(I'll focus on using the LLM to populate the quiz section, rather than using it for code)

There are several degrees of granularity which you could use an LLM for in this context. Starting from the broadest usage (with the most manual followup) and working down from there:

Identify Topics

As implemented, currently each question has its own topic. For example, in my implementation specifically, this was one SIT305 question, followed by four maths questions (each with their own mini-category based on the operation being performed).

An LLM could be used as a brainstorming tool for categories of questions. A variation of this would be to supply a question that you already have in order to identify an appropriate topic for that given question (which would make it easier to get – whether via LLM or not – further questions on the same topic).

Create new questions for a supplied topic

Incrementing the granularity, we could supply a topic to the LLM in order to generate appropriate questions. This could perhaps be done in a guided way, such as by supplying supplemental instructions (about e.g., length of the question), or in a more free-form depending on preference.

A variation of this would be to again supply the LLM with a question of our own, but to then ask it to either:

- Generate questions on the same topic, or
- Generate questions of a similar style, length, format, etc (not necessarily on the same topic)

Generate answers for a supplied question

The obvious point for this level of granularity would be to get the LLM to generate the *correct* answer for the question, but LLMs are infamously not good at doing that reliably – at least for some topics. So you *could* use an LLM for this, but if you care about the answers being correct you would need to double-check them yourself anyway, which wouldn't really be saving time or effort.

Without burning a lot of time and electricity figuring out which topics the given LLM is good/bad at answering correctly (or simply sticking to likely-safe topics without

explicitly testing it), **I think a more suitable use here would actually be to generate *incorrect* answers.**

Our quiz app has one question with three answers, two of which are wrong and one of which is correct. If we manually get the answers (regardless of the source of the question) in order to maintain answer accuracy, we can get an LLM to make the other two. This could be particularly useful for topics in which the developer is not familiar with, and thus might not be good at generating plausible-sounding – but ultimately incorrect – answers.

Generate entire quiz contents

While I wouldn't personally recommend this for a production app, you could theoretically use an LLM to, e.g.:

- Generate a number of topics
- Generate a number of questions for these topics
- Generate correct and incorrect answers to each of these questions

And then auto-populate the quiz app's contents using the above. Instead of pointing to a hardcoded string or a string resource, the relevant code could point to the result of an LLM response function. You might need to have a parsing function as an intermediate, but nonetheless the concept should be doable.