].*

**Example output:** The LLM transforms your notes into:

```
You have demonstrated a
strong grasp of Python
libraries in this assignment;
your implementation was
notably efficient and well-
structured. However, I
noticed that the error-
handling requirement was
missing around line 42, which
is critical for program
stability. For your next
project, I encourage you to
focus on modularising your
functions further to improve
code reusability.
```

### Reflection: Ethics and the "Human-in-the-Loop"

When using a "Feedback Persona", data privacy is paramount. Never input a student's full name, ID number, or sensitive personal details into a public LLM. As mentioned earlier, some institutions may provide you with a more secure, local version of an LLM for safer use. Please read your institution's policy on GenAI and LLM usage before attempting this.

It is also important to consider what staff-generated materials are shared with commercial LLMs. Uploading assignment briefs, marking schemes, or model solutions to public systems may result in this content being incorporated into future model training data, depending on the provider's policies. This creates a risk that students could later access high-quality or even complete solutions via the same tools. Where possible, opt out of data sharing (sometimes referred to as incognito or private mode), use institutionally approved systems, or avoid uploading full solutions altogether.

Treat the AI-generated text as a "first draft" — always review it to ensure it accurately reflects your marking notes before sending it to the student. Transparency is also key; informing students that you use GenAI to help synthesise your handwritten notes into formal feedback builds trust and models professional GenAI usage. In our experience, students have been okay with this, and when they have challenged it, we have provided them with the raw feedback notes to show that marking has been done manually.

## Conclusion

That's it; using those tips, you can adapt your assessment process to integrate this new tool into your practice, regardless of assessment type or field. We expect to see this field change and evolve. Many of the tips and prompts we have suggested can also be used to design simple GenAI agents — small, task-focused systems built on top of an LLM. In tools like Copilot these agents can be created in an accessible, streamlined form that effectively "pre-trains" the model to understand the structure, style, and type of output you want, without you having to repeat detailed instructions each time. Once you feel comfortable with the basics of using GenAI, exploring these lightweight agents can be a powerful way to simplify your workflow.

## References


Banihashem, S. K., Nafiseh Taghizadeh Kerman, O. Noroozi, Jewoong Moon, and Hendrik Drachsler. 2024. “Feedback Sources in Essay Writing: Peer-Generated or AI-Generated Feedback?” *International Journal of Educational Technology in Higher Education* 21: 23. https://doi.org/10.1186/s41239-024-00455-4.

Biancini, G., A. Ferrato, and C. Limongelli. 2024. “Multiple-Choice Question Generation Using Large Language Models: Methodology and Educator Insights.” In *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization (UMAP Adjunct’24)*, 584–90. New York (NY): Association for Computing Machinery. https://doi.org/10.1145/3631700.3665233.

Clift, L., and O. Petrovska. 2025. “Learning Without Limits: Analysing the Usage of Generative AI in a Summative Assessment.” In *Proceedings of the 9th Conference on Computing Education Practice (CEP’25)*, 5–8. New York (NY): Association for Computing Machinery. https://doi.org/10.1145/3702212.3702214.

Doughty, J., Z. Wan, A. Bompelli, J. Qayum, T. Wang, J. Zhang, Y. Zheng, et al. 2024. “A Comparative Study of AI-Generated (GPT-4) and Human-Crafted MCQs in Programming Education.” In *Proceedings of the 26th Australasian Computing Education Conference (ACE’24)*, 114–23. New York (NY): Association for Computing Machinery. https://doi.org/10.1145/3636243.3636256.

Engelhard, G. 1988. “Thorndike’s and Wood’s Principles of Educational Measurement: A View from the 1980’s.” Washington (DC): National Academy of Education. https://eric.gov/?id=ED295961.

Liu, R., C. Zenke, C. Liu, A. Holmes, P. Thornton, and D. J. Malan. 2024. “Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education.” In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education. Volume 1 (SIGCSE 2024)*, 750–56. New York (NY): Association for Computing Machinery. https://doi.org/10.1145/3626252.3630938.

Petrovska, O., L. Clift, and F. Moller. 2023. “Generative AI in Software Development Education: Insights from a Degree Apprenticeship Programme.” In *Proceedings of the 2023 Conference on United Kingdom & Ireland Computing Education Research (UKICER’23)*. New York (NY): Association for Computing Machinery. https://doi.org/10.1145/3610969.3611132.

Petrovska, O., L. Clift, F. Moller, and R. Pearsall. 2024. “Incorporating Generative AI into Software Development Education.” In *Proceedings of the 8th Conference on Computing Education Practice (CEP’24)*, 37–40. New York (NY): Association for Computing Machinery. https://doi.org/10.1145/3633053.3633057.

Petrovska, O., R. Pearsall, and L. Clift. 2025. “Assessing Software Engineering Students’ Analytical Skills in the Era of Generative AI.” In *Proceedings of the 9th Conference on Computing Education Practice (CEP’25)*. New York (NY): Association for Computing Machinery. https://doi.org/10.1145/3702212.3702223.

Tran, Andrew, Kenneth Angelikas, Egi Rama, Chiku Okechukwu, David H. Smith IV, and Stephen Macneil. 2023. “Generating Multiple Choice Questions for Computing Courses Using Large Language Models.” In *2023 IEEE Frontiers in Education Conference (FIE)*, 1–8. College Station, TX, USA: IEEE. https://doi.org/10.1109/FIE58773.2023.10342898.